File system vs DBMS :-

(1) Let a 1KB data is in a file of size 1GB. If we use file system and search for that data, then we will get that 1GB file. Now we have to search for that data from that large file. so this is more time consuming.

But in DBMS if we will search for that data through SQL server and we will get that data only not the file. So it is less time consuming.

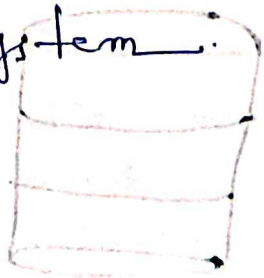
(2) we can record attributes of file for file system.

But DBMS is attribute independent.

(3) There are no Concurrency Protocal in file syst.

But DBMS has Protocal for Concurrency.

(4) There are Role Based Security in DBMS but not in file system.



1-a

1a) Starvation:- Starvation, or Livelock is the situation when a transaction has to wait for a indefinite period of time as a resource is locked.

Solution of starvation:-

↳ Increasing priority ⇒

Starvation occurs when a transaction has to wait for an indefinite time, in this situation, we can increase priority of that particular transaction.

↳ Modification in victim selection Algorithms ⇒

If a transaction has been a victim of repeated selections, then the algorithm can be modified by increasing its priority over other transactions.

↳ First come First come approach ⇒

FCFS can be adopted, in which the transaction has acquired a lock on an item in the order, in which requested lock.

↳ wait, die and wound wait schemes ⇒

There are schemes that use the timestamp ordering mechanism of transaction.

1-b

Third normal form (3NF):

- Table must be in 2NF
- There should be no transitive dependency

Transitive Dependency:
non prime attribute is dependent on another non prime attribute.

Roll no.	state	city
1	Punjab	Mohali
2	Haryana	Ambala
3	Punjab	Mohali
4	Haryana	Ambala
5	Bihar	Patna

Candidate key → Roll no.

FD ⇒ Roll no. → state ⇒ Prime → non Prime
state → city ⇒ non prime → non prime
∴ Roll no. → city ⇒ Transitive dependency

The table is not in 3NF.

Roll no.	state
1	Punjab
2	Haryana
3	Punjab
4	Haryana
5	Bihar

In 3NF

Roll no.	city
1	Mohali
2	Ambala
3	Mohali
4	Ambala
5	Patna

In 3NF

Boyce Codd Normal form: (BCNF)

- Stronger form of 3NF
- Left hand side of all functional dependency should be candidate key as super key

Roll no.	name	voter ID	age
1	A	K0123	20
2	B	M034	21
3	A	K186	23
4	C	D286	21

CK : { Roll no., voter ID }

FD :
- Roll no. → name ①
- Roll no. → voter ID ②
- voter ID → age ③
- voter ID → Roll no. ④

① ⇒ Roll no. ∈ CK ✓
② ⇒ Roll no. ∈ CK ✓
③ ⇒ voter ID ∈ CK ✓
④ ⇒ voter ID ∈ CK ✓

∴ The table is in BCNF.

1-C

d) Foreign Key:- It is a field in one table, that refers to the primary key of another table.

Importance of Foreign key:-

↳ Referential integrity:-

↳ An advantage of the FK is that the database constantly maintains referential integrity. ~~This means the database~~

↳ Easier Detection work

↳ one of the important roles of the FK is how much easier it is for DBA's and database developers to determine how the database is designed.

↳ Better Performence:-

↳ Another obvious advantage for using FK is improved Performence. By including information on how tables are joined, SQL server can easily determine how it is going to retrieve data when using a join.

Referential Integrity:-

Referential Integrity refers to the accuracy and consistency of data within a relationship.

In relationship, data is linked between two or more tables. This is achived by having the foreign key reference a primary key of the other table.

So, referential integrity requires that, whenever a FK value is used it must reference a valid primary key.

1-e b) When does transaction enters the "partially committed" state?

After Completation of all the read and write operation the changes are made in main memory and the transaction enters in "partially Commited" state.

Is it possible for a "partially commited" transaction to be aborted?

Yes ⇒

Why?

After "partially Commited" state the transaction will go to either "commited state" or "Failed state".

In case of failure it will go to "failed state". After having any type of failure the transaction goes from "failed state" to "aborted state".

Hence it is possible for a partially commited transaction to be aborted.

2) a) Data independence :-

Data independence refers the characteristic of being able to modify the scheme at one level of the database system without altering the scheme at the next higher level.

Types of data independence :-

There are two types of data independence —

↳ Logical data independence :-

↳ Logical data independence refers characteristic of being able to change the conceptual schema without having to change the external schemas.

↳ It is used to separate the external level from the conceptual view

↳ It occurs at the user interface level

↳ Physical data interface :-

↳ It can be defined as the capacity to change the internal scheme without having to change the conceptual schemas.

↳ It is used to separate conceptual level from internal level

↳ It occurs at the logical interface level.

b) Activities of Database Administrator:

↳ Decides hardware:
They decides economical hardware, based upon cost, performance and efficiency of hardware, and best suits organisations. It is hardware which is interface between end users and database.

↳ Manage data integrity and security:-
Data integrity need to be checked and managed accurately as it protects and restricts data from unauthorized use.

↳ Database design:-
DBA is held responsible and accountable for logical, physical design, external model design and integrity and security control.

↳ Query processing performance:-
DBA enhances query processing by improving their speed, performance and accuracy.

↳ Database implementation:-
DBA implements DBMS and checks database loading at time of its implementation.

↳ Tuning Database performance:-
If user is not able to get data speedily and accurately then it may loss organization business. So by tuning SQL commands DBA can enhance performance of database.

7/a) Serializability :-

Scheduling in DBMS is majorly classified into serial and non-serial schedules. As a part of transaction management, it is important to verify if a non-serial schedule is serializable especially when the transactions have concurrent execution. A given non-serial schedule of 'n' transactions is said to be serializable if there exists some kind of equivalent serial schedule to the same 'n' transactions.

A schedule can be checked for serializability in one of 3 methods mentioned below -

(i) Result Equivalent schedule :-

↳ Two schedules, S_1, S_2 are said to be result equivalent if they produce the same output obtained when the schedules are serially executed.

↳ often, this kind of schedule is given the least significance since the result derived are mainly focused on output which in some cases may vary for the same set of inputs.

(ii) Conflict Equivalent Schedule :-

When either of a conflict operation such as Read-Write or Write-Read or write-write is implemented on the same data item within different transactions at the same time, then the schedule holding such transaction is called conflict schedule.

(iii) View Equivalent Schedule :-

Two schedules (one being serial schedule and another being non-serial schedule) are said to be view serializable if the rules of being view equivalent to one another.

the rules are -

① Initial values of data items involved within a schedule must be same.

② Final values of data item involved within a schedule must be same.

③ The number of WR operations performed must be equivalent for the schedules involved.

c) Dense index vs Sparse index:-

↳ Dense index

↳ An index record appears for every search key value in file

↳ This record contains search key value and a pointer to the actual record.

Sparse index

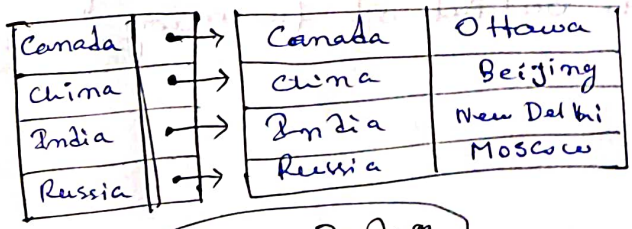
↳ Index records are created only for some of the records

↳ To locate a record, we find the index record with the largest search key value less than or equal to the search key value we are looking for.

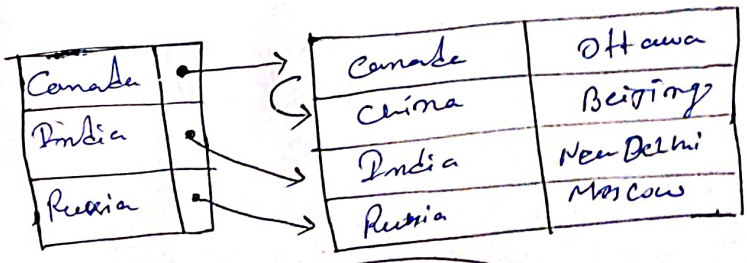
↳ We start at the record pointed by the index record and proceed along the pointers in the file until we find the desired record.

↳ Dense indices are faster in general, but sparse indices require less space and impose less maintenence for insertions and deletions.

↳



Dense Index



Sparse Index

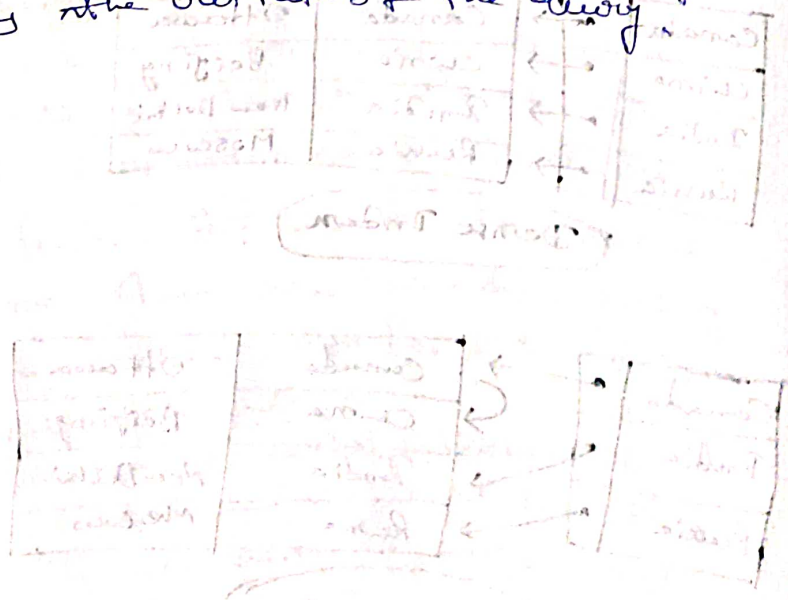
d) Query optimization :-

↳ The cost of the query evaluation can vary for different types of queries. Although the system is responsible for constructing the evaluation plan, the user does not need to write their query efficiently.

↳ Usually, a database system generates an efficient query evaluation plan, which minimizes its cost. This type of task performed by the database system and is known as query optimization.

↳ For optimizing a query, the query optimizer should have an estimated cost analysis of each operation. It is because the overall operation cost depends on the memory allocations to several operations, execution costs and so on.

↳ Finally, after selecting an evaluation plan, the system evaluates the query and produces the output of the query.



c) Dynamic Hashing :-

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

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

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

↳ Searching a key :-

↳ First, calculate the hash address of the key

↳ Check how many bits are used in the ~~dict~~ directory, and these bits are called as 'i'.

↳ Take the least significant 'i' bits of hash address. This gives an index of directory.

↳ Now using the index, go to the directory and find bucket address where the record might be.

↳ Insertion of a new record :-

↳ Firstly, we have to follow the same procedure for retrieval, ending up in some bucket.

↳ If there is still space in that bucket, then place the record in it.

↳ If bucket is full, then we will split the bucket and redistribute the records.

~~Advantages~~

f) Shadow Paging :-

Shadow paging is recovery technique that is used to recover database. In this recovery technique, database is considered as made up of fixed size of logical units of storage which are referred as pages. Pages are mapped into physical blocks of storage, with help of the page table which allows one entry for each logical page of database. This method uses two page tables named current page table and shadow page table.

The entries which are present in current page table are used to point to most recent database pages on disk. Another table i.e., shadow page table is used when the transaction starts which is copying current page table. After this, shadow page table gets saved on disk and current page table is going to be used for transaction. Entries present in current page table may be changed during execution but in shadow page table it never get changed. After transaction both table become identical.

To commit transaction following steps should be done —

(i) All the modifications which are done by transaction which are present in buffers are transferred to physical database.

(ii) Output current page table to disk.

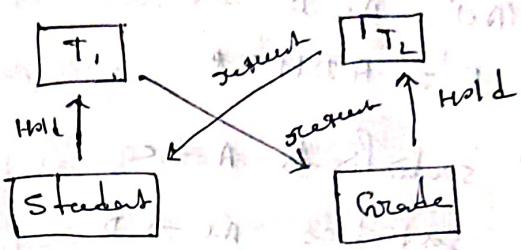
(iib) Disk address of current page table to fixed location which is stable storage containing address of the shadow page table. This operation overwrites address of old shadow page table. With this current page table becomes same as shadow page table and transaction is committed.

9) Deadlock:-

A Deadlock is condition where two or more transactions are waiting for one another to give up locks. Deadlock is said to be one of the most feared complications in DBMS as no task ever gets finished and is in waiting state forever.

Example

In a student table, transaction T_1 holds a lock on some rows and need to update some rows of grade table. Simultaneously, transaction T_2 holds locks on some rows in the grade table and need to update some rows of student table held by transaction T_1.



Now, the main problem arises. Now transaction T_1 is waiting for transaction T_2 to release its lock and similarly, transaction T_2 is waiting for transaction T_1 to release its lock. All activities come to a halt state and remain at a standstill. It will remain in a standstill until the DBMS detects the deadlock and aborts one of the transactions.

Deadlock Avoidance:-

↳ When a database is stuck in a deadlock state, then it is better to avoid the database rather than aborting or restarting the database.

↳ Deadlock avoidance mechanism is used to detect any deadlock situation in advance. A method like "wait for graph" is used for detecting the ~~deadlock~~ deadlock situation but this method is suitable only for the smaller database. For larger database deadlock prevention method is used.

5) a) Check whether following FDs are equivalent or not.

$F = \{ A \to C,\ AC \to D,\ E \to AD,\ E \to H \}$

$G = \{ A \to CD,\ E \to AH \}$

2) Checking for $F \subseteq G$,

(A) $A^+ = AC\ D$

$(AC)^+ = AC\ D$

$E^+ = ADHC$

$\therefore$ A^+ satisfies $A \to C$

$(AC)^+$ satisfies $AC \to D$

E^+ satisfies $E \to AD$ and $E \to H$

$\therefore F \subseteq G$ ——— (1)

Checking $G \subseteq F$

$A^+ = AC\ D$

$E^+ = ADHC$

$\therefore$ A^+ satisfies $A \to CD$,

E^+ satisfies $E \to AH$

$\therefore G \subseteq F$ ——— (2)

From (1) & (2), we can say that,

$F = G$

$\therefore$ F & G are equivalent.

b) Relation ⇒ Book (Book-title , Authorname , Book-type, Listprice , Author-affil , Publisher)

FDs ⇒

Book-title → Publisher , BookType —— (1)

Book-type → List price —— (2)

Author-name → Author-affil —— (3)

(i) Q. what normal form is the relation in?

⇒ Book-title$^+$ = Book-title Publisher BookType List price

Author-name$^+$ = Author-name Author-affil

∴ Candidate key = { Book-title Author-name }

∴ Prime attributes = { Book-title , Author-name }

∴ non-Prime attributes = { Book-type, List price , Publisher, Author-affil }

Checking for BCNF

~~(1) Book-title → Publisher , Author-name, Book-type~~

~~(2) Book-type~~

In (1), L.H.S. ⇒ 'Book-title' is not a candidate key.

∴ (1) does not satisfy the conditions of BCNF

∴ Relation is not in BCNF.

Checking for 3NF

In (1), LHS ⇒ 'Book-title' is not candidate-key

RHS ⇒ None of them are prime attribute

∴ (1) does not satisfy condition of 3NF

∴ Relation is not in 3NF

Checking for 2NF

In (1), LHS ⇒ is a proper subset of Candidate key

RHS ⇒ all of them are non-prime.

∴ (1) does not satisfy condition of 2NF

∴ Relation is not in 2NF

∴ The Relation is in 1NF.