IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015

Performance Measurement using CASE Expression over PIVOT Operator in SQL Server 2008 r2

Mr. Sudhakar Panigrahy[1], Mr. Pragnyaban Mishra[2]

Dept. of CSE, Gandhi Institute of Engineering & Technology, Gunupur, Rayagada, India[1,2]

Abstract: Data aggregation using PIVOT operator is useful in many aspects by converting data from rows into column as per the common values. The execution can be faster by using CASE expression in SQL Server 2008 r2. So, CASE expression can be used in the place of PIVOT operator. Some of the major issues in PIVOT operator are the grouping elements can't be defined explicitly. Thus, when database schema changes by adding new column in the existing tables results NULL value in the report. Getting row and column wise aggregated value not possible with PIVOT operator. At a time only one grouping function can be used i.e Different grouping functions can't be applied in different pivot columns. These problems can be solved by using CASE expression. In this paper it is explained the difficulties observed in PIVOT operator and solved using CASE Expression.

Keywords: SQL Server 2008 r2, CASE expression, NULL value, PIVOT operator.

1. INTRODUCTION

A pivot table is a frequently used method of summarizing and displaying especially report data by means of grouping and aggregating values. The main use of PIVOT operator in SQL Server is to rotate rows into columns without changing the data base schema. Pivoting (or producing a "cross-tab") is a common reporting requirement and data stored in rows are presented it in columns. PIVOT compares unique value from row and aggregates the values of the specified column. These aggregated value represented in cross tab format (PIVOT table). Converting from row to column can be done manually through a traditional CASE expression [7, 8, 9]. Generating Dynamic column in both PIVOTE operator and CASE expression is a common requirement for developer as sometimes dynamic column retrieves from another schema or depending on the different values of row data.

The three phases which are used in the CASE expression are Grouping Phase, Spreading Phase and Aggregation Phase. In the Grouping Phase it compacts the records into distinct database entities and performs the GROUP BY. The Spreading Phase which is a second phase where multiple CASE expressions rotates data from rows into columns and finally in the aggregation phase the GROUP function can be used to get grouping values in the pivot column.

In this paper it is explained the details of SQL statements in order to execute complex queries in both the PIVOT operator and CASE expression with dynamic pivot column.

2. RELATED WORK

Valentin Dinu et al. [1] explained the Entity–Attribute–Value (EAV) data, as present in repositories of clinical patient data, and show how it must be transformed (pivoted) into one-column-per-parameter format before it can be used by a variety of analytical programs. Guiyi Wei et al. [2] explained an adaptive discovery frame work called as PIVOT which is called as an active information discovery mechanism and it was found, PIVOT is highly adaptive and necessary utility for grid system.

Damianos Chatziantoniou [3] Transpose a table row data as columns with correlated aggregation and used in many data analysis application like medical informatics, biogenetics etc.

The relational model can be converted to pivot concepts and Pivot is basically used for grouping variables and to make certain complex data analysis queries [4, 5, 6].

CASE expression [7, 8, 9] can be used to create one or more numbers of columns, where clause and order by clause of SQL statement. It can also be used in different kind of statements in DML.

The CASE expression compares set of values may be static or dynamic and can be converted into columns without changing the data base structure.

Performance Measurement between PIVOT operator and CASE expression:

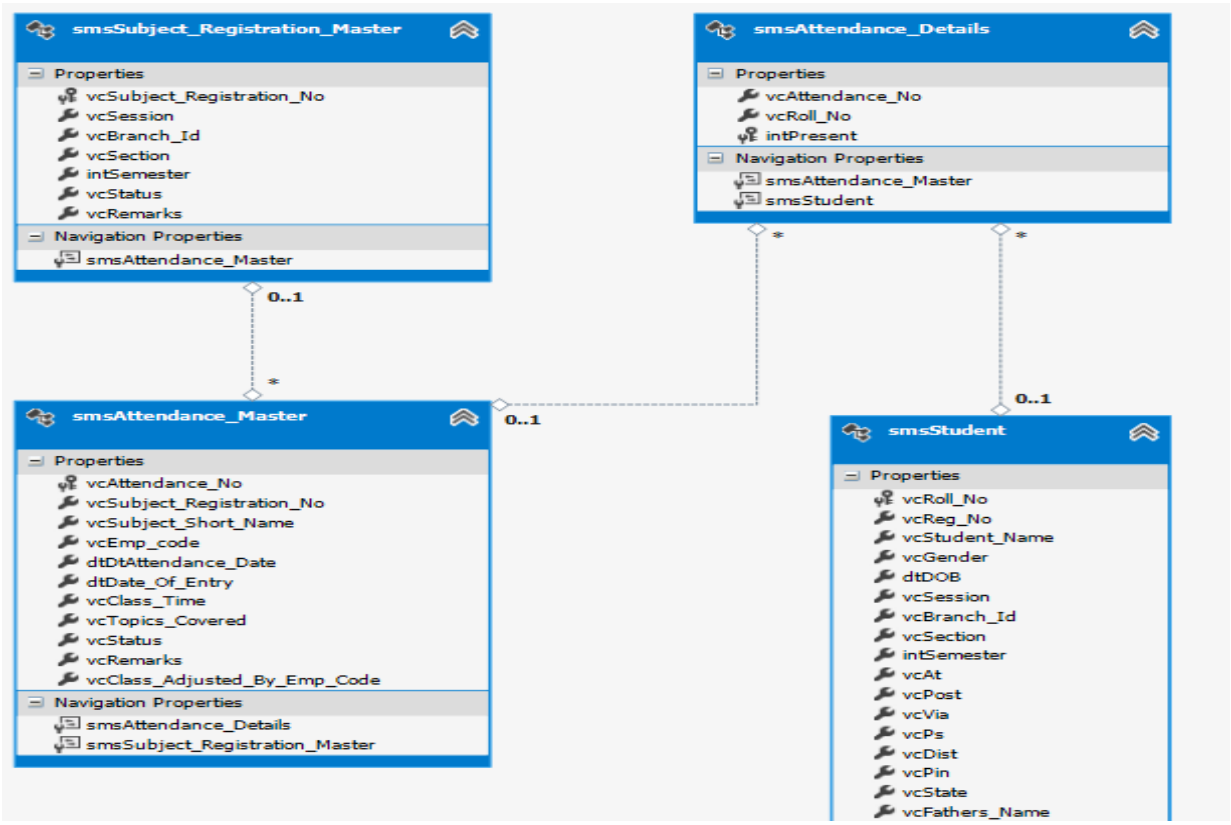
To measure the performance of the both PIVOT operator and CASE expressions following four tables are used and shown along with their relationship and navigation properties.

To measure the performance among the PIVOT operator and CASE expression two methods have been used one is the Query Execution Plan, the other one is execution cost in terms of Ticks (10,000 Ticks = 1Milli Sec).

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015



(Figure-01: The Schema of Database Tables used in the Paper)

The data store as follows while the teachers enter the attendance.

Roll No	Student Name	Parent Mobile Number	Subject	Present
15CSE061	ANUPAM ALOK	9439271311	CP	1
15CSE061	ANUPAM ALOK	9439271311	MECH	1
15CSE061	ANUPAM ALOK	9439271311	MATH 1	1
15CSE061	ANUPAM ALOK	9439271311	BE	0
15CSE061	ANUPAM ALOK	9439271311	PHY 1	1
15CSE061	ANUPAM ALOK	9439271311	CE	0

(Table -01: Data in Database entered by six Subject Teachers)

When the PIVOT operator is used the row wise data are converted into columns with aggregated values as follows:

Sl No	Roll No	Student Name	Parent Mobile Number	BE	CE	CP	MATH 1	MECH	PHY 1
1	15CSE061	ANUPAM ALOK	9439271311	0	0	1	1	1	1

(Table -02: Data in Database by after PIVOT Operation of Table-01)

The query to get above data with dynamic aggregated column as follows:

```
DECLARE @ColumnNames nvarchar(max)=''
DECLARE @sql nvarchar(max)=''
select @ColumnNames += QUOTENAME(vcSubject_Short_Name) + ',' from
( SELECT Distinct smsAttendance_Master.vcSubject_Short_Name FROM
smsAttendance_Master,smsSubject_Registration_Master WHERE
smsAttendance_Master.vcSubject_Registration_No=smsSubject_Registration_Master.vcSubject_Registration_No
AND smsSubject_Registration_Master.vcBranch_Id='BSH' and intSemester= 1 and
vcSection='D' and smsAttendance_Master.dtDtAttendance_Date='11/2/2015' ) as test
set @ColumnNames=LEFT(@ColumnNames, NULLIF(LEN(@ColumnNames)-1,-1))
set @sql= 'select * from (select smsAttendance_Details.vcRoll_No as Roll_No,smsStudent.vcStudent_Name as
Student_Name,
```


DOI 10.17148/IJARCCE.2015.41219

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015

```
smsStudent.vcFathers_No as Parent_Mobile_Number, smsAttendance_Master.vcSubject_Short_Name as
Subject_Name,
intpresent from smsSubject_Registration_Master, smsAttendance_Master, smsAttendance_Details, smsStudent where
smsSubject_Registration_Master.vcSubject_Registration_No=smsAttendance_Master.vcSubject_Registration_No and
smsAttendance_Master.vcAttendance_No =smsAttendance_Details.vcAttendance_No and
smsAttendance_Details.vcRoll_No = smsStudent.vcRoll_No   AND smsStudent.vcBranch_Id="BSH"  AND
smsStudent.intSemester=1 and  smsStudent.vcSection="D"  and smsAttendance_Master.dtDtAttendance_Date
="11/2/2015") as basedata
pivot (SUM(intpresent) for Subject_Name in(' + @ColumnNames + ') )AS PIVOTETABLE'
execute sp_executesql @sql
```

(Query-01: Query using PIVOT operator with dynamic aggregated column)

For the five subjects namely CP, BE, MECH, MATH-1,PHY-1 in the table-04 becames null as the student information was not set while entering where as in the subject CE it indicated the student is present.

Roll No	Student Name	Parent Mobile Number	Subject	Present
15CSE061	ANUPAM ALOK	9439271311	CP	1
15CSE061	ANUPAM ALOK	9439271311	MECH	1
15CSE061	ANUPAM ALOK	9439271311	MATH 1	1
15CSE061	ANUPAM ALOK	9439271311	BE	0
15CSE061	ANUPAM ALOK	9439271311	PHY 1	1
15CSE061	ANUPAM ALOK	9439271311	CE	0
15CSE062	SOMA SAHOO	8018339927	CE	1

(Table -03: Data after inserting a new record of Soma Sahoo for the subject CE)

Sl No	Roll No	Student Name	Parent Mobile Number	BE	CE	CP	MATH 1	MECH	PHY 1
1	15CSE061	ANUPAM ALOK	9439271311	0	0	0	0	0	0
2	15CSE062	SOMA SAHOO	8018339927	NULL	1	NULL	NULL	NULL	NULL

(Table -04: Report after applying PIVOT operator of table -03)

It is also experimented that, when the structure of the table changes it results unrelated output as because in the grouping phase the new column name doesn't appear. Secondly getting row wise sum using PIVOT operator is not possible.

CASE expression can be used in the place of PIVOT operator with following query:

```
SELECT
smsAttendance_Details.vcRoll_No as Roll_No,smsStudent.vcStudent_Name as Student_Name,
smsStudent.vcFathers_No as Parent_Mobile_Number,

SUM(CASE WHEN smsAttendance_Master.vcSubject_Short_Name = 'CP' THEN intpresent END) AS 'CP',
SUM(CASE WHEN smsAttendance_Master.vcSubject_Short_Name = 'BE' THEN intpresent END) AS 'BE',
SUM(CASE WHEN smsAttendance_Master.vcSubject_Short_Name = 'CE' THEN intpresent END) AS 'CE',
SUM(CASE WHEN smsAttendance_Master.vcSubject_Short_Name = 'MATH 1' THEN intpresent END) AS 'MATH
1',
SUM(CASE WHEN smsAttendance_Master.vcSubject_Short_Name = 'MECH' THEN intpresent END) AS 'MECH',
SUM(CASE WHEN smsAttendance_Master.vcSubject_Short_Name = 'PHY 1' THEN intpresent END) AS 'PHY 1',
SUM(intpresent) AS Total
FROM smsSubject_Registration_Master,smsAttendance_Master,smsAttendance_Details,
smsStudent where
smsSubject_Registration_Master.vcSubject_Registration_No=smsAttendance_Master.vcSubject_Registration_No and
smsAttendance_Master.vcAttendance_No=smsAttendance_Details.vcAttendance_No and
smsAttendance_Details.vcRoll_No=smsStudent.vcRoll_No   AND smsStudent.vcBranch_Id='BSH'
AND smsStudent.intSemester=1 and  smsStudent.vcSection='D'
and smsAttendance_Master.dtDtAttendance_Date='11/2/2015'
GROUP BY smsAttendance_Details.vcRoll_No ,smsStudent.vcStudent_Name ,
smsStudent.vcFathers_No
-- Grouping phase
GO
```

(Query-02: Query using CASE expression with static cross tab columns)

DOI 10.17148/IJARCCE.2015.41219

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015

As the requirement is dynamic in nature it is necessary to generate dynamic CASE expression depending on the diffrent subjects taught, as a result cursor is used to get dynamic cross tab colum in CASE expression. The related query is given below:

```
Declare @str1 as varchar(1000)=''
Declare @Name as varchar(100)=''
DECLARE @sql1 nvarchar(max)=''
Declare MY_data1 CURSOR FOR
SELECT Distinct smsAttendance_Master.vcSubject_Short_Name FROM
smsAttendance_Master,smsSubject_Registration_Master WHERE
smsAttendance_Master.vcSubject_Registration_No=smsSubject_Registration_Master.vcSubject_Registration_No
AND  smsSubject_Registration_Master.vcBranch_Id='BSH' and intSemester= 1   and
vcSection='D' and smsAttendance_Master.dtDtAttendance_Date='11/2/2015'

OPEN MY_data1
          FETCH NEXT FROM MY_data1 INTO @Name
      WHILE @@FETCH_STATUS = 0
      BEGIN
                              SET @str1 += 'COALESCE(SUM(CASE WHEN
smsAttendance_Master.vcSubject_Short_Name = ' + QUOTENAME(@Name,'')  +' THEN intpresent   END),0) AS '
+ QUOTENAME(@Name,'') + ',';
                              print @Name;
                              FETCH NEXT FROM MY_data1 INTO @Name
      END
   CLOSE MY_data1;
   DEALLOCATE MY_data1;
set @sql1=  'SELECT smsAttendance_Details.vcRoll_No as Roll_No, smsStudent.vcStudent_Name as
Student_Name,smsStudent.vcFathers_No as Parent_Mobile_Number, ' + @str1 + ' SUM(intpresent) AS Total
FROM smsSubject_Registration_Master,smsAttendance_Master,smsAttendance_Details,
smsStudent where smsSubject_Registration_Master.vcSubject_Registration_No
=smsAttendance_Master.vcSubject_Registration_No and smsAttendance_Master.vcAttendance_No
=smsAttendance_Details.vcAttendance_No and
smsAttendance_Details.vcRoll_No=smsStudent.vcRoll_No   AND smsStudent.vcBranch_Id="BSH" AND
smsStudent.intSemester=1 and  smsStudent.vcSection="D" and  smsAttendance_Master.dtDtAttendance_Date
="11/3/2015" GROUP BY smsAttendance_Details.vcRoll_No ,smsStudent.vcStudent_Name
,smsStudent.vcFathers_No ';
execute sp_executesql @sql1
```

(Query-03: Query using CASE expression with dynamic cross tab columns)

3. EXPERIMENTAL RESULTS

We took random data set and executed using the code segment of query-01 and measured the execution cost in terms of Ticks (10,000Ticks=1Milli Second) for the total of 2922 records of 10 different execution with different data set. The execution cost found to be 15600027 Ticks. This is represented as follows.

Using Dynamic PIVOT Operator				
Sl. No	**Number of Data Set**	**Start Time (In Ticks)**	**End Time(In Ticks)**	**Time Required (In Ticks)**
1	360	635849210629289000	635849210631785000	2496004
2	360	635849211935635000	635849211938287000	2652004
3	360	635849212946985000	635849212948233000	1248002
4	360	635849214004199000	635849214005603000	1404002
5	96	635849215455781000	635849215456717000	936002
6	96	635849216525943000	635849216526879000	936002
7	210	635849217585029000	635849217585965000	936002
8	360	635849218936460000	635849218937552000	1092002
9	360	635849220800975000	635849220803003000	2028003
10	360	635849222914477000	635849222916349000	1872004
Total	2922			15600027

(Table -02: Execution cost of PIVOT Operator of Query-01, measured in TICKS)

DOI 10.17148/IJARCCE.2015.41219

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015

The same data executed using the dynamic CASE expression (query-03) and measured the execution cost in terms of Ticks for the same total numbers of 2922 records, the execution cost found to be 12792021 Ticks.

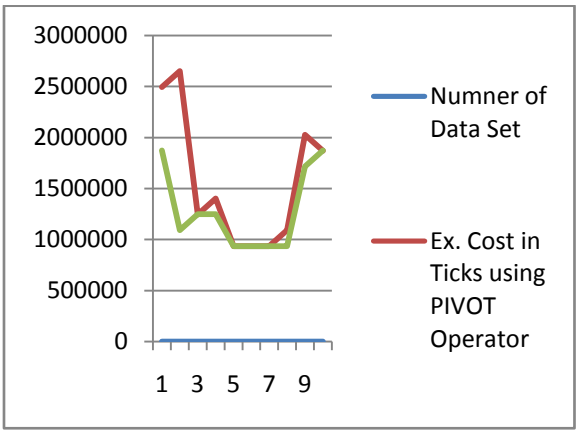
Using Dynamic CASE Expression				
Sl. No.	**Number of Data Set**	**Start Time (In Ticks)**	**End Time(In Ticks)**	**Time Required (In Ticks)**
1	360	635849210851121000	635849210852993000	1872003
2	360	635849212610492000	635849212611584000	1092002
3	360	6358492135411650000	635849213542438000	1248002
4	360	635849214453012000	635849214454260000	1248002
5	96	635849215892582000	635849215893518000	936001
6	96	635849216832328000	635849216833264000	936002
7	210	635849218347090000	635849218348026000	936001
8	360	635849219005412000	635849219006348000	936002
9	360	635849221892977000	635849221894693000	1716003
10	360	635849222864712000	635849222866584000	1872003
Total	2922			12792021

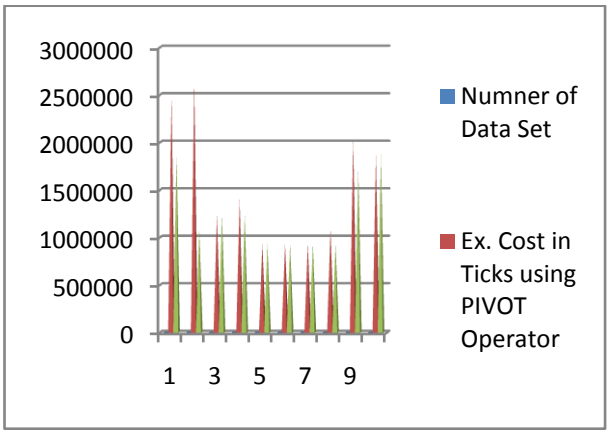
(Table -03: Execution cost of CASE expression of Query-03, measured in TICKS)

The difference in performance is explained as follows:

Sl. No.	**Number of Data Set**	**Ex. Cost in Ticks using PIVOT Operator**	**Ex. Cost in Ticks using CASE Expression**	**Difference in Tics**
1	360	2496004	1872003	624001
2	360	2652004	1092002	1560002
3	360	1248002	1248002	0
4	360	1404002	1248002	156000
5	96	936002	936001	1
6	96	936002	936002	0
7	210	936002	936001	1
8	360	1092002	936002	156000
9	360	2028003	1716003	312000
10	360	1872004	1872003	1

(Table -04: Difference in performance using PIVOT operator and CASE expression)





(Graph-01, 02: Comparison graph of Table-04)

It is observed that, for any length of random data set the execution cost of CASE expression is either smaller or same to the execution cost of PIVOT operator. In this random sampled data set we found that CASE expression is 960.99 Ticks per record faster than Pivot operator.

Here we found the esteemated operator cost in the case of PIVOT operator is 66% and the corresponding cost in case of CASE expression is 59%. It indicates that, CASE expression differed by less 7%. So CASE expression can be substituted in PIVOT operator for better performance.

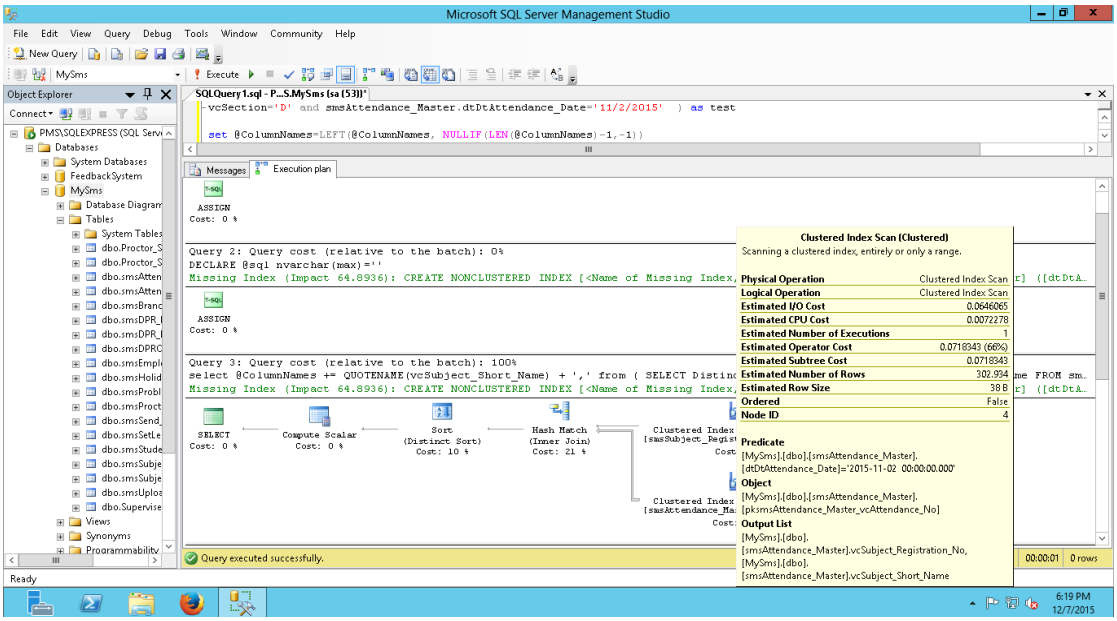
The graphical representation of query execution plan of the PIVOT operator (query-01) is as follows:

DOI 10.17148/IJARCCE.2015.41219

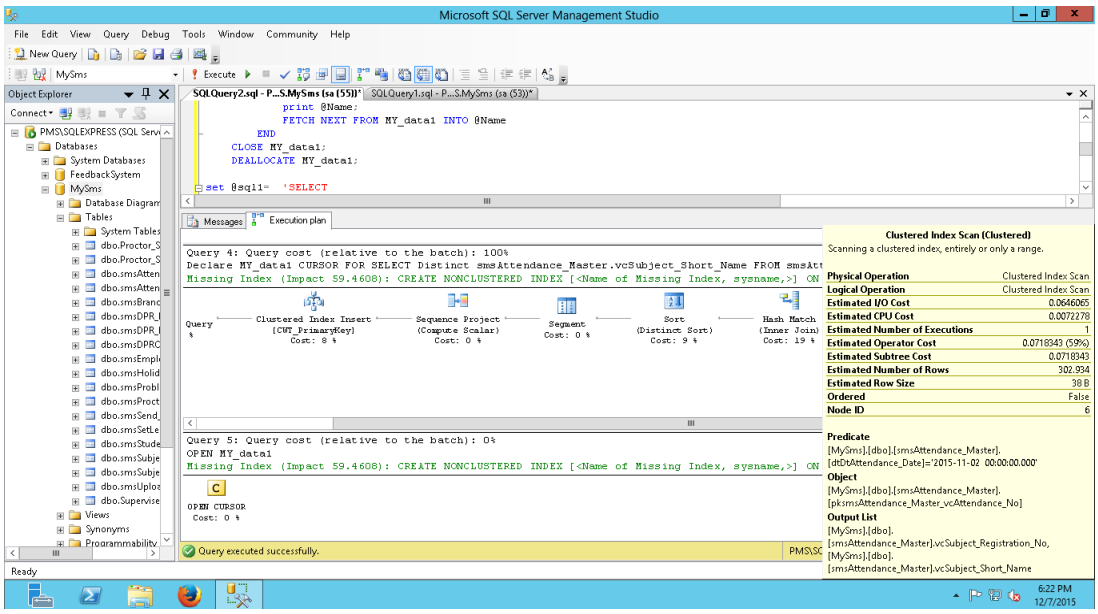
IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015



The graphical representation of query execution plan of the CASE expression (query-02) is as follows:



4. CONCLUSION

To represent data in cross tab report it is suggested to use CASE expression in place of PIVOT operator. The difficulties like using different group functions, getting row wise sum, NULL value when database schema changes and specifying explicit column name while grouping can be overcome using CASE expression. It has been observed that the performance of CASE expression is better than PIVOT operator. Using tuning the experimented queries can be analyzed and the performance of the execution shall be kept for further study and research.

REFERENCES

[1] "Pivoting approaches for bulk extraction of Entity–Attribute–Value data" by Valentin Dinu, Prakash Nadkarni, Cynthia Brandt, Computer Methods and Programs in Biomedicine 82(2006), PP 38–43.

[2] "PIVOT: An adaptive information discovery framework for computational grids" by Guiyi Wei, Yun Ling, Athanasios V. Vasilakos, Bin Xiao, Yao Zheng Information Sciences 180 (2010) PP 4543–4556.

[3] "Using grouping variables to express complex decision support queries" by Damianos Chatziantoniou Data & Knowledge Engineering 61 (2007) PP 114–136.

[4] "A system for query, analysis, and visualization of multi-dimensional relational databases" by C. Stolte, P. Hanrahan, Polaris Proceedings of the IEEE Information Visualization Symposium, 2000.

[5] "A data stream language and system designed for power and extensibility" by Bai, Y., Thakkar, H., Wang, H., Luo, C., Zaniolo, C., ACM Conference on Information and Knowledge Management (CIKM) 2006. , PP. 337–346.

[6] "RFID data processing with a data stream query language" by Bai, Y., Wang, F., Liu, P., Zaniolo, C., Liu, S., IEEE International Conference on Data Engineering (ICDE) 2007. PP 1184–1193.

[7] https://docs.oracle.com/cd/B28359_01/server.111/b28286/expressions004. htm

[8] "Introducing Microsoft SQL Server 2008 r2" by Ross Mistry and Stacia Misner.

[9] "Microsoft SQL Server 2008 Reporting Service" by Stacia Misner.