MASTER OF COMPUTER APPLICATIONS

SEMESTER 1

RELATIONAL DATABASE MANAGEMENT SYSTEM

Unit 4

Query Optimisation

Table of Contents

1. INTRODUCTION

You have already studied DBMS and SQL in the previous units. Hence, query optimisation which we are going to cover in this unit will not be new to you as it is related to DBMS.

Query optimisation is a technique that helps the DBMS to reduce the query execution time. Nowadays, every database software supplies optimising SQL compilers that firstly analyses the SQL query, if required then rewrites the query, and finally develops an optimal query to retrieve the data from the database. This module of SQL compiler is called Query Optimiser. This optimisation is based on the different optimisation rules devised on the criteria of cost of each operation on the query.

We start our discussion with the overview of various algorithms for query operations in the context of an RDBMS. It will also cover the discussion on the heuristic in query optimisation. We will further study a brief overview of the semantic query optimisation, multi-query optimisation and application. The latter part of the unit deals with execution strategies for SQL sub queries and query processing for SQL updates.

1.1 Objectives

After studying this unit, you should be able to:

- *describe the algorithms for executing query operations*
- *discuss the heuristics in query optimisation*
- *explain briefly semantic query optimisation*
- *identify multi-query optimisation and application*
- *explain the execution strategies for SQL sub queries*
- *discuss query processing for SQL updates*

2. QUERY EXECUTION ALGORITHM

RDBMS provides various algorithms for implementing the different types of relational operations that appear in a query execution strategy.

Different types of algorithms that are used by many relational operations are like external sorting merge sort and many more to implement operations like SELECT, JOIN, PROJECT (UNION, INTERSECTION, SET DIFFERENCE)

and aggregate operations MIN, MAX, COUNT, AVERAGE and SUM. Let's discuss these in detail.

2.1 External Sorting

Sorting is one of the primary algorithms used in query processing. It is used to reorder data into a new desired sequence. It may be avoided if an index already exists to facilitate ordered access to the records.

Internal sorting uses main memory so it is fast but also expensive while on the other hand, external sorting is slow and cheaper as it uses secondary storage devices. External sorting is appropriate for large files stored on the disk as can't be completely fitted in the main memory. Internal sorting algorithm is suitable for sorting data structures that can fit completely in memory.

An external sorting algorithm makes use of a sort-merge strategy. Sort- merge strategy divides the main file into sub-files of smaller size termed as runs. Now these runs are sorted first and then merged to make larger runs. These larger runs are again sorted in turn. The external sorting needs buffer space in main memory to execute the actual sorting and merging of the runs.

External Sorting algorithm carries out the operation in following two phases: Sorting Phase and Merging Phase.

Phase 1: Sorting phase: In this phase, the runs are read into the main memory. Over there the runs are sorted by using internal sorting algorithm and the result is written back to a

disk as temporary sorted runs. The number of initial runs and the size of a run (n_R) are governed by the number of file blocks (b) and available buffer space (n_B).

For instance,
if n_B = 5 blocks
b = 1024 blocks

Then, $n_R = [(b/n_B)]$ = 205 initial runs each of size 5 blocks.

Hence after the sort phase, 205 sorted-runs are stored as temporary sub- files on disk.

Phase 2: Merging phase: In this phase, merging of the sorted runs is carried out over one or more phases. The number of runs that can be merged together in each pass is termed as the degree of merging (dM). In each pass, a buffer block is required to hold one single block from every runs being merged and one block is required for containing one block of the final result.

Therefore, it can be concluded that

- dM is smaller of (n_B – 1) and n_R, and
- number of passes = [log dM (n_R)].

Let's continue with the above example;
with dM =4 (four way merging) our above calculated 205 initial sorted runs, in the first phase would be merged into 52 runs which are further merged into 13 runs, later on 4 and then finally 1 run. So it means that total 4 passes are required.

The worst case performance comes with minimum value of dM as 2. And the number of block accesses will be [(2*b) + (2*(b*(log dM b))]. Here, the 1st term constitutes the number of block accesses in sort phase since each file block is accessed at two times, first for reading into memory and second for writing the record blocks, after sorting, to disk. The 2nd term symbolises the number of blocks accesses for the merge phase, presuming the worst-case scenario of dM 2.

2.2 Implementing the SELECT Operation

SELECT (represented by symbol σ) operation performs the task of retrieving the desired records from the database.

Now, let's generate a query for selecting all the records of EMPNO 3276 from table EMP (Table 4.1

(Query1): $\sigma_{EMPNO='3276'(EMP)}$

Table 4.1: Instance of EMP Table

ENAME	EMP NO	DNO	SEX	SALARY
Mahesh	1234	1	M	30000
Kartik	3276	4	M	4000
Gaurav	4278	5	M	10000
Jaya	2753	3	F	5000
Kiran	3721	6	F	15000

If you want to retrieve all the records with DNO more than 4 from table DEPT (Table 4.2), then the query will be framed like:

(Query2): $\sigma_{DNO>4(EMP)}$

Table 4.2: Instance of DEPT Table

DNAME	DNO	MGRENO
Research	4	4001
Production	5	5001
Sales	1	1001

If you want to select all the records from table EMP where DNO field is 2, then it can be written as:

(Query3): $\sigma_{DNO=2(EMP)}$

SELECT operation can retrieve data with multiple criteria's as well. For example, to select all the records about the female employees whose SALARY is more than 10000 from department 3 with reference to EMP table; will be framed like:

(Query4): $\sigma_{DNO=3\ AND\ SALARY>10000\ AND\ SEX='F'}(EMP)$

Search Methods for Simple Selection: To select records from the file a possibilities of numerous search algorithms exists. These are termed as file scans as they scan entire file and then retrieve the records satisfying the user defined condition. And if the search algorithm carries an index, then that search is termed as index scan. The basic search algorithms are explained below.

- ***Linear search (brute force):*** It retrieves all the records from the file and checks whether the attribute values qualify the criteria or not. This process is carried out till the end of the file. So we can say above mentioned query 1

  σ EMPNO='3276'$^{(EMP)}$may be executed using the linea search algorithm.
- ***Binary search:*** It does not take care of the ordering done on some field of the file. It checks for the equality on which ever key attribute file is ordered. Binary search is faster and more efficient than the linear search as the search space is reduced to half in each comparison.

So, we can say that if in EMP file the EMPNO is the ordering attribute

then query 1 σEMPNO='3276'$^{(EMP)}$ might have used binary search.

- ***Using primary index (or Hash Key):*** If the query checks for equality between the key attribute and the primary index (or hash key), then hash search might be used. If in query 1, in EMP file, the EMPNO is hash indexed, then hash search may be used on this field. On using the primary index (or hash key) at most a single record is only retrieved.

Depending upon the cost associated with each method, the query optimiser picks up the most appropriate method for executing a SELECT operation. Now let us extend our discussion to JOIN operation.

2.3 Methods to Implement JOIN Operation

JOIN operation is used to join two database tables/relations. It is a time consuming operation. JOIN operation is in the form of $X \bowtie_{M=N} Y$; where M and N are domain compatible attributes of X and Y files respectively.

There are various algorithms for implementing JOIN operations. Some of these are discussed below:

- ***Nested loop join (brute force):*** Retrieve each and every record from (inner loop) Y of each record t in (outer loop) X. After this, check if both records meet the condition; t [M]= s[N].
- ***Single loop join (using an access structure to retrieve the matching records):*** If an hash key or index is there for any one of the two join attributes (let us say N of Y) retrieve each and every record t in X, (one at one time). After that utilise the access structure to directly retrieve all matching records s from S satisfying condition; s [N] = t [M].
- ***Sort-merge join:*** JOIN can be implemented more efficiently if the records from X and Y files are physically sorted by the join attribute value M and N respectively

  $X \bowtie_{M=N} Y$

- ***Hashjoin:*** All the records from both the files R and S are hashed to a single hash file by utilising the same function on the join attributes M of X and N of Y as hash keys.

Firstly, the partitioned phase takes places where the records of the file with less entries/records (say X) hash its entries to the hash file bucket. Here, the records of R are sent into the hash buckets.

Secondly, the probing phase takes place, where each record of another file (say Y) is hashed and added to the bucket. Now the matching records from X are combined in that bucket.

2.4 Project and Set Operations Implementation

By defining the required attributes, PROJECT operation can select subset of attributes from a relation. In the project operation $\Pi <$ attribute list > (R) implementation if < attribute list > carries the key of relation R then the output produces same number of tuples as that of R but with only the attribute values in the <attribute list>.

Now, if the < attribute list> is without key of R, then the output will produce duplicate tuples. These duplicate tuples must be deleted by sorting. After sorting, wherever duplicate tuples appear consecutively, it is eliminated.

Among the Set operations (UNION, INTERSECTION, SET DIFFERENCE and CARTESIAN PRODUCT), the CARTESIAN PRODUCT operation R x S is most costly. It is because of two reasons, firstly its output carries a combination of records from R and S. Secondly, all the attributes of R and S are present in the output.

For example, if we have two relations R and S as shown in the table,

Relation	Records	Attributes
R	i	x
S	j	y

CARTESIAN PRODUCT output of the above table will carry (i*j) records and (x+y) attributes. As it carries many records so it's wise to avoid CARTESIAN PRODUCT operation and go for some different equivalent operation.

Set operations like UNION, INTERSECTION and SET DIFFERENCE can be applicable only to union compatible relations. These relations must have same number of attributes and must be from same attribute domain. Relations where Union operation can be performed must have same attributes and that to from the same domain.

2.5 Aggregate Operations Implementation

Sometimes you need to find out certain statistical values from the table. Aggregate Functions are functions that provide mathematical operations. If you need to add, count or perform basic statistics, then these functions are of great help.

SQL has many built-in functions for performing calculations on data. SQL Aggregate Functions are the functions that return a single value, calculated from values in a column which is selected from the table. Some of the useful aggregate functions are:

AVG() - The AVG() function returns the average value of a numeric column.

COUNT() - Returns the number of rows

MAX() - The MAX() function returns the largest value of the selected column

MIN() - The MIN() function returns the smallest value of the selected column

SUM() - The SUM() function returns the total sum of a numeric column. Let us take one example. The SQL syntax for using MAX() is:

SELECT MAX(column_name) FROM table_name

Table STD.

STD_Id	STUDENTS	SUBJECTS	MARKS
1	Seema	Maths	95
2	Sangeetha	Maths	91
3	Seema	Physics	97
4	Monika	Physics	55
5	Sangeetha	Physics	63
6	Seema	Chemistry	70

As an example, let's consider a SQL query for the table STD:

```
SELECT MAX (MARKS) AS MAXIMMARKS
FROM STD
```

Where STD is the table name and MARKS is the column name of the table STD. The above statement would select the highest (maximum) Marks from the column MARKS of the table STD and result would look like this:

MIN operation will work in the same way. If a SQL query is made for table STD as:

```
SELECT MIN (MARKS) AS MINMARKS
FROM STD
```

Then the above statement would select the lowest (minimum) Marks from the column MARKS of the table STD and result would look like this:

MAXIMMARKS
97

Similarly AVG(),COUNT(),SUM() would give the result as stated above.

Aggregate functions often need an added GROUP BY statement. The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns.

The SQL syntax of GROUP BY statement is:

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
```

If you want to find the sum total of each students from the table STD, then you have to use the GROUP BY statement to group the STUDENTS.

If you use the following SQL statement:

```
SELECT STUDENTS, SUM (MARKS) FROM STD
 GROUP BY STUDENTS
```

Then the result will look like this:

STUDENTS	SUM (MARKS)
Monika	55
Sangeetha	154
Seema	262

SELF-ASSESSMENT QUESTIONS – 1

1. _____ may be avoided if an appropriate index exists to allow ordered access to the records.
2. Relations are said to be Union compatible if they have same _____ and that to from same domain.
3. JOIN operation is the most time consuming operation. (True/False)
4. _____ is another name given to search algorithms.

Activity 1

Discuss the reasons for converting SQL queries into relational algebra queries before optimization is done.

3. HEURISTICS IN QUERY OPTIMIZATION

In general, heuristics means a good idea to make best decision. We use heuristics to solve our daily life problems. Heuristics can be defined as rules governed from common sense rather than from an exhaustive methodology.

In query optimisation heuristic rule defines internal representation of the query. This representation may be in the form of query graph or query tree. It is done with the objective of performance improvement. All high level programs initially generate an internal representation, which is further optimised as per the heuristic rules. Later on, depending upon the access path specified in the query, the query execution plan is generated.

The basic heuristic rule says that before applying JOIN operation or any other binary operation, SELECT and PROJECT operations must be applied. This is done because SELECT and PROJECT operations reduce the size of a file while JOIN operation increases the file size.

Query graph and query tree are the data structures used for queries internal representation. Query graph represents relational calculus expression. And query tree represents relational algebra.

3.1 Notation for Query Trees and Query Graphs

A query tree is employed to represent relational algebra expressions. Here, the leaf nodes symbolize query's input relations, and the internal nodes denotes the relational algebra operations.

Query tree execution consists of an internal node operation. On the operation execution the operands are replaced from the internal nodes by the resultant relation. The operation reaches its final stage on the execution of the root node and generates the resultant relation for the query.

Figure 4.1 illustrates one query tree for query block 'Q'. For all projects from the 'Sikkim', retrieve project number, controlling dept. Number, and manager's surname, address and birth date.

Q: SELECT P.PNUMBER, P.DNUM, E.ENAME, E.ADDRESS, E.BDATE

FROM PROJECT AS P, DEPT AS D, EMP AS E

WHERE P.DNUM=D.DEPTNO

AND D.MGREMPNO=E.EMPNO

AND P.PLOCATION='Chennai';

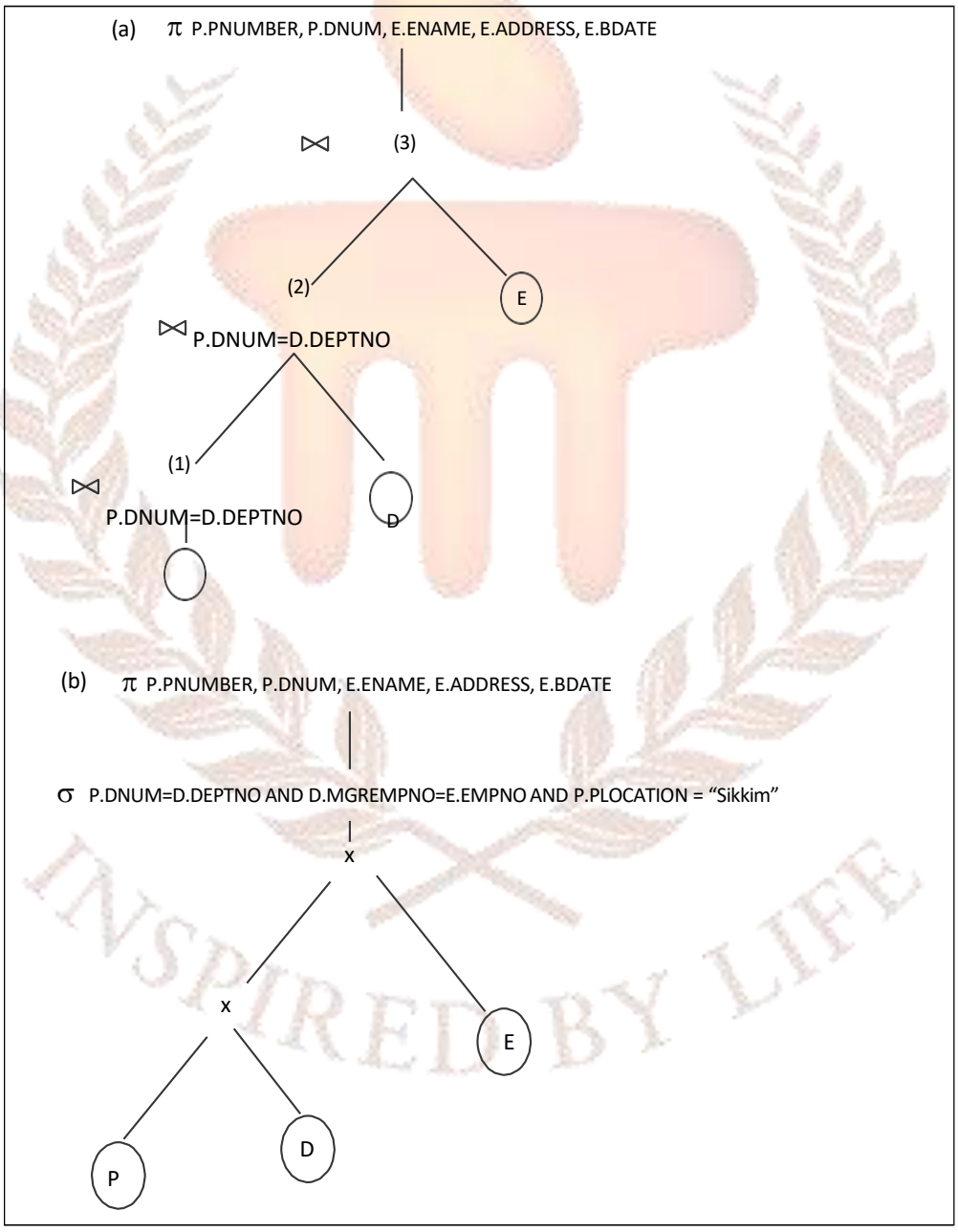

Fig 4.1: Two Query Trees for the Query Q

In Figure 4.1(a), the leaf nodes P, D and E represents PROJECT, DEPT and EMP relations respectively. And the operations are represented by the internal tree nodes. On the execution of this query tree, all the nodes marked (1), (2) and (3) in Figure 4.1(a) will operate sequentially as the resulting tuples of preceding node will be the input of the following node. Query tree arranges the operations in specific order for query execution.

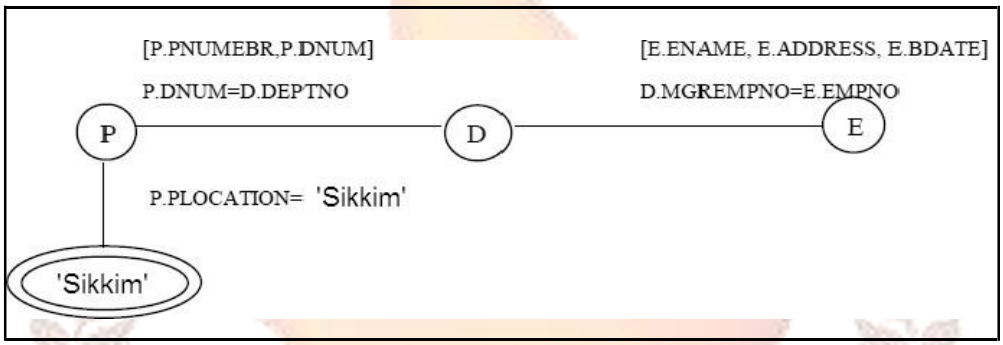

Fig 4.2: Query Graph for Q2

Figure 4.2 represents the query graph for the relational algebra expression given below:

◊ PNUMBER, DNUM, ENAME, ADDRESS, BDATE

(((PLOCATION= 'Chennai' (PROJECT))

⋈ DNUM=DEPTNO (DEPARTMENT))

⋈ MGREMPNO=EMPNO(EMP))

In the query graph as shown in Figure 4.2 the single circles represents the relations whereas the double nodes denotes the constant values. The graph edges represent the selection and join conditions. And the square brackets specify the attributes to be retrieved from each relation.

Moreover, there are no order preferences for executing an operation in case of query graph. In correspondence to every query only a single graph can be created.

3.2 General Transformation Rules for Relational Algebraic Operations

If two operations produce same result they are said to be equivalent. Interestingly, two relations can also be considered as equivalent if they have the same set of attributes in a different order but represents the same information.

Many rules exist to transform relational algebra operations into equivalent ones. The symbols used over there are defined in the table below.

Table 4.3: Notations for Various Relational Algebra Operations

Relational Algebra Operation	Symbol
SELECT	**σ**
PROJECT	**Π**
JOIN	⋈
Union	**U**
Interaction	∩
Cartesian Product	***x***

Let's study some of the transformation rules used in query optimisation:

Transformation rule	Explanation
Sequence or Cascade of σ	A conjunctive selection condition can be broken up into a cascade (sequence) of individual ***σ*** operation $\sigma_{c1}\text{AND}\sigma_{c2}\text{AND}\ldots\text{AND}\sigma_{cn}(R) = \sigma_{c1}(\sigma_{c2}(\ldots(\sigma_{cn}(R))\ldots))$ here $c_1 \ldots c_n$ are the relational conditions and R is the Relation.
Commutativity of σ	The ***σ*** operation is commutative. $\sigma_{c1}(\sigma_{c2}(R)) = \sigma_{c2}(\sigma_{c1}(R))$
Sequence or Cascade of π	In a cascade (sequence) of π operations, all but the last one can be ignored: $\pi_{List1}(\pi_{List2}(\ldots..(\pi_{Listn}(r(N)))\ldots\ldots)) = \pi_{List1}(r(N))$
Commuting π with π	If the selection condition c involves only those attributes Al,…, An in the projection list, the two operations can be commuted: A1, A2…An (σc(R)) = σc (πA1, A2… An (R))
Commutativity of ⋈ (and x)	The ⋈ operation is commutative, as is the x operation: $R \bowtie_c S = S \bowtie_c R$ $R \times S = S \times R$ Note: The order of attributes can vary in the resulting relation as compared to the original relations.

Associativel y of ⋈, x, U and ∩:	These four operations are individually associative; that is, if q stands for any one of these four operations (throughout the expression), we have: R q (S q T) = (R q S) q T
Distribution σ with set operations	The operation commutes with È, Ç and x. If q stands for any one of these three operations (throughout the expression), we have: σc (R S) = (σc (R)) q (σc (S))
Converting a (σ, x) sequence into ⋈	If the condition c of a ***σ*** that follows an x corresponds to a join condition, convert the (***σ***, x) sequence into ⋈ as follows: (***σ*** c (R x S)) = (R ⋈ c S) There are other possible transformations. For example, a selection or join condition c can be converted into an equivalent condition by using the following rules (DeMorgan's laws): NOT (c1 AND c2) = (NOT c1) OR (NOT c2) NOT (c1 OR c2) = (NOT c1) AND (NOT c2)

3.3 Conversion of Query Trees into the Query Execution Plans

Query tree is represented by execution plan for a relational algebra expression. It holds information about the algorithm to be used and the access methods available. This information is helpful to compute the relational operators existing in the query tree.

To understand this, let's consider a query Q2:

SELECT Fname, LName, Address

FROM DEPARTMENT, EMPLOYEE

WHERE D.Dname='Research'

AND D.DNumber=E.DNO

The transformation of query Q2 into the relational algebra expression will be like:

π FNAME, LNAME, ADDRESS (σDNAME='RESEARCH' (DEPARTMENT) ⋈

DNUMBER=DNDEMPLOYEE)

Moving ahead, the query tree for query Q2 is shown in Figure 4.3.

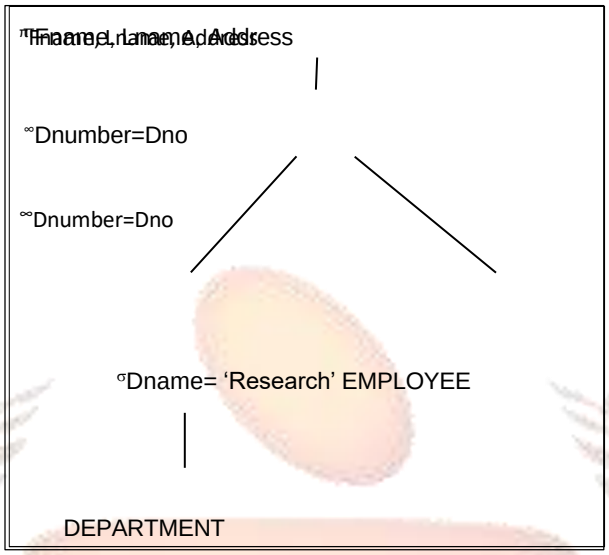

Fig 4.3: A Query tree for query Q2

For the conversion of this tree into an execution plan, following are the requirements of the optimiser:

- index search for the SELECT operation
- table scan as access method for EMPLOYEE,
- nested loop join algorithm for JOIN,
- scan of JOIN result for the PROJECT operator.

Additionally, for query execution a pipelined or materialised evaluation can also be taken into account.

Materialised (stored) evolution specifies that the result of operations can be stored as temporary relation (table). For example, the output of JOIN operation can be stored as a temporary relation (table), which further is read as input for the PROJECT operation, to produce the resultant query table.

Pipelined evaluation results into the cost savings. This is because the intermediate results need not to be saved to the disk and not having to read them back for the next operation.

SELF-ASSESSMENT QUESTIONS – 2

5. The size of the file can be reduced by SELECT and _____ operations.
6. _____ represents a relational calculus expression.
7. The query graph representation also indicates an order in which operations perform first. (True/False).

4. SEMANTIC QUERY OPTIMISATION

Semantic query optimisation is a technique to modify one query into another query by using the relational database constraints. These constraints may be unique attributes or much more complex constraints. This technique is used for the efficient execution of the query.

Let's discuss this approach with the help of an example given below:

```
Select E.LNAME, M.LNAME
FROM EMPLOYEE AS E, EMPLOYEE AS M
WHERE E.SUPERNO=M.ENO AND E.SALARY>M>SALARY
```

This query retrieves the names of employees who earn more than their supervisors. If a constraint is applied on the database schema to check that none of the employee can earn more than his reporting supervisor; the semantic query optimiser checks for this constraint and may not execute the query if it knows that the resultant query will be empty. If the constraint check is done efficiently then this approach can save a lot of time.

SELF-ASSESSMENT QUESTIONS – 3

8. Semantic query optimisation helps in efficient query _____ by modifying one query into another.
9. Relational database constraints are used in semantic query optimisation technique. (True/False)

5. MULTI-QUERY OPTIMISATION AND APPLICATION

Query optimisers have been very useful for the success of relational database technology. For data stream systems the risk is even higher. As compared to relational DBMS one-shot queries, the stream system processes multiple continuous queries simultaneously. The risk involved in data stream systems is higher. These queries can process massive streams in real time and are active for longer time period. A query performance can be affected depending upon the query implementation method. For good stream processing performance the key is to optimise multiple queries together, instead of optimising them individually.

In stream query processing, the workload is shared among concurrently multiple active queries by sharing computation and state. Query evaluation techniques that do not follow this property are known as MQO (Multi- Query Optimisation) techniques. MQO saves the evaluation cost and execution time by executing the common operations once over a set of queries. MQO offers significant improvement to the system performance.

Consider the following two queries that retrieve information from an order processing database.

a)

```
SELECT name, custkey, orderkey, orderdate, totalprice
FROM customer, orders, lineitem
WHERE orders.custkey = customer.custkey
AND lineitem.orderkey = orders.orderkey
AND lineitem.quantity = '24';
```

b)

```
SELECT name, custkey, orderkey, orderdate, totalprice
FROM customer, orders, lineitem
WHERE orders.custkey = customer.custkey
AND lineitem.orderkey = orders.orderkey
AND lineitem.quantity = '24'
AND orders.orderstatus = ‘shipping’;
```

The first query retrieves customer and order information for the specific quantity of items ordered. The second query also retrieves the same information but only for those whose order status is shipping.

Second query's output is a subset of the first query output, so its computation is fast. MQP (Multiple Query processing) helps in optimising the result by first finding out the customers whose lineitem quantity is 24 and then utilising this information by applying additional constraint to check that the orderstatus is shipping.

Multi-query optimisation technique can be used for framing proficient algorithms for problems like view/index selection, query result caching and maintenance.

For example, multiple query optimisations can be applied to mobile database system to pull (on-demand) batches requests. Several queries can be answered at once by the resulting view. This is broadcasted over a view channel dedicated to common answers of multiple queries instead of transmitting over individual downlink channels.

Efficient and extensible algorithm for multi-query optimisation

Multi-query optimisation targets common sub-expressions to be removed to minimize the evaluation cost. This can be achieved with the help of multiple cost-based heuristic algorithms designed particularly for multi-query optimisation. Greedy algorithm is the simplest algorithm that falls under this category.

Greedy algorithm is a cost-based heuristic algorithm. Depending upon the current situation it makes the decision and never reconsiders this decision again, whatever situation may arise later. To find an optimal solution, Greedy algorithm selects a set of nodes to be materialised and then concludes the decision. It is a repetitive task to be carried over different sets of nodes to find the best set of nodes to be materialised.

A greedy strategy works in a top-down manner. It reduces each problem into a smaller one by making one greedy choice after another. This approach turns out to be good strategy in some cases and sometimes does not offer optimal solutions, but only provides a compromise that produces acceptable approximations.

SELF-ASSESSMENT QUESTIONS – 4

10. The key to achieving good stream processing performance is to optimise _____ together.
11. MQO (Multi Query Optimisation) saves the evaluation cost and execution time by executing the common operations once over a set of queries (True/False)

Activity 2

With the help of internet find out some more practical applications of multi query optimisation.

6. EXECUTION STRATEGIES FOR SQL SUB QUERIES

As discussed in unit-2 sub queries are a powerful addition to the SQL language. In different application they help in query creation for example decision support or automatic query formulation by ad-hoc reporting tools. Different physical execution strategies are employed by query optimiser for the various logical query plan options.

Two types of strategies are there for sub query execution:

(1) navigational strategies
(2) set-oriented strategies

- ***Navigational strategies:*** For executing sub- query, navigational strategies depends on the nested loops joins. Basically there are two classes of navigational strategies: *forward lookup and reverse lookup. Forward lookup firstly starts executing the outer query and when outer rows are generated then it invokes the sub-query*. Reverse lookup starts with the sub-query and processes one sub-query row at one time.
- ***Set-oriented processing*** finally needs that the query could be effectively de-correlated. If this is the situation, set operations for example hash, merge and join can execute the query.

SELF-ASSESSMENT QUESTIONS – 5

12. _____ rely on nested loop joins for implementation.

13. works reversely it starts with sub query first and after that executes the outer query.

7. QUERY PROCESSING FOR SQL UPDATES

Data update operations are considered as the most important part of user applications. The proficient processing of these operations is equally important as the processing of data retrieval operations. When data insertion, deletion or modification is performed in a database, the DBMS is required to confirm the changes made by the stated constraints. Also it preserves the basic storage structures in order to maintain a consistent as well as correct representation of data.

It is required to provide validation to update operations against stated constraints like Check, uniqueness, etc. Also these operations are required to preserve the basic storage structures.

Modelling updates in query processing

An algebraic model is a base for SQL query processor. It supports new operator addition as per the requirement. The trees related to those operators are manipulated by the structure. Now let us discuss some concepts related to update processing approach:

Delta stream: It is defined as a set of rows which is used to encode the changes to a specific base table. This is just a relation with a precise schema. Any relational operator can process it.

Application of delta stream: Stream Update operator is considered as a side-effecting operator. This operator for every input row, sends data modification instruction to the storage engine. An update instruction carries Stream Update multiple instances. This is further used for maintaining physical structures of different types.

In Figure 4.4, we have shown a general template used for the implementation plan of update statements.

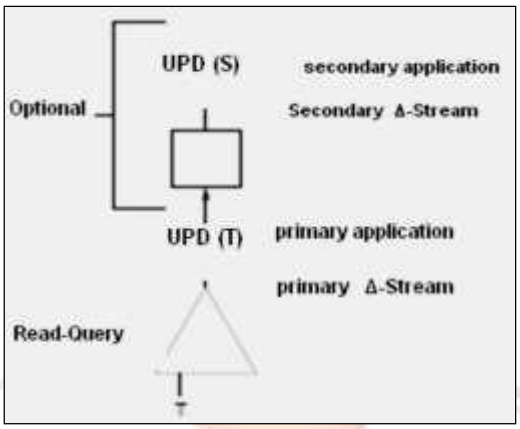

Fig 4.4: General Template for the Execution Plan of Update Statement

It incorporates two components:

- The first is read-only and in charge for delivering a delta stream that specifies the changes to be applied to the target table.
- The second component consumes the delta stream, applying the changes to the base table, and then performs all the actions that the DML (**Data Manipulation Language**) statement implicitly fires.

The action series to be executed is evaluated by checking the entire active dependencies opposed to the target table, and then filtering depending upon the current statement requirement.

SELF-ASSESSMENT QUESTIONS – 6

14. It is required to validate update operations against stated relational database constraints (True/False)
15. _____ is defined as a set of rows that encode the changes made to a specific base table.

8. SUMMARY

Let us recapitulate the important points of this unit:

- Sorting is one of the primary algorithms used in query processing. It is of two types: internal sorting and external sorting.
- There are multiple categories of query execution algorithms such as external sorting, binary search, linear search, hash-key search or primary index etc.
- SELECT (represented by symbol) operation performs the task of retrieving the desired records from the database. There are various search methods such as linear search, binary search and primary index.
- JOIN operation is used to join two database tables/relations. There are various algorithms for implementing the JOIN operations such as Nested loop join, single loop join, sort-merge join and hash join.
- Query graph and query tree are the data structures used for queries internal representation.
- A Multi Query Optimisation (MQO) is a methodology of optimising a group of SQL queries altogether with the help of common sub- expressions to save cost and time.
- There are two types of sub- query optimisation strategies. First, is the navigational strategy and second is the set-oriented strategy.

9. GLOSSARY

- ***Greedy algorithm:*** A Greedy algorithms is a cost-based heuristic algorithm.
- ***Mobile database system:*** A mobile database is a database that can be connected to by a mobile computing device over a mobile network.
- ***Multi-query optimisation:*** A Multi Query Optimisation (MQO) is a methodology of optimising a group of SQL queries altogether with the help of common sub-expressions to save cost and time.
- ***Query optimisation:*** Query optimization refers to the procedure for selecting the best execution strategy amongst the various options available
- ***Query tree:*** A query tree is a data structure to represent relational algebra expressions.

10. TERMINAL QUESTIONS

1. Explain the various heuristics involved in query optimisation.
2. Explain the various algorithms for executing query operations
3. Write a short note on Semantic query optimisation.
4. Describe multi-query optimisation and its application.
5. What are the various execution strategies for SQL sub-queries?

11. ANSWERS

Self-Assessment Questions

1. Sorting
2. Attributes
3. True
4. File scans
5. PROJECT
6. Query graph
7. False
8. Execution
9. True
10. Multiple queries
11. True
12. Navigational strategies
13. Reverse lookup
14. True
15. Delta stream

Terminal Questions

1. Heuristics approach to query optimisation uses heuristic rules and algebraic techniques to improve the efficiency of query execution. Refer Section 3 for more details.
2. SELECT, JOIN, PROJECT (UNION, INTERSECTION, SET DIFFERENCE), and aggregate operations (MIN, MAX, COUNT, AVERAGE, SUM) are various algorithms for query execution. Refer Section 2 for more details.
3. Semantic query optimisation is a different approach to query optimisation that uses various constraints to modify one query into another to make it more efficient to execute. Refer Section 4 for more details.
4. To achieve good stream processing performance, multiple queries are optimised together, rather than individually. This is multi-query optimisation. Refer Section 5 for more details.

5. There are mainly two techniques for SQL sub-queries execution namely navigational strategies and set-oriented strategies. Refer Section 6 for more details.

12. REFERENCES

- Elmasri, Navathe, Somayajulu, Gupta, (2006) *Fundamentals of Database Systems,* (6th Ed.), India: Pearson Education.
- Peter Rob, Carlos Coronel, (2004). *Database Systems: Design, Implementation, and Management*, (7th Ed.), US: Thomson Learning.
- Silberschatz, Korth, Sudarshan, (2011). *Database System Concepts,* (6th Ed.), McGraw-Hill.

E-references

- http://www.cs.iusb.edu/technical_reports/TR-20080105-1.pdf
- http://research.microsoft.com/pubs/76059/pods98-tutorial.pdf
- http://infolab.stanford.edu/~hyunjung/cs346/ioannidis.pdf