Page No.:
Date:
YOUVA

UNIT IV
Implementation Techniques

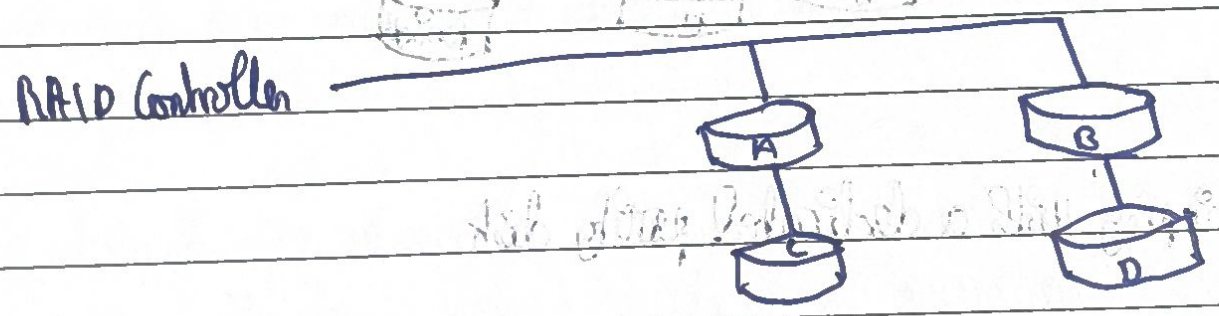
RAID (Redundant Array of Independence disks)

*) It is way of connecting multiple secondary disk as one
*) The OS see these as one logical disk
*) Data is distributed across these disks, in case of failure parity information can help recover the data

Types of RAID:

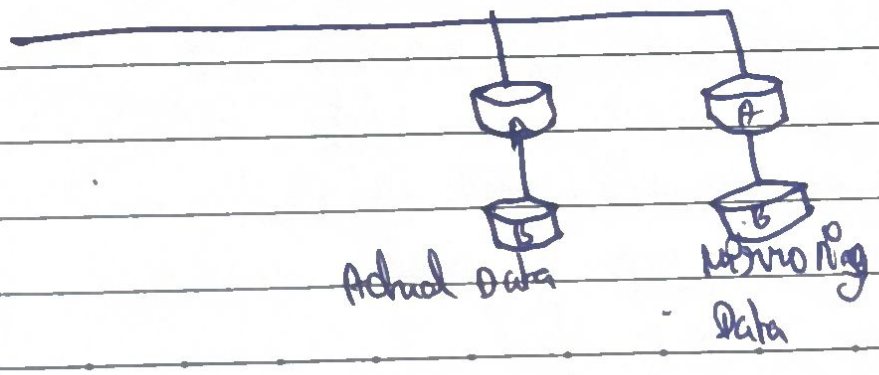
o) RAID 0 (Striping):

o) Splits data into blocks and spreads them across all disks
o) No replicates of data
o) Focus performance



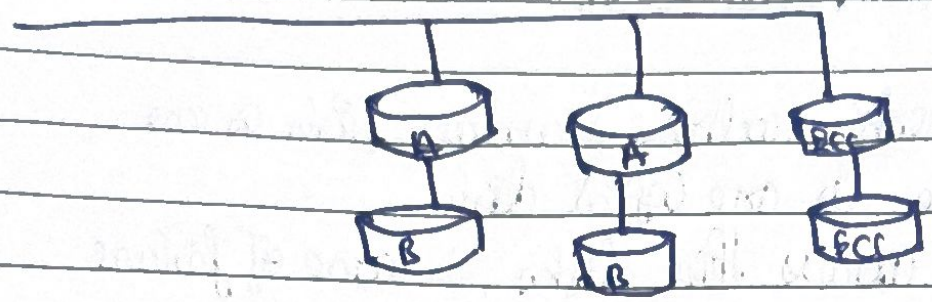
o) RAID 1:

o) All Data are duplicated in other disk
o) 100% redundancy
o) Half used for actual data, other half is used for mirroring
o) If one disks fails other has same data



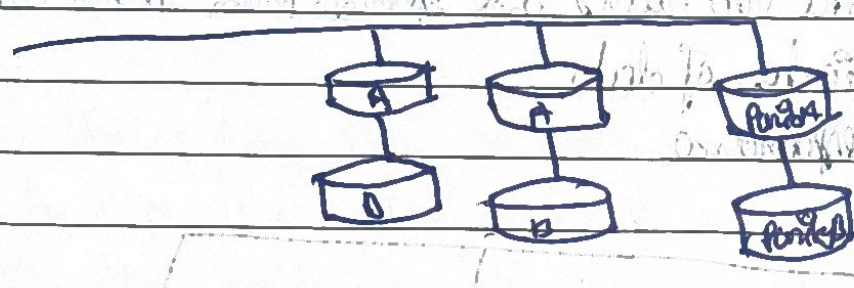
RAID 2:

*) uses Memories and Error Correcting Codes
*) Data is stripped across different disks, ECC on other disks
*) Complex and Expensive



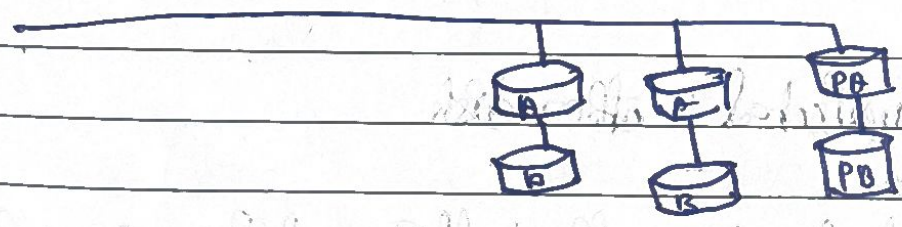
RAID 3:

*) Byte-level Stripping with a dedicated Parity disk
*) Parity disk is used to recovers data of last
*) If one disk fails, remaining disk and Parity recovers data



RAID 4:

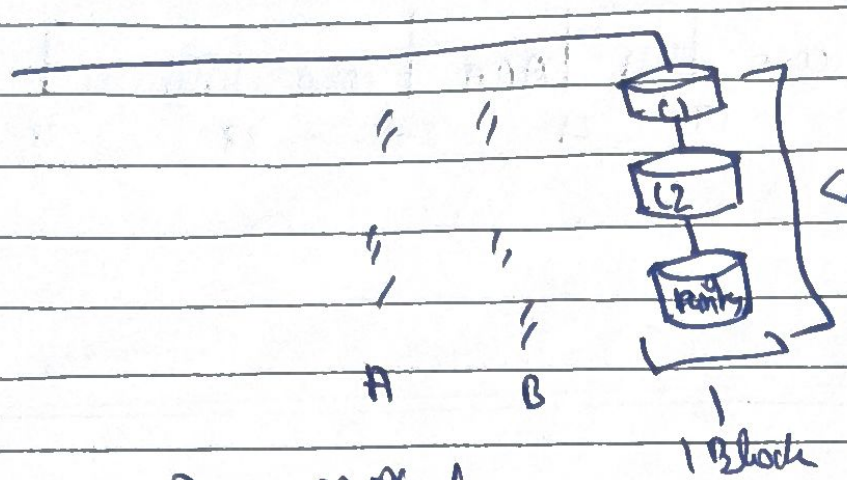
a) Block-level Stripping with a dedicated parity disk



Same as RAID-3

RAID 5:

* Parity disks are distributed Among the other disks
* Improved performance



Draw similarly

RAID 6:

Same as RAID 5 but has two parity disks each
Diagram also same but each Block has two parity disk

File organization

* A file organization is a method of arranging records in a file when the file is stored on disk. A file is organized logically as a sequence of records is a sequence of fields

Types of of Records in file organization

* Fixed length Record : the length of the records are fixed nothing more or less
* A Variable length Record : It can be flexible and

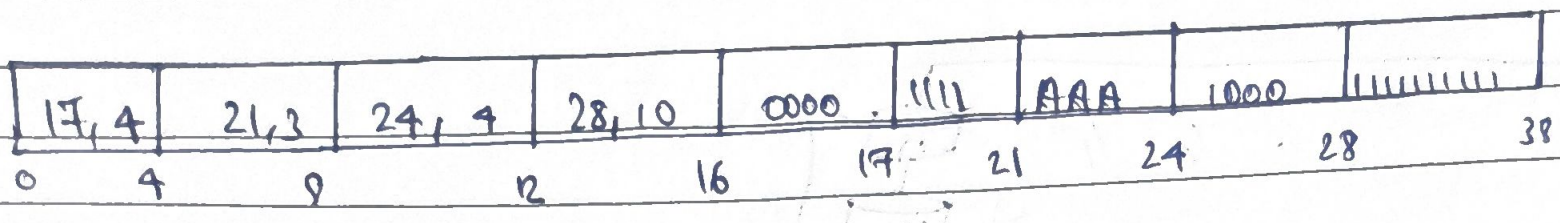
Fixed length Record Example: Emp

EMPNO	ENAME	Salary	Phone
1111	AAA	1000	1111111111

EMPNO(4)
ENAME(3)
Salary(5)
phone(10)
} lengths are fixed

Variable length Record:

Null Bit map



Organization of Records in Files

There are three commonly used approaches

1) Heap file organizing:

*) Any record can be placed whenever there is space in the file
*) There is no ordering for record placement
*) used when insertion are frequent and searching order is not important

2) Sequential file organization:

*) Records are stored in sequential order based on Searchkey
*) Inserting or deleting requires adjusting pointers or Adding records to overflow blocks
*) Maintaining strict physical sequential order is difficult after multiple insertion and deletions
*) Deletions can be managed using pointer chains
*) Insertion use free space if present, otherwise record goes in an overflow block and pointer chain is updated

M T W T F S S

Page No.: YOUVA

Date:

3) Hashing file organization:

*) A hash function determines the location for each record in the file.
*) Efficient for direct access based on key value, since searching involves applying the hash function

4) Multitable clustering file organization:

*) Records of several different relations stored together in a single file
*) Useful for join operations between tables
*) Results in variable size records; pointer chains can be added to track addresses of next records

Data Dictionary Storage

* A data dictionary is a part of database management system that stores and manages metadata, which is a data about database itself

purpose & use:

*) Assists database administrators in managing the database
*) Stores critical information about the database's structure design, users, transactions and usage statistics
*) Controls and documents all aspects of the database
*) Generates reports as needed

Types of Data Dictionary:

a) Active: managed automatically by the DBMS.
- *) Always consistent with current database structure
- *) Modifications in DBMS are automatically reflected in the data dictionary
- *) usually derived from the System Catalog.

b) Passive Data Dictionary:
- *) used mainly for documentation purposes
- *) It is a self-contained application or files set for documenting the database environment
- *) maintenance are manual
- *) changes in DB requires manual updates

Column oriented Storage

*) Store data by columns instead by rows

Features and advantages:

- *) only required columns for query are retrieved
- *) Data for each column is stored together
- *) Efficient for queries that involves operations on columns
- *) Capable of handling very large amounts of data

Row oriented → Column oriented

ID | Name | Subject | Marks → ID | Subject | Marks or ID | Name

Indexing And Hashing

* An Index is a data structure that organizes disk data records for efficient retrival
* The Search key for an Index comprises one or more fields used to locate data quickly
* Indexes speed up search operation but add space and maintanence cost

Types of Indices:

1) Ordered Indices: Based on sorted ordering values
2) Hash Indices: use a hash function for uniform distribution of values across buckets

Ordered Indices:

Primary Index: Index that has fields of primary key, sorted in order
Clustering Index: Index on non-primary key columns; records with similar characteristics are grouped
Dense Index: holds pointers to actual record
Sparse Index: Index records created for some records
Single Level Indexing: Auxiliary file using binary search; can be primary clustering or secondary
Multilevel Indexing: Large indexes are broked into several similar indices the top level can reside in memory for faster access

M T W T F S S

Page No.: YOUVA

Date:

B+ tree

*) In a B+ tree, we have to store the data in leaf nodes only.

*) It does not waste space

*) Search operations of any data is very easy in B+ tree as all data is found in leaf node

*) B+ tree stores redundant search key

Characteristics:

① The B+ tree is a balanced tree and the operations insertion & deletion keeps the tree balanced

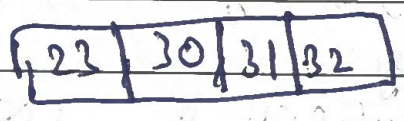
② A minimum occupancy of 50 percent is guranteed for each node except root node

③ Searching for a record requires just traversal from the root to appropriate leaf

Example. B+ tree for the following data:

30, 31, 23, 32, 22, 28, 24, 27 where number of pointer that fit in one node is 5

Step 1: Insert 30, 31, 23, 22



M T W T F S S

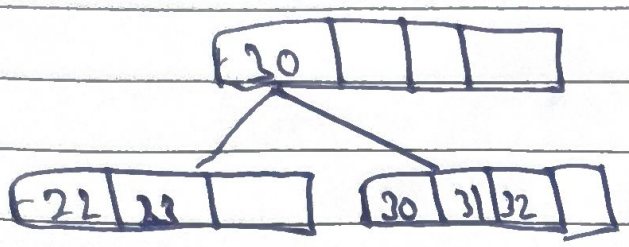
Page No.:

Date:

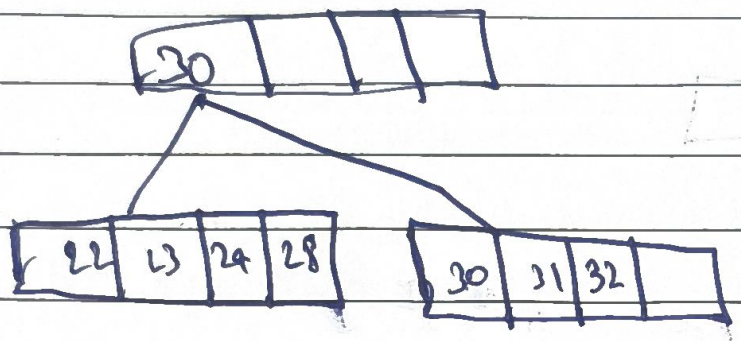
YOUVA

Step2: Now Insert 22 the sequence will be

22, 23, 30, 31, 32

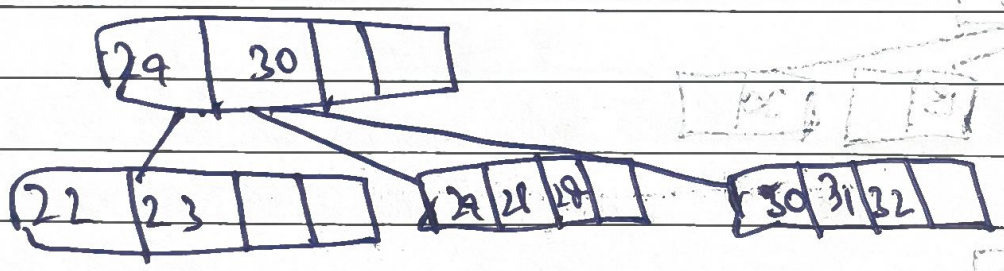


Step 3: Insert 28, 24 in ascending order



Step4: The sequence become 2223, 24, 28, 29,

the middle will go up



deleting something will reverse there process

B Tree

*) On B tree we can store both the key and data are stored in the internal leaf node

*) It waste space

*) Searching becomes difficult

*) B-tree does not store redundant search key