11. Explain the algorithms for implementing Join methods.

There are two algorithms to compute natural join and conditional join of two relations in database: Nested loop Join, and Block nested loop join. To understand these algorithms we will assume there are two relations, relation R and relation S. Relation R has T_R tuples and occupies B_R blocks. Relation S and T_S tuples and occupies B_S blocks. We will also assume relation R is the outer relation and S in the inner relation.

Nested loop Join:-

In the nested loop join algorithm, for each tuple in outer relation, we have to compare it with all the tuples in the inner relation then only the next tuple of outer relation is considered. All pairs of tuples which satisfy the condition are added in the result of the join.

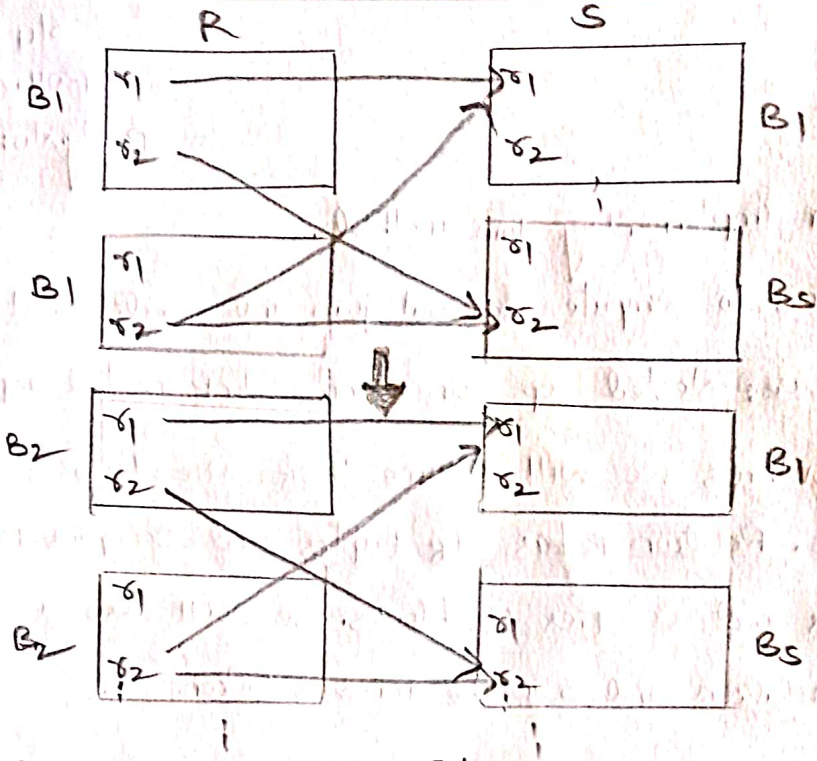
```
for each tuple tR in TR do
    for each tuple ts in Ts do
        Compare (tR, ts) if they satisfies then add them in the result of the join
    end
end
```

This algorithm is called nested join because it consists of nested for loops.

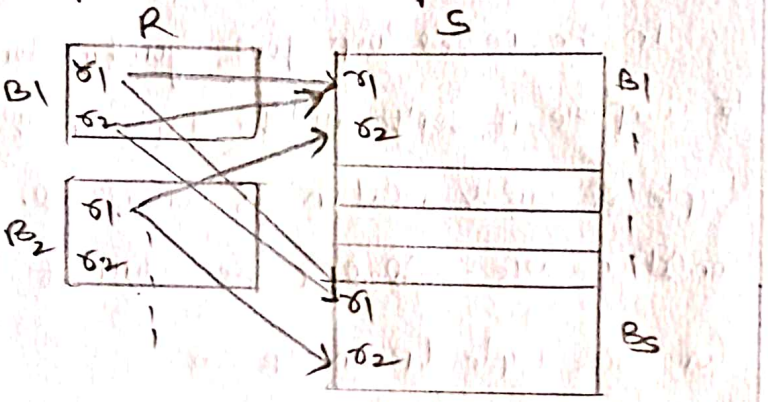
Case-1:-

Assume only two blocks of main memory are available to store blocks from R and S Relation.

For each tuple in relation to R_1 we have to transfer all blocks of relation S and each block of Relation R should be transferred only once. So, the total block transfers needed $= T_R * B_S + B_R$.



Case-2:- Assume one relation fits entirely the main memory and there is atleast space for one extrablock.



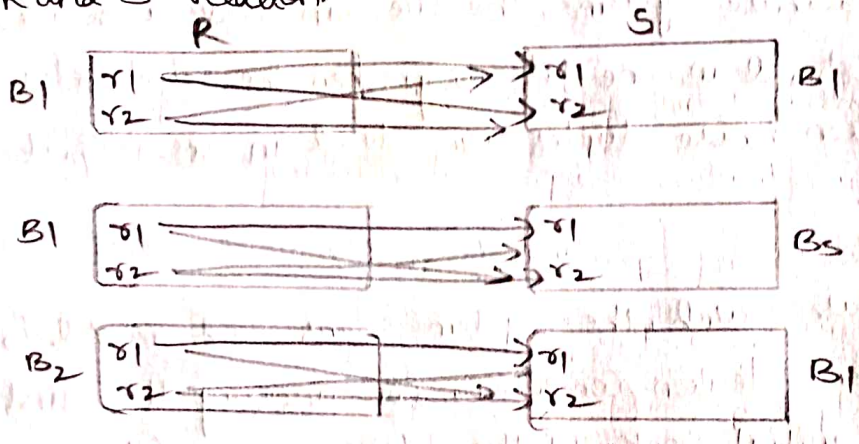
In this case, the blocks of relations (this is, the inner relation) are only transferred once and kept in the main memory and the blocks of relation R are transferred sequentially. So, all the blocks of both the relation are transfers = $B_R + B_S$.

Block Nested loop Join:-

In block nested loop join, for a block of outer relation, all the tuples in that block are compared with all the tuples of the inner relation, then only the neset block of outer relation is considered.

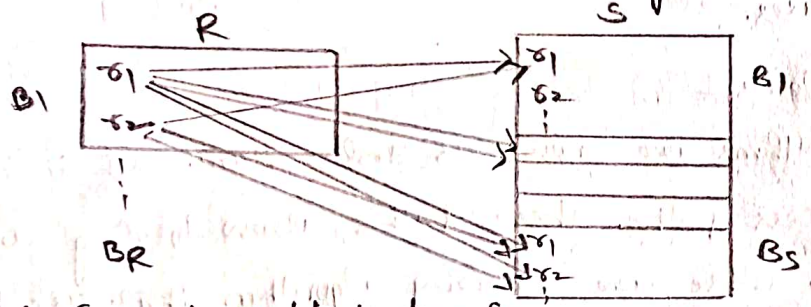
```
for each block bR in BR do
 for each block bs in Bs do
   for each tuple tR in TR do
     for each tuple ts in Ts do
     Compare(tR, ts) if they satisfies the satisfies the Condition
     add them in the result of the join
   end
  end
 end
end
```

Case-1:- Assume only two blocks of main memory are available to store blocks from R and S relation.



for each block of relation R, transfers needed = $B_R + B_R * B_S$.

Case-2:- Assume one relation fits entirely in the main memory.



In this case, total block transfers needed are similar to nested loop join. Block nested loop join algorithm reduces the access cost compared to nested loop join if the main memory space allocated for join is limited.

—o—

2) a) Explain the methods for implementing selecting operations?

Ans:- We understood that estimating the cost of a query plan should be done by measuring the total resource consumption.

In this section, we will understand how the selection operation is performed in the query execution plan.

Generally, the selection operation is performed by the filescan. filescans are the search algorithms that are used for locating and accessing the data. It is the lowest-level operator used in query processing.

Selection using file scans and Indices:-

In RDBMS or relational database systems, the file scan reads a relation only if the whole relation is stored in one file only. when the selection operation is performed on a relation whose tuples are stored in one file, it uses the following algorithms.

- **Linear search:-** In a linear search, the system scans each record to test whether satisfy the given selection condition. for accessing the first block of a file, it needs as initial seek. If the blocks in the file are not stored in contiguous order, then it needs some extra seeks. However, linear search is the slowest algorithm used for searching, but it is applicable in all types of cases.

- **selection operation with Indexes:-**

The index-based search algorithms are known as Index scans. Such index structures are known as access paths. these paths allow locating and accessing the data in the file. There are following algorithm that use the index in query processing

  - **Primary Index:-** we use the index to retrieve a single record that satisfies the quality condition for making the selection. The quality comparision is performed on the key attribute carrying a primary key.

- Primary index, on nonkey: the difference b/w equality on key and nonkey is that in this, we can fetch multiple records. we can fetch multiple records through a primary key when the selection criteria specify the equality comparison on a non key.

- Secondary index, equality on key or nonkey:- The selection that specifies an equality condition can use the secondary index. using secondary index strategy, we can either retrieve a single record when equality condition is nonkey.

• Selection operation with Comparisons:

• Primary index, Comparison: When the selection condition given by the user is a comparison, then we use a primary ordered index, such as the primary B^+-tree index.

• Secondary index, Comparison:— The secondary ordered index is used for secondary ordered index is user satisfying the selection operation that involves $<, >, \geq, \leq$. In this, the files scan searches the blocks of the lowest-level index.

Working on more complex selection involves the selection predicates known as conjuction, Disjunction and Negation.

Conjunction: A conjuctive selection is the selection having the form as: $\sigma_{\theta_1 \theta_2 n \cdots n \theta_n}(r)$. A conjunction is the intersection of all records that satisfies the above selection condition.

Disjunction: A disjunction is the union of all records that satisfies the given selection condition θ_1.

Negation: The result of a selection $\sigma_{\theta}(r)$ is the set of tuples of given relation r where the selection condition evaluates to false. But nulls are not present, and this set is only the set of tuples in relation r that are not in $\sigma_{\theta}(r)$.

—o—

2(b) ☆, ⇒ Explain the steps involved in the query processing with a neat diagram.

Ans.

query in a high-level language

scanning, parsing and validating

Immediate form of query

query optimizer

Execution plan

query code generator

Code to execute the query

Runtime database processor

Result of query.

Query processing includes translations on high level queries into low level expressions that can be used at physical level of file system, query optimizati and actual execution of query to get actual result.

Step-1:-

Parser:- During parse call, the database performs the following checks syntax check, semantic check and shared pool check, after converting the query into relational algebra.

1; Syntax check- Concludes SQL syntax validity.

Ex:- SELECT * from Employee.

Here error of wrong spelling of FROM is given by this check.

2; Semantic Check:- Determines whether the statement is meaningful or not.

Ex:- query contains a table name which does not exist.

3; Shared pool check:- Every query process a hash code during its execution. So, this check determines existence of written hash code in shared pool if code exists in shared pool.

Hard parse and Soft parse:- If there is a fresh query and its hash code does not exist in shared pool then that query has to pass through from the additional steps known as hard parsing. Hard parse includes following steps- optimizer and row source generation.

Step 2:-

Optimizer: During optimization stage, database must perform a hard parse atleast for one unique DML statement and perform optimization during its parse. This database never optimizes DDL unless it includes a DML component such as subquery that require optimization.

Row Source Generation: The row source generation is a software that receives optimal execution plan from the optimized and produces an interactive execution plan that is usable by the rest of the database. The iterative plan is the binary program that when executed by the SQL Engine produces the result set.

Step-3:-

Execution Engine: finally runs the query & display the result.

3) Explain the algorithm for implementing
a; External sorting.
b; Outer join.

a; External Sorting:- External sorting refers to sorting algorithms that are suitable for large files of records stored on disk that do not fit entirely in main memory, such as most database files. The Typical External sorting algorithm uses a sortmerge strategy, which starts by sorting algorithm uses small subfiles called runs of the main file and then merges the sorted runs, creating the large sorted subfiles that are merged in turn.

The sort merge algorithm, like other database algorithms, requires buffer space in main memory, where the actual sorting and merging of the runs is performed. The basic algorithm, outlined consists of two phases: the sorting phase and the merging phase the buffer space in main memory is part of DBMS Cache an area in the Computer is main memory is part of DBMS controlled by the DBMS. the buffer space is divided into individual better, where each buffer space is divided and is same size in bytes as the size of one disk block.

b; Outer Join:-

In a DBMS, we follow the principles of Normalization that allows us to minimize the large tables into small tables. By using a select statement in joins we retrieve the big table back.

1. left outer join.
2. Right outer join.
3. Full outer join

Creating a database: Run the following command to create a database

```
create database testdb;
```

using the database:- Runs the following command to use a database

```
use testdb;
```

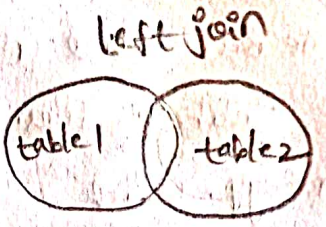
Adding table to the database: Runs the following command to add tables to a database

```
CREATE TABLE students (
  studentID int; lastName varchar(255), firstName varchar(255), Address
  varchar(255), city varchar(225));
```

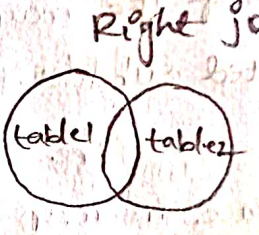
Types of outer join:-

1); Left outer join: The left outer join operation returns all record from left table a matching record from the right table. on a matching element not found in the right table, Null is represented.

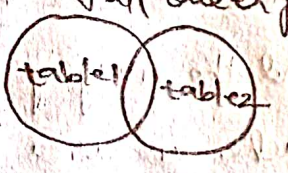
```
SELECT Column_name(s) from table1
LEFT JOIN Table2 on table1. column_
Name = table2. Column_Name;
```



2); Right outer join:- The right join operation returns all the record from right table and matching records from the left table.



3); Full outer join:- The full outer join keyword returns all records when there is a match in left or right table records.



—o—

4) Explain the Notation for query trees and query graph and also Explain the heuristic optimization of query trees?

My:- A query tree is a tree datastructure that corresponds to a relational algebra Expression. It represents the input relations of the query as leaf nodes of the tree and represents the relational algebra operations as internal nodes. An execution of the query tree consists of executing an internal node operation whenever its operands are available and then replacing that internal node by the relation that results from executing of operations start at the leaf nodes, which represents the input database relations for the query, and ends at the root node, which represents the final operation of the query

$$\pi_{Pnumber, Dnum, Lname, Address, Bdate}(((\sigma_{Plocation="Stafford"}(PROJECT))$$

$$DNum = Dnumber(DEPARTMENT))$$

$$mgr\text{-}SSN = SSN(EMPLOYEE))$$

The query tree represents a specific order of operations for executing a query. A more neutral data structure for representing of a query is the query,

The query graph representation does not indicate an order on which operations to perform first. There is only a single graph corresponding to each query. Although some optimization techniques were based on query graphs, it is now generally accepted that query trees are preferable because, in practise the query optimizer needs to show the order of operations for query execution, which is not possible in query graph.

optimization of heuristic in query trees:-

Heuristic optimization transforms the expression-tree by using a set of rules which improve the performance.

Steps in heuristic optimization:

- Deconstruct the conjunctive selections into a sequence of single selection operation
- move the selection operations down the query trees for the earliest possible execution.
- first execute those selections and join operations which will produce smallest relations.
- Replace the cartesian product operation followed by selection operation with join operation.
- Deconstructive and move the tree down as far as possible.
- Identify those subtrees whose operations are pipelined.

—•—

5) Write the algorithms for implementing the union, intersection and set difference?

Ans:- A union-algorithm is an algorithm that performs two useful operations on such as data structure: find: Determine which subset a particular element is in. This can be used for determining if two elements are in the same subset.

union: Join two subsets into a single subset.

Syntax:

```
SELECT Column names FROM table1
UNION
SELECT Column names FROM table2;
```

UNION operator provides only unique values by default.

2. UNION ALL Syntax:

SELECT Column names FROM table1
UNION ALL
SELECT Column names FROM table2;

Example:- select city FROM Greek1 UNION select city FROM Greek2 order by city;

O/P:- city
Delhi
Gurugram
Jaipur
Noida.

Intersection operation: It displays the common values in $R_1 \& R_2$. It is denoted by $\cap$.

Syntax:- $\Pi_{regno}(R1) \cap \Pi_{stdgno}(R2)$

Consider two sets,

$A = \{1, 2, 4, 6\}$ and $B = \{1, 2, 7\}$

Intersection of A and B

$A \cap B = \{1, 2\}$

Elements that are present in both sets A and B are present in the set obtained by intersection of A and B.

In relational algebra if R_1 and R_2 are two instances of relation

$R_1 \cap R_2 = \{ x | x \in R_1 \text{ and } x \in R_2 \}$

this is the intersection of R_1 and R_2.

Example:-

$\Pi_{name}(Depositor) \cap \Pi_{Name}(Borrower)$

Depositor

ID	Name
1	A
2	B
3	C

Borrower

ID	Name
2	B
3	A
5	D

So, the Intersection of deposit and Borrower is as

A
B

Set Difference: set difference operation in relational algebra, purpose of set difference operation, example of set difference relational algebra operation, relational algebra in dbms.

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation $r - s$

finds all the tuples that are present in r but not in s.

$\Pi_{author}(BOOKS) - \Pi_{author}(Articles)$

Rules:-

1', Both the relations R_1 and R_2 (or the result of expression 1 and expression 2) must have the same number of Attributes.

2', The domain of ith attribute of R_1 and ith attribute of R_2 must be same for all i.

example:-

student indoor	regno	sport
3 records	B1T001	chess
	B1T023	Carrom
	BCE020	Badminton

student-outdoor	regno	sport
3 records	B1T001	soccer
	B1T023	soccer
	BME023	soccer

SQL: (Select regno FROM stu-indoor) MINUS (SELECT regno FROM stu-outdoor);

Result:-

$(\Pi$ regno (stu-indoor)) $-(\Pi$ regno (stu-outdoor))

Regno
BCE020