INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 3, Issue 6)
Available online at www.ijariit.com

A Comparative Study on Query Optimization and Performance Analysis using Different Data Values

Murali Krishna Senapaty
muralisenapaty@gmail.com
Gandhi Institute of Engineering and Technology, Gunupur, Odisha

Sudhakar Panigrahy
sudhakarpanigrahy@gmail.com
Gandhi Institute of Engineering and Technology, Gunupur, Odisha

A. Sreelakshmi
srilak1@rediffmail.com
Maharshi Gurukul, Gunupur, Odisha

ABSTRACT

A database system will respond to requests for information from the users with the help of queries. In this paper we focussed on query optimization and analysed the queries with different case studies for improving the performance. We have taken a sample data for query processing before and after query optimization and then made a comparative performance analysis.

Keywords: *Query Optimization, Parsing, Cost Estimation, Query Analysis, Query Metrics, and Cartesian product.*

1. INTRODUCTION

The Query Processor

A query goes through 3 phases in DBMS, they are:
1. Parsing and translation
2. Optimization process
3. Evaluation process

Most of the queries are submitted to a DBMS are in a high-level language such as SQL. Now during the parsing phase and translation phase, the human readable form of the query is translated into forms usable by the database system. These can be in the forms of a relational algebraic expression, query tree and query graph. [1][7]

After that the query tree or graph that can be handled by an optimization engine. The optimization engine then performs various analyses on the query data, generating a number of valid evaluation plans. From there, it find the most appropriate evaluation plan for execution. A query goes through 3 phases in DBMS, they are:

1. Parsing and translation
2. Optimization process
3. Evaluation process

Most of the queries are submitted to a DBMS are in a high-level language such as SQL. Now during the parsing phase and translation phase, the human readable form of the query is translated into forms usable by the database system. These can be in the forms of a relational algebraic expression, query tree and query graph. [13]

After that the query tree or graph that can be handled by an optimization engine. The optimization engine then performs various analyses on the query data, generating a number of valid evaluation plans. From there, it find the most appropriate evaluation plan for execution. [10]

1.1 Parsing and Translating the Query

The SQL represents the query as a string or sequence of characters. The parser is to extract the tokens from the sequence of characters and translate them into corresponding internal data elements i.e. relational algebra operations and operands and structures i.e. query tree and query graph. [15][1]

1.2 Query Optimization Process

Query Plans

A query execution plan is an ordered set of steps for accessing data in a SQL relational database management system.

1.2.1 Query Optimization: A query can be executed through different algorithms, written in different forms and structures. So, the query optimization plays very important

role for retrieval of data in terms of time and memory space. The query optimizer attempts to find the efficient way to execute a given query by considering the possible query plans.

1.2.2 Importance of Query Optimization: The goal of query optimization is to reduce usage of system resources and to provide the user with the correct result set within minimum span of time.

- Optimization gives results quickly and in turn it makes the application faster to the user.
- Optimization allows the system to execute more queries in less amount of time, because each optimized query request takes less time than un-optimized queries.
- Query optimization ultimately reduces the amount memory usage and allows the server to run more efficiently with less power consumption.

1.2.3. A Query Optimization Generally Classified in Two Ways: [9] [11]

- We can try to minimize the column counts and tuples in the intermediate query processing and in final query processes
- We can use some algorithms on each operation to determine how tuples are accessed from the data structures.

1.2.4. The Query Optimization in Relational System

We can reduce the multi block query into a single block using Merging Views: Consider a query of conjunctive type and if one or more relations in the query are views then the view definitions are to be unfolded to obtain a single block SQL query. But if the views are complex then the unfolding may not work.

1.3 Evaluating the Query

The best process is to select an optimization engine and execute. Sometime the queries can be processed in parallel as independent processes or threads.

2. ANALYSING THE EQUIVALENT RELATIONAL EXPRESSIONS [1]

2.1 Conjunctive selection operations as: While joining the two tuples which are having less no. of records have to join first. This results less no. of tuples so that the same cane be extended for another inner join.

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

2.2 Selection is Commutative: It is always best practice to apply the selection first and then go for outer selection as the selection is commutative.

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

2.3 Selections on Cartesian Products can be as

Equivalence: While applying the cross production operation It matches the E1 with E2 which produces m*n entries. Then if we apply selection operation on that entries then we have to scan m*n entries to find the required tuples which satisfies out condition.

Without doing it if we go for a theta join to select only those entries then it will be more optimal.

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

Equivalence: The theta join reduces the no. of resulting tuples, if we go for applying the intersection of both join conditions then we will get very few scans.

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

2.4 Join Operations are Associative

Natural Join: Here we need to take care that the two tables which have less no. of entries they need to be joined first and then apply join on other table.

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

Theta Join: As above the theta join also can be implemented as it is associative.

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

3. COST ESTIMATION

We can find many logical algebraic expressions and many ways to implement them as operators. Here the operator tree uses resources like CPU time, memory, communication band width etc. So an operator tree of a query needs to be accurate. Here the frame work estimation can be known based on collecting the summery of data stored, output data stream, and cost estimation of execution process.

3.1 Statistical Information on Base Data

For a relation the statistical information contains the no. of tuples which determines the cost of data scans, joined and their memory requirements, the no. of physical pages used by the table. [3][4]

3.2 Estimating Statistics on Base Data

Generally an enterprise database have large volume of data so to get the statistics for improving accuracy we need to estimate the statistical parameters also.

3.3 Propagation of Statistical Information

A query may contain many operators so it is important to propagate the statistical information. It is possible using operators. e.g.: selection. The inability to capture correlation is a key source of error. Similarly if the multiple predicates are present, then independence assumptions are made and the product of the selectivity is considered. Anyways, some systems only use the selectivity of the most selective predicate. [8]

4. OPTIMIZATION OBJECTIVES

In the optimization we focus on either maximize the output or minimize the usage of resources. The main objective of query optimization is to minimize the response time for a given query language. [12][13]

Here the direct cost minimization of technical resources also important. So, we shall have a comparison of efficiency the functional capabilities of the query evaluation systems to be

4.1 Communication Cost: It includes costs for the communication, time the line is open, and costs for the delay in processing caused by transmission.

4.2 Secondary Storage Access Cost: The cost of loading data pages from secondary storage into main memory. e.g the number of data to be retrieved, the clustering of data on physical pages, the size of the available buffer

4.3 Storage Cost: It is the cost of occupying secondary storage and memory buffers.

4.4. Computation Cost: The cost for using the CPU.

In centralized systems, the costs are dominated by the time for secondary storage accesses although the CPU costs may be quite high for complex Queries. In locally distributed DBMSs, all factors have similar weights, which results in very complex cost functions and optimization procedures.

For the optimization of single queries, storage costs are usually also assumed to be of secondary importance. A number of ideas underlie most techniques developed to reduce these costs. They try to (1) avoid duplication of effort, (2) use standardized parts, (3) to avoid unnecessary operations, (4) choose the cheapest way to execute operations, and (5) sequence them in an optimal fashion. [7]

5. QUERY ANALYSIS FOR PERFORMANCE MEASUREMENT

Considered the sample tables along With records for analysis:

Table Name: smsAttendance_Master
No. of rows: 81917

vcAttendance_No	varchar(14)	Unchecked
vcSubject_Registration_No	varchar(14)	Checked
vcSubject_Short_Name	varchar(20)	Unchecked
vcEmp_code	varchar(10)	Checked
dtDtAttendance_Date	datetime	Unchecked
dtDate_Of_Entry	Datetime	Unchecked
vcClass_Time	varchar(30)	Checked
vcTopics_Covered	varchar(MAX)	Checked
vcStatus	varchar(7)	Unchecked
vcRemarks	varchar(MAX)	Checked
vcClass_Adjusted_By_Emp_Code	varchar(10)	Checked
vcSchedule_No	varchar(15)	Checked
vcAttendance_For	varchar(30)	Checked
vcCurrent	varchar(20)	Checked

Table Name: smsAttendance_Details
No. of rows: 3877464

vcAttendance_No	varchar(14)	Checked
vcRoll_No	varchar(10)	Checked
intPresent	Int	Unchecked

measured. Here the total cost to be minimized is =communication cost+ secondary memory access cost+ storage cost+ computation cost

4.5. Query metrics: The execution time of a query depends on the resources needed to perform the operations such as: disk accesses, CPU cycles, RAM etc. Since the data transfer to/from disks is substantially slower than memory-based transfers, the disk accesses usually represent an overwhelming majority of the total cost. The cost to access a disk is usually measured in terms of the number of blocks transferred from and to a disk. [2][5]

Table Name: smsSubject_Registration_Master
No. of rows: 419

vcSubject_Registration_No	varchar(14)	Unchecked
vcSession	varchar(20)	Unchecked
vcBranch_Id	varchar(20)	Checked
vcSection	varchar(1)	Unchecked
intSemester	Int	Unchecked
vcStatus	varchar(7)	Unchecked
vcRemarks	varchar(MAX)	Checked
vcSchedule_No	varchar(15)	Checked
vcAdmission_Batch	varchar(9)	Checked
dtStart_Date	Date	Checked
dtEnd_Date	Date	Checked
vcRequired_Status	varchar(20)	Checked

Table Name: smsEmployee
No. of rows: 280

vcEmp_code	varchar(10)	Unchecked
vcEmp_Branch_Id	varchar(20)	Checked
vcEmp_Designation	varchar(20)	Checked
dtDOJ	date	Checked
vcEmp_Name	varchar(30)	Unchecked
dtDOB	date	Checked
vcStatus	varchar(7)	Unchecked

Table Name: smsStudent
No. of rows: 5612

vcRoll_No	varchar(10)	Unchecked
vcReg_No	varchar(20)	Checked
vcStudent_Name	varchar(30)	Unchecked
vcGender	varchar(30)	Checked
dtDOB	date	Checked
vcSession	varchar(9)	Checked
vcBranch_Id	varchar(20)	Checked
vcSection	varchar(1)	Unchecked
intSemester	Int	Unchecked
vcAt	varchar(30)	Checked
vcPost	varchar(20)	Checked
vcVia	varchar(20)	Checked
vcPs	varchar(20)	Checked
vcDist	varchar(20)	Checked
vcPin	varchar(6)	Checked
vcState	varchar(20)	Checked
vcFathers_Name	varchar(30)	Checked
vcMothers_Name	varchar(30)	Checked
vcFathers_No	varchar(30)	Checked
vcMothers_No	varchar(30)	Checked
vcStudent_ Mobile_No	varchar(30)	Checked
vcStudent_ Email_id	varchar(100)	Checked
vcParent_ Email_id	varchar(30)	Checked
int10th_Per	real	Checked
int12th_Per	real	Checked
int10th_Eng_Marks	real	Checked
int12_Eng_Marks	real	Checked
vc10thBoard	varchar(30)	Checked
vc12thOrDiloma Board	varchar(30)	Checked
int10thYOP	int	Checked
int12thYOP	int	Checked
vcStatus	varchar(6)	Checked
vcRemarks	varchar(MAX)	Checked
vcImagePath	varchar(200)	Checked
realDueAmount	real	Checked
vcAdmission_Batch	varchar(9)	Checked
dtDue_Amount_ As_On_Date	date	Checked
intRegistration	int	Checked

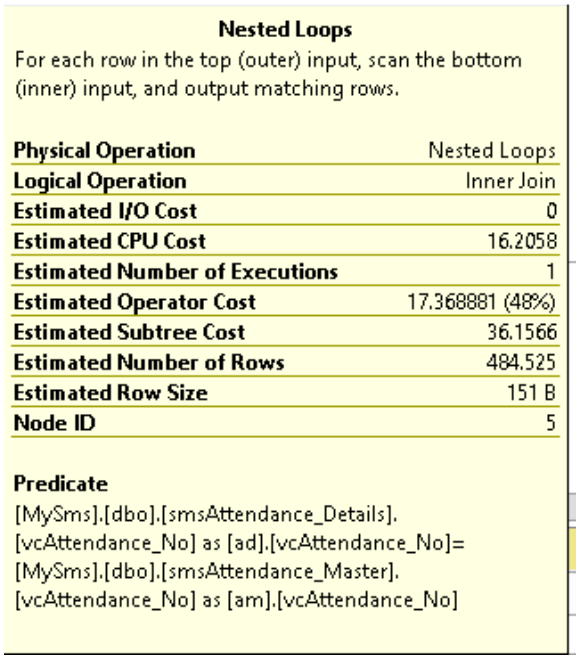
Table scan smsAttendance_Details : 52%

5.1 Example and analysis on query before optimization

Query 1

```
SELECT  am.vcAttendance_No, am.dtDtAttendance_Date,
am.vcAttendance_For, am.vcSubject_Short_Name,
am.vcClass_Time, am.vcEmp_code, e.vcEmp_Name,
e.vcEmp_Branch_Id, am.vcSubject_Registration_No,
srm.vcAdmission_Batch, srm.vcBranch_Id, srm.intSemester,
srm.vcSection, srm.dtStart_Date, ad.vcRoll_No,
s.vcStudent_Name, s.vcStudent_Mobile_No, s.vcFathers_No,
ad.intPresent, am.vcStatus AS smsAttendance_Master_Status,
srm.vcStatus AS smsSubject_Registration_Master_Status,
e.vcStatus AS smsEmployee_Status, s.vcStatus AS
smsStudent_Status
FROM smsAttendance_Master am
,smsSubject_Registration_Master srm, smsAttendance_Details
ad,smsStudent s,smsEmployee e
where
ad.vcAttendance_No=am.vcAttendance_No and
am.vcSubject_Registration_No=srm.vcSubject_Registration_N
o and
am.vcEmp_code = e.vcEmp_code and
ad.vcRoll_No = s.vcRoll_No and
dtDtAttendance_Date='2018/01/17'
```

Query Output
Estemeted Execution plan
Nested loop inner join cost : 48%



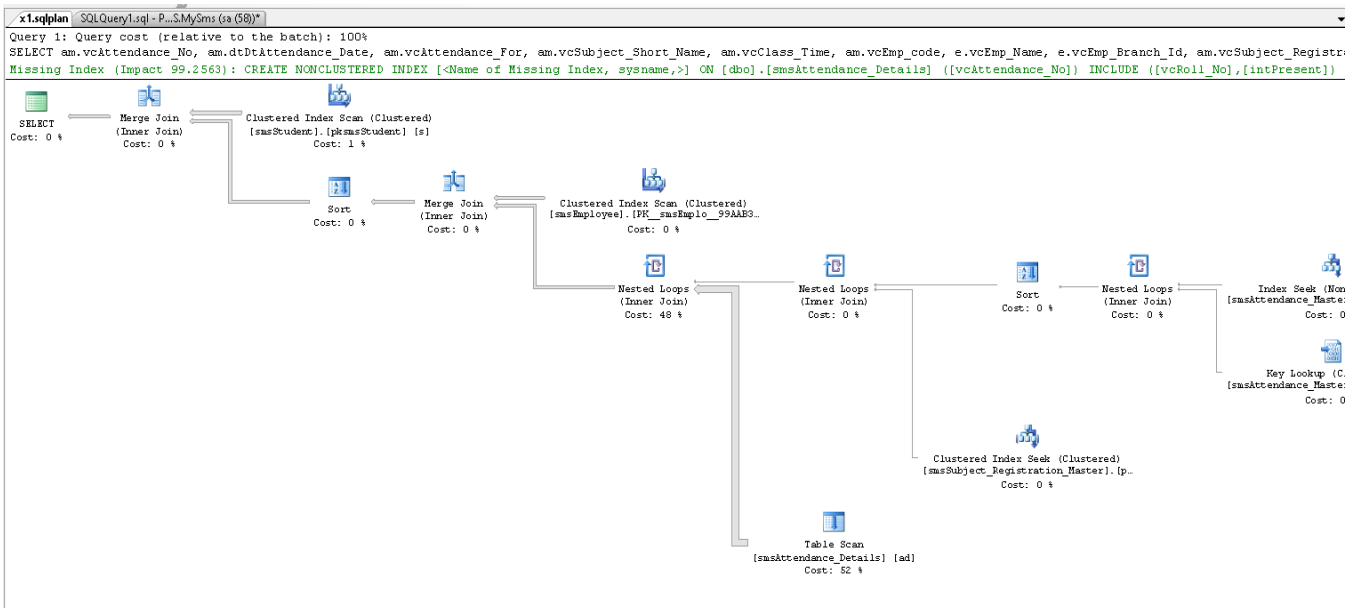


Table Scan

Scan rows from a table.

Physical Operation	Table Scan
Logical Operation	Table Scan
Estimated I/O Cost	14.5016
Estimated CPU Cost	4.26485
Estimated Number of Executions	1
Estimated Operator Cost	18.7665 (52%)
Estimated Subtree Cost	18.7665
Estimated Number of Rows	3876990
Estimated Row Size	38 B
Ordered	False
Node ID	26

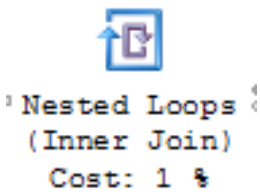
Object
[MySms].[dbo].[smsAttendance_Details] [ad]
Output List
[MySms].[dbo].
[smsAttendance_Details].vcAttendance_No, [MySms].
[dbo].[smsAttendance_Details].vcRoll_No, [MySms].
[dbo].[smsAttendance_Details].intPresent

Index Seek (NonClustered)

Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Estimated I/O Cost	0.0053472
Estimated CPU Cost	0.0007702
Estimated Number of Executions	1
Estimated Operator Cost	0.0061174 (2%)
Estimated Subtree Cost	0.0061174
Estimated Number of Rows	557.466
Estimated Row Size	22 B
Ordered	True
Node ID	25

Object
[MySms].[dbo].[smsAttendance_Details].
[smsAttendance_Details_vcAttendance_No] [ad]
Output List
[MySms].[dbo].[smsAttendance_Details].vcRoll_No,
[MySms].[dbo].[smsAttendance_Details].intPresent
Seek Predicates
Seek Keys[1]: Prefix: [MySms].[dbo].
[smsAttendance_Details].vcAttendance_No = Scalar
Operator([MySms].[dbo].[smsAttendance_Master].
[vcAttendance_No] as [am].[vcAttendance_No])



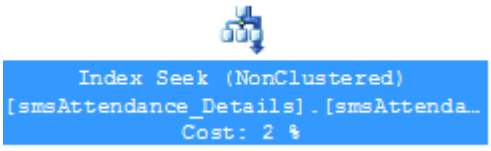
Nested Loops

For each row in the top (outer) input, scan the bottom (inner) input, and output matching rows.

Physical Operation	Nested Loops
Logical Operation	Inner Join
Estimated I/O Cost	0
Estimated CPU Cost	0.0023302
Estimated Number of Executions	1
Estimated Operator Cost	0.0023303 (1%)
Estimated Subtree Cost	0.0215926
Estimated Number of Rows	557.466
Estimated Row Size	186 B
Node ID	2

Outer References

[MySms].[dbo].
[smsAttendance_Master].vcAttendance_No



Clustered Index Seek (Clustered)

Scanning a particular range of rows from a clustered index.

Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Estimated Number of Executions	1
Estimated Operator Cost	0.0032831 (1%)
Estimated Subtree Cost	0.0032831
Estimated Number of Rows	1
Estimated Row Size	41 B
Ordered	True
Node ID	24

Object

[MySms].[dbo].[smsSubject_Registration_Master].
[pksmsSubject_Registration_Master_vcSubject_Registration_
No] [srm]

Output List

[MySms].[dbo].
[smsSubject_Registration_Master].vcBranch_Id, [MySms].
[dbo].[smsSubject_Registration_Master].vcSection, [MySms].
[dbo].[smsSubject_Registration_Master].intSemester,
[MySms].[dbo].[smsSubject_Registration_Master].vcStatus,
[MySms].[dbo].
[smsSubject_Registration_Master].vcAdmission_Batch,
[MySms].[dbo].
[smsSubject_Registration_Master].dtStart_Date

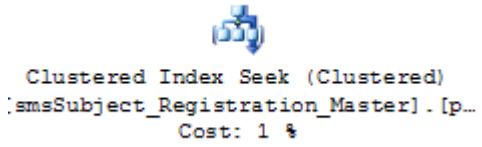
Seek Predicates

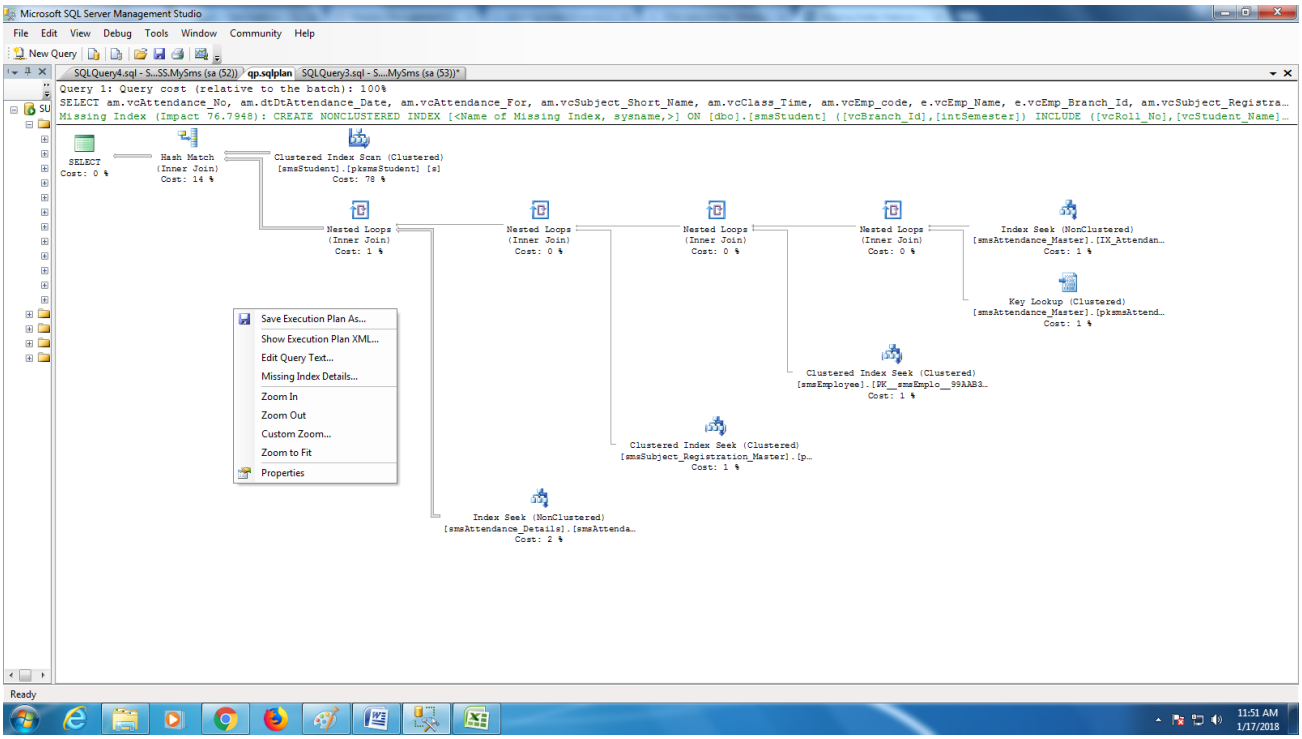
Seek Keys[1]: Prefix: [MySms].[dbo].
[smsSubject_Registration_Master].vcSubject_Registration_N
o = Scalar Operator([MySms].[dbo].
[smsAttendance_Master].[vcSubject_Registration_No] as
[am].[vcSubject_Registration_No])

5.2 Example and analysis on query after optimization

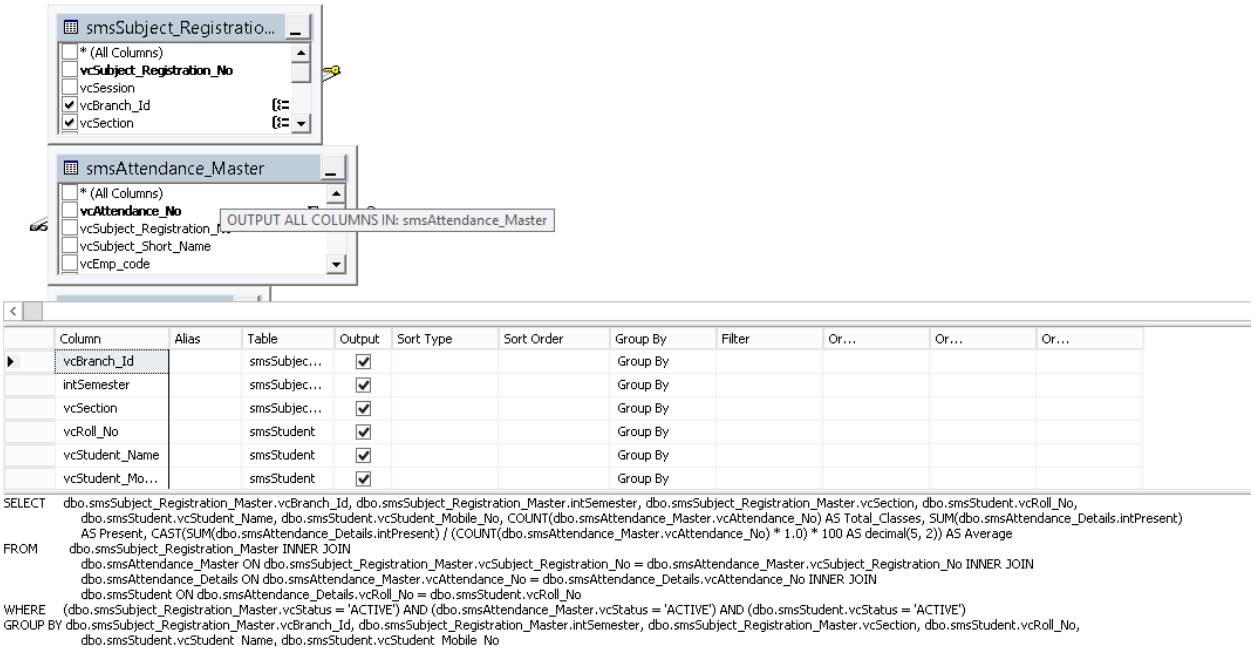
CREATE NONCLUSTERED INDEX
smsAttendance_Details_vcAttendance_No
on smsAttendance_Details ([vcAttendance_No]) include
([vcRoll_No],[intPresent])

Execution plan
Nested loop inner join : 1%





5.3 Creating View



5.4 Retrieving Data from View

```
select vcAdmission_Batch Addmission_Batch, vcBranch_Id Branch, intSemester Semester,
vcSection Section,vcSubject_Short_Name Subject,vcClass_Time Clas_Time,  COUNT(intpresent)
Total_Strength,sum(intpresent) Class_Strength  from viewAttendance_Deatils
 where dtDtAttendance_Date='1/17/2018 12:00:00 AM'   group by vcAdmission_Batch, vcBranch_Id,
 intSemester, vcSection,vcSubject_Short_Name,vcClass_Time  order by vcAdmission_Batch, vcBranch_Id,
 intSemester, vcSection,vcSubject_Short_Name,vcClass_Time
```


Inforamation Of Class, Date : 17-01-2018 Display

Sl No	Addmission_Batch	Branch	Semester	Section	Subject	Clas_Time	Total_Strength	Class_Strength
1	2014-2018	MECHANICAL	8	A	MM	9:10 AM	61	33
2	2015-2019	CSE	6	B	IWT	9:10 AM	60	48
3	2015-2019	EE	6	A	CNDC	9:10 AM	59	56
4	2016-2020	CSE	4	A	MATH III	9:10 AM	61	46
5	2016-2020	EE	4	A	MATH III	9:10 AM	59	41
6	2017-2021	BSH	2	C	DS	8:40 AM	60	34
7	2017-2021	BSH	2	D	CHE	8:40 AM	50	31
8	2017-2021	BSH	2	E	PHY	8:40 AM	62	54

6. CONCLUSION

The most important functional requirements of a DBMS is its ability to process queries in a timely manner. This is particularly true when a very large, mission critical applications such as weather forecasting, banking systems and aeronautical applications, which can contain millions and even trillions of records. The need for faster and faster, "immediate" results never ceases. Thus, a great deal of research and resources is spent on creating smarter, highly efficient query optimization engines. Here we have analysed a sample database of an organisation where some few different data maintained. Some of the basic techniques of query processing and optimization have been presented in this paper. Other, more advanced topics are the subjects of many research papers and projects.

7. REFERENCES

[1] https://www.geeksforgeeks.org/query-optimization/

[2] Henk Ernst Blok, Djoerd Hiemstra and Sunil Choenni, Franciska de Jong, Henk M. Blanken and Peter M.G. Apers. Predicting the cost-quality Trade-off for information retrieval queries: Facilitating database design and query optimization. Proceedings of the tenth international Conference on Information and knowledge Management, October 2001, Pages 207-214.

[3] D. Calvanese, G. De Giacomo, M. Lenzerini and M. Y. Vardi. Reasoning on Regular Path Queries. ACM SIGMOD Record, Vol. 32, No. 4, December 2003.

[4] Andrew Eisenberg and Jim Melton. Advancements in SQL/XML. ACM SIGMOD Record, Vol. 33, No. 3, September 2004.

[5] Andrew Eis enberg and Jim Melton. An Early Look at XQuery API for Java™ (XQJ). ACM SIGMOD Record, Vol. 33, No. 2, June 2004.

[6] Ramez Elmasri and Shamkant B. Navathe. Fundamentals of Database Systems, second Edition. Addison-Wesley Publishing Company, 1994.

[7] Donald Kossmann and Konrad Stocker. Iterative Dynamic Programming: A new Class of Query Optimization Algorithms. ACM Transactions on Database Systems, Vol. 25,
No. 1, March 2000, Pages 43-82.

[8] Chiang Lee, Chi-Sheng Shih and Yaw-Huei Chen. A Graph-theoritic model for optimizing queries involving methods. The VLDB Journal — The International Journal on Very Large Data Bases, Vol. 9, Issue 4,
April 2001, Pages 327-343.

[9] Hsiao-Fei Liu, Ya-Hui Chang and Kun-Mao Chao. An Optimal Algorithm for Querying Tree Structures and its Applications in Bioinformatics. ACM SIGMOD Record Vol. 33, No. 2, June 2004.

[10] Reza Sadri, Carlo Zaniolo, Amir Zarkesh and Jafar Adibi. Expressing and Optimizing Sequence Queries in Database Systems. ACM Transactions on Database Systems, Vol. 29,
Issue 2, June 2004, Pages 282-318.

[11] On Query Optimization in Relational Databases By John Ngubiri, PGDCS(Mak), BSc/Ed(Mak), ngubiri@ics.mak.ac.ug, 071921969

[12] A Performance Study of Query Optimization Algorithms on a Database System Supporting Procedures, Anant Jhingran, EECS Department, University of California, Berkeley

[13] A Study on Optimization Techniques and Query Execution Operators That Enhances Query Performance,
By Tejy Johnson, Dr. S. K.Srivatsa, International Journal of Advanced Research in Computer Science, Volume 3, No. 3, May-June 2012

[14] An Overview of Query Optimization in Relational Systems, Surajit Chaudhuri, Microsoft Research, One Microsoft Way, Redmond, WA 98052, +1-(425)-703-l 938
surajitc@ microsofkcom

[15] Query Optimization in Database Systems MATTHIAS JARKE, Graduate School of Business Administration, New York Uniuersity, New York, New York 10006, JijRGEN KOCH, Fachbereich Znformatik, Johann Wolfgang Goethe-Universitiit, 6000 Frankfurt 1, West Germany

[16] Introduction to Query Processing and Optimization
Michael L. Rupley, Jr. Indiana University at South Bend
mrupleyj@iusb.edu

