DBMS Mid-2 Important Questions & Answer

1) What is File Organization? Explain about different file organization techniques?

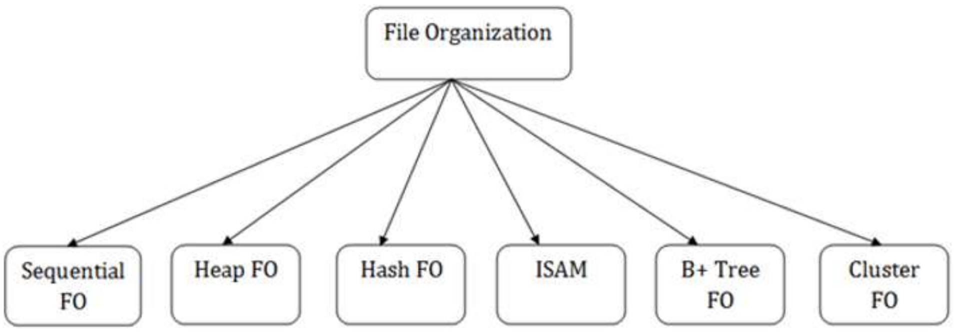
Ans:-

File Organization

- The **File** is a collection of records. Using the primary key, we can access the records.
- File organization is a logical relationship among various records. This method defines how file records are mapped on to disk blocks.
- File organization is used to describe the way in which the records are stored in terms of blocks, and the blocks are placed on the storage medium.
- The first approach to map the database to the file is to use the several files and store only one fixed length record in any given file.
- An alternative approach is to structure our files so that we can contain multiple lengths for records.
- Files of fixed length records are easier to implement than the files of variable length records.

Types of file organization:

- File organization contains various methods. These particular methods have pros and cons on the basis of access or selection.
- In the file organization, the programmer decides the best-suited file organization method according to his requirement.



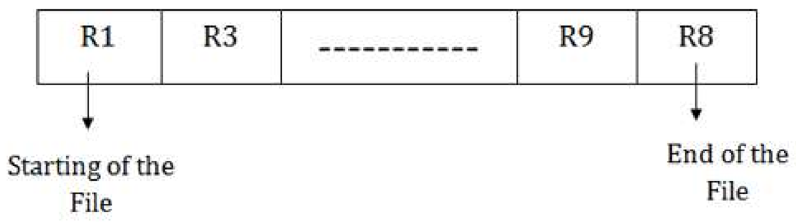
Sequential File Organization

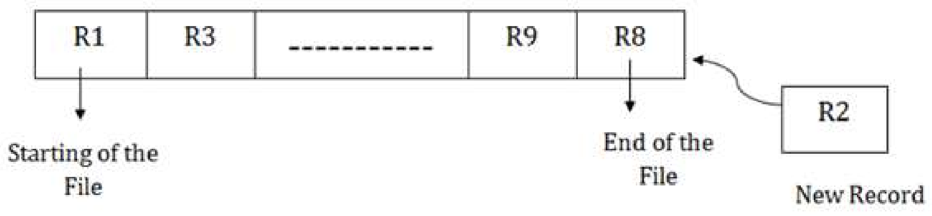
- This method is the easiest method for file organization.
- In this method, files are stored sequentially.
- This method can be implemented in two ways:

- **1. Pile File Method**
- **2. Sorted File Method**

1. Pile File Method

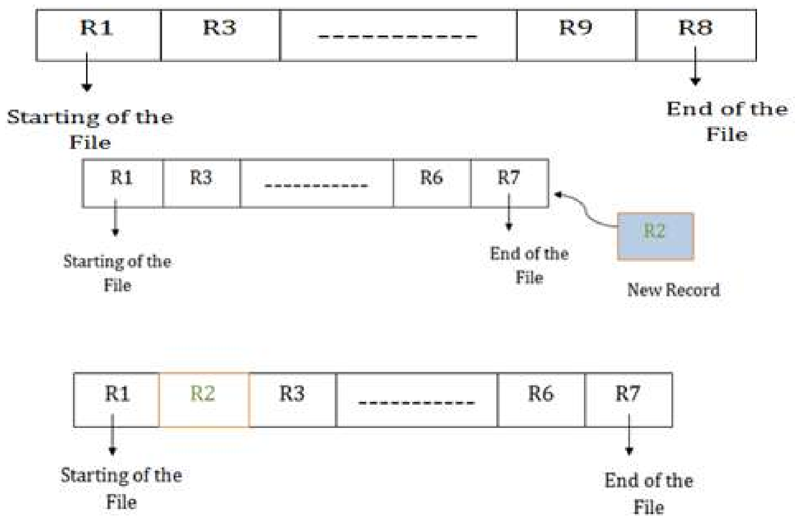
- It is a quite simple method. In this method, we store the record in a sequence, i.e., one after another. Here, the record will be inserted in the order in which they are inserted into tables.
- In case of updating or deleting of any record, the record will be searched in the memory blocks. When it is found, then it will be marked for deleting, and the new record is inserted.





2. Sorted File Method

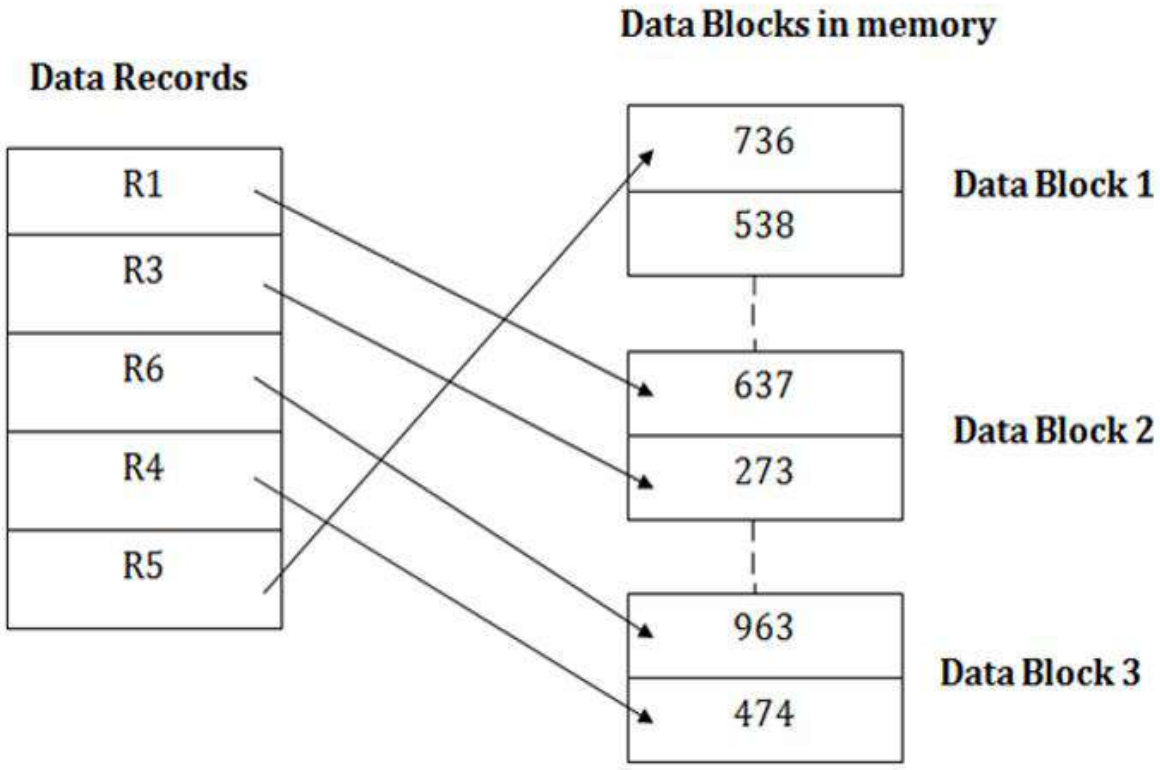
- In this method, the new record is always inserted at the file's end, and then it will sort the sequence in ascending or descending order.
- Sorting of records is based on any primary key or any other key.
- In the case of modification of any record, it will update the record and then sort the file, and lastly, the updated record is placed in the right place.



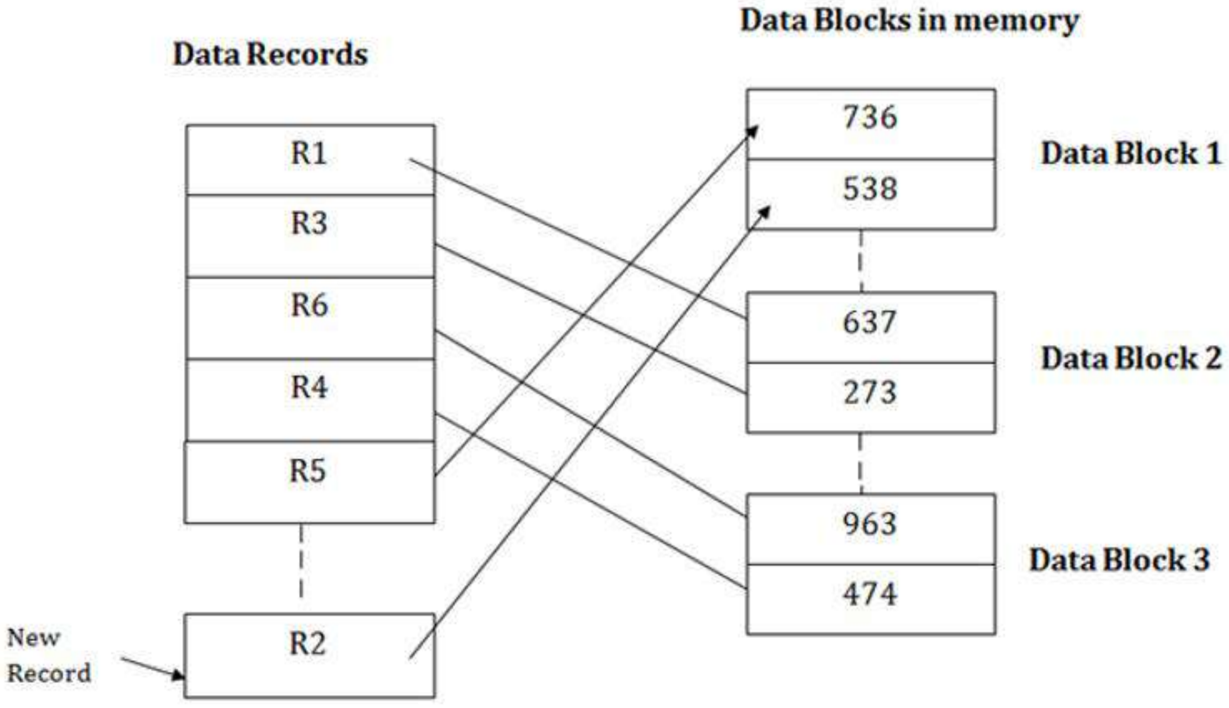
2. Heap file organization

- It is the simplest and most basic type of organization. It works with data blocks.
- In heap file organization, the records are inserted at the file's end. When the records are inserted, it doesn't require the sorting and ordering of records.
- When the data block is full, the new record is stored in some other block.
- This new data block need not to be the very next data block, but it can select any data block in the memory to store new records. The heap file is also known as an unordered file.
- In the file, every record has a unique id, and every page in a file is of the same size. It is the DBMS responsibility to store and manage the new records.

Heap file organization(cont..)



Insert a new record

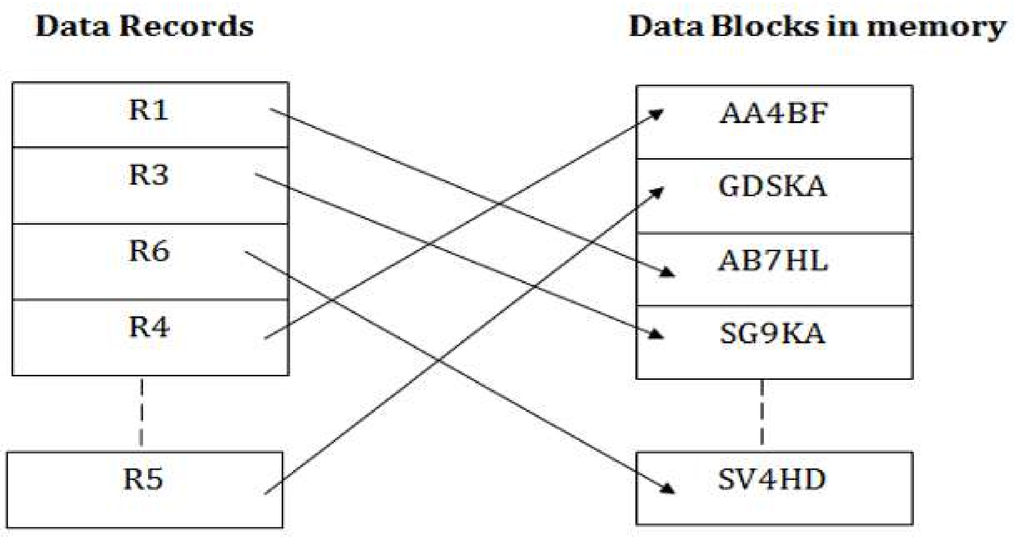


Heap file organization(cont..)

- If we want to search, update or delete the data in heap file organization, then we need to traverse the data from staring of the file till we get the requested record.
- If the database is very large then searching, updating or deleting of record will be time-consuming because there is no sorting or ordering of records.
- In the heap file organization, we need to check all the data until we get the requested record.

3. Hash File Organization

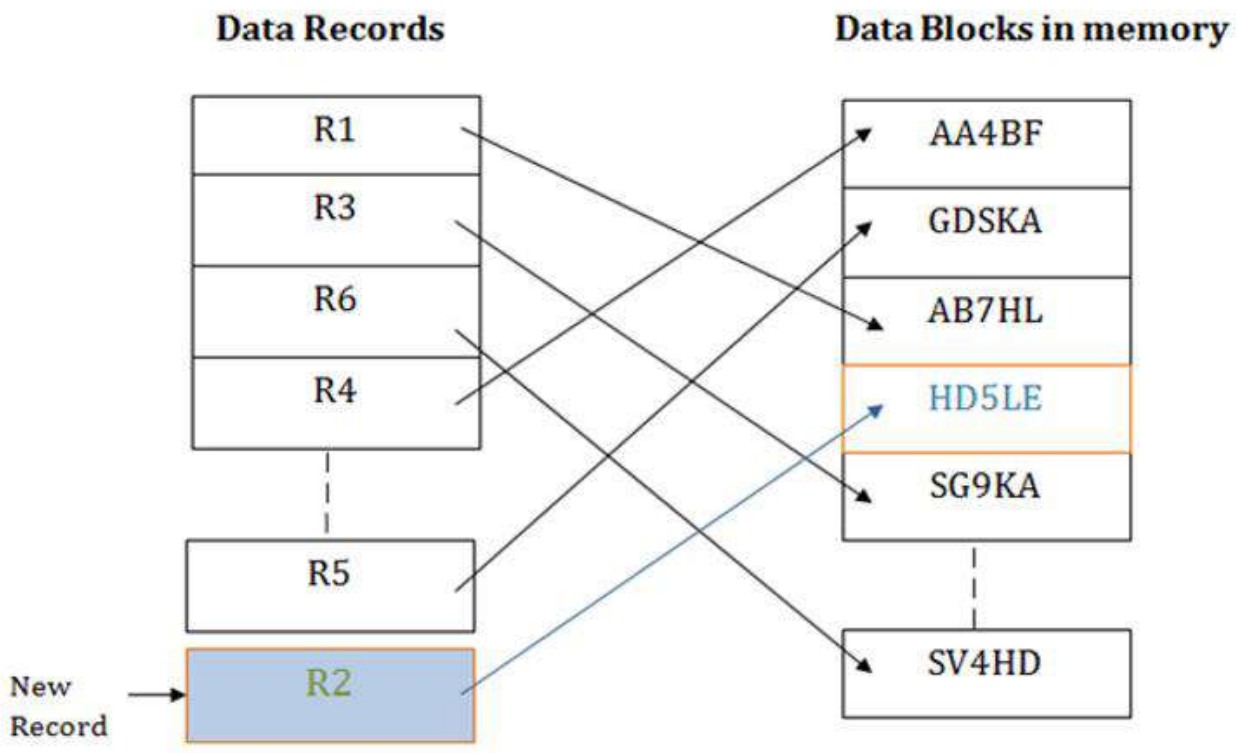
- Hash File Organization uses the computation of hash function on some fields of the records.
- The hash function's output determines the location of disk block where the records are to be placed.



Hash File Organization(cont..)

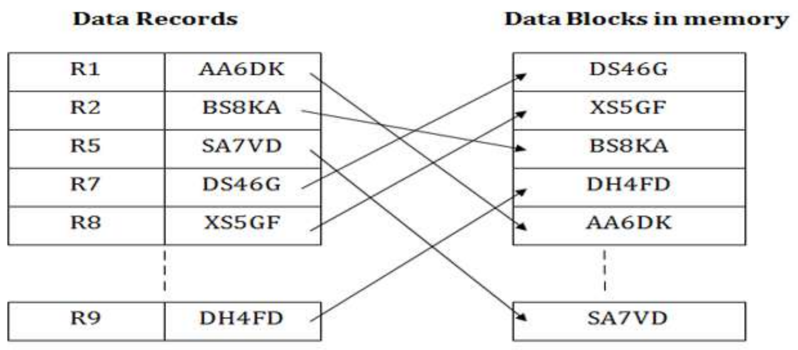
- When a record has to be received using the hash key columns, then the address is generated, and the whole record is retrieved using that address.
- In the same way, when a new record has to be inserted, then the address is generated using the hash key and record is directly inserted.
- The same process is applied in the case of delete and update.
- In this method, there is no effort for searching and sorting the entire file. In this method, each record will be stored randomly in the memory.

Hash File Organization(cont..)



4. Indexed Sequential Access Method (ISAM)

- ISAM method is an advanced sequential file organization.
- In this method, records are stored in the file using the primary key.
- An index value is generated for each primary key and mapped with the record.
- This index contains the address of the record in the file.



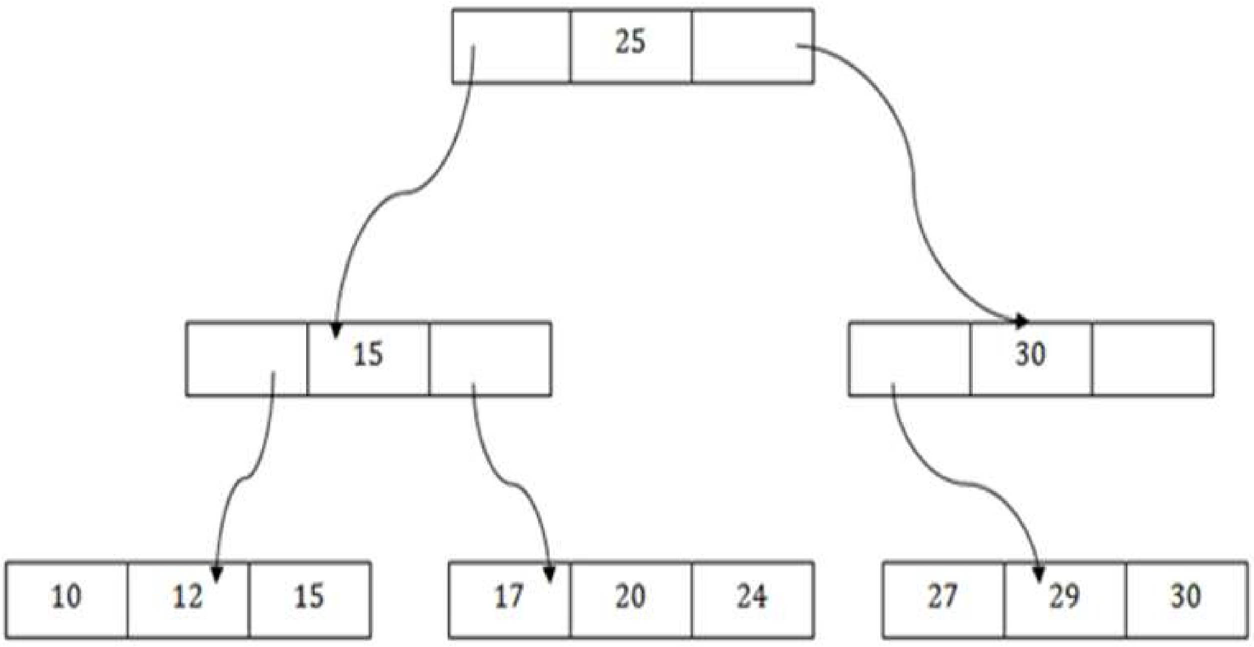
Indexed Sequential Access Method (ISAM)-Cont..

- If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.

5.B+ File Organization

- B+ tree file organization is the advanced method of an indexed sequential access method.
- It uses a tree-like structure to store records in File.
- It uses the same concept of key-index where the primary key is used to sort the records.
- For each primary key, the value of the index is generated and mapped with the record.
- The B+ tree is similar to a binary search tree (BST), but it can have more than two children.
- In this method, all the records are stored only at the leaf node.
- Intermediate nodes act as a pointer to the leaf nodes. They do not contain any records.

5.B+ File Organization(Cont..)



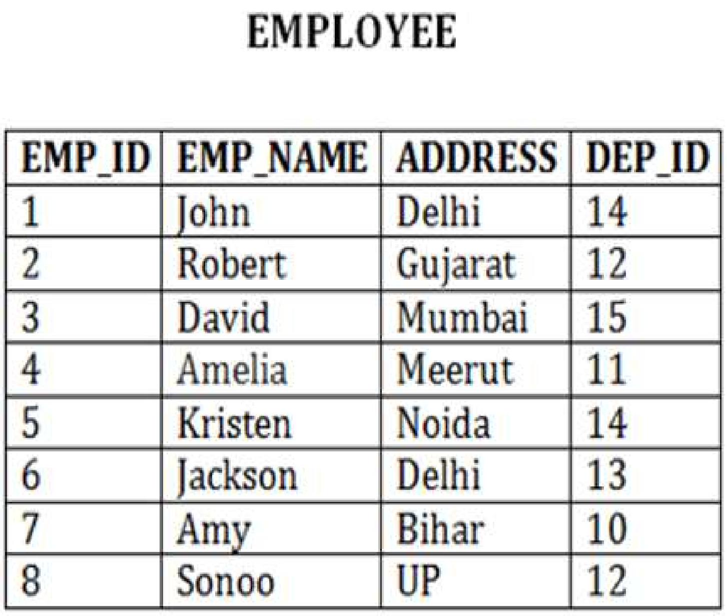
The above B+ tree shows that:

- There is one root node of the tree, i.e., 25.
- There is an intermediary layer with nodes. They do not store the actual record. They have only pointers to the leaf node.
- The nodes to the left of the root node contain the prior value of the root and nodes to the right contain next value of the root, i.e., 15 and 30 respectively.
- There is only one leaf node which has only values, i.e., 10, 12, 17, 20, 24, 27 and 29.
- Searching for any record is easier as all the leaf nodes are balanced.
- In this method, searching any record can be traversed through the single path and accessed easily.

6.Cluster File Organization

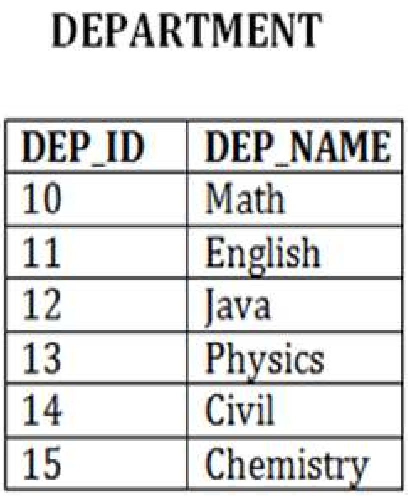
- When the two or more records are stored in the same file, it is known as clusters.
- These files will have two or more tables in the same data block, and key attributes which are used to map these tables together are stored only once.
- This method reduces the cost of searching for various records in different files.
- The cluster file organization is used when there is a frequent need for joining the tables with the same condition.
- These joins will give only a few records from both tables. In the given example, we are retrieving the record for only particular departments.
- This method can't be used to retrieve the record for the entire department.

Cluster File Organization(cont..)



EMPLOYEE

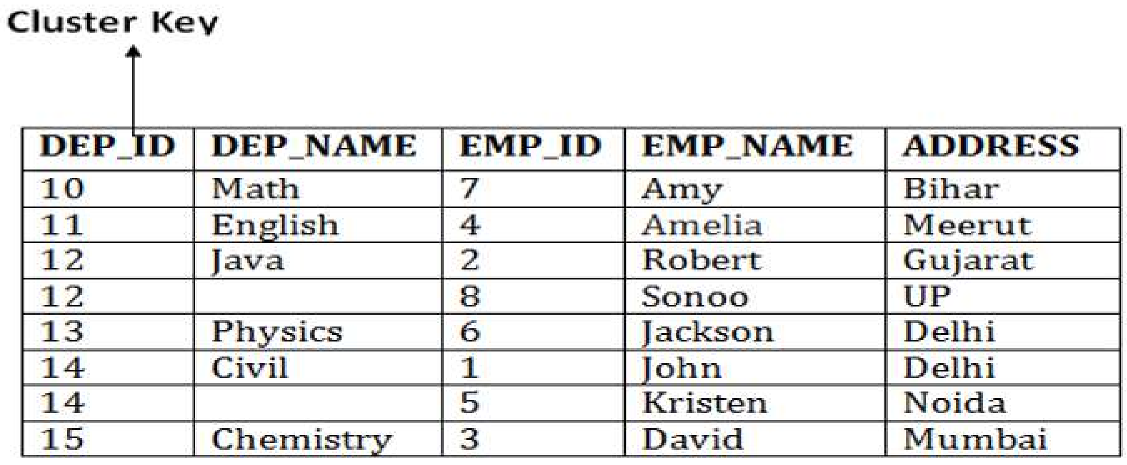
EMP_ID	EMP_NAME	ADDRESS	DEP_ID
1	John	Delhi	14
2	Robert	Gujarat	12
3	David	Mumbai	15
4	Amelia	Meerut	11
5	Kristen	Noida	14
6	Jackson	Delhi	13
7	Amy	Bihar	10
8	Sonoo	UP	12



DEPARTMENT

DEP_ID	DEP_NAME
10	Math
11	English
12	Java
13	Physics
14	Civil
15	Chemistry

Cluster File Organization(cont..)


Cluster Key

DEP_ID	DEP_NAME	EMP_ID	EMP_NAME	ADDRESS
10	Math	7	Amy	Bihar
11	English	4	Amelia	Meerut
12	Java	2	Robert	Gujarat
12		8	Sonoo	UP
13	Physics	6	Jackson	Delhi
14	Civil	1	John	Delhi
14		5	Kristen	Noida
15	Chemistry	3	David	Mumbai

•In this method, we can directly insert, update or delete any record. Data is sorted based on the key with which searching is done.

• Cluster key is a type of key with which joining of the table is performed.

Types of Cluster file organization:

- Cluster file organization is of two types:

- **1. Indexed Clusters:**
- In indexed cluster, records are grouped based on the cluster key and stored together.
- The above EMPLOYEE and DEPARTMENT relationship is an example of an indexed cluster. Here, all the records are grouped based on the cluster key- DEP_ID and all the records are grouped.

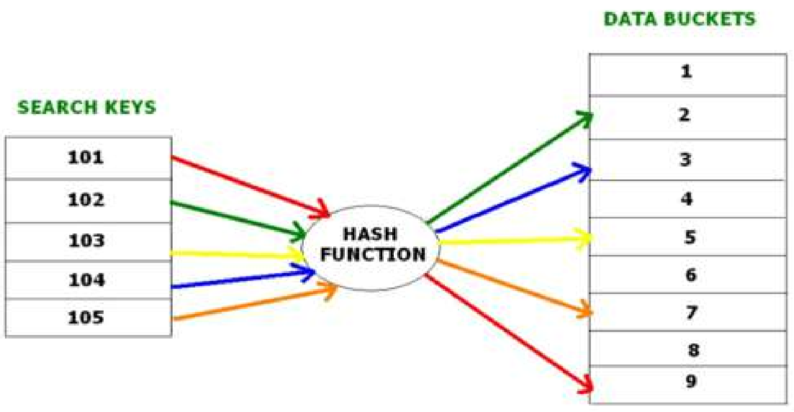
- **2. Hash Clusters:**
- It is similar to the indexed cluster. In hash cluster, instead of storing the records based on the cluster key, we generate the value of the hash key for the cluster key and store the records with the same hash key value.

2) Explain about Hash based indexing with suitable examples.

Ans:-

1. Hash-Based indexes

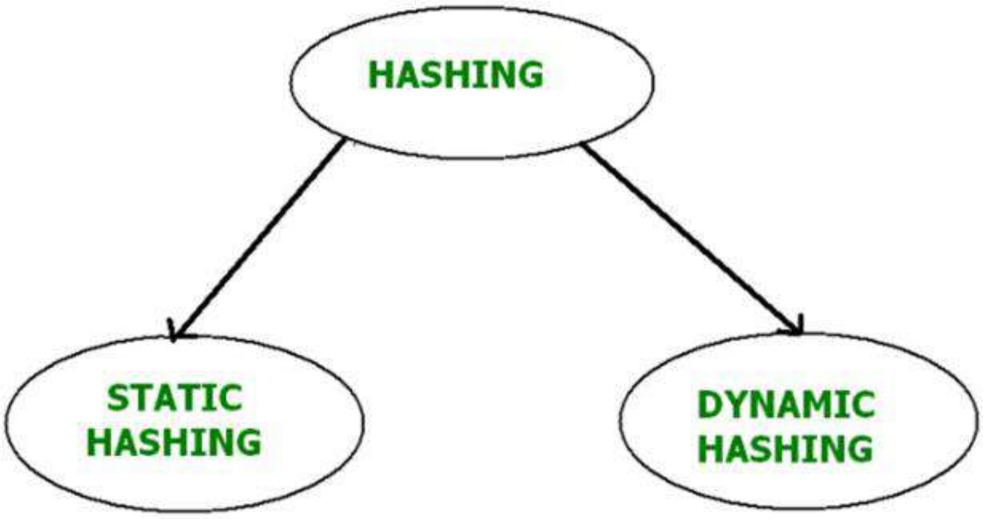
- **In the Hash File Organization we have**
- **Data bucket** – Data buckets are the memory locations where the records are stored. These buckets are also considered as *Unit Of Storage*.
- **Hash Function** – Hash function is a mapping function that maps all the set of search keys to actual record address.
- **Hash Index**-The prefix of an entire hash value is taken as a hash index.



1. Hash-Based indexes

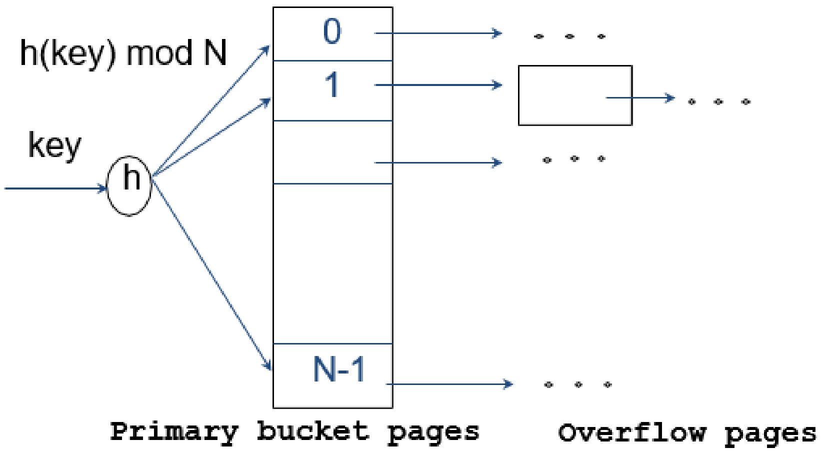
- Index is a collection of **buckets**.
  - Bucket = primary page plus zero or more overflow pages.
  - Buckets contain data entries.
- **Hashing function h:**
  - **h**(r) = bucket in which data entry for record r belongs.
  - **h** looks at **search key** fields of r.

- Hashing is further divided into two sub categories :



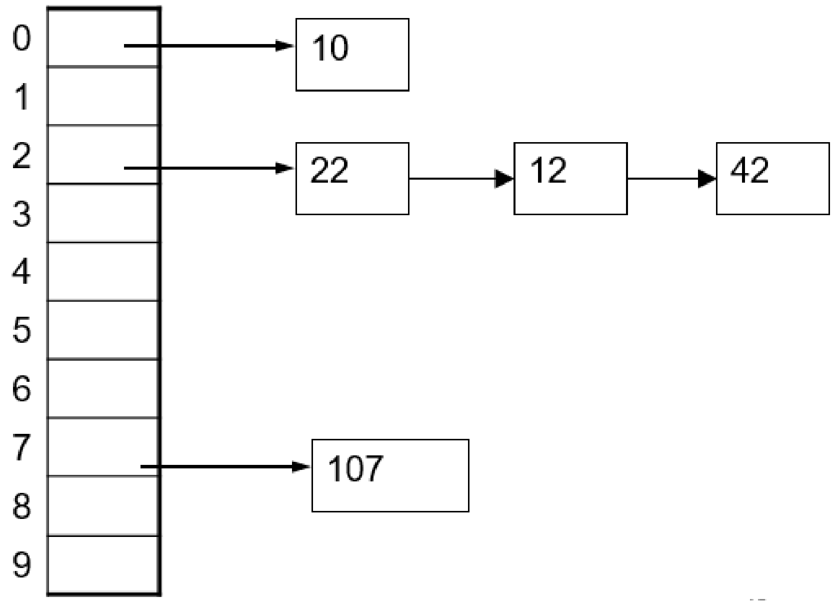
1.Static Hashing

- # primary pages fixed, allocated sequentially, never de-allocated; overflow pages if needed.
- **h(*k*) mod N** = bucket to which data entry with key *k* belongs. (N = # of buckets)



Chaining

- **Chaining**: All keys that map to the same hash value are kept in a linked list

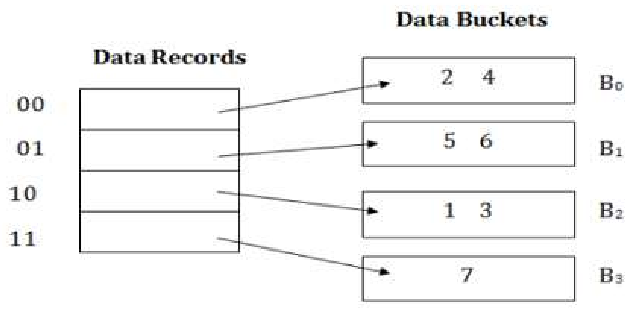


2.Dynamic Hashing

- The dynamic hashing method is used to overcome the problems of static hashing like bucket overflow.

- In this method, data buckets grow or shrink as the records increases or decreases. This method is also known as Extendable hashing method.

- This method makes hashing dynamic, i.e., it allows insertion or deletion without resulting in poor performance.



3) Explain about the 5 Normal Forms?

Ans:-

Normal forms

Well structured relation

- A relation that contain minimum redundancy and allow users to safely insert, delete, update records in a table is called well structured.

Normal form

- It is a state of a relation obtained by applying simple rules regarding FDs.

Normalization

- The process of decomposing a relation which having the anomalies into smaller relation to produce well structured relations is called normalization.

First Normal Form (1 N F)

Each attribute must be atomic

- No repeating columns within a row.
- No multi-valued columns.

1NF simplifies attributes

- Queries become easier.

1 N F

Employee (unnormalized)

emp_no	name	dept_no	dept_name	skills
1	Kevin Jacobs	201	R&D	C, Perl, Java
2	Barbara Jones	224	IT	Linux, Mac
3	Jake Rivera	201	R&D	DB2, Oracle, Java

Employee (1NF)

emp_no	name	dept_no	dept_name	skills
1	Kevin Jacobs	201	R&D	C
1	Kevin Jacobs	201	R&D	Perl
1	Kevin Jacobs	201	R&D	Java
2	Barbara Jones	224	IT	Linux
2	Barbara Jones	224	IT	Mac
3	Jake Rivera	201	R&D	DB2
3	Jake Rivera	201	R&D	Oracle
3	Jake Rivera	201	R&D	Java

Second Normal Form (2 N F)

Each attribute must be functionally dependent on the primary key.

- Functional dependence - the property of one or more attributes that uniquely determines the value of other attributes.
- Any non-dependent attributes are moved into a smaller (subset) table.

2NF improves data integrity.

- Prevents update, insert, and delete anomalies.

Functional Dependence

Employee (1NF)

emp_no	name	dept_no	dept_name	skills
1	Kevin Jacobs	201	R&D	C
1	Kevin Jacobs	201	R&D	Perl
1	Kevin Jacobs	201	R&D	Java
2	Barbara Jones	224	IT	Linux
2	Barbara Jones	224	IT	Mac
3	Jake Rivera	201	R&D	DB2
3	Jake Rivera	201	R&D	Oracle
3	Jake Rivera	201	R&D	Java

Name, dept_no, and dept_name are functionally dependent on emp_no.

(emp_no -> name, dept_no, dept_name)

Skills is not functionally dependent on emp_no since it is not unique to each emp_no.

2 N F

Employee (1NF)

emp_no	name	dept_no	dept_name	skills
1	Kevin Jacobs	201	R&D	C
1	Kevin Jacobs	201	R&D	Perl
1	Kevin Jacobs	201	R&D	Java
2	Barbara Jones	224	IT	Linux
2	Barbara Jones	224	IT	Mac
3	Jake Rivera	201	R&D	DB2
3	Jake Rivera	201	R&D	Oracle
3	Jake Rivera	201	R&D	Java

Employee (2NF)

emp_no	name	dept_no	dept_name
1	Kevin Jacobs	201	R&D
2	Barbara Jones	224	IT
3	Jake Rivera	201	R&D

Skills (2NF)

emp_no	skills
1	C
1	Perl
1	Java
2	Linux
2	Mac
3	DB2
3	Oracle
3	Java

Third Normal Form (3NF)

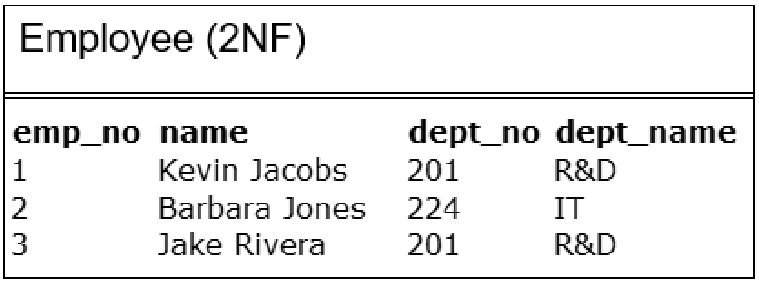
Remove transitive dependencies.

- Transitive dependence - two separate entities exist within one table.
- Any transitive dependencies are moved into a smaller (subset) table.

3NF further improves data integrity.

- Prevents update, insert, and delete anomalies.

Transitive Dependence



Employee (2NF)

emp_no	name	dept_no	dept_name
1	Kevin Jacobs	201	R&D
2	Barbara Jones	224	IT
3	Jake Rivera	201	R&D

Dept_no and dept_name are functionally dependent on emp_no however, department can be considered a separate entity.

3 N F

Employee (2NF)

emp_no	name	dept_no	dept_name
1	Kevin Jacobs	201	R&D
2	Barbara Jones	224	IT
3	Jake Rivera	201	R&D

Employee (3NF)

emp_no	name	dept_no
1	Kevin Jacobs	201
2	Barbara Jones	224
3	Jake Rivera	201

Department (3NF)

dept_no	dept_name
201	R&D
224	IT

BCNF(Boyce Codd Normal form)

- BCNF is a Advanced version of 3NF.It is Stricter than 3NF.
- A table id Functional Dependency X->Y is the Super key of the Table.
- For BCNF Table should be in 3NF.

Fourth Normal Form (4 N F)

- 4th Normal Form
  - BCNF with no multi valued dependencies
  - Create separate tables for each separate functional dependency

(a) The EMP relation with two MVDs: ENAME —>> PNAME and ENAME —>> DNAME. (b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

(a) **EMP**

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

(b) **EMP_PROJECTS**

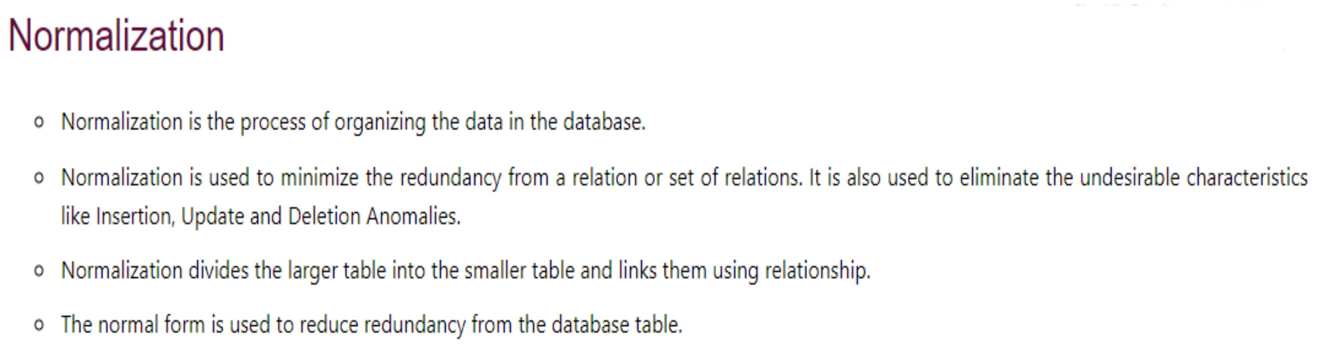
ENAME	PNAME
Smith	X
Smith	Y

EMP_DEPENDENTS

ENAME	DNAME
Smith	John
Smith	Anna

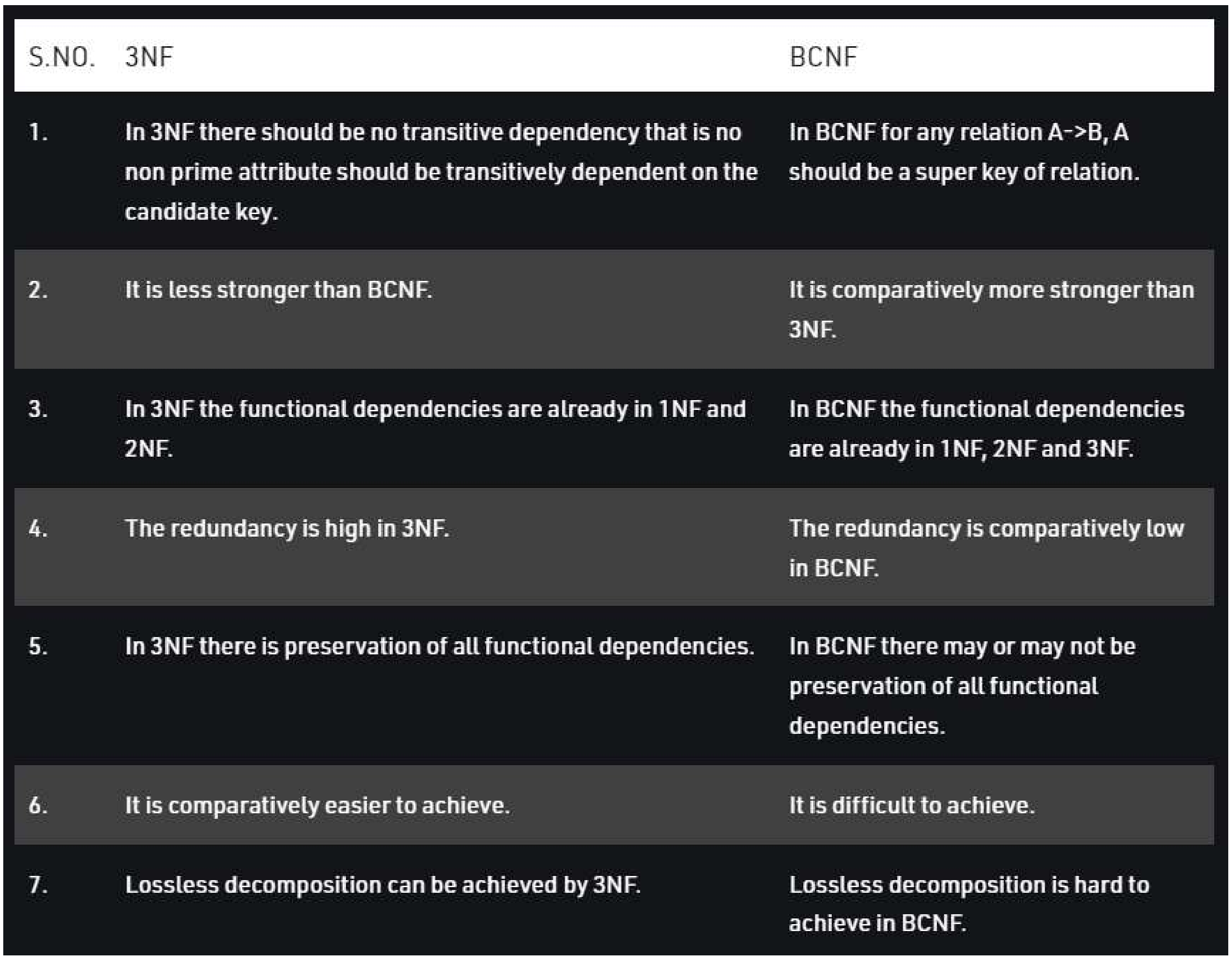
4) Define Normalization. Differentiate between 3NF and BCNF.

Ans:-



Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.



S.NO.	3NF	BCNF
1.	In 3NF there should be no transitive dependency that is no non prime attribute should be transitively dependent on the candidate key.	In BCNF for any relation A->B, A should be a super key of relation.
2.	It is less stronger than BCNF.	It is comparatively more stronger than 3NF.
3.	In 3NF the functional dependencies are already in 1NF and 2NF.	In BCNF the functional dependencies are already in 1NF, 2NF and 3NF.
4.	The redundancy is high in 3NF.	The redundancy is comparatively low in BCNF.
5.	In 3NF there is preservation of all functional dependencies.	In BCNF there may or may not be preservation of all functional dependencies.
6.	It is comparatively easier to achieve.	It is difficult to achieve.
7.	Lossless decomposition can be achieved by 3NF.	Lossless decomposition is hard to achieve in BCNF.

5) Explain 2-phase locking protocol in detail.

Ans:-

Two-phase locking defines how transactions acquire and relinquish locks. Two-phase locking guarantees serializability, but it does not prevent deadlocks. The two phases are:

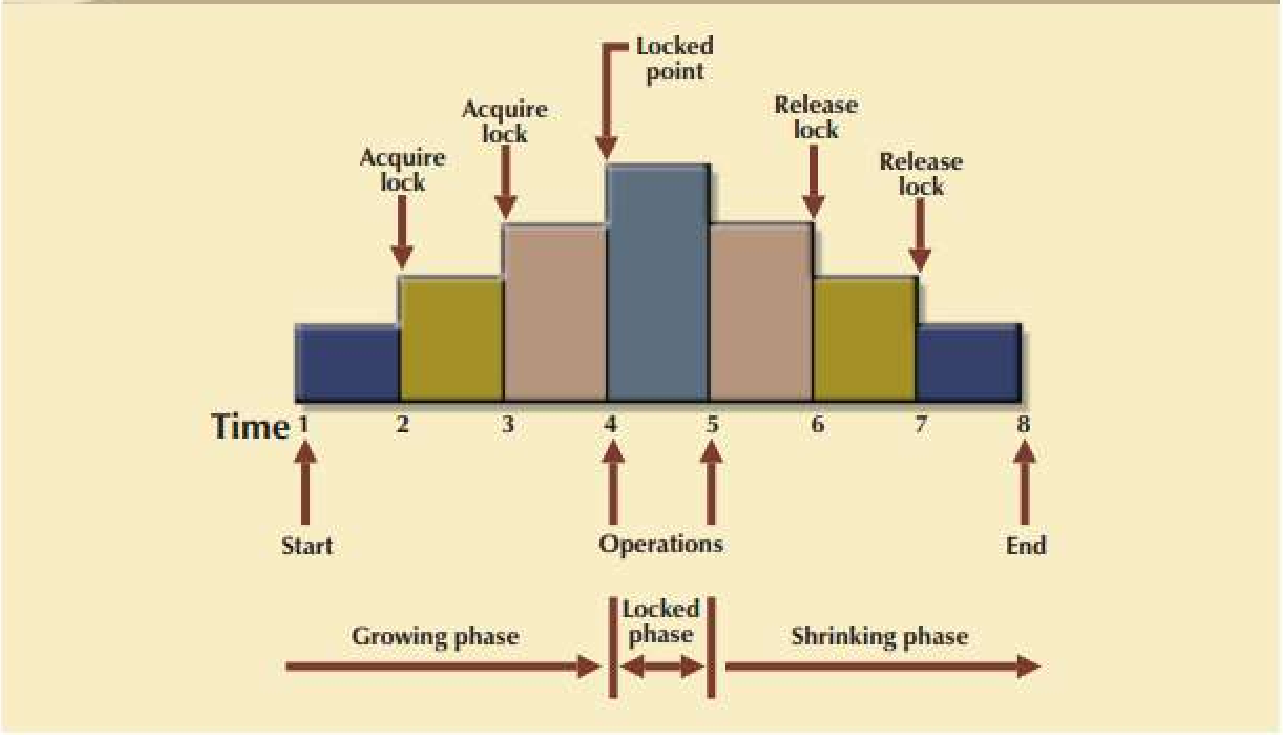
1. A growing phase, in which a transaction acquires all required locks without unlocking any data. Once all locks have been acquired, the transaction is in its locked point.
2. A shrinking phase, in which a transaction releases all locks and cannot obtain any new lock.

The two-phase locking protocol is governed by the following rules:

- Two transactions cannot have conflicting locks.
- No unlock operation can precede a lock operation in the same transaction.
- No data are affected until all locks are obtained—that is, until the transaction is in its locked point.

Figure 10.7 depicts the two-phase locking protocol.

FIGURE 10.7 Two-phase locking protocol

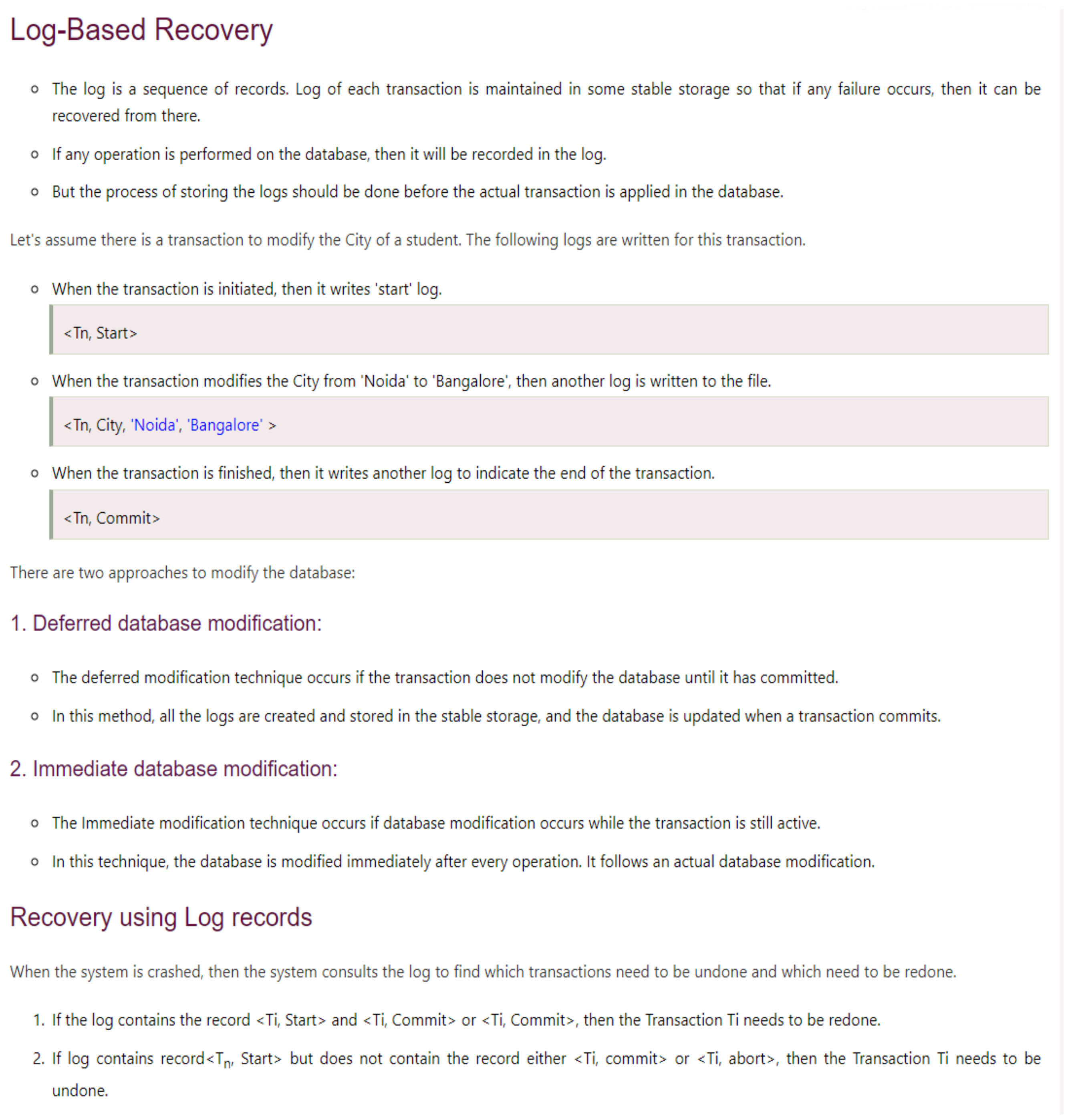


In this example, the transaction acquires all of the locks it needs until it reaches its locked point. (In this example, the transaction requires two locks.) When the locked point is reached, the data are modified to conform to the transaction requirements. Finally, the transaction is completed as it releases all of the locks it acquired in the first phase.

Two-phase locking increases the transaction processing cost and might cause additional undesirable effects. One undesirable effect is the possibility of creating deadlocks.

6) Explain Log based Recovery.

Ans:-



Log-Based Recovery

- The log is a sequence of records. Log of each transaction is maintained in some stable storage so that if any failure occurs, then it can be recovered from there.
- If any operation is performed on the database, then it will be recorded in the log.
- But the process of storing the logs should be done before the actual transaction is applied in the database.

Let's assume there is a transaction to modify the City of a student. The following logs are written for this transaction.

- When the transaction is initiated, then it writes 'start' log.

  ```
  <Tn, Start>
  ```

- When the transaction modifies the City from 'Noida' to 'Bangalore', then another log is written to the file.

  ```
  <Tn, City, 'Noida', 'Bangalore' >
  ```

- When the transaction is finished, then it writes another log to indicate the end of the transaction.

  ```
  <Tn, Commit>
  ```

There are two approaches to modify the database:

1. Deferred database modification:

- The deferred modification technique occurs if the transaction does not modify the database until it has committed.
- In this method, all the logs are created and stored in the stable storage, and the database is updated when a transaction commits.

2. Immediate database modification:

- The Immediate modification technique occurs if database modification occurs while the transaction is still active.
- In this technique, the database is modified immediately after every operation. It follows an actual database modification.

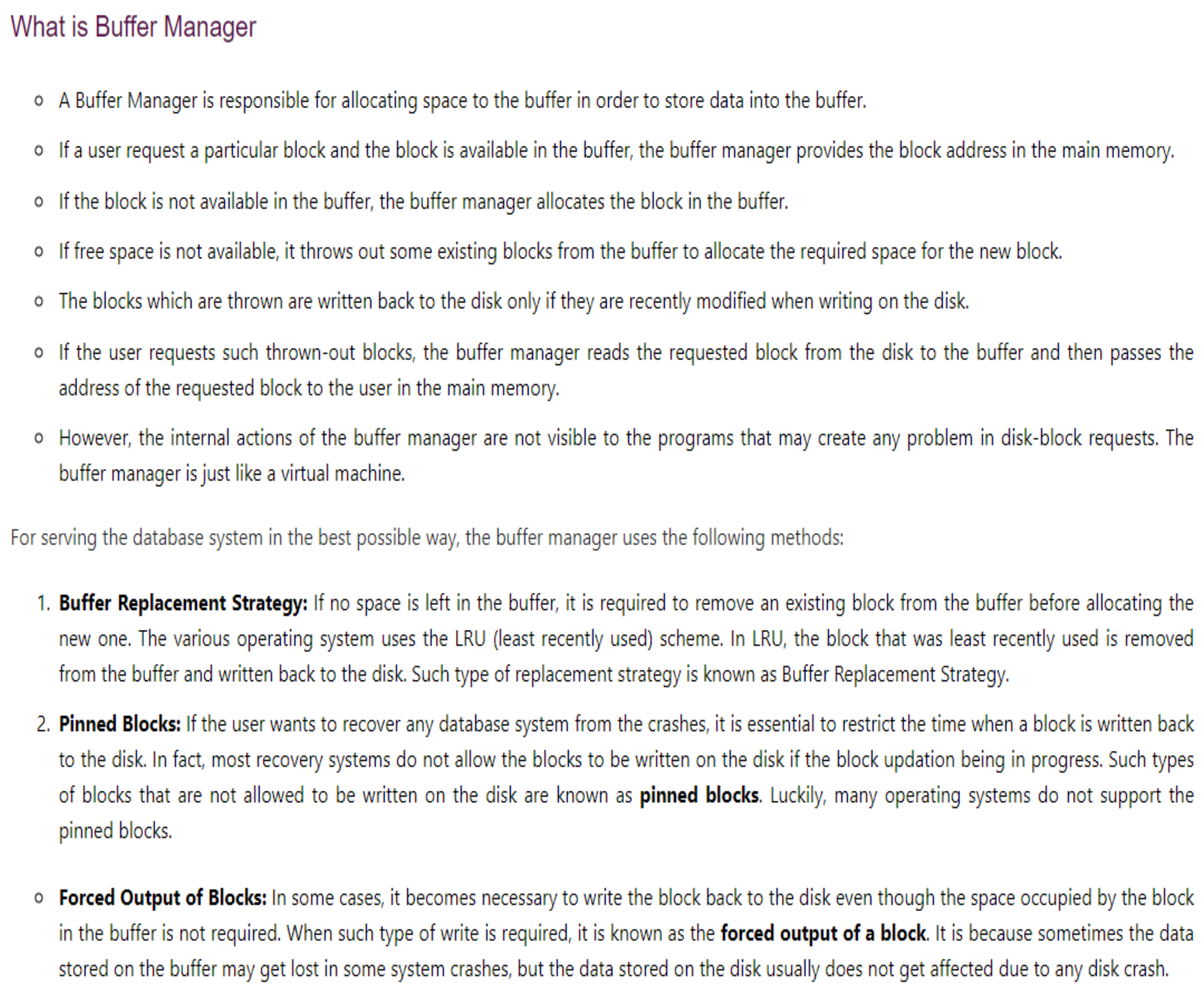
Recovery using Log records

When the system is crashed, then the system consults the log to find which transactions need to be undone and which need to be redone.

1. If the log contains the record <Ti, Start> and <Ti, Commit> or <Ti, Commit>, then the Transaction Ti needs to be redone.
2. If log contains record<T_n, Start> but does not contain the record either <Ti, commit> or <Ti, abort>, then the Transaction Ti needs to be undone.

7) Explain about Buffer Management?

Ans:-



What is Buffer Manager

- A Buffer Manager is responsible for allocating space to the buffer in order to store data into the buffer.
- If a user request a particular block and the block is available in the buffer, the buffer manager provides the block address in the main memory.
- If the block is not available in the buffer, the buffer manager allocates the block in the buffer.
- If free space is not available, it throws out some existing blocks from the buffer to allocate the required space for the new block.
- The blocks which are thrown are written back to the disk only if they are recently modified when writing on the disk.
- If the user requests such thrown-out blocks, the buffer manager reads the requested block from the disk to the buffer and then passes the address of the requested block to the user in the main memory.
- However, the internal actions of the buffer manager are not visible to the programs that may create any problem in disk-block requests. The buffer manager is just like a virtual machine.

For serving the database system in the best possible way, the buffer manager uses the following methods:

1. **Buffer Replacement Strategy:** If no space is left in the buffer, it is required to remove an existing block from the buffer before allocating the new one. The various operating system uses the LRU (least recently used) scheme. In LRU, the block that was least recently used is removed from the buffer and written back to the disk. Such type of replacement strategy is known as Buffer Replacement Strategy.
2. **Pinned Blocks:** If the user wants to recover any database system from the crashes, it is essential to restrict the time when a block is written back to the disk. In fact, most recovery systems do not allow the blocks to be written on the disk if the block updation being in progress. Such types of blocks that are not allowed to be written on the disk are known as **pinned blocks**. Luckily, many operating systems do not support the pinned blocks.

- **Forced Output of Blocks:** In some cases, it becomes necessary to write the block back to the disk even though the space occupied by the block in the buffer is not required. When such type of write is required, it is known as the **forced output of a block**. It is because sometimes the data stored on the buffer may get lost in some system crashes, but the data stored on the disk usually does not get affected due to any disk crash.

8) Write a short note on Serializability and Recoverability.

Ans:-

Types of serializability

There are two types of serializability −

View serializability

A schedule is view-serializability if it is viewed equivalent to a serial schedule.

The rules it follows are as follows −

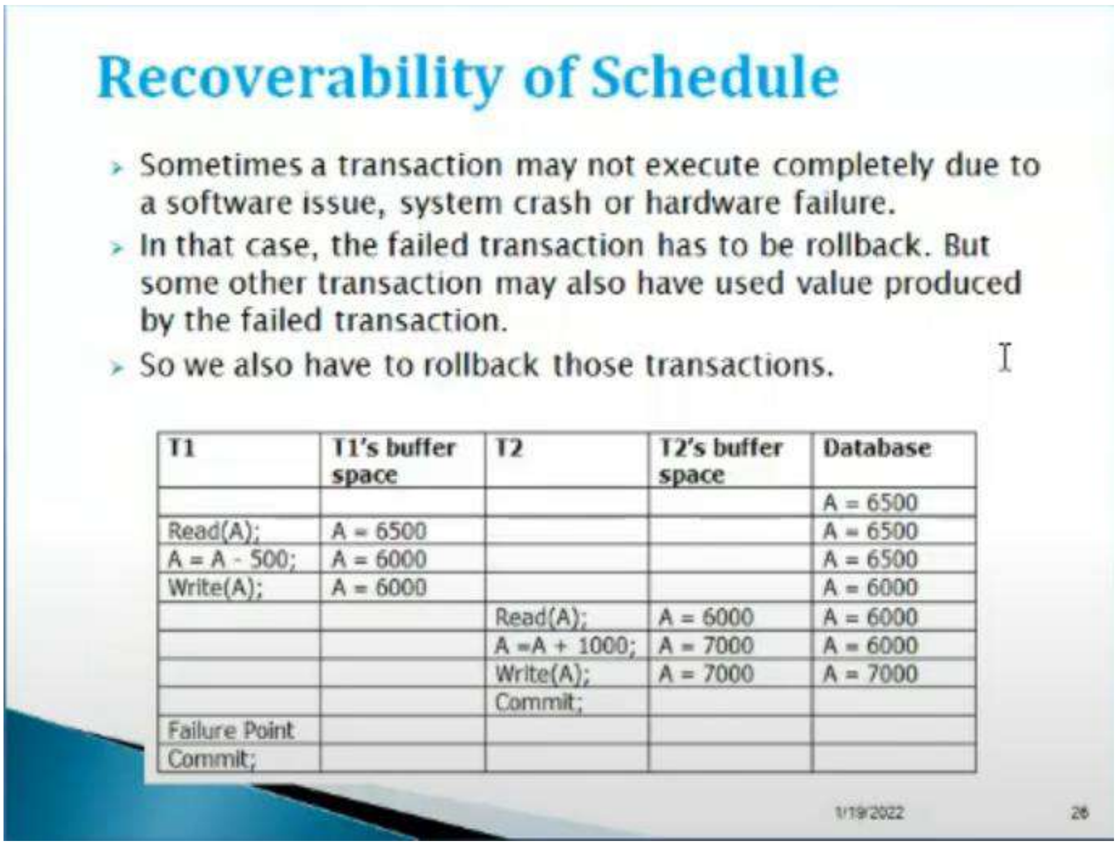
- T1 is reading the initial value of A, then T2 also reads the initial value of A.
- T1 is the reading value written by T2, then T2 also reads the value written by T1.
- T1 is writing the final value, and then T2 also has the write operation as the final value.

Conflict serializability

It orders any conflicting operations in the same way as some serial execution. A pair of operations is said to conflict if they operate on the same data item and one of them is a write operation.

That means

- Readi(x) readj(x) - non conflict read-read operation
- Readi(x) writej(x) - conflict   read-write operation.
- Writei(x) readj(x) - conflict   write-read operation.
- Writei(x) writej(x) - conflict   write-write operation.



Recoverability of Schedule

- Sometimes a transaction may not execute completely due to a software issue, system crash or hardware failure.
- In that case, the failed transaction has to be rollback. But some other transaction may also have used value produced by the failed transaction.
- So we also have to rollback those transactions.

T1	T1's buffer space	T2	T2's buffer space	Database
				A = 6500
Read(A);	A = 6500			A = 6500
A = A - 500;	A = 6000			A = 6500
Write(A);	A = 6000			A = 6000
		Read(A);	A = 6000	A = 6000
		A =A + 1000;	A = 7000	A = 6000
		Write(A);	A = 7000	A = 7000
		Commit;		
Failure Point				
Commit;				

1/19/2022

26

All the best
Now do the
rest...