16.1. What is the difference between primary and secondary storage?

Following are the differences between primary and secondary storage:

Primary storage	**Secondary storage**
The CPU can directly access the primary storage devices.	The CPU cannot directly access the secondary storage devices.
Fast access to data is provided by the primary storage devices.	Slower access to data is provided by the secondary storage devices.
The storage capacity is limited.	The storage capacity is larger.
Cost of primary storage devices is high than the secondary storage devices.	Cost of secondary storage devices is low than the primary storage devices.
Examples of primary storage are main memory and cache memory.	Examples of secondary storage are hard disk drive, magnetic disks, magnetic tapes, optical disks and flash memory.

16.2. Why are disks, not tapes, used to store online database files?

To store online database files, we use disks. Disks are secondary storage device; a disk is a random access addressable device.

Data us stored and retrieved in units called disk blocks while come to the tapes.

Tapes are the sequential access addressable device.

16.3. Define the following terms: *disk, disk pack, track, block, cylinder, sector, interblock gap, and read/write head.*

Disk: The disk is the secondary storage device that is used to store the huge amount of data. The disk stores the data in the digital form i.e., 0's and 1's. The most basic unit of data that can be stored in the disk is bit.

Disk pack: The disk pack contains the layers of hard disks to increase the storage capacity i.e., it includes many disks.

Track: In the disk, the information is stored on the surface in the form of circles with various diameters. Each circle of the surface is called a track.

Block: Each track of the disk is divided into equal sized slices. One or more such slices are grouped together to form a disk block. The block may contain single slice (sector). The size of the block is fixed at the time of disk formatting.

Cylinder: In the disk pack, the tracks with the same diameter forms a cylinder.

Sector: Each track of the disk is divided into small slices. Each slice is called as sector.

Interblock gap: The interblock gap separates the disk blocks. The data cannot be stored in the interblock gap.

Read/write head: The read/write head is used to read or write the block.

16.4. Discuss the process of disk initialization.

Process of disk initialization:-

In the disk formatting / initialization process, tracks are divided into equal size. It is set by the operating system.

Initialization means,

The process of defining the tracks and sectors, so that data and programs can be stored and retrieved.

While initialization of the disk, block size is fixed, and it can not be changed dynamically.

16.5. Discuss the mechanism used to read data from or write data to the disk.

When the disk drive begins to rotate the disk when ever a particular read or write request is initiated and once the read/unit head is positioned on the right track and the block specified in the block address moves under the read/write head. The electronic component of the read/write head is activated to transfer the data.

Below procedure is follows when the data is read for or write from disk.

(1) The head seeks to the correct track

(2) The correct head is turned on

(3) The correct sector is located.

(4) The data is read from the hard disk and transferred to a buffer RAM

16.6. What are the components of a disk block address?

Disk block address:-

Data is stored and retrieved in units called disk blocks or pages.

Address of a block:-

Consists of a combination of cylinder number, track number (Surface number with in the cylinder on which the track is located. Block number (with in the track) is supplied to the disk

16.7. Why is accessing a disk block expensive? Discuss the time components involved in accessing a disk block.

The data is arranged in an order and then stored in a block of the disk is said to be known as blocking. The data can be transferred from the disk to the main memory in units.

Accessing the data in the main memory is less expensive than accessing the data in the disk. This is due to the following components:

- Seek time.
- Rotational latency.
- Block transfer time.

The access of the data in the disk is more expensive because of the time components. The time components are explained as follows:

- **Seek time:**

o The disk contains a set of tracks. Each track has one head. This is said to be known as disk head. The track is formed by sectors of fixed size.

o A sector is said to be known as a small sub-division of a track present on the disk.

o Each sector can store up to 512 bytes of data in which the user can access the data.

o For reading the data present in the disk, there is an arm on the disk. This is used to read a record from the disk.

o Seek time is said to be known as the total time taken to position the arms to the disk head present on the track.

o Accessing a disk block takes more seek time. Therefore, this is one of the major reasons for the expensiveness of accessing a disk block.

- **Rotational latency:**

o Latency is said to be known as time delay.

o The total amount of time taken between the request for an information and how long it takes the disk to position the sector where that data is available. This is said to be known as rotational latency.

o This is also said to be a waiting time in which if this time increases, then the expensiveness of accessing a disk block will also increase.

• **Block transfer time:**

o If there is need to transfer the data in the disk from one block to another block then it will take some time.

o This time is said to be known as block transfer time. At the time of accessing a block of data from the disk, the transfer time may increase. This will result in the expensiveness of accessing a disk block.

16.8. How does double buffering improve block access time?

Improve block access time using duffer buffer:-

Double buffering is used to read a continuous stream of blocks from disk to memory.

Double buffering permits continuous reading or writing of data on consecutive

Disk blocks, which eliminates the seek time and rotational delay for all but the first block transfer. Moreover, in the programs data is kept ready for processing and it reducing the waiting time.

Double buffering processing time is np

Where n blocks

p processing time/block

16.9. What are the reasons for having variable-length records? What types of separator characters are needed for each?

Variable-length records (Reasons) A file is a sequence of records. All records in a file are of the same type and in same size. If different records in the file have different size, the file is said to be made up of variable – length records.

A file may have variable-length records for sever reasons.

* File records are of the same record type, but one or more of the fields are of different size.

* File records are same record type, but one or more of the fields are optimal.

Means, they may have some values but not for all.

* File contains records of different record types and varying size. If related records of different types were placed together on the disk block it would be occur.

Type of separator characters are need for each:-

In the variable length fields, each record has a value for each field. But we do not know the exact length of some field values. To determine the bytes with in that record it represent each field. Then we can use separator characters like

? or % or $

Types of separator characters:-

Separating the field name from the field value and separating one field from the next field.

For this we use three types of characters.

= Separate field name from field value

▌ Separate fields

▩ Terminates record

Name=John, peeter ▌ Ssn=012345 ▌ DEPARTMENT=CS ▩

Example

= Separate field name from field value

▌ Separate fields

▩ Terminates record

Name=John, peeter ▌ Ssn=012345 ▌ DEPARTMENT=CS ▩

Here we use three separator characters. That are

= Separate field name from field value

▌ Separate fields

▩ Terminates record

Name=John, peeter ▌ Ssn=012345 ▌ DEPARTMENT=CS ▩

= Separate field name from field value

▌ Separate fields

▩ Terminates record

Name=John, peeter ▌ Ssn=012345 ▌ DEPARTMENT=CS ▩

and

= Separate field name from field value

▌ Separate fields

▩ Terminates record

Name=John, peeter ▌ Ssn=012345 ▌ DEPARTMENT=CS ▩

16.10. Discuss the techniques for allocating file blocks on disk.

Techniques for allocating file Blocks on Disk:

Many techniques are there fore allocating the blocks of a file on Disk. In that

→ Contiguous allocation

→ Linked allocation

→ Clusters

→ Indexed allocation.

Contiguous allocation:-

File blocks are allocated to consecutive disk blocks. This makes reading the whole file very fast using double buffering.

Linked allocation:-

Each file block contains a pointer to the next file block. It is easy to expand the file but makes it slow to read the whole file.

Clusters:-

Combination of two allocates of consecutive disk blocks. Clusters are sometime called as file segments/extends.

Indexed allocation:-

One or more index blocks contain pointers to the actual file blocks.

16.11. What is the difference between a file organization and an access method?

Difference between a file organization and an access method

File organization:-

- It shows “how the physical records in a file are arranged on the disk.

A file organization refers to the organization of the data of a file in to records, blocks, and access structures

- In this, records and blocks are placed on the storage medium and interlinked.

Access methods:-

How the data can be retrieved based on the file organization.

It provides a group of operations and that can be applied to a file.

Some access methods apply to a file organization and can be applied only to file organization in certain way.

16.12. What is the difference between static and dynamic files?

Difference between static and dynamic files:-

Static file:- in the file organization update operations are rarely performed.

While come to dynamic files,

→ It may change frequently,

→ Up date operations are constantly applied to them.

16.13. What are the typical record-at-a-time operations for accessing a file? Which of these depend on the current file record?

Typical record at a time operations are:

1.) Reset: Set the file pointer to the beginning of file.

2.) Find (locate): Searches for the first record that satisfies a search condition. Transfer the block containing that record into memory buffer. The file pointer points to the record in buffer and it becomes the current record.

3.) Read (Get): Copies current record from the buffer to the program variable in the user program. This command may also advance the current record pointer to the next record in the file, which may necessitate reading the next file block from disk.

4.) FindNext: Searches for next record in file that satisfies the search condition. Transfer the block containing that record into main memory buffer. The record is located in the buffer and becomes current record.

5.) Delete: Delete current record and updates file on disk to reflect the deletion.

6.) Modify: Modifies some field values for current record and eventually update file on disk to reflect the modification.

7.) Insert: Insert new record in the file by locating the block where record is to be inserted, transferring the block into main memory buffer, writing the record into the buffer, and eventually writing buffer to disk to reflect insertion.

Operations that are dependent on current record are:

1.) Read

2.) FindNext

3.) Delete

4.) Modify

16.14. Discuss the techniques for record deletion.

Techniques for record deletion:-

We may delete a record from the file using following techniques.

(1) → A program must first find it's block

→ Copy the block in to a buffer

Delete the record from the buffer and then,

Rewrite the block back to the disk.

By using this, it leaves the unused space in the disk block. When we use this technique for deleting the large number of records result is wasted in storage space.

(2) Another technique for record deletion is deletion marker (deletion is to have an extra byte or bit). In this

Setting the deletion marker to a certain (deleted) value.

A different value of the marker indicates a valid record

→ Search programs and consider only valid records in a block.

These two deletion techniques requires periodic reorganization.

During this reorganization. The file blocks are accessed consecutively and records are packed by removing deleted records.

For un ordered file, we use either spanned or un spanned organization and it is used with either fixed-length or variable-length records.

16.15. Discuss the advantages and disadvantages of using (a) an unordered file, (b) an ordered file, and (c) a static hash file with buckets and chaining. Which operations can be performed efficiently on each of these organizations, and which operations are expensive?

(a) **an unordered file:**

It can be defined as the collection of records, those are placed in file in the same order as they are inserted.

Advantages:

• It is fast, and Insertion of simple records are added at the end of the last page of the file.

• It is easy to get the records from the unordered file.

Disadvantages:

• Blank spaces may appear in the unordered file.

• It will take time to sort the records in unordered file.

b) an ordered file:

An ordered file, it is stores records in order and it will changes the file when records are inserted.

Advantages:

• Recording a sequential based file is more capable as all the files are being stored as the order.

• Helpful when large volume of data is present.

Disadvantage:

• Rearranging of file would be needed for storing or modifying or deleting any records.

c) static file hashing:

Advantages:

• The speed is the biggest advantage and it is efficient when huge volume of data is present.

Disadvantages:

• Difficult to implement the static file hashing.

The hashing technique is the most efferent to be executed and is expensive process because of is sophisticated structures.

• The extendable hashing is a type of dynamic hashing, which splits and associate the bucket of the database size for change, because when a hash function is to be adjusted on a dynamic basis.

• There is a cache which is an added advantage for faster improvement of information's.

16.16. Discuss the techniques for allowing a hash file to expand and shrink dynamically. What are the advantages and disadvantages of each?

Hashing techniques are allow the techniques for dynamic growth and shrinking of the number of the file records.

Techniques that are include the dynamic hashing, extendible hashing and linear hashing.

In the static hashing the primary pages are fixed and allocated sequentially, pages are never de-allocated and if needed pages of overflowed.

This technique use the binary representation of the hash value $h(k)$.

In the dynamic hashing, the directory is a binary tree, the directories can be stored on disk, and they expand or shrink dynamically. Directory entries point to the disk blocks and that contain the stored records.

Dynamic hashing is good for the database and that grows and shrinks in size, and hash function that allows dynamically.

In dynamic hashing, extendable hashing is the one form. It generates the values over a large range typically b-bit integers with $b = 32$

Hash function tht allows only prefix to index into a table of bucket addresses.

Example

Let the length of the prefix be I fits,

I must be in the limts between 0 and 32

Bucket address table size is 2^i here initially i is 0.

Now, value of *i* grows and shrinks as the size of the database.

The number of buckets also changes dynamically because of coalescing and spilitting of buckets.

Advantages and disadvantages of hashing techniques:-

(1) Advantages of static hashing:-

→ Static hashing uses a fixed address space, and perform computation on the internal binary representation of the search key.

→ Using bucket overflow, static hashing is redused, and it can not be eliminated.

Disadvantages:-

→ Here data base grow is with in time. And if initial number of buckets is too small, then performance will degraded.

→ If data base is shrinks, than again space will be wasted.

Extendible hashing:-

Advantages:-

→ It is a type of directory

→ Hash performance dose not degrade with growth of a file

→ Minimal space is over headed.

Disadvantages:-

→ Bucket address table may it self become big.

→ Changing size of bucket address table is an expensive operation.

Linear hashing:-

Advantages:-

→ It avoids the directory by splitting.

→ Overflow pages not likely to be long

→ Duplicates handled easily

→ It allows a hash file to expand to shrink its number of buckets dynamically with out a directory file.

Disadvantages:-

Linear hashing handles the problem of long overflow chains without using a directory and handles duplicates.

16.17. What is the difference between the directories of extendible and dynamic hashing?

The differences between directories of extendible and dynamic hashing are as follows:

Extendible hashing	Dynamic hashing
Access structure along with the file is stored.	Access structure based on binary tree data structure is stored.
A flat directory is maintained using the notation of global depth d with an array of 2^d bucket addresses.	A directory is maintained as a tree structure with two types of nodes, internal node with two pointers and leaf nodes.
The integer number formed from the first d bits of the hash value form the address.	The n or n-1 high order bits form the addresses of the buckets.

16.18. What are mixed files used for? What are other types of primary file organizations?

A mixed file refers to a file in which contains records of different file types.

• An additional field known as record type field is added as the first field along with the fields of the records to distinguish the file to which it belongs.

• The records in a mixed file will be of varying size.

The uses of mixed file are as follows:

• To place related records of different record types together on disk block.

• To increase the efficiency of the system while retrieving related records.

The other types of primary file organization are as follows:

• Unordered file organization

• Ordered file organization

• Hashed/file organization

• Sorted file organization

• Hashed file organization

• Indexed (b-tree) file organization

16.19. Describe the mismatch between processor and disk technologies.

In computer systems, the collection of data can be stored physically in the storage medium.

• From the **DBMS** (DataBase Management System), the data can be processed, retrieved, and updated whenever it is needed.

• The storage medium structure in a computer will have some storage hierarchy to make the process of collections of data.

There are two main divisions in the storage hierarchy of a computer system.

• Primary storage

• Secondary and tertiary storage

Primary storage:

• This storage medium in the computer can be **directly accessed by the CPU** (Central Processing Unit), it can be stored only as temporarily.

• The primary storage is also called as **main memory (RAM)**.

• In main memory, the data can be accessed faster with faster cache memories but less storage capacity and cost-effective.

• Please note that in case of any power failures or browser crash, the contents of the main memory will be erased automatically.

Secondary and tertiary storage:

• This storage medium in the computer can be **stored permanently in the way of disks**, tapes, CD-ROMs, or DVDs.

• The secondary storage is also called as secondary memory or Hard Disk Drives (ROM).

• In today's world, the data can be stored in offline considered as removable media, it is called as tertiary storage.

• It will store the data as a **permanent medium of choice**.

• The data cannot be accessed directly in this type of storage, at first it will be copied to primary storage and then the CPU processes the data.

The Mismatch between processor and disk technologies:

In computer systems, the processing can be done by RAM which is having a series of chips.

• For an efficient performance, the faster memory is provided to the processor.

• Also, the processor has the support of cache memory to retrieve the information faster which will be an added advantage.

In computer systems, the disk technologies need the space to accumulate the data.

• In disk technologies, the collection of data can be stored physically.

• The data cannot be accessed directly in disk type technologies, at first it will be copied to primary storage and then the CPU processes the data.

• When it is compared to the processor, the time consumption will be more, and the processor is better to run the processes.

Hence, the processor will provide efficient performance better than the disk technologies.

16.20. What are the main goals of the RAID technology? How does it achieve them?

To increase reliability of database when using the redundant array of independent disks by introducing redundancy

Disk mirroring:-

It is the technique for introducing redundancy in a database is called mirroring/ shadowing. Is store data redundantly on two identical physical disks that are treated as one logical disk.

In case of mirroed data, the data items can be read from any disk, hot for writing the data item must be written on both. Means, When data is read, it can be retrieved from the disk with shorter queuing, seek, and rotational delays.

If one disk fails, the other disk is still there to continuously provide the data. It improves the reliability.

Quantities example from book:-

The mean time to failure of a mired disk depends on the man time to failure of the individual disks, as well as on the mean time to repair, which is the time it takes (an average) to replace a failed disk and to restore the data an it. Suppose that, the failures of the two disks are independent; means there is no connection between the failure of one disk and the failure of the other.

It the system has 100 disks in an array. The mean to repair is 24 hours, and the MTTF if 200,000 hours on each disk.

The mean time to data loss of a mirrored system is $\frac{(MTBF)^2}{(2 \times MTTR)}$

16.21. How does disk mirroring help improve reliability? Give a quantitative example.

The technique of data striping to achieve higher transfer rates and improves the performance of disk in RAID, which has two levels (i) bit-level data striping (ii) block level data striping.

16.22. What characterizes the levels in RAID organization?

Raid Levels:-

In the RAID organization, one solution that presents it self because of the increased size and reduced cost of hard drives is to built in redundancy. RAID can be implemented in hardware and software and it is a set of physical disk drives viewed by the operating system as a single logical drive.

Levels:-

Depends on the data redundancy introduced and correctness checking technique used in the schema.

Level 0:-

Uses data striping and it has no redundancy and no correctness checking.

Level 1:-

Redundancy through mirroring and no correctness checking.

Level 2:-

In this level; mirroring and no mirroring combined with memory like correctness checking.

For example:

Using parity hit:

Various versions of level 2 are possible.

Level 3:-

Level 3 is seems like as level 2, but uses the single disk for parity. Level 3 is some time called as bit-interleaved. Disk controller can detect whether a sector has been read correctly. A single parity bit can be used for error correction as well as detection.

Level 4:-

Block level data striping and parity like level 3 and in this level stores blocks.

Level 5:-

Block level data striping but data and parity are distributed across all disks.

Level 6:-

Uses the P+Q redundancy scheme, and P+Q redundancy using Reed-suloman codes to recover from multiple disk failures.

16.23. What are the highlights of the popular RAID levels 0, 1, and 5?

Different RAID (Redundant Array of Inexpensive Disks) organizations were defined based on different combinations of the two factors,

1. Granularity of data interleaving (striping)

2. Pattern used to compute redundant information.

There are various levels of RAID from 0 to 6. The popularly used RAID organization is level 0 with striping, level 1 with mirroring, and level 5 with an extra drive for parity.

RAID level 0

• It uses data striping.

• It has no redundant data and hence it provides best write performance as updates are not required to be duplicated.

• It splits data evenly across multiple disks.

RAID level 1

• It provides good read performance as it uses mirrored disks.

• Performance improvement is possible by scheduling a read request to the disk with shortest expected seek and rotational delay.

RAID level 5

• It uses block level data striping.

• Data and parity information are distributed across all the disks. If any one disk fails, the data lost is due to any changes is determined by using the information of the parity available from the remaining disks.

16.24. What are storage area networks? What flexibility and advantages do they offer?

There is a demand for storage and management of cost all data as data are integrated across organization and it is necessary to move from a static fixed data which are used from centered architecture operation to a more flexible and dynamic infrastructure for the processing of information requirements, most of the organizations moved to the better criterion of storage area networks (SANs).

• In SAN, online storage peripherals are configured as nodes on a high-speed network and can be attached and removed from servers in a very flexible manner.

• They allow storage systems to be placed at longer distances from the servers and provide good performance and different connectivity options

• It provides point-to-point (every devices are connected to every other device) connections between servers and storage systems through fiber channel; it allows connecting multiple RAID systems, tape libraries to servers.

Advantages

1. It is more flexible as it provides flexible connection with many devices that is many-to-many connectivity among servers and storage devices using fiber channel hubs and switches.

2. Between a server and storage system there is a distance separation of up to 10km provided by using fiber optic cables.

3. It provides better isolation capabilities by allowing non-interruptive addition of new peripheral devices and servers.

16.25. Describe the main features of network-attached storage as an enterprise storage solution.

In enterprise applications it is necessary to maintain solutions at a very low cost to provide high performance. Network-attached storage (NAS) devices are used for this purpose. It does not provide any of the services common to the server, but it allows the addition of storage for file sharing.

Features

• It provides very large amount of hard-disk storage space and it is attached to a network and multiple or more number of servers can make use of those space without shutting them down so that it ensure better maintenance and improve the performance.

• It can be located at anywhere in the local area network (LAN) and used with different configuration.

• A hardware device called as NAS box or NAS head acts as a gateway between the NAS systems and clients who are connected in the network.

• It does not use any of the devices such as monitor, keyboard, or mouse, disk drives that are connected to many NAS systems to increase total capacity.

• It can store any data that appears in the form of files, such as e-mails, web content includes text, image or videos, and remote system backups.

• It works to provide reliable operation and for easy maintenance.

• It includes built-in features such as security (authenticate the access) or automatic sending of alerts through mail in case of error occurred on the device that are connected.

• It contributes to provide high degree of scalability, reliability, flexibility and performance.

16.26. How have new iSCSI systems improved the applicability of storage area networks?

Internet SCSI (iSCSI) is a protocol proposed to issue commands that allows clients (initiators) to send SCSI commands to SCSI storage devices through remote channels.

• The main feature is that, it does not require any special cabling connections as needed by Fiber Channel and it can run for longer distances using existing network infrastructure.

• iSCSI allows data transfers over intranets and manages storage over long distances.

• It can transfer data over variety of networks includes local area networks (LANs), wide area networks (WANs) or the Internet.

• It is bidirectional; when the request is given, it is processed and the resultant data is sent in response to the original request.

• It combines different features such as simplicity, low cost, and the functionality of iSCSI devices provides good upgrades and hence applied in small and medium sized business applications.

16.27. What are SATA, SAS, and FC protocols?

SATA Protocol:

SATA stands for serial ATA, wherein ATA represents attachment; therefore SATA becomes serial AT attachment.

SATA is a modern storage protocol that has fully replaced the most commonly used SCSI (small computer system interface) and parallel ATA in laptops and small personal computers. SATA overcomes design limitations of previous storage protocol.

• SATA is suitable for tiered storage environment.

• SATA can be used for small and medium sized enterprises.

• SATA support interchangeability.

SAS Protocol:

SAS stands for serial attached SCSI. SAS overcomes design limitations of previous storage protocol and also considered superior to SATA.

• SAS was designed to replace SCSI interfaces in Storage area network (SAN).

• SAS drives are faster than SATA drives and has dual portability.

• SATA can be used for small and medium sized enterprises.

• SATA support interchangeability.

FC Protocol:

FC stands for serial Fiber channel protocol. Fiber channel is used to connect multiple RAID systems, taps, which have different configurations.

• Fiber channel supports point to point connection between server and storage system. It also Provide flexibility to connect too many connections between servers and storage devices.

• Fiber channel has almost the same performance like SAS. It uses fiber optic cables, so high speed data transfer supported.

• No distance limitation. Low cost alternative for devices.

16.28. What are solid-state drives (SSDs) and what advantage do they offer over HDDs?

Solid-state drives (SSD):

SSD is abbreviation for solid-state drives, which uses integrated circuit assemblies as storage to store data permanently. It is a nonvolatile memory, means it will not forget the data on system memory when the system is turned off.

SSD is based on flash memory technology, that's why sometimes it is known as flash memory, and they don't require continuous power supply to store data on secondary storage, so they are known as solid state disk or solid state drives.

SSD does not have read and write head like traditional electromagnetic disk, instead it has controller (embedded processor) for various operations. It makes speed of data retrieval faster than magnetic disks. Commonly in SSDs, interconnected NAND flash memory cards are used.

SSD uses wear leveling technique to store data that extend the life of SSD by storing data to separate NAND cell, instead of overwriting it.

Advantages of SSDs over HDDs are as follows:

• **Faster access time and higher transfer rate:**

In SSD data can be accessed directly from different locations on flash memory, so access time in SSD is 100 times faster than HDD and latency time is low, consequently data transfer rate is high and system boot up time is low.

• **More reliable:**

SSD does not have a moving mechanical arm for read and write operations. Data is stored on integrated circuit chips. SSD has controller to manage all the operations on flash cells, and data can be written and erased on flash cell, only limited number of time before it fails. The controller manages these activities, so that SSD can work for many years under normal use.

• **No moving component (durable):**

As SSD does not have moving component, so data on SSD is safer, even when equipment is being handled roughly.

• **Uses less power:**

As in SSD, there is no head rotation to read and write data, so power consumption is lower than HDD and saves battery life. SDD uses only 2-3 watts whereas HDD uses 6-7 watts of power.

• **No noise and generate less heat:**

As no moving head rotation is there, so SSD generate less heat and doesn't make noise that helps to increase life and reliability of the drive.

• **Light weight:**

As SSDs are mounted on circuit board and they don't have moving head and spindle, so they are light weight and small in size.

16.29. What is the function of a buffer manager? What does it do to serve a request for data?

The buffer manager is a software module of DBMS whose responsibility is to serve to all the data requests and take decision about choosing a buffer and to manage page replacement.

The main functions of buffer manager are:

• To speed up the processing and increase efficiency.

• To increase the possibility that the requested page is found in main memory.

• To find an appropriate replacement for a page while reading a new disk block from disk, such that the replacement page will not be required soon.

• The buffer manager must ensure that the number of buffers fits in the main memory.

• Buffer manager functions according to the buffer replacement policy and selects the buffers that must be emptied, when the requested amount of data surpasses the available space in buffer.

The buffer manager handles two types of operations in buffer pool to fulfill its functionality:

1. **Pin count:** This is the counter to track the number of page requests or corresponding number of users who requested that page. Initially counter value is set to zero. If the counter value is always zero, the page is unpinned. Only unpinned blocks are allowed to be written on the disk. As the value of counter is incremented the pages are called pinned.

2. **Dirty bit:** Initially its values is set to zero for all pages. When the page is updated, its value is updated to 1.

Buffer manager processes the page requests in following steps:

• Buffer manager checks the availability of the page in buffer. If the page is available, it increments the pin count and sends the page.

• If page is not in buffer, than buffer manager takes the following steps:

• Buffer manager decides a page according to the replacement policy and increments page's pin count.

• If the dirty bit of replacement page is on, buffer manager writes that page onto disk and replaces the old copy.

• If the dirty bit is not on, buffer manager does not write the page back to disk.

• Buffer manager reads the new page and conveys the memory location of the page to the demanding application.

16.30. What are some of the commonly used buffer replacement strategies?

Buffer replacement strategies:

In large DBMSs, files contain so many pages and it is not possible to keep all the data in memory at the same time. To overcome this storage problem and improve efficiency of DBMS transactions, buffer manager (software) uses buffer replacement strategies that decide what buffer to use and which pages are to be replaced in the buffer to give a space to newly requested pages.

Some commonly used buffer replacement strategies are as follows:

- **LRU (Least recently used):**

The LRU strategy keeps track of page usages for specific period of time and it removes the oldest used page.

LRU works on the principle that the pages which are frequently used are most likely to be used in further processing too. To maintain the strategy the buffer manager has to maintain a table where the frequency of the page usage is recorded for every page. This is very common and simple policy.

It has problem of sequential flooding, which means that there are frequent scanning and repeated use of I/O for each page.

- **Clock policy:**

This is an approximate LRU technique. It is like Round robin strategy. In clock replacement policy buffers are arranged in a circle like a clock with a single clock hand. The buffer manager sets “use bit” on each reference. If “use bit” is not set (flag 0) for any buffer that means it is not used in a long time and is vulnerable for replacement. It replaces the old page not the oldest.

• **FIFO (First In First Out):**

This is the simplest buffer replacement technique. When buffer is required to store new pages, the oldest arrived page is swapped out. The pages are arranged into the buffer in a queue in a fashion that most recent page is the tail and oldest arrival is the head.

During replacement the page at the head of the queue is replaced first. This strategy is simple and easy to implement but not desirable, because it replaces the oldest page which may be most frequently used page and in future it can be needed, so again it will be swapped in. It creates processing overhead.

• **MRU (Most recently used):**

It removes most recently used pages first. This is also called **fetch and discard.** This is useful in sequential scanning when most recently used page, won't be used in future for a period of time.

In situation of sequential scanning LRU and CLOCK strategies don't perform well. To enhance performance of FIFO, it can be modified by using some pinned block like root index block, and make sure that they can't be replaced and always remain in buffer.

16.31. What are optical and tape jukeboxes? What are the different types of optical media served by optical drives?

Optical jukeboxes:

Optical jukebox is an intelligent data storage device that uses an array of optical disk platters, and automatically load and unload these disks like according to the storage need. Jukeboxes has high capacity storage and it supports up to terabytes and even petabytes of tertiary storage.

• Optical jukeboxes have up to 2000 different disk slots. As optical jukeboxes keep traversing different disk storage according data requirement, so it create time overhead and affect processing.

• Jukeboxes are cost effective and provide random access of data.

• The process of dynamically loading and uploading of disk drives is called migration.

Magnetic jukeboxes:

Magnetic tape jukeboxes uses a number of tapes as a storage and automatically load and unload taps on tape drives. This is a popular tertiary storage medium that can handle data up to terabytes.

16.32. What is automatic storage tiering? Why is it useful?

Automated storage tiering (AST):

AST is the one of the storage types that filters and transfers the data among different types of storage like SATA, SAS, SSDs based on the storage requirement, dynamically.

Automated tiering mechanism is managed by the storage administrator. According to the tiering policy, less used data is transferred to the SATA drives, as it is slower and is not much expensive, and frequently used data is transferred to high speed SAS or solid state drives.

The automated tiering highly improves performance of the DBMS.

EMC implements FAST (fully automated storage tiering). It automatically monitors data activeness, and moves active data to high performance storage like SSD and inactive data to inexpensive and slower storage like SATA. Therefore, **AST is useful as it results in high performance and low cost.**

16.33. What is object-based storage? How is it superior to conventional storage systems?

Object - based storage:

In object based storage system data is organized in units called object instead of blocks in file. In this storage system, data is not stored in hierarchy rather than all the data is stored in the form of objects, and required object can be searched directly using unique global identifier, without overhead.

Every object in object based storage has three parts:

• **Data:** It is the information that is to be stored in the objects.

• **Variable Meta data:** This field has the information about main data like location of the data, usability, confidentiality and other information required to manage the data.

• **Unique global identifier:** This identifier stores the address information of the data so that data can be located easily.

Object storage system is better than conventional storage system in following ways:

• As the organizations are expanding, their data is also increasing day by day. If the file system is used as a data storage system and data is stored in the blocks, it would become very difficult to manage huge amount of data. In conventional file systems, data is stored in hierarchical fashion and all these data are stored into blocks with their own unique address.

To solve this management overhead, data is stored in the form of objects with additional metadata information.

• Object based storage provides security of data. In object based systems, the objects can be accessed directly by the applications through unique global identifier. While in the file storage system data need to be searched in linear or binary fashion that generates processing overhead and is time consuming.

• Object based storage system supports features like replication, encapsulation and distribution of objects, that makes data secure, manageable and easily accessible. However, conventional file based storage system does not supports replication and distribution of objects.

16.34. Consider a disk with the following characteristics (these are not parameters of any particular disk unit): block size $B = 512$ bytes; interblock gap size $G = 128$ bytes; number of blocks per track = 20; number of tracks per surface = 400. A disk pack consists of 15 double-sided disks.

Given data

Block size $B = 512$ bytes

Inter block gap size $G = 128$ bytes

Number of blocks per track $= 20$

Number of tracks per surface $= 400$

Disk pack consists of 15 double – sided disks

a. What is the total capacity of a track, and what is its useful capacity (excluding interblock gaps)?

(a) Total track size $= 20$

Block per track $\times$(block size $+$ block gap size)

$= 20 \times (512 + 128)$

$= 12800$ Bytes

$= 12.8$ k bytes

Useful capacity of a track = block per tract $\times$ block size

$\Rightarrow 20 \times 512 = 10240$

Bytes $= 10.24$ k bytes

b. How many cylinders are there?

(b) Number of cylinders

= Numbers of tracks

= 400

c. What are the total capacity and the useful capacity of a cylinder?

(c) Total cylinder capacity

$$= 15 \times 2 \times 20 \times 512$$

$$= 307200 \text{ bytes}$$

$$= 307.2 \text{ k bytes}$$

d. What are the total capacity and the useful capacity of a disk pack?

(d) Total capacity of a disk pack $= 4$

$$= 15 \times 2 \times 400 \times 20 \times (512 + 128)$$

$$= 153600000 \text{ Bytes}$$

$$= 153.6 \text{ m bytes}$$

Useful capacity of a disk pack

$$= 15 \times 2 \times 4000 \times 20 \times 512$$

$$= 122.88 \text{ m bytes}$$

e. Suppose that the disk drive rotates the disk pack at a speed of 2,400 rpm (revolutions per minute); what are the transfer rate (*tr*) in bytes/msec and the block transfer time (*btt*) in msec? What is the average rotational delay (*rd*) in msec? What is the bulk transfer rate? (See Appendix B.)

(e) Transfer rate $tr = \dfrac{\text{total track size in bytes}}{\text{time for one disk revolution in msec}}$

$$Tr = \frac{12800}{\left(\frac{(60 \times 1000)}{(2400)}\right)}$$

$$= \frac{12800}{25}$$

$$= 512 \frac{\text{bytes}}{\text{m sec}}$$

Block transfer time $btt = \dfrac{B}{tr}$

$$= \frac{512}{512}$$

$$= 1 \text{ msc}$$

Average rotational delay

$$rd = \left(\text{time torone disk revolution in } \frac{\text{msec}}{2} \right)$$

$$= \frac{25}{2}$$

$$= 12.5 \text{ msec}$$

f. Suppose that the average seek time is 30 msec. How much time does it take (on the average) in msec to locate and transfer a single block, given its block address?

(f) Average time to locate and transfer a block

$= S + rd + btt$

$= 30 + 12.5 + 1$

$= 43.5 \text{ msec}$

g. Calculate the average time it would take to transfer 20 random blocks, and compare this with the time it would take to transfer 20 consecutive blocks using double buffering to save seek time and rotational delay.

(g) Time to transfer 20 random blocks

$= 20 \times (S + rd + btt)$

$= 20 \times 43.5$

$= 870 \text{ msec}$

Time to transfer 20 consecutive blocks using double

Buffering $= S + rd + 20 \times btt$

$= 30 + 12.5 + (20 \times 1)$

$= 62.5 \text{ msec}$

16.35. A file has $r = 20{,}000$ STUDENT records of *fixed length*. Each record has the following fields: Name (30 bytes), Ssn (9 bytes), Address (40 bytes), PHONE (10 bytes), Birth_date (8 bytes), Sex (1 byte), Major_dept_code (4 bytes), Minor_dept_code (4 bytes), Class_code (4 bytes, integer), and Degree_program (3 bytes). An additional byte is used as a deletion marker. The file is stored on the disk whose parameters are given in Exercise 16.27.

a. Calculate the record size R in bytes.

b. Calculate the blocking factor bfr and the number of file blocks b, assuming an unspanned organization.

Consider the following parameter of a disk:
Seek time $s = 20$ msec
Rotational delay $rd = 10$ msec
Block transfer time $btt = 1$ msec
Block size $B = 2400$ bytes
Inter block gap size $G = 600$ bytes
Consider a file STUDENT is having records such that $r = 20{,}000$.
Different fields common in each record are as follows:

Field name	Size (in bytes)
Name	30
Ssn	9
Address	40
Phone	10
Birth_date	8
Sex	1
Major_dept_code	4
Minor_dept_code	4
Class_code	4
Degree_pro	3
deletion marker	1

a.

The record size R can be calculated as,

$$R = \text{sum of all field size}$$
$$= 30+9+40+10+8+1+4+4+4+3+1$$
$$= 114$$

The record size is $\boxed{114 \text{ bytes}}$.

b.

Assume the organization of records is unspanned.
The blocking factor *bfr*, which represents the average number of records per block, can be calculated as,

$$\begin{aligned} bfr &= \lfloor B / R \rfloor \\ &= 2400 / 114 \\ &= 21.05 \\ &= 21 \end{aligned}$$

The blocking factor is 21 records per block.

The number of file blocks can be calculated as,

$$\begin{aligned} b &= \lceil r / bfr \rceil \\ &= 20000 / 21 \\ &= 952.3 \\ &= 953 \end{aligned}$$

The numbers of file blocks are 953 blocks.

c. Calculate the average time it takes to find a record by doing a linear search on the file if (i) the file blocks are stored contiguously, and double buffering is used; (ii) the file blocks are not stored contiguously.

c.

To calculate the average time to find a record by doing linear search on the file, the search is performed on average half of the file blocks.
Half of 952 file blocks is approximately, 952/2 = 476 blocks.

i.

If the file blocks are stored contiguously and double buffering is used, the time taken to read 476 blocks is,

$$s + rd + \left(476 \cdot \left(B / btr\right)\right) \quad \text{..... (1)}$$

Where s = seek time, rd = rotational delay, B = block size and btr = block transfer rate.
First calculate transfer rate btr as follows,

$$btr = tr \cdot \left(B / \left(B + G\right)\right)$$

Where $tr = B / btt$.
Hence, $tr = 2400$ bytes per millisecond.
Therefore,

$$\begin{aligned} btr &= 2400 \cdot \left(2400 / \left(2400 + 600\right)\right) \\ &= 2400 \cdot \left(2400 / 3000\right) \\ &= 1920 \end{aligned}$$

The value of btr is 1920 bytes.
Substitute the value of btr in equation 1.

$$\begin{aligned} s + rd + \left(476 \cdot \left(B / btr\right)\right) &= 20 + 10 + \left(476 \cdot \left(2400 / 1920\right)\right) \\ &= 30 + 595 \\ &= 625 \end{aligned}$$

If the file blocks are stored contiguously and double buffering is used, then the average time taken to find a record by doing linear search on the file is 0.625 sec.

ii.

If the file blocks are not stored contiguously, the time taken to read 476 blocks is,

$$\begin{aligned} \left(s + rd + btt\right) \cdot 476 &= \left(20 + 10 + 1\right) \cdot 476 \\ &= \left(31\right) \cdot \left(476\right) \\ &= 14756 \end{aligned}$$

If the file blocks are not stored contiguously, then the average time taken to find a record by doing linear search on the file is 14.756 sec.

d. Assume that the file is ordered by Ssn; by doing a binary search, calculate the time it takes to search for a record given its Ssn value.

d.

If it is assumed that file is ordered by Ssn, the time taken to search a record, with its Ssn value provided, is calculated as,

$$\begin{aligned}\log_2(b)\cdot(s+rd+btt) &= \log_2(952)\cdot(20+10+1)\\ &= (9.8)\cdot 31\\ &= (10)\cdot(31)\\ &= 310\end{aligned}$$

The time taken to search a record, with its Ssn value provided, is 310 ms.

16.36. Suppose that only 80% of the STUDENT records from Exercise 16.28 have a value for Phone, 85% for Major_dept_code, 15% for Minor_dept_code, and 90% for Degree_program; and suppose that we use a variable-length record file. Each record has a 1-byte *field type* for each field in the record, plus the 1-byte deletion marker and a 1-byte end-of-record marker. Suppose that we use a *spanned* record organization, where each block has a 5-byte pointer to the next block (this space is not used for record storage).

a. Calculate the average record length R in bytes.

b. Calculate the number of blocks needed for the file.

Assume that a variable length record file is being used.

It is provided that each record has 1 byte *field type*, along with 1 byte deletion marker and 1 byte end of record marker.

So the fixed record size would be calculated for fields not mentioned in the question, that is Name, Ssn, Address, Birth_date, Sex, Class_code.

Therefore,

$$\begin{aligned}\text{Fixed record size} &= (30+1)+(9+1)+(40+1)+(8+1)+(1+1)+(4+1)+1+1\\ &= 100 \text{ bytes}\end{aligned}$$

And for the remaining variable length fields, that is Phone, Major_dept_code, Minor_dept_code, Degree_program), the number of bytes per record can be calculated as,

$$\begin{aligned}\text{Variable record size} &= ((10+1)\times 0.8)+((4+1)\times 0.85)+((4+1)\times 0.15)+\\ &\quad ((3+1)\times 0.9)\\ &= 8.8+4.25+0.75+3.6\\ &= 17.4 \text{ bytes}\end{aligned}$$

a.

Therefore, the average record length R is,

$$\begin{aligned}R &= \text{fixed record size} + \text{variable record size}\\ &= 100+17.4\\ &= 117.4\end{aligned}$$

The average record length is $\boxed{117.4 \text{ bytes}}$.

b.

Since a spanned record-file organization is being used, where each block has unused space of 5-bytes pointer, so the usable bytes in each block are $2400-5=2395$ bytes.

The number of blocks required for the file can be calculated as,

$$\begin{aligned}\lceil (r\times R)/(B-5) \rceil &= (117.4\times 20000)/2395\\ &= 980.3\\ &= 981\end{aligned}$$

The numbers of blocks required for file are $\boxed{981 \text{ blocks}}$.

16.37. Suppose that a disk unit has the following parameters: seek time $s = 20$ msec; rotational delay $rd = 10$ msec; block transfer time $btt = 1$ msec; block size $B = 2400$ bytes; interblock gap size $G = 600$ bytes. An EMPLOYEE file has the following fields: Ssn, 9 bytes; Last_name, 20 bytes; First_name, 20 bytes; Middle_init, 1 byte; Birth_date, 10 bytes; Address, 35 bytes; Phone, 12 bytes; Supervisor_ssn, 9 bytes; Department, 4 bytes; Job_code, 4 bytes; deletion marker, 1 byte. The EMPLOYEE file has $r = 30{,}000$ records, fixed-length format, and unspanned blocking. Write appropriate formulas *and* calculate the following values for the above EMPLOYEE file:

a. Calculate the record size R (including the deletion marker), the blocking factor bfr, and the number of disk blocks b.

b. Calculate the wasted space in each disk block because of the unspanned organization.

c. Calculate the transfer rate tr and the bulk transfer rate btr for this disk unit (see Appendix B for definitions of tr and btr).

d. Calculate the average *number of block accesses* needed to search for an arbitrary record in the file, using linear search.

e. Calculate in msec the average *time* needed to search for an arbitrary record in the file, using linear search, if the file blocks are stored on consecutive disk blocks and double buffering is used.

f. Calculate in msec the average *time* needed to search for an arbitrary record in the file, using linear search, if the file blocks are *not* stored on consecutive disk blocks.

g. Assume that the records are ordered via some key field. Calculate the average *number of block accesses* and the *average time* needed to search for an arbitrary record in the file, using binary search.

Consider the following parameter of a disk:

Seek time *s* = 20 msec

Rotational delay *rd* = 10 msec

Block transfer time *btt* = 1 msec

Block size *B* = 2400 bytes

Inter block gap size *G* = 600 bytes

Consider a file EMPLOYEE is having records such that *r* = 30,000.

Different fields common in each record are as follows:

Field name	**Size (in bytes)**
Ssn	9
First_name	20
Last_name	20
Middle_init	1
Address	35
Phone	12
Birth_date	10
Supervisor_ssn	9
Department	4
Job_code	4
deletion marker	1

The record size *R* can be calculated as,

$$\begin{aligned} R &= \text{sum of all field size} \\ &= 9+20+20+1+35+12+10+9+4+4+1 \\ &= 125 \end{aligned}$$

The record size is $\boxed{125 \text{ bytes}}$.

Since the file is unspanned so the blocking factor *bfr* can be calculated as,

$$\begin{aligned} bfr &= \lfloor B / R \rfloor \\ &= 2400/125 \\ &= 19.2 \\ &= 19 \end{aligned}$$

The blocking factor is $\boxed{19 \text{ records per block}}$.

In an unspanned organization of records, the number of file blocks can be calculated as,

$$\begin{aligned} b &= \lceil r / bfr \rceil \\ &= 30000/19 \\ &= 1578.9 \\ &= 1579 \end{aligned}$$

The numbers of file blocks are $\boxed{1579 \text{ blocks}}$.

As the file has unspanned organization, so wasted space in each block can be calculated as,

$$\begin{aligned} \text{Wasted space} &= B-(R \times bfr) \\ &= 2400-(125 \times 9) \\ &= 2400-2375 \\ &= 25 \end{aligned}$$

The wasted space in each disk block is $\boxed{25 \text{ bytes}}$.

The transfer rate *tr* can be calculated as,

$$\begin{aligned} tr &= B / btt \\ &= 2400 / 1 \\ &= 2400 \end{aligned}$$

The transfer rate for the disk is $\boxed{2400 \text{ bytes per msec}}$.

The bulk transfer rate *btr* can be calculated as,

$$\begin{aligned} btr &= tr \cdot \left(B / \left(B + G \right) \right) \\ &= 2400 \cdot \left(2400 / 3000 \right) \\ &= 1920 \end{aligned}$$

The bulk transfer rate for the disk is $\boxed{1920 \text{ bytes per msec}}$.

While searching for an arbitrary record in a file using the liner search the average number of block accesses can be found as follows:

• Records are searched on key fields.

If one record satisfies the search condition, on average half of the blocks are to be searched, that is $b / 2 = 1579 / 2 \text{ blocks}$.

If the record does not satisfies the search condition, all blocks are to be searched, that is $b = 1579 \text{ blocks}$.

• Records are searched on non-key fields.

In this case all blocks are to be searched, that is $b = 1579 \text{ blocks}$.

To calculate the average time to find a record using linear search on the file, the search is performed on average half of the file blocks.

Half of 1579 file blocks is approximately, 1579/2 = 789.5 blocks.

If the blocks are stored on consecutive disk block and double buffering is used, the average time taken to read 789.5 blocks is,

$$\begin{aligned} s + rd + \left(789 \cdot \left(B / btr \right) \right) &= 20 + 10 + \left(\left(789.5 \right) \cdot \left(2400 / 1920 \right) \right) \\ &= 30 + 986.8 \\ &= 1017 \end{aligned}$$

If the file blocks are stored consecutively and double buffering is used, then the average time taken to find a record by doing linear search on the file is $\boxed{1.017 \text{ sec}}$.

If the file blocks are not stored in consecutive disk blocks, the time taken to read 789.5 blocks is,

$$\begin{aligned}(s+rd+btt)\cdot(789.5) &= (20+10+1)\cdot(789.5)\\ &= (31)\cdot(789.5)\\ &= 24475\end{aligned}$$

If the file blocks are not stored consecutively, then the average time taken to find a record by doing linear search on the file is $\boxed{24.475\ \text{sec}}$.

While the records are ordered via some key field and binary search is going on, then the average number of block accesses can be found as follows

• If record is found then on an average half of the blocks are to be accessed, that is $b/2 = 1579/2\,\text{blocks}$.

• If the record is not found then all blocks are to be accessed, that is $b = 1579\,\text{blocks}$.

If it is assumed that records are ordered through some key field, the time taken to search a record, using binary search, is calculated as,

$$\begin{aligned}\log_2(b)\cdot(s+rd+btt) &= \log_2(1579)\cdot(20+10+1)\\ &= (10.6)\cdot 31\\ &= (11)\cdot(31)\\ &= 341\end{aligned}$$

The average time taken to search a record via some key field is $\boxed{0.341\ \text{sec}}$.

16.38. A PARTS file with Part# as the hash key includes records with the following Part# values: 2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, 6975, 4981, and 9208. The file uses eight buckets, numbered 0 to 7. Each bucket is one disk block and holds two records. Load these records into the file in the given order, using the hash function $h(K) = K \bmod 8$. Calculate the average number of block accesses for a random retrieval on Part#.

Consider the data:

The part values are 2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, 6975, 4981, 9208.

The number of buckets used by file = 8, numbered from 0 to 7

One bucket = one disk block and two records.

The hash function $h(k) = k \bmod 8$

The value of hash function when k = 2369 is obtained as follows:

$h(k) = k \bmod 8$

= 2369 mod 8

=1

The value of hash function when k = 3760 is obtained as follows:

$h(k) = k \bmod 8$

= 3760 mod 8

= 0

The value of hash function when k = 4692 is obtained as follows:

$$h(k) = k \bmod 8$$

=4692 mod 8

=4

The value of hash function when *k* = 4871 is obtained as follows:

$$h(k) = k \bmod 8$$

=4871 mod 8

=7

The value of hash function when *k* = 5659 is obtained as follows:

$$h(k) = k \bmod 8$$

=5659 mod 8

=3

The value of hash function when *k* = 1821 is obtained as follows:

$$h(k) = k \bmod 8$$

=1821 mod 8

=5

The value of hash function when k = 1074 is obtained as follows:

$$h(k) = k \bmod 8$$

=1074 mod 8

=2

The value of hash function when k = 7115 is obtained as follows:

$$h(k) = k \bmod 8$$

=7115 md 8

=3

The value of hash function when k = 1620 is obtained as follows:

$$h(k) = k \bmod 8$$

=1620 mod 8

= 4

The value of hash function when k = 2428 is obtained as follows:

$$h(k) = k \bmod 8$$

=2428 mod 8

=4 (Over flow)

The value of hash function when k = 3943 is obtained as follows:

$$h(k) = k \bmod 8$$

=2428 mod 8

=4 (Over flow)

The value of hash function when k = 3943 is obtained as follows:

$$h(k) = k \bmod 8$$

=3943 mod 8

= 7

The value of hash function when k = 4750 is obtained as follows:

$$h(k) = k \bmod 8$$

=4750 mod 8

=6

The value of hash function when k = 6975 is obtained as follows:

$$h(k) = k \bmod 8$$

=6975 mod 8

=7(Over flow)

The value of hash function when k = 4981 is obtained as follows:

$$h(k) = k \bmod 8$$

=4981 mod 8

= 5

The value of hash function when k = 9208 is obtained as follows:

$$h(k) = k \bmod 8$$

=9208 mod 8

= 0

The value of hash function when k = 9209 is obtained as follows:

$$h(k) = k \bmod 8$$

=9209 mod 8

= 1

The hash table for the part values is as follows:

Bin Number	Key 1	Key 2	Overflow
0	3760	9208	
1	2369	9209	
2	1074		
3	5659	7115	
4	4692	1620	Yes (2428)
5	1821	4981	
6	4750		
7	4781	3943	Yes (6975)

In this, two records of 15 are in overflow, and which will require an additional block access. The other records require only one block access.

Thus, the average time = $\left(1\left(\frac{13}{15}\right)\right)+\left(2\left(\frac{2}{15}\right)\right)$

$= 0.867 + 0.266$

$= 1.133 \text{ block access}.$

16.39. Load the records of Exercise 16.31 into expandable hash files based on extendible hashing. Show the structure of the directory at each step, and the global and local depths. Use the hash function $h(K) = K \bmod 128$.

Part#-> Hash values-> binary values

2369 ->65 -> 1000001

3760 ->48 -> 110000

4692 ->84 -> 1010100

4871 ->7 -> 111

5659 ->27 -> 11011

1821 ->29 -> 11101

1074 ->50 -> 110010

7115 ->75 -> 1001011

1620 ->84 -> 1010100

2428 ->124 -> 1111100

3943 ->103 -> 1100111

4750 ->14 -> 1110

6975 ->63 -> 111111

4981 ->117 -> 1110101

9208 ->120 -> 1111000

Table in office.

d = 4
0000111
0001110
0011011
0011101
0110000
0110010
0111111
1000001
1001011
1010100
1100111
1110101
1111000
1111100

Global depth is 4 and local depth is written with each level.in figure above.

16.40. Load the records of Exercise 16.31 into an expandable hash file, using linear hashing. Start with a single disk block, using the hash function $h_0 = K \bmod 2^0$, and show how the file grows and how the hash functions change as the records are inserted. Assume that blocks are split whenever an overflow occurs, and show the value of n at each stage.

When we apply hash function K Mod 2^0

we get a single bucket.

We split this bucket into two buckets with new function K Mod 2^1

Bucket1:2369, 4871, 5659, 1821, 7115, 3943, 6975, 4981

Bucket2:3760,4692, 1074, 1620, 2428, 4750, 9208

Now we can split bucket into four buckets:

B1a:2369, 1821,4981

B1b:1074,4750

B1c:4871,5659,7115,3943,6975,

B1d:3760, 4692,1620,2428,9208

Since some bucket more than 2 elements they can be split using function K Mod 2^3

B1:2369,

B5:1821,4981

B7:4871,3943,6975

B3:5659, 7115

B8:3760,9208

B4:4692,1620,2428

B2: 1074

B6:4750

Since some buckets are still greater in size so we apply another function on them K Mod 2^4

B1: 2369

B5:4981

B7:4871,3943,

B8:9208

B15:6975

B4:4692,1620

B11:5659,7115

B12:2428

B13:1821

B16:3760

B14:4750

B2:1074

Now we have all buckets of correct size.

16.41. Compare the file commands listed in Section 16.5 to those available on a file access method you are familiar with.

File commends listed in Files of Unordered Records, on a file access methods.

Records are placed in the file in the order in which they are inserted. Records are inserted at the end of the file. This record organization is called heap/ pile file. File commends in the files of unordered records:-

Inserting a new record:-

→Delete a record

→External sating

Inserting a record:-

New record insertion is very efficient. It is done by when new record is inserted. Then the last block of the file is copied in to a butter than the new record is added then block is rewriters back to the disk.

Delete a record:-

Program must find it's block first, and copy the block into a buffer, then delete the record from the buffer and finally rewrite the block back to it disk. In this record deletion. We use the technique of deletion marker.

External sorting:-

When we want to read all records in order of the value of some fields. Then we create a sorted copy of the file. For a large disk file it is an expensive. So, for this we use external sorting.

16.42. Suppose that we have an unordered file of fixed-length records that uses an unspanned record organization. Outline algorithms for insertion, deletion, and modification of a file record. State any assumptions you make.

Compare the heap file (unordered files) and file access methods.

Heap file:-

- The simplest and basic type of organization.

- Records are placed in the file in the order in which are inserted.

- Inserting a new record is very efficient.

- New records are inserted at the end of the file.

- Searching is done by only search procedure. Mainly involves a linear search, and it is an expensive procedure.

Fine access methods:-

- In the file organization, organization of the data of a file into records, blocks, and access structures.

- Records and blocks are placed on the storage medium and they are interlinked. Example: sorted file.

Access methods:-

- Provide a group of operations and that can be applied to a file.

Example: Open, find, delete, modify, insert close ……etc.

- An organization is consists of several access methods. It is possible to apply.

- Some access methods can be applied only to file organized in certain ways. That are

→ Records organized by serially, (sequential)

→ Relative record number based on organization. (Relative)

→ Indexed based organization (indexed)

Method access refers to the way that is, in which records are accessed. A file with an organization of indexed or relative may still have its records accessed sequentially. But records in a file with an organization of sequential. Cannot be accessed directly.

16.43. Suppose that we have an ordered file of fixed-length records and an unordered overflow file to handle insertion. Both files use unspanned records. Outline algorithms for insertion, deletion, and modification of a file record and for reorganizing the file. State any assumptions you make.

For ordered file of fixed length:

Algorithms: Consider that file name is abc and file is ordered on Key field that is a numeric fiels and in increasing order.

For insertion: Let for record that is to be inserted value of Key field be n

1. Open file abc and take file pointer in variable fp
2. Find record where fp.key>n
3. Insert current record at this position.
4. Save the file data
5. Close file

For deletion: let record to be deleted has value for key field = n

1. Open file abc and take file pointer in variable fp
2. Find record where fp.key = n
3. Delete the record.
4. Save result
5. Close file

For modification: let record to be modified has value of key field = n and value of Name is to be modified to xyz.

1. Open file abc and take file pointer in variable fp
2. Find record where fp.key=n
3. Set fp.name = 'xyz'
4. Save result
5. Close file.

For am unordered file:

For insertion: Let for record that is to be inserted value of Key field be n

1. Open file abc and take file pointer in variable fp
2. Seek end of file
3. Insert current record at this position.
4. Save the file data
5. Close file

For deletion: let record to be deleted has value for key field = n

1. Open file abc and take file pointer in variable fp
2. Find record where fp.key = n
3. Delete the record.
4. Save result
5. Close file

For modification: let record to be modified has value of key field = n and value of Name is to be modified to xyz.

1. Open file abc and take file pointer in variable fp
2. Find record where fp.key = n
3. Set fp.name = ‘xyz’
4. Save result

16.44. Can you think of techniques other than an unordered overflow file that can be used to make insertions in an ordered file more efficient?

Yes, we may think that it is possible to use an overflow file in which the records are chained together in a manner similar to the overflow for static hash files. The overflow records that should be inserted in each block of the ordered file are linked together in the overflow file, and a pointer to the first record in the linked list, that is kept in the block of the main file.

The list may or may not be kept ordered.

16.45. Suppose that we have a hash file of fixed-length records, and suppose that overflow is handled by chaining. Outline algorithms for insertion, deletion, and modification of a file record. State any assumptions you make.

Over flow is handled by chaining. Means, in a bucket. Multiple blocks are chained together and attached by a number of over flow buckets together.

In a hash structure. The insertion is done like this

Step 1:

Each bucket j stores a value ij all the entries that point to the same bucket have the same values on the first ij; bits

Step 2:

To locate the bucket containing search key kj;

- Compute $H(kj) = X$
- Use the first i high order nits of X as a displacement in to the bucket address table and follow the pointer to the appropriate bucket.

Step 3: T inserts a record with search key value k;

- Follow lookup procedure to locate the bucket, say j
- If there is room in bucket j, insert the record
- Otherwise the bucket must be split and insertion reattempted.

Deletion in hash file:-

To delete a key value,

Sept 1.

Locate it in its bucket and remove it

Step 2.

The bucket it self can be removed if it becomes empty

Step 3.

Coalescing of buckets is possible-can only coalesce with a “buddy” bucket having the same value of $i\ j$ and same $1_i - 1$ prefix, if one such bucket exists

Assumptions:-

→ Each key in the record is unique

→ Data file in the record is open

→ Overflow file is open

→ A bucket record has been defined

16.46. Can you think of techniques other than chaining to handle bucket overflow in external hashing?

To handle a bucket overflow in external hashing, there is a techniques like chaining and Trie-Based hashing.

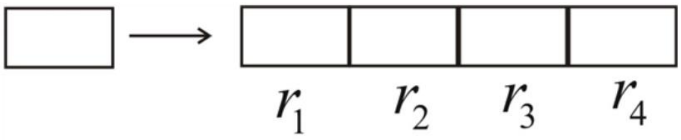
Through this technique:

- it allow the number of allocated buckets to grow and shrink as needed.

- Distributes records among buckets based on the values of the leading bits in their hash values.

We can show this technique by the following.

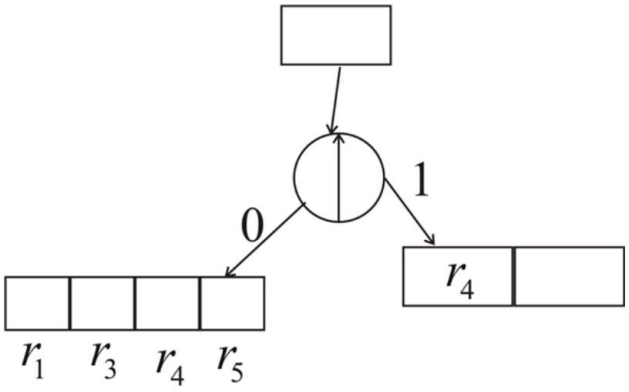
Let bucket of disk address is



Over flow is done by,

the bucket (block) based on the first binary digit of the hash address.

So, the address is split into



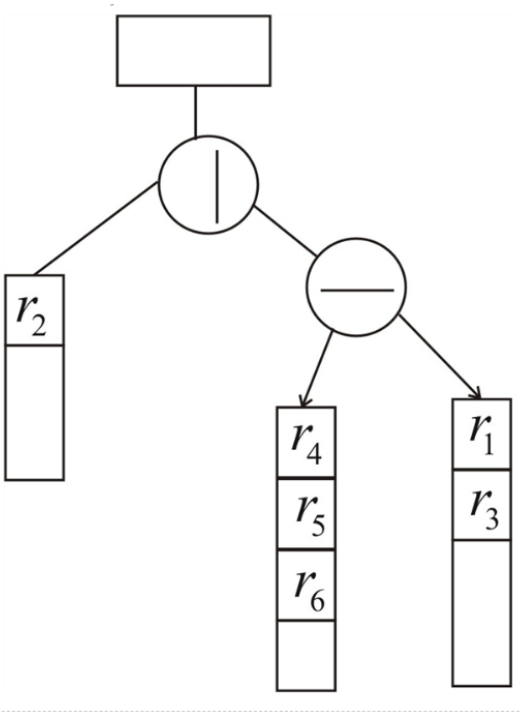
Here bulk flow is done and now again it is split on 2nd bit in the hash address

Ti show this,

Suppose we have:

Record Number	Hash address
r_1	0000
r_2	1000
r_3	0010
r_4	0100
r_5	0110

If we want to inset $r_3 = 0111$ in the previous structure thour the structure is comes like this



16.47. Write pseudocode for the insertion algorithms for linear hashing and for extendible hashing.

Pseudo code for the insertion algorithms:-

We assume that the elements in the hash table T are keys with no information.

The key *K* is identical to the element containing key *K*. Every slot contains either a key or Nil.

HASH – INSERT (T, K)

$i \leftarrow 0$

Report $j \leftarrow h(k,i)$

If $T[i] = \text{NIL}$

Then $T[i] \leftarrow k$

Return j

Else $i \leftarrow i+1$

Unitl $i = m$

Error “hash table over flow”

Pseudo code for the insertion algorithms for extendible mashing:-

Insertion

Algorithm: initialize (num buckets)

Input: desired number of buckets

1. Initialize array of linked lists;

Algorithm: in sert (key, value)

Input: key – value pair

// compute table entry:

Entry = key. Has code ( ) mod num buckets

If table [entry] is null

//no list present, so create one

Table [Entry] = new linked list;

Table [Entry].add (key. value)

Else

//otherwise, add to existing list

Table [entry].add (key. value)

End if.

16.49. Suppose that a file initially contains $r = 120{,}000$ records of $R = 200$ bytes each in an unsorted (heap) file. The block size $B = 2{,}400$ bytes, the average seek time $s = 16$ ms, the average rotational latency $rd = 8.3$ ms, and the block transfer time $btt = 0.8$ ms. Assume that 1 record is deleted for every 2 records added until the total number of active records is 240,000.

a. How many block transfers are needed to reorganize the file?

b. How long does it take to find a record right before reorganization?

c. How long does it take to find a record right after reorganization?

Let X = # of records are deleted and 2X= # of records added.

So, total active records = 240,000

= 120,000 - X + 2X.

X = 120,000

Physically records may deleting for reorganization is = 360,000.

(a)

No. of blocks for Reorganization = Blocks Read + Blocks Written.

-200 bytes/record and 2400 bytes/block gives us 12 records per block

- involves 360,000 records

360,000/12 = 30K blocks

-Writing involves 240,000 records

240000/12 = 20K blocks.

Total blocks transferred during reorganization = 30K + 20K

= 50K blocks.

(b)

On an average we assume that half the file will be read.

So, Time = (b/2)* btt = 15000 * 0.8 ms

= 12000 ms.

= 12 sec.

(c)

Time to locate a record after reorganization = (b/2) * btt

= 10000 * 0.8

= 8 sec.

16.50. Suppose we have a sequential (ordered) file of 100,000 records where each record is 240 bytes. Assume that $B = 2{,}400$ bytes, $s = 16$ ms, $rd = 8.3$ ms, and $btt = 0.8$ ms. Suppose we want to make X independent random record reads from the file. We could make X random block reads or we could perform one exhaustive read of the entire file looking for those X records. The question is to decide when it would be more efficient to perform one exhaustive read of the entire file than to perform X individual random reads. That is, what is the value for X when an exhaustive read of the file is more efficient than random X reads? Develop this as a function of X.

The records in the file are ordered sequentially.

Total number of records in the file (T_r) = 100000.

Size of each record (*rs*) = 240 bytes.

Size of each block (B) = 2400 bytes.

Average seek time (*s*) = 16 ms.

Average rotational latency (*rd*) = 8.3 ms.

Block transfer time (*btt*) = 0.8 ms.

Calculate the total number of blocks (T_B) in file using the formula $\frac{T_r \times rs}{B}$.

$$T_B = \frac{100000 \times 240}{2400}$$
$$= 10000 \text{ blocks}$$

Hence, total number of blocks in file (T_B) = 10000 blocks.

Calculate the time required for exhaustive reads (*er*) using the formula $s + rd + T_B \times btt$.

$$\begin{aligned} e_r &= 16 + 8.3 + 10000 \times 0.8 \\ &= 8024.3\,\text{ms} \end{aligned}$$

Hence, the time required for exhaustive read (*er*) = 8024.3 ms.

Consider X be the number of records need to be read.

The equation to decide the performance of one exhaustive read of the entire file is more efficient than performing *X* individual random reads follows:

Time required to perform *X* individual random reads > time required for exhaustive read

$$X(s + rd + btt) > er$$

$$\begin{aligned} X &> \frac{er}{s + rd + btt} \\ &> \frac{8024.3}{16 + 8.3 + 0.8} \\ &> \frac{8024.3}{25.1} \\ &> 319.69 \\ &\simeq 320 \end{aligned}$$

Therefore, when 320 or more individual random reads are required, then it is better to read the file exhaustively.

The function in X that relates the individual random reads and exhaustive reads is given by the following equation:

$$\boxed{X(s + rd + btt) > er}$$