Explain system catalog and Discuss in detail the information stored in system catalog.

A relational DBMS maintain information about every table and index that it contains. the descriptive information is itself stored in a collection of special tables called the catalog tables.

The catalog tables are also called the data dictionary, system catalog or simply the catalog.

Information in system catalog are as follows:-

The system catalog has system wide information, such as the size of the buffer pool and the page size, and following information about individual tables, indexes and views.

* For each table:

- Its table name, file name and the file structure of the file in which it is stored.

- The attribute name and type of each of its attributes.

- The index name of each index on the table.

- The integrity constraints on the table.

* For each Index:

- The index name and the structure of the index.

- The search key attributes.

* For each view:-

→ Its view name and definition.

- In addition, statistics about tables and indexes are stored in the system catalogs and updated periodically.

- The following information is commonly stored:

- Cardinality :→ The number of tuples Ntuple(R) for each table R.

- size :→ The number of pages Npages(R) for table R.

- Index Cardinality :→ The number of distinct key values Nkeys(I) for each index I.

- Index size :→ The number of pages (I) for each Index I.

- Index height :→ The number of non leaf levels for each tree index.

- Index Range :→ The minimum present key values Ilow(I) and maximum Present key value Ihigh(I) for each index I.

⇒ The catalog also contain information about users such as accounting information and authorization information.

Explain Three techniques commonly used in algorithms to evaluate relational operator.

A few simple techniques are used to develop algorithm for each operator.

① Indexing

If a selection or join condition is specified, use an index to examine just the tuples that satisfy the condition.

② Iteration

Examine all tuples in an input table one after the other. If we need only a few fields from each tuple and there is an index whose key contains all these fields, instead of examining data tuples, we can scan all index data entries.

⑧ Partitioning

By Partitioning tuples on a sort key, we can often decompose an operation into a less expensive collection of operations on partitions sorting and hasing are two commonly used Partitioning techniques.

Discuss Access path and selectivity of Access path with reference to the role of Indexing in algorithm development

An access path is a way of retriving tuples from a table and consists of either

① a file scan

② an index plus a matching selection condition.

- Consider a simple selection that is a conjuction of conditions of the form attr op value, where op is one of the comparison operators.

$<, \leq, =, \neq, \geq, or >$.

- such selection are said to be conjuctive normal form (CNF) and each condition is called a conjuct.

⇒ The following example illustrate access Path.

If we have a hash index H on the search key (bid, sid) and we also have B+ tree index on day. the selection condition day 8/9/2002 ∧ bid = 5 ∧ sid = 3 offers us a choice. Both indexes match the selection condition. and we can use either to retrieve Reserves tuples. Whichever index we use the conjucts in the selection condition that are not match by the index must be check for each retrieved tuples.

Explain Query optimization with Diagram.

Query optimization is one of the most important tasks of a relational DBMS. One of the strength of relational Query language is the wide variety of ways in which a user can express and thus the system can evaluate a query.

- Although this flexibility makes it easy to write queries, good performance relies greatly on the quality of the query optimizer.

- Queries are passed and then presented to a query optimizer which is responsible for identifying an efficient execution plan.

- The optimizer generates alternative plans and choose the plan with the least estimated cost.

- Estimating the cost of each enumerated plan and choosing the plan with the lowest estimated cost.

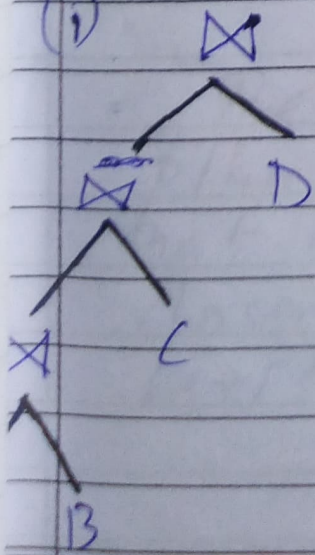
Discuss Evaluation plans with example and how the alternative plans considered.

A query evaluation plan consists of an extended relational algebra tree, with additional annotations at each node, indicating the access methods to use for each table and the implementation method to use for each relational operator

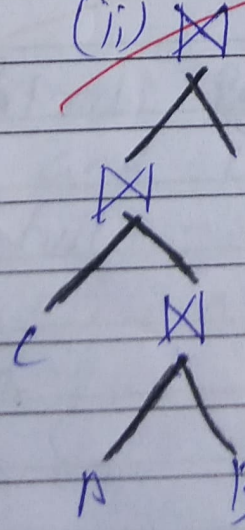
- Consider the following SQL query.

```
SELECT  S.sname
FROM    Reserves R, Sailors S.
WHERE   R.sid = S.sid AND
        R.bid = 100 AND
        S.Rating > 5.
```

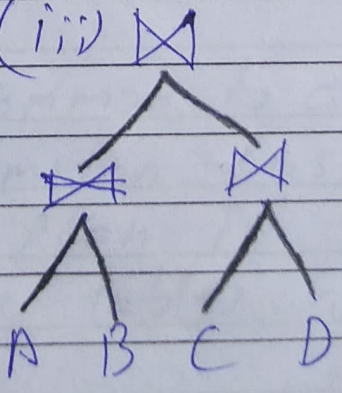
(i)



(ii)



(iii)



Explain the cost estimation of a plan.

The cost of a plan is the sum of costs for the operations it contains. The cost of individual operators in the plan is estimated using information obtained from the system catalog about properties of their input tables.

- If we focus on the metric of I/O costs the cost of a plan can be broken down into three parts.

(1) reading the input tables.

(2) writing intermediate tables.

(3) costing the final result.

- The third part is common to all plans and in the common case that a fully-pipelined plan is chosen, no intermediate tables are written.

- The cost for fully-pipelined plan is dominated by part.

(i) This cost depend, greatly on the access paths used to read input tables: of course, access paths that are used repeatedly to retrive matching tuples in a join algorithm are especially important.

- This query can be expressed in relational algebra as follows:

$$\pi_{sname}(\sigma_{bid=100 \wedge rating>5}(Reserves \bowtie_{sid=sid} Sailor))$$

⇒ Alternative plans considered:

* Left-Deep plans.

- Consider a query of the form $A \bowtie B \bowtie C \bowtie D$ that is, the natural join of four tables.

- Three relational algebra operator trees that are equivalent to this query shown in Figure.

- By convention, the left child of a join node is the outer table and the right child is the inner table.

- The number of tuples in the result of a projection is the same as the input, assuming that duplicates are not eliminated.

- The result size for a join can be estimated by multiplying the maximum result size, which is the product of the input table sizes by the reduction factor of the join condition.

- Thus, the number of tuples that have the same value in column A as a give value in column in R1 key(R).

How System catalogs are stored?

An elegant aspect of a relation DBMS is that is that the system catalog is itself a collection of tables. For example, we might store information about the attributes of tables in a catalog table called Attribute_cat:

Attribute_Cat (attr_name : string,
rel_name : String,
type : string, position : integer)

Suppose that the database contains the two tables that we introduced at the begining of this chapter.

Sailors (sid : integer, sname : string,
rating : integer, age : real)

Reserves (sid : integer, bid : integer,
day : dates, name : string)

attrname	rel	type	
attr_name	Attribute_Cat	string	1
rel_name	"	"	2
type	"	"	3
position	"	integer	4
sid	Sailors	"	1
sname	"	string	2
rating	"	integer	3
age	"	real	4
sid	Reserves	integer	1
bid	"	"	2
day	"	dates	3
rname	"	string	4

The fact that the system catalog is also a collection of tables is very useful. For example, catalog tables can be queried just like any other table, using the query language of the DBMS! Further, all the techniques available for implementing and managing tables apply directly to catalog tables. The choice of catalog tables and their schemas is

not unique and is made by the implementor of the DBMS. Real systems vary in their catalog schema design, but the catalog is always implemented as a collection of tables, and it essentially describes all the data stored in the database.

1. Difference of Dense Index vs Spare Index.

Dense Index	Spare Index
- An Index records appears for every search key value in File.	- An index record are created only for some of the records.
- Indicies are Faster	- Indicies are slower
- Indicies require more space	- Indicies require less space.
- Indicies impose more maintainance for insertion and deletions	- Indicies impose less maintaince for insertion and deletion.

e.g.?

Discuss Iteration and partitioning technique with reference to σ π ⋈.

Iteration :

Examine all tuples in an input table, one after the other. If we need only a few fields from each tuple and there is an index whose key contains all these fields, instead of examining data tuples, we can scan all index data entries.

Partitioning :

By partitioning tuples on a sort key, we can often decompose an operation into a less expensive collection of operations on partitions. Sorting and hashing are two commonly used partitioning techniques.

Reference with Selection

Consider a selection of the form rname < 'C%' on the Reserves table.

Assuming that names are uniformly distributed with respect to the initial letter, for simplicity, we estimate that roughly 1/26 of Reserves tuples are in the Result. This is a total of 10000 tuples, or 100 pages. If we have a clustered B+ tree index on the sname field of Reserves, we can retrieve the qualifying tuples with 100 I/Os. However, if the index is unclustered, we could have up to 10000 I/Os in the worst case. Since each tuple could cause us to read a page.

What is pipelined Evaluation? What its benefits?

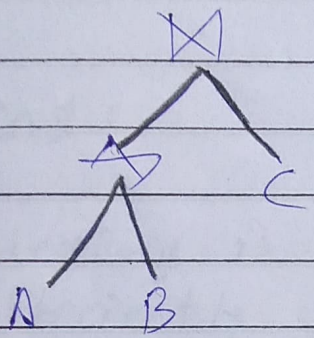
→ When a query is composed of several operators, the result of one operator is sometimes pipelines to another operator without creating temporary table to hold the intermediate result.

→ Pipelined the output of operator into the reset operator saves the cost of sorting out the intermediate result and reading it back in.

→ If the output of an operator saved in a temporary, we say that the tuple are materialized.

→ pipelined evaluation has overhead costs than materialized and is chosen the algorithm for the operator calculation permit it.

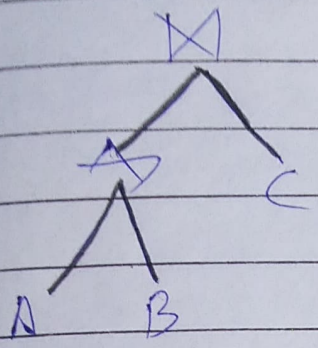
→ Example: Consider a join of the form $(A \bowtie B) \bowtie C$, shown in figure as tree of join operation



Result of type of First join Pipelined into join with C

→ A query tree, illustrating pipelining Both joins can be evaluated as version of a nested loop join.

→ Conceptually the evaluation is initiated from the root, and the node joining A and B produces tuples of as and when they are really by it Parent node, when they root node gets a page of tuples from its left child all matching inner tuples is disclosed.



Result of type of First join Pipelined into join with C

→ A query tree, illustrating pipelining Both joins can be evaluated as a version of a nested loop join.

→ Conceptually the evaluation is initiated from the root, and the node joining A and B produces tuples of as and when they are ready by it Parent node, when they root node gets a page of tuples from its left child all matching inner tuples is disclosed.

Define

i) Indexing :

- Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done.

- Indexing in database system is similar what we see in books.

- Indexing is defined based on its indexing attributes.

- Indexing can be of the following types.

* Primary index

Primary index is defined on an ordered data file. The data file is ordered on a key field. The key field is generally the primary key of the relation

* Secandary Index

Secandary Index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.

* clustering Index

clustering index is defined an ordered data file. The data file is ordered on a non key field.

⇒ Ordered Indexing is of two types:-

(1) Dense Index
(2) Sparse Index

- **Dense Index** -

An index record appears for every search key in the file.

- **Sparse Index** -

An index record appears for only some of the search key value

- In the B^+ tree, the leaf nodes are linked using a link list. Therefore, a B^+ tree can support random access as well as sequential access.

④ CNF :-

Conjunctive normal form (CNF) is an approach to boolean login that express formulas as conjustions of clauses with an AND or OR.

- Each clause connected by a conjunction, or AND must be either a literal or contain a disjunction or OR. Operator. CNF is usertul for avtomated theorem proving.

- In conjunctive normal form, statemed in Boolean logic are conjuctions of clauses with clauses of disjunctions.

- In other words, a statement is a series of ORs connected by ANDs.

⑥ Materialization

- It is easiest to understand intuitively how to evaluate an expression by looking at a pictorial ~~representation~~ representation of the expression in an operator tree.

- If we apply the materialization approach, we start from the lowest level operations in the expression

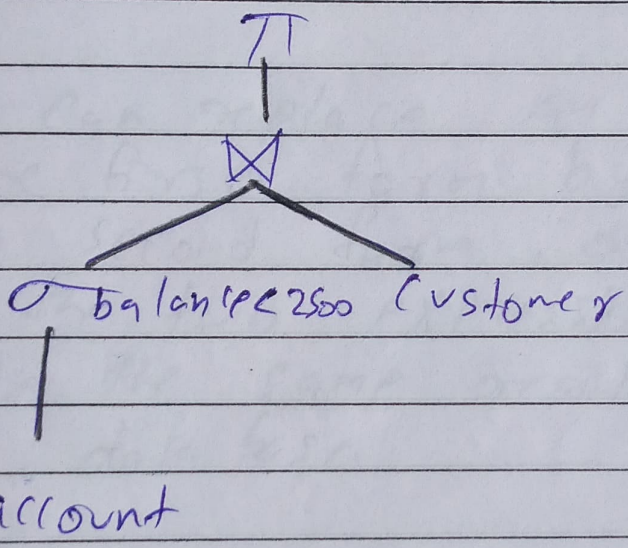
- There is only one such operation: the selection operation on account.

- The inputs to the lowest level operations are relations in the database. We stores the result in the temporary results.

- In our example, the inputs to the join are customer relation and the temporary relation created by the selection on account.

- The join can now be evaluated, creating another temporary relation.

- By repeating the process, we will eventually evaluate the operation at the root of the tree, giving the final result of the expression.



- In our Example, we get the final result by executing the projection operation at the root of the tree, using as input the temporary relation created by the join.

- Evaluation as just described is called materialized evaluation, since the result of each intermediate

operation are created and then are used for evaluation of next level operation.

② Equivalance Rule

- An Equivalance rule says that expressions of two forms are equivalent.

- We can replace an expression of the first form by an expression of the second form, or vice versa. Since the two expressions would generate the same result on any valid data base.

- We now list the number of general equivalence rules on relational algebra expression.

①. Conjuctive selection operations can be deconstructed into a sequence of individual selections. This information is referred to as a cascade of σ.

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

(2) selection operations are commutative.

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

(3) Only the final operations in a sequence of projection operations are needed the others can be omitted. This transaction can also be referred to as a cascade Π

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{L_n}(E))\ldots)) = \Pi_{L_1}(E)$$

(4) selection can be combined with Cartesian products and theta joins.

a. $\sigma_\theta(E_1 \times E_2) = E_1 \bowtie_\theta E_2$

This Expression is just the definition of the theta join.

b. $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$

(5) Theta Join operations are commutative

$$E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$$

(6) Natural join operations are associative

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

Theta joins are associative in the following manner:

$$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$$

(7) The selection operation distributes over the theta-join operation under the following two conditions:

(a) It distributes when all the attributes in selection condition θ_0 involve only the attributes of one of the expressions being joined.

$$\sigma_{\theta_0}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_0}(E_1)) \bowtie_\theta E_2$$

(b) It distributes when selection condition θ_1 involves only the attributes of E_1 and θ_2 involves only the attributes of E_2

$$\sigma_{\theta_1 \cap \theta_2}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_\theta (\sigma_{\theta_2}(E_2))$$

(3) The Projection operation distributes over the theta join operation under the following condition.

(a) $\Pi_{L_1, \theta L_2}(E_1 \bowtie E_2) = (\Pi_{\theta L_1}(E_1)) \bowtie_\theta (\Pi_{L_2}(E_2))$

(b) $\Pi_{L_1, \theta L_4}(E_1 \bowtie_\theta E_2) = \Pi_{L_1, \theta L_3}((\Pi_{L_1, \theta L_3}(E_1)) \bowtie_\theta (\Pi_{L_2 \theta L_4}(E_2)))$

(4) The set operations union and intersection are commutative.

$$E_1 \cup E_2 = E_2 \cup E_1$$
$$E_1 \cap E_2 = E_2 \cap E_1$$

Set difference is not commutative.

(10) Set union and intersection are associative.

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$

$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

(11) The selection operation distributes over the union intersection and set difference operations.

$$\sigma_p (E_1 - E_2) = \sigma_p(E_1) - \sigma_p (E_2)$$

- similarly the preceding equivalence with replaced with either '$\cup$' or '$\cap$', also holds further,

$$\sigma_p (E_1 - E_2) = \sigma_p (E_1) - E_2$$

The proceding equivalence, with replaced by $\cap$, also holds, but does not hold if - is replaced by '$\cup$'.

(12) The projection operation distributes over the union operation

$$\Pi_L (E_1 \cup E_2) = (\Pi_L (E_1)) \cup (\Pi_L (E_2))$$

Q14 Discuss selection operation/algorithm for calculating I/o cost in terms of no. of Pages Per I/o.

ns

If One or more indexes on R match the selection, we can use the index to retrieve matching tuples, and apply any remaining selection condition to further restrict the result set.

As an example, consider a selection of the form rname < 'c%' on the Reserves table. Assuming that names are uniformly distributed with respect to the initial letter, for simplicity, we estimate that roughly 10% of Reserves tuples are the result. This is total of 10000 tuples, or 100 Pages. If we have a clustered B+ tree index on the rname field of Reserves, we can retrieve the qualifying tuples with 100 I/os. however if the index is unclustered, we could have up to 10000 I/O's in the worst case. Since each tuple could cause us to

read a page.

As a rule of thumb, it is probably cheaper to simply scan the entire table if over 5% of the tuples are to be retrieved.

Discuss algorithm for computing the join of relations.

Joins are expensive operations and very common. Therefore, they have been widely studied and systems typically support several algorithms to carry out joins.

Consider the join of Reserves and Sailors, with the join condition Reserves.sid = Sailors.sid. Suppose that one of the tables, say Sailors, has an index on the sid column. We can scan Reserves and, for each tuple, use the index to go to the Sailors

for matching tuples. This approach is called index nested loops Join.

suppose that we have a hash-based index using Alternative on the sid attribute of sailors and that it takes about 1.2 I/Os on average to retrieve the appropriate page of the index. Since sid is a key for sailors, we have at most one matching tuple, Indeed, sid in Reserves is a foreign key referring to sailors, and therefore we have exactly one matching sailors tuple for each Reserves tuple; let us consider the cost of scanning Reserves and using the index to retrieve the matching Sailors tuple for each Reserves tuple, The cost of scanning Reserves is 1000. There are 100*1000 tuples in Reserves. For each of these tuples retrieving the index page containing the rid of the matching sailors tuple costs 1.2 I/Os; in addition, we have to retrieve the sailors page containing the qualifying tuple, Therefore we have 100000*(1+1.2) I/Os to

retrieve matching sailors tuples.
The total cost is 221000 I/Os.

If we do not have an index that matches the join condition on either table, we cannot use index nested loops. In this case, we can sort both tables on the join column and then scan them to find matches. This is called sort-merge join.

Q16 Discuss the step in Query processing with diagram?

Ans Query processing refers to the range of activities involved in extracting data from a database. The activities include translation of queries in high-level database language into expressions that can be used at the physical level of the file system, a variety of query optimizing transformation and actual evaluation of queries.

- The steps involved in processing a query appear in the figure.

- The basic step are:

(1) Parsing and translation

(2) Optimization

(3) Evaluation

- Before query processing can begin, the system must translate the ~~query~~ query into a usable form.

- Once the query plan is chosen, the query is evaluated with that plan, and the result of the query is output.

Q17 Name the two scan algorithm to implement selection operation.

① A1 (Linear search)

- In a linear search, the system scans each file block and tests all records to see whether they satisfy the selection condition.

- An initial seek is required to access the first block of the file. In case blocks of the file are not stored contiguously, extra seeks may be required, but we ignore this effect for simplicity.

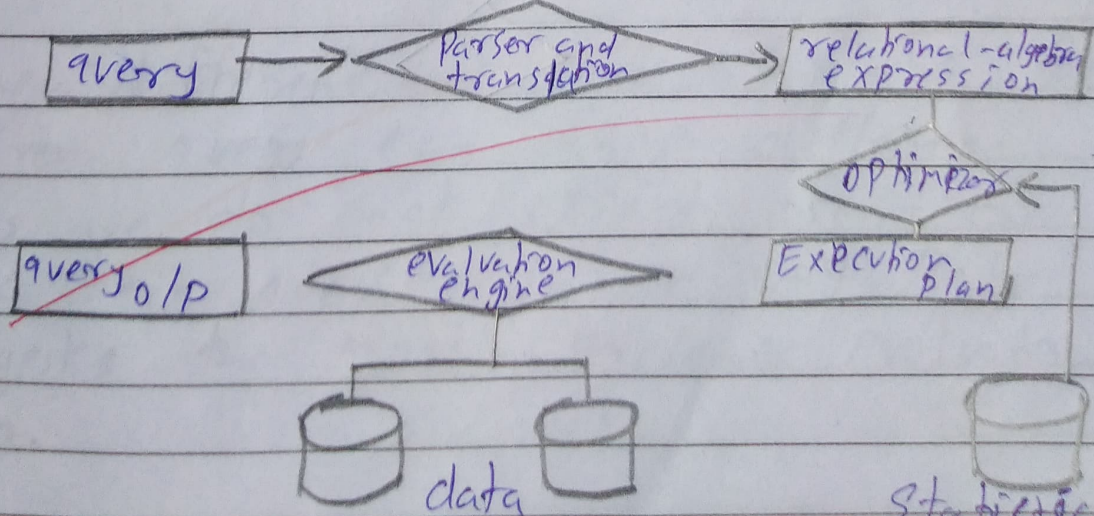
- The cost of linear search, in terms of number of disk operations, is one seek plus denotes the number of blocks in the file, or equivalently

- Thus the first action the system must take in query processing is to translate a given query into it's internal form.

- This translation process is similar to work performed by the parser of a compiler

- In generating the internal form of the query the parser checks of user's query verifies that the relation names appearing in the query are names of the relations in the database and so on.

- The system constructs a parse-tree representation of the query, which it then translate into a relational algebra expression.



- To specify fully how to evaluate a query, we need not only to provide the relational algebra expression, but also to annotate it with instructions specifying how to evaluate each operation.

- Annotations may state the algorithm to be used for a specific operation or the particular index or indices to use.

- A relational algebra operation annotated with instruction on how to evaluate is called an evaluation primitive.

- A sequence of primitive operation that can be used to evaluate a query is a query execution plan or query-evaluation plan.

- The different evaluation plans for a given query can have different costs. We do not expect users to write their queries in a way that suggests the most efficient evaluation plan.

the time cost is $t_S + b_r * t_T$

- For a selection on a key attribute have an average transfer cost of $b_r/2$, but still have a worst case cost of b_r block transfer in addition to one seek.

- Although it may be slower than other algorithm for implementing selection, the linear search algorithm can be applied to any file, regardless of the ordering of the file, or the availability of indices, or the nature of the selection operation.

→ A2 (binary search):

- If the file is ordered on an attribute, and the selection condition is equality comparison on the attribute we can use a binary search to locate records that satisfy the selection. The system perform the binary search on the blocks of the file.

- In the worst case; the number of blocks that need to be examined to find a block containing the required records is $\lceil \log_2(b_r) \rceil$ where b_r denotes the number of blocks in the file.

- Each of these block accesses require a disk seek in addition to a block transfer and the time cost is thus $\lceil \log_2(b_r) \rceil * (t_r + t_s)$

- If the selection is on a nonkey attributes, more than one block may contain required records, and the cost of reading the extra blocks has to be added to the cost estimate we can estimate this number by estimating the size of the selection result and dividing it by the average number of records that are stored contigously, so we only pay a transfer cost t_T per extra block.

Q18 Define Transaction what is a transaction state and what are the different states a transaction could be in? Explain with diagram.

Ans Transaction :-

- Collections of operations that form a single logical unit of work are called transaction

⇒ Transaction state :-

- Transaction State in DBMS are the states through which a transaction goes throughout its lifetime.

⇒ Different states of transaction :-

- Active :-

In this state, the transaction is being executed. This is the Initial state of every transaction.

- Partially committed :-

When a transaction executes its final operation, it is said to be in a partially committed state.

- Failed :-

A transaction is said to be in a failed state if any of the checks made by the database recovery system fails.

- A fail transaction can no longer proceed further.

- Aborted :-

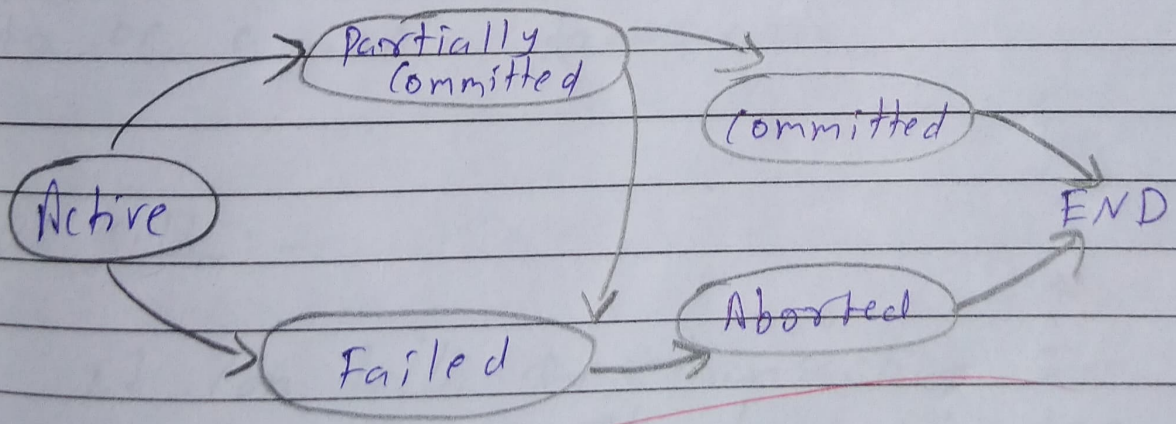
If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operation on the database back to its original state when it was prior to the execution of the transaction. Transaction in this state are called

aborted. The database recovery module can select one of the two operations after a transaction aborts.

- Re start the transaction
- kill the transaction.

- Committed :-

If a transaction executes all its operation successfully it is said to be committed. All its effects are now permanently established on the database system.



Q19 What are two options a transaction has after entering abort?

Ans

When the transaction enters aborted state the system has two options.

① Restart :-

It can restart the transaction, but only if the transaction was aborted as a result of some hardware or software error that was not created through the internal logic of the transaction. A restored transaction is considered to be a new transaction.

② Kill :-

It can kill the transaction. It usually does so because of some internal logical error that can be corrected only by rewriting the application program or because the input was bad, or

because the desired data was not found in the database.

Discuss is anomalies due to interleaved transactions.

(i) Reading Uncommitted Data (WR Conflicts)

- The first source of anomalies is that a transaction T2 could read a database object A that has been modified by another transaction T1, which has not yet committed. Such a read is called a dirty read. A simple example illustrates how such a schedule could lead to an inconsistent database state. Consider two transaction T1 and T2 each of which, run alone, preserves database consistency:

T1 transfers $100 from A to B and T2 increments both A and B by 6%.

Suppose that the actions are interleaved so that

(1) The account transfer program T1 deducts $100 from account A, then

(2) The interest deposit Program T2 reads the current values of accounts A and B and adds 6% interest to each, and then

(3) The account transfer program credits $100 to account B.

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
	R(B)
	W(B)
	Commit
R(B)	
W(B)	
Commit	

=> Unrepeatable Reads (RW conflicts)

- The second way in which anomalous behavior could result is that a transaction T2 could change the value of an object A that has been read by a transaction T1, while T1 is still in progress.

If T1 tries to read the value of A again, it will get a different result, even though it has not modified A in the meantime. This situation could not arise in a serial execution of two transactions; it is called an unrepeatable read.

This situation can never arise in a serial execution of T1 and T2; the second transaction would read A and see 0 and therefore not proceed with the order.

Overwriting Uncommitted Data
(ww conflicts).

The third source of anomalous behavior is that a transaction T2 could overwrite the value of an object A, which has already been modified by a transaction T1, while T1 is still in progress. Even if T2 does not read the value of A written by T1, a potential problem exists as the following example illustrates.

Suppose that Harry and Larry are two employees, and their salaries must be kept equal. Transaction T1 sets their salaries to $2000 and transaction T2 sets their salaries to $1000. If we execute these in the serial order T1 followed by T2, both receive the salary $1000. the serial order T2 followed by T1 gives each the salary $2000. Either of these is acceptable from a consistency standpoint. Note that

neither transaction reads a salary value before writing it such a write is called a blind write, for obvious reasons.

The problem is that we have a lost update. The first transaction to commit, T2 overwrote Larry's salary as set by T1. In the serial order T2 followed by T1, Larry's salary should reflect T1's update rather than T2's but T1's update is lost.

2) Explain schedules involving aborted transactions & Recoverable schedules.

Ans A Serializable schedule over a set S of transactions is a schedule whose effect on any consistent database instance is guaranteed to be identical to that of some complete Serial schedule over the set of Committed transaction in S.

- The definition of serializability relies on the actions of aborted

transactions being undone completely, which may be impossible in some situations.

- Suppose that,

(1) An account transfer program T1 deducts $100 from account A, then

(2) An interest deposit program T2 reads the current values of accounts A and B and add 6% interest to each, then commits and then

(3) T1 is aborted.

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
	R(B)
	W(B)
	Commit
Abort	

- Now, T_2 has read a value for A that should never have been there. If T_2 had not yet committed, we could deal with the situation by cascading the abort of T_1 and also aborting T_2, this process recursively aborts any transaction that read data written by T_2 and so on. But T_2 has already commited and so we cannot undo its actions we say that such a schedule is unrecoverable.

- In a recoverable schedule, transaction commit only after all transaction whose changes they read commit.

- If transaction read only the changes of commited transaction, not only is the schedule recoverable but also aborting a transaction can be accomplished without cascading the abort to other transaction such a schedule is said to avoid cascading abort.

Q22 Compare Index Sequential File and B+ tree Index files.

ns

Index Sequential File	B+ tree Index files
- Performance degrads as file grows since many overflow blocks get created.	- Automatically reorganised itself with small, local changes in the face of insertion and deletions.
- Periodic Reorganisation of file is required	- Reorganization of file is not required to maintain performance
- Less insertion and deletion overhead, space overhead.	- Extra insertion & deletion overhead, space overhead.

Explain cascading aborts

Even if a schedule is recoverable to recover correctly from the failure of a ~~transaction~~ transaction T_i, we may have to roll back several transactions. Such transaction occur when transaction have read data written by T_i

T_{10}	T_{11}	
read (A)		
read (B)		
write (A)		
	read (A)	
	write (A)	
		read (A)

- Transaction T_{10} ~~writes values~~ of A that is read by transaction T_{11}. transaction T_{11} writes value of A which is again Read by T_{10}.

- suppose at this point T_{10} fails T_{10} must roll back. since T_{11} is dependent on T_{10}. T_{11} must be roll back. since T_{12} is dependent on T_{11}, T_{12} must roll back

- This phenomen in which a single transaction failure leads to a series of transaction roll back / failure is called cascading roll back / cascading aborted.

- Cascading roll back is undesirable. since leads to the undoing of a significant amount of work. It is desirable to restrict the schedules to those where cascading roll back cannot occur.

12/8/2020