

References

- [1] G. Ramakrishnan. Database Management Systems, Third Edition. McGraw-Hill, 2003
- [2] Fox, B. 2011. “Leveraging Big Data for Big Impact”, Health Management Technology, <http://www.healthmgttech.com/>.
- [3] A. Hameurlain, “Evolution of Query Optimization Methods: From Centralized Database Systems to Data Grid Systems”, Proceedings of the 20th International Conference on Database and Expert Systems Applications.
- [4]. Andrew N.K. Chen. Robust optimization for performance tuning of modern database Systems, European Journal of Operational Research. 171, 412--429 (2006)
- [5] The State of the Art in Distributed Query Processing DONALD KOSSMANN, University of Passau, ACM Computing Surveys, Vol. 32, No. 4, December 2000.
- [6] Jacobs, A. 2009. “Pathologies of Big Data”, Communications of the ACM, 52(8):36-44.
- [7] JASON. 2008. “Data Analysis Challenges”, The Mitre Corporation, McLean, VA, JSR-08-142
- [8] Kaisler, S. 2012. “Advanced Analytics”, CATALYST Technical Report, i_SW Corporation, Arlington, VA
- [9]. Rasha Osman, Irfan Awan, Michael E. et al.: QuePED-Revisiting queuing networks for the Performance evaluation of database designs. Simulation Modeling Practice and Theory, 19, 251--270 (2011)
- [10]. Balsamo, S., A. Di Marco, P. Inverardi and M. Simeoni, Model-based performance Prediction in software development: a survey, IEEE Transactions on Software Engineering. 30 ,5, 295--310 (2004)
- [11] K. Ono and G.M.Lohman. Measuring the complexity of join enumeration in query optimization. In D.McLeod, R Sacks-Davis, and J.-J. schek, editors,16th International Conference on Very Large Data Bases, August 13-16,1990, Brisbane, Queensland, Australia, Proceedings, pages 314-325. Morgan Kaufmann, 1990.
- [12] A. Hameurlain, “Evolution of Query Optimization Methods: From Centralized Database Systems to Data Grid Systems”, Proceedings of the 20th International Conference on Database and Expert Systems Applications.

- [13] Fox, B. 2011. "Leveraging Big Data for Big Impact", Health Management Technology, <http://www.healthmgttech.com/>.
- [14] https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html.
- [15] Gantz, J. and E. Reinsel. 2011. "Extracting Value from Chaos", IDC's Digital Universe Study, sponsored by EMC.
- [16] <https://cs.uwaterloo.ca/~kmsalem/courses/.../Chalamalla-HadoopDB.pdf>
- [17] <https://en.wikipedia.org/wiki/Greenplum>
- [18] <https://pig.apache.org/>
- [19] www.cs.rutgers.edu/~zz124/cs671.../srikanth_mapreducemerge.pdf. Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters. Hung-chih Yang, Ali Dasdan. Yahoo! Ruey-Lung Hsiao, D. Sto Parker.
- [20] <http://www.journalofcloudcomputing.com/content/3/1/12>. Improving the performance of Hadoop Hive by sharing scan and computation tasks Tansel Dokero glu1, Serkan Ozal1, Murat Ali BayMuhammetSerkanCinar3 and Ahmet Cosar1.
- [21] Liu et al. "An Investigation of Practical Approximate Nearest Neighbor Algorithms", 2004. Carnegie-Mellon University, pp. 1-8.
- [22] www.elsevier.com/locate/jcss, Journal of Computer and System Sciences 77 (2011) 637-651.
- [23] Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis, IJCAI-07 1606, Evgeniy Gabrilovich and Shaul Markovitch Department of Computer Science Technion—Israel Institute of Technology, 32000 Haifa, Israel
- [24] {gabr,shaulm}@cs.technion.ac.il.
- [25] <https://en.wikipedia.org/wiki/MapReduce>.
- [26] Applied Spatial Data Analysis with R Authors: Roger S. Bivand, Edzer Pebesma, Virgilio Gómez-Rubio.
- [27] A twelve-analyzer detector system for high-resolution powder diffraction P. L. Lee, D. Shu, M. Ramanathan, C. Preissner, J. Wang, M. A. Beno, R. B. Von Dreele, L. Ribaud, C. Kurtz, S. M. Antao, X. Jiao and B. H. Toby. J. Synchrotron Rad. (2008). 15, 427-432.

Appendices

Appendix A - User interface and architecture diagram of the system.

Security Module - Authentication

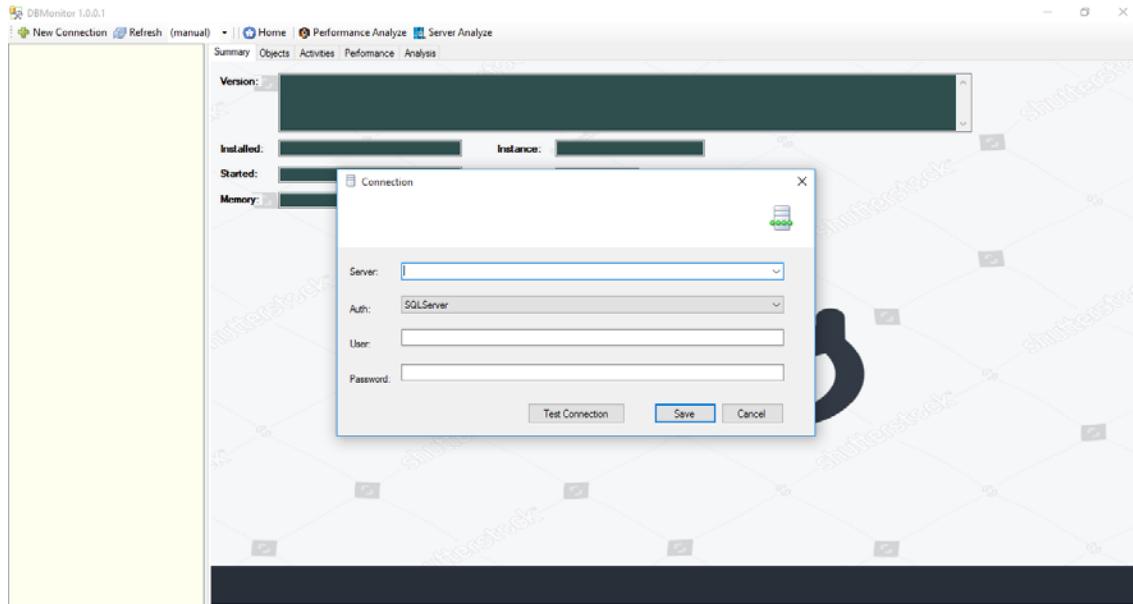


Figure 5.1 – Security Module - Authentication

Control Module – Server and Database Information

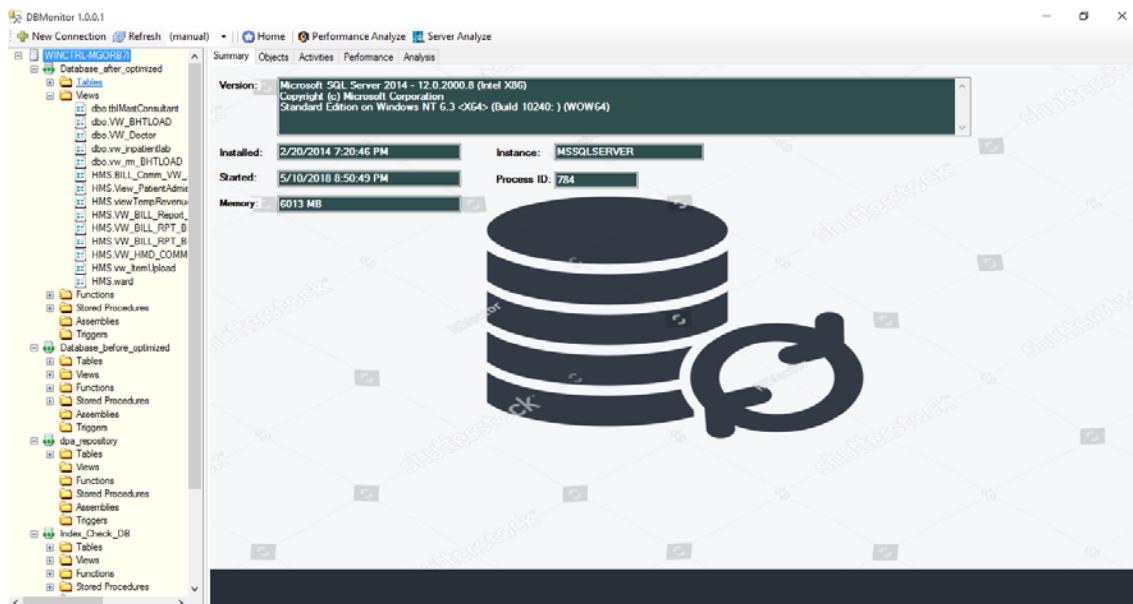


Figure 5.2 – Control Module-Server and Database Information

Server Configuration

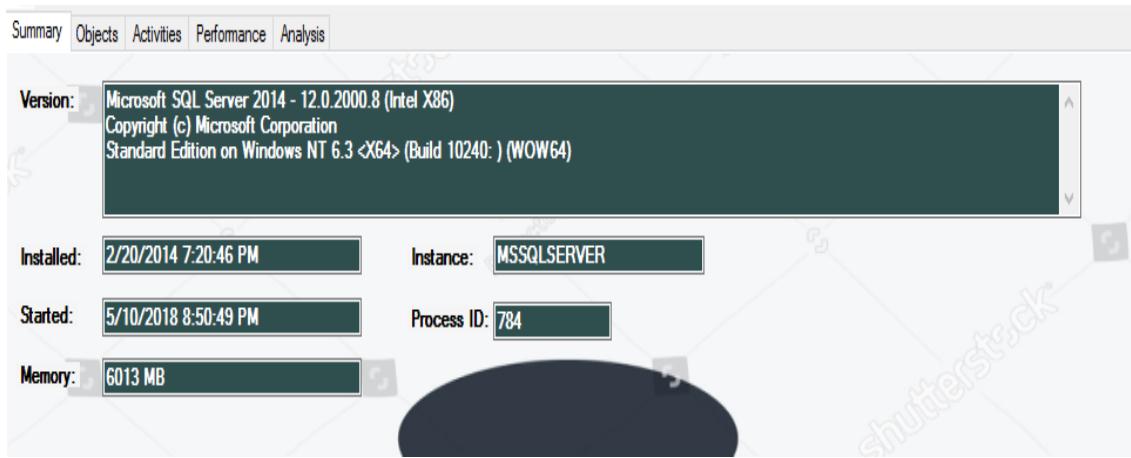


Figure 5.3 – Server Configuration

Database Server Performance Analyzer

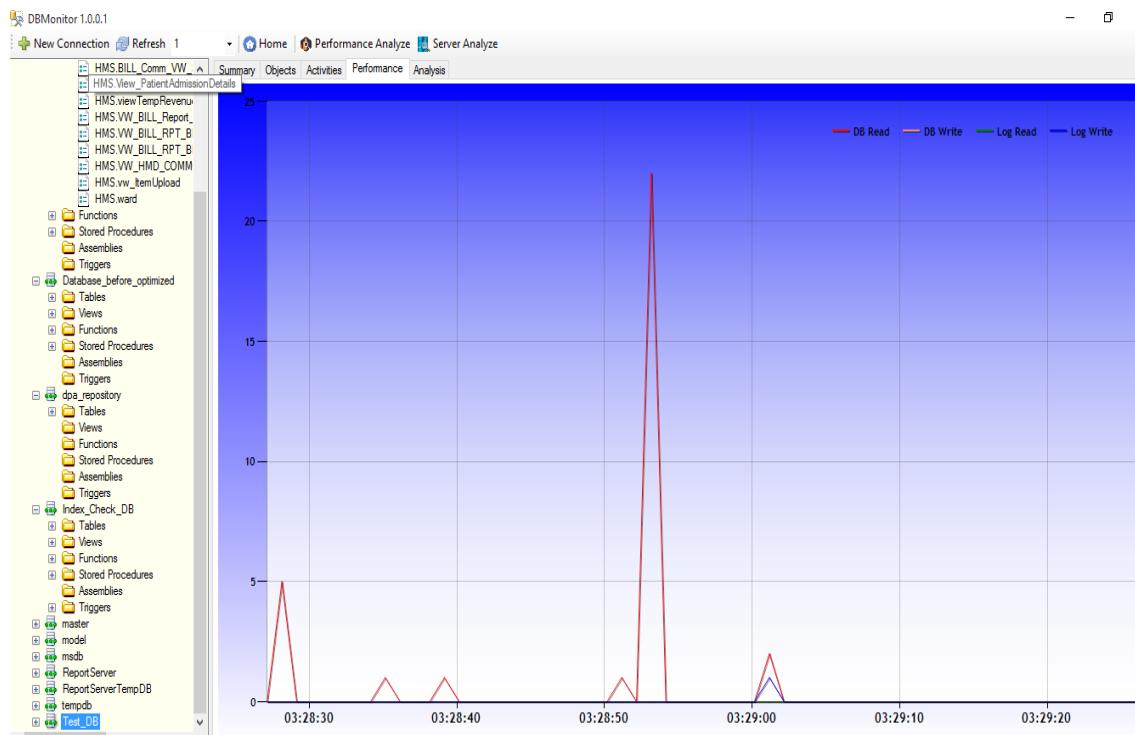


Figure 5.4 – Database Server Performance Analyzer

Database Log Information and Suggestions

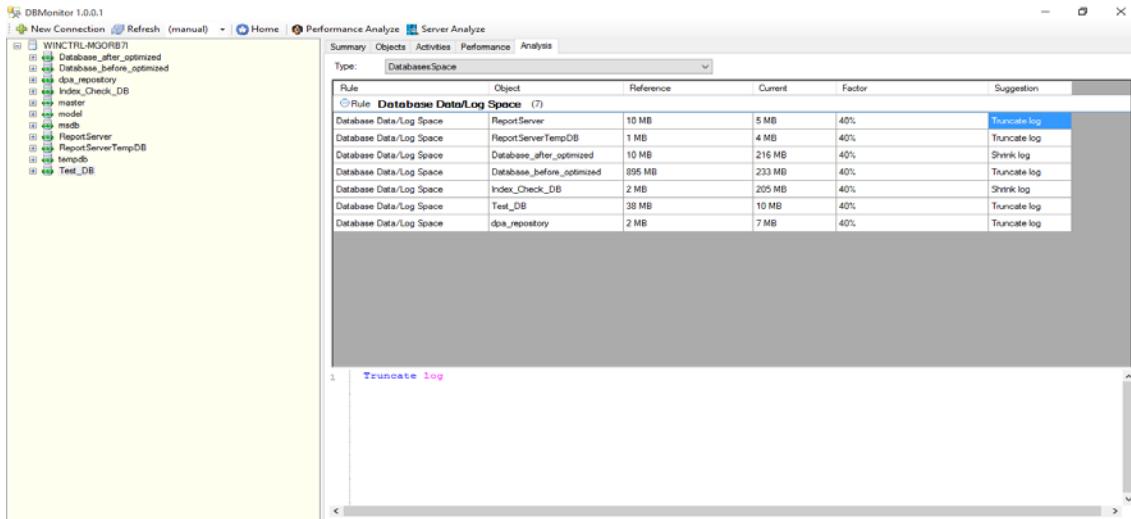


Figure 5.5 – Database Log Information and Suggestions.

Database Performance Improvement Suggestions

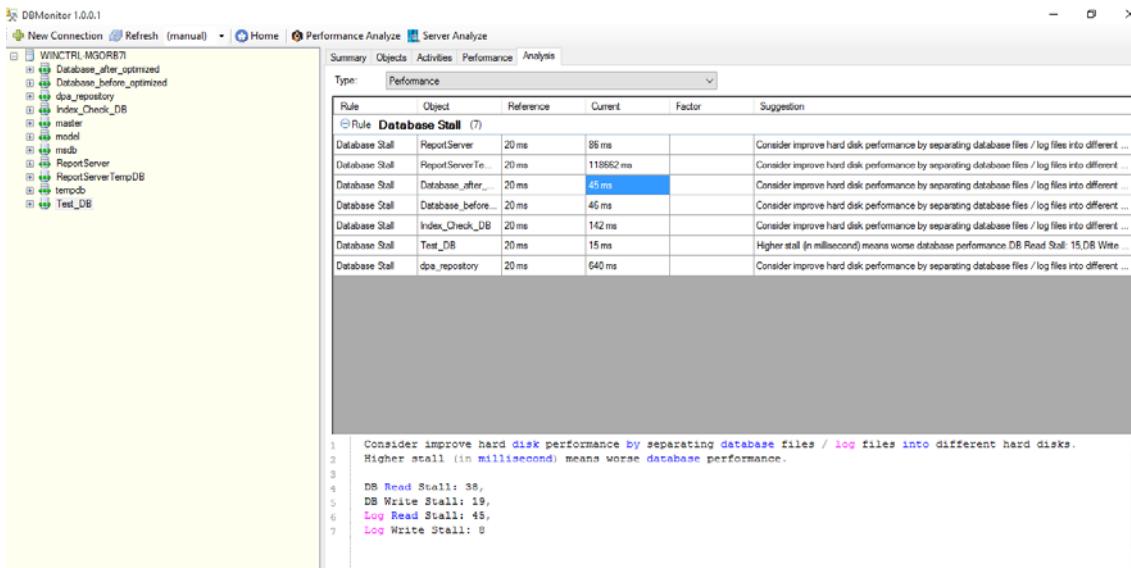


Figure 5.6 – Database Performance Improvement Suggestions.

Database Waiting Tasks

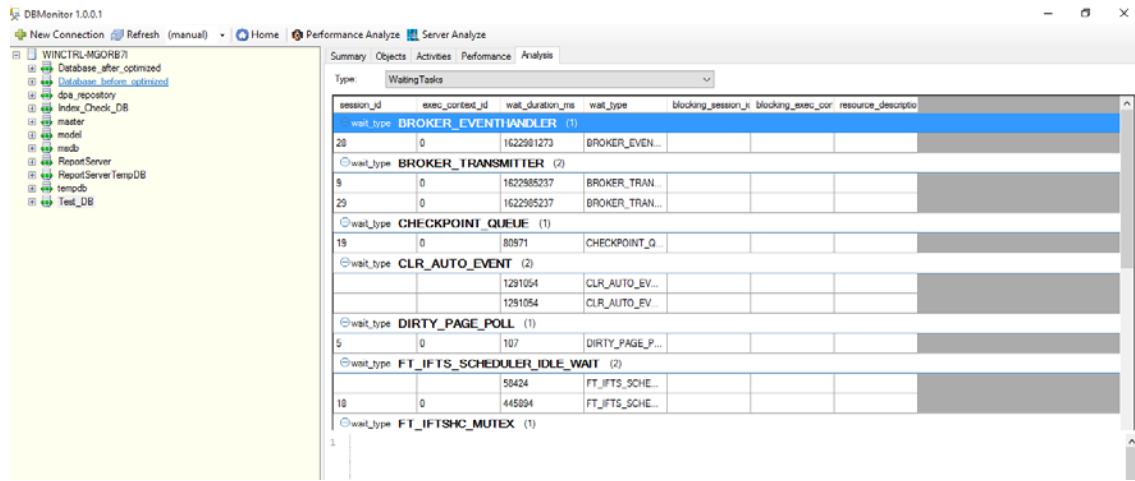


Figure 5.7 – Database Waiting Tasks.

Database Missing Index Details and Suggestions

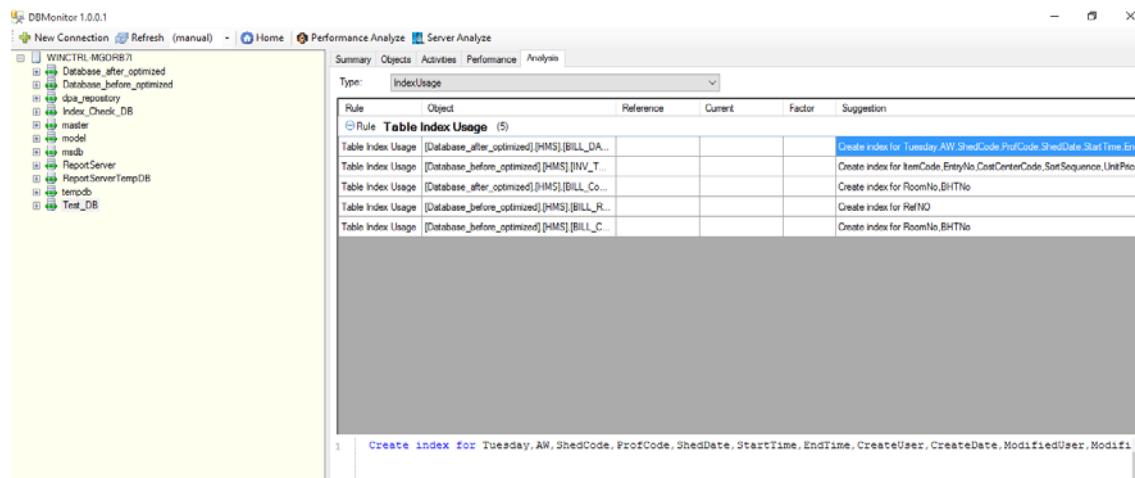


Figure 5.8 – Database Missing Index Details and Suggestions

Database IO operations

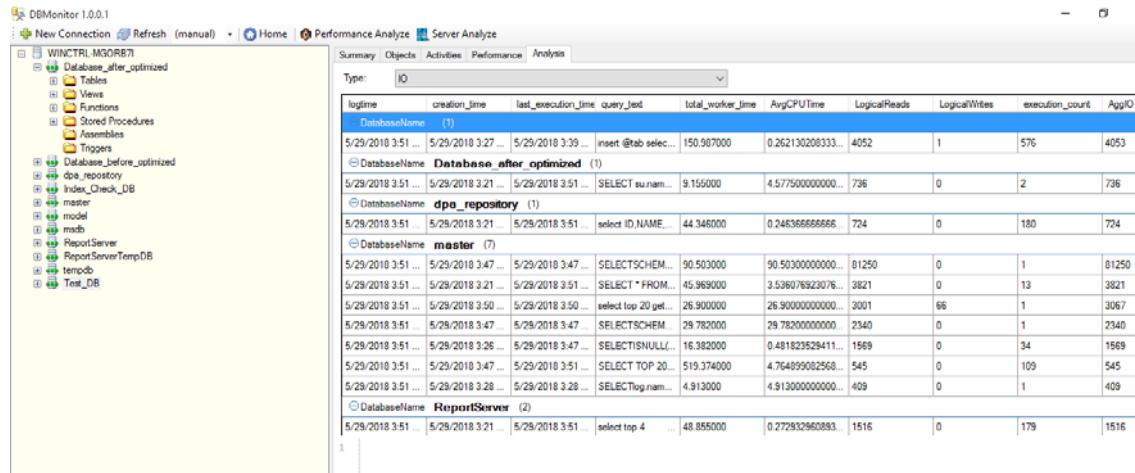


Figure 5.9 – Database IO Operations

Database Objects and Details

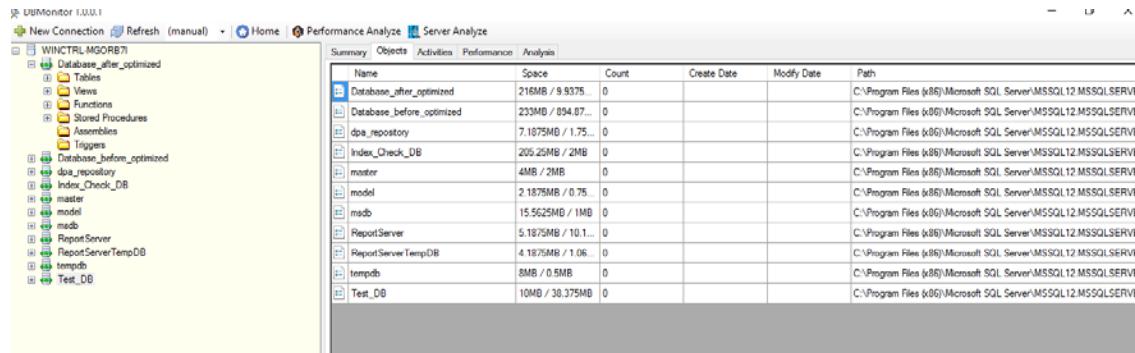


Figure 5.10 – Database Objects and Details

Database Monitoring Application Options

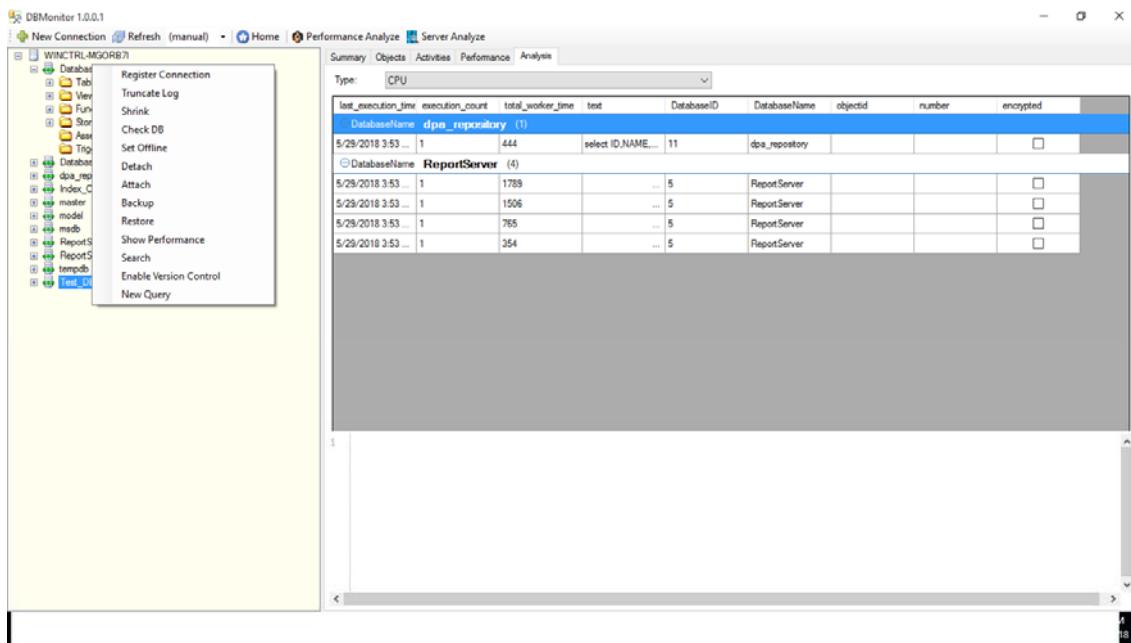


Figure 5.11 – Database Monitoring Application Options

Appendix B – Evaluation of Database Monitoring Application

Functionality	Result in our approach	The result from manually check in the database	Are the both results are same?
Server information	Please refer figure 7.4.1.1	Please refer figure 7.4.1.2	Yes
Database Table Count	Please refer figure 7.4.1.3	Please refer figure 7.4.1.4	Yes
Database current reading count	Please refer figure 7.4.1.3	Please refer figure 7.4.1.4	Yes
Database current writing count	Please refer figure 7.4.1.3	Please refer figure 7.4.1.4	Yes
Missing index details	Please refer figure 7.4.1.5	Please refer figure 7.4.1.6	Yes
Database current reading count	Please refer figure 7.4.1.3	Please refer figure 7.4.1.4	Yes
Database memory utilization details	Please refer figure 7.4.1.7	Manually checked	Yes
Database lock	Please refer figure 7.4.1.8	Manually checked	Yes
Currently running Processors	Please refer figure 7.4.1.9	Manually checked	Yes

Table 7.1 – Evaluation functionality in database monitoring application

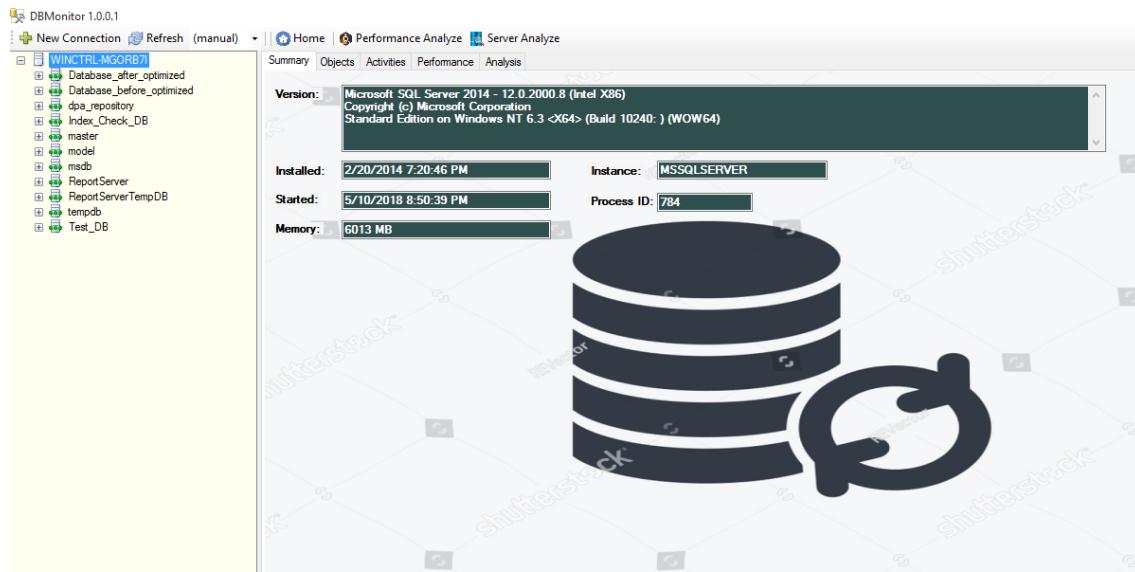


Figure 7.4.1.1 – Database server information from newly developed database monitoring application

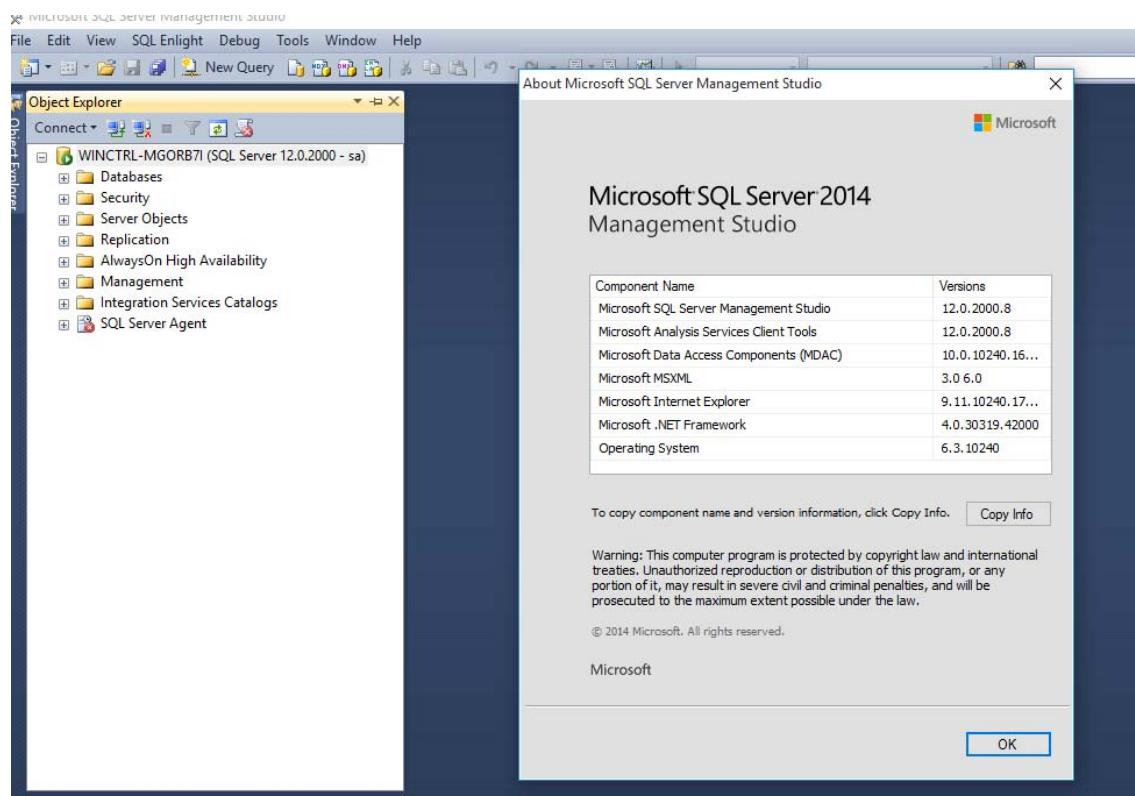




Figure 7.4.1.3 - Database server statistics from newly developed database monitoring application

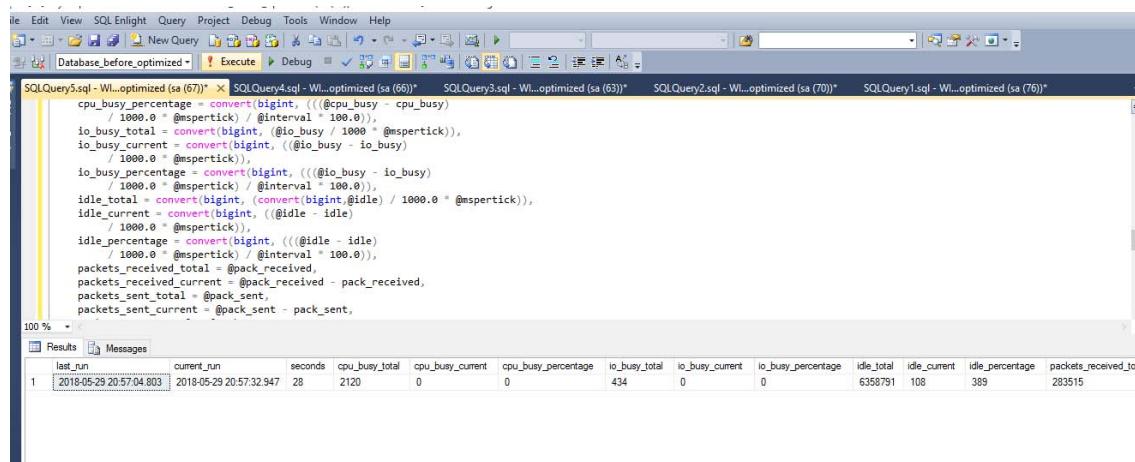


Figure 7.4.1.4 - Database server statistics

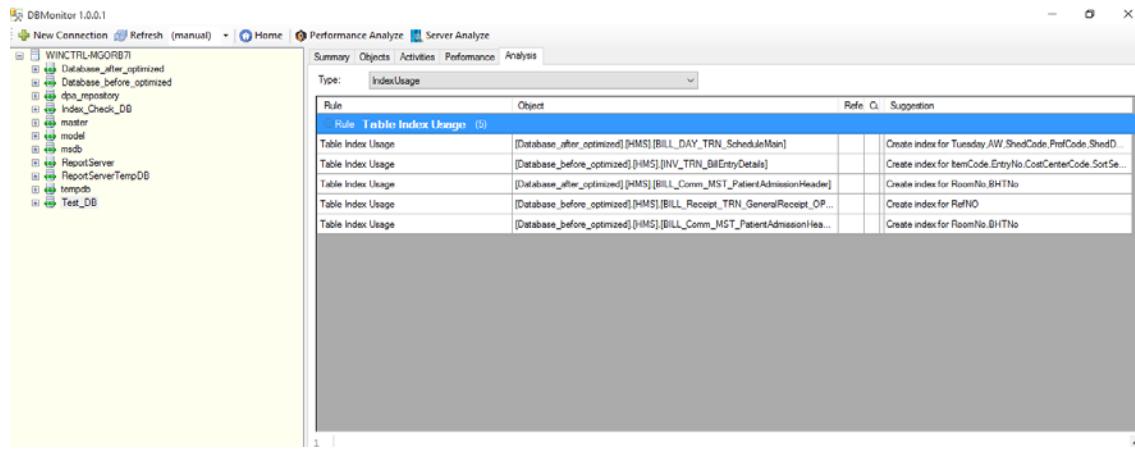


Figure 7.4.1.5 – Missing index suggestions from newly developed database monitoring application

The screenshot shows Microsoft SQL Server Management Studio (SSMS) with a query window titled 'SQLQuery2.sql - WINCTRL-MGORB71.Database_before_optimized (sa (70))'. The query is:

```

SELECT D.statement AS Objectname, column_name, column_usage
FROM sys.dm_db_missing_index_groups G
JOIN sys.dm_db_missing_index_group_stats GS ON G.index_group_handle = GS.group_handle
JOIN sys.dm_db_missing_index_details D ON G.index_handle = D.index_handle
CROSS APPLY sys.dm_db_missing_index_columns (D.index_handle) DC
WHERE column_usage='EQUALITY'
ORDER BY D.index_handle, D.statement
    
```

The results grid shows the following data:

ObjectName	column_name	column_usage
1 [Database_after_optimized] [HMS] [BILL_DAY_TRN_ScheduleMain]	Tuesday	EQUALITY
2 [Database_after_optimized] [HMS] [BILL_DAY_TRN_ScheduleMain]	AW	EQUALITY
3 [Database_before_optimized] [HMS] [INV_TRN_BillEntryDetail]	ItemCode	EQUALITY
4 [Database_after_optimized] [HMS] [BILL_Comm_MST_PatientAdmissionHeader]	RoomNo	EQUALITY
5 [Database_before_optimized] [HMS] [BILL_Receipt_TRN_GeneralReceipt_OPDProfCharge]	RefNO	EQUALITY
6 [Database_before_optimized] [HMS] [BILL_Comm_MST_PatientAdmissionHeader]	RoomNo	EQUALITY

At the bottom, a message bar indicates: 'Query executed successfully.' and 'WINCTRL-MGORB71 (12.0 RTM) sa (70) Database_before_optimized 00:00:00 6 rows'.

Figure 7.4.1.6 – Missing index suggestions by manually

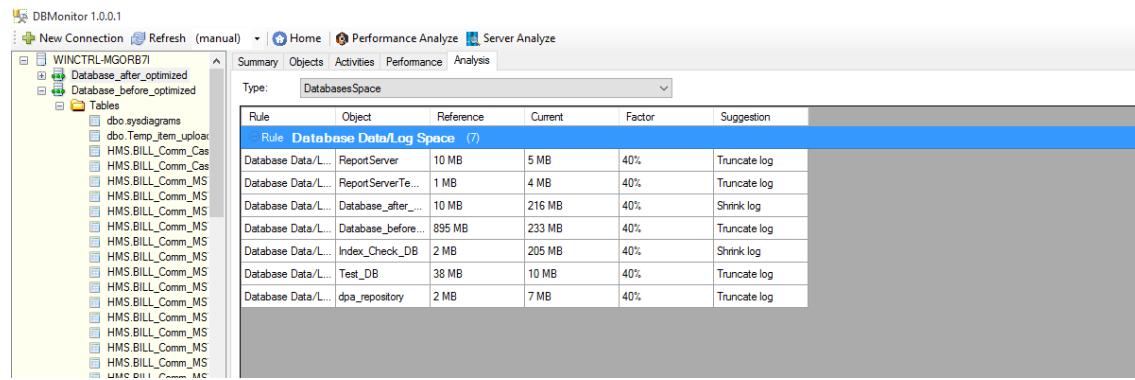


Figure 7.4.1.7 – Database memory utilization details

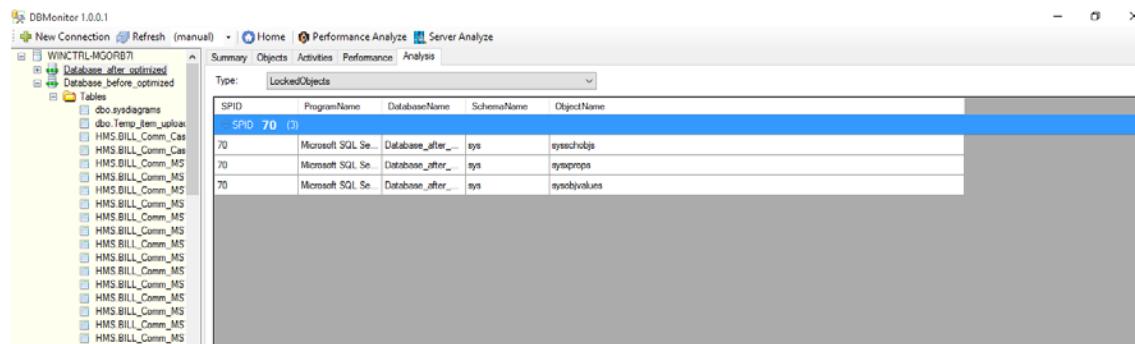


Figure 7.4.1.7 – Database lock

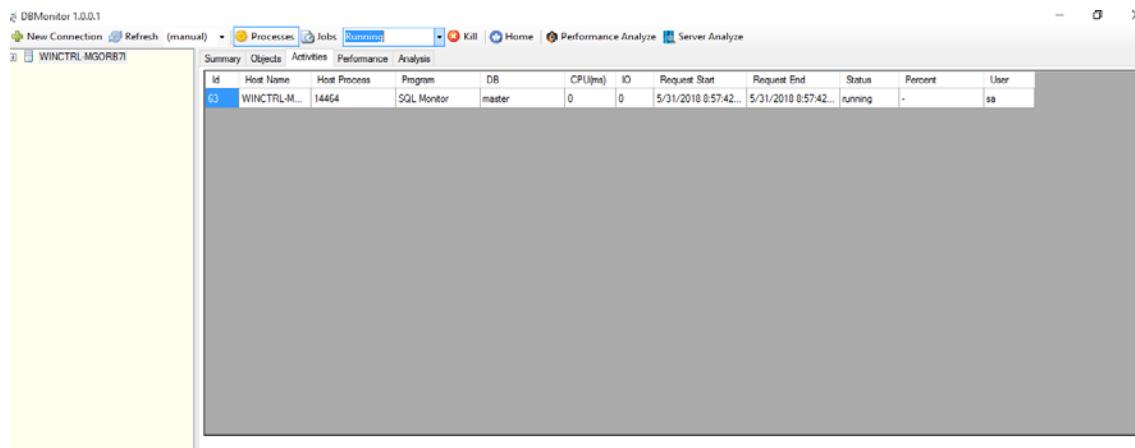


Figure 7.4.1.8 – Currently running Processors

Appendix C – Evaluation of proposed optimization techniques

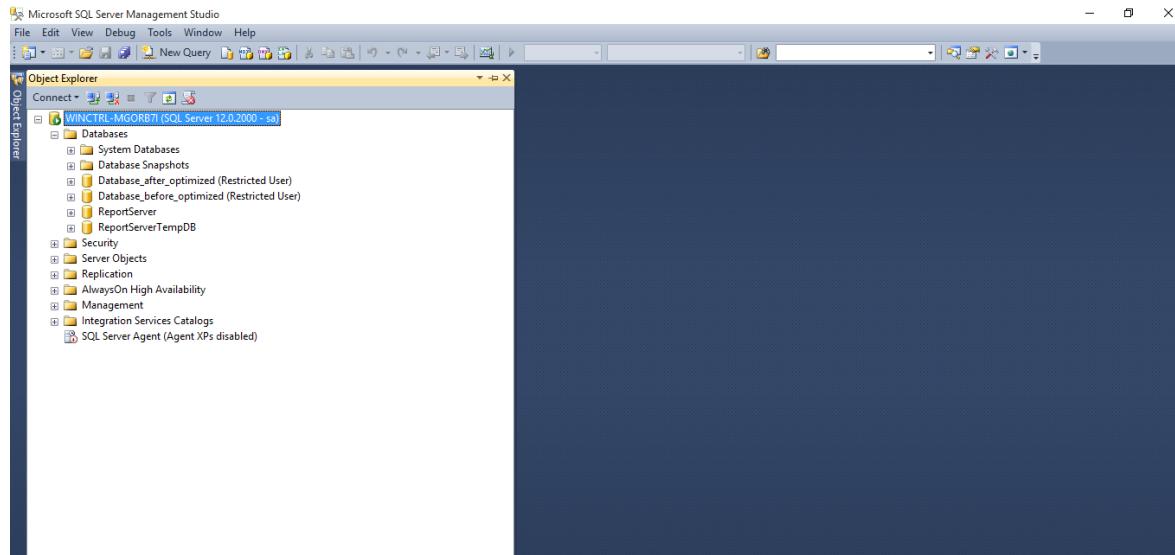


Figure 6.1 – Database Configuration

The screenshot shows the Microsoft SQL Server Management Studio interface with the title bar "01-Script-Index.sql - WINCTRL-MGORB7I.Database_before_optimized (sa (61)) - Microsoft SQL Server Management Studio". The Query Editor window contains a complex SQL query with numerous joins and conditions. Below the query is the "Results" tab showing the execution results. The results table has columns: TmTypeCode, ReceiptNo, BHTNo, ReferenceNo, ReceiptAmount, PaidAmount, MachineCode, MachineBillNo, AdvanceReceiptType, CreateUser, and CreateDate. The data shows 10 rows of records. A yellow circle highlights the last row in the results table. At the bottom of the screen, a status bar displays "Query executed successfully.", the connection information "WINCTRL-MGORB7I (12.0 RTM) | sa (61) | Database_before_optimized", the execution time "00:00:03", and the number of rows "3781 rows".

Figure 6.2 – Complex Query Execution Time

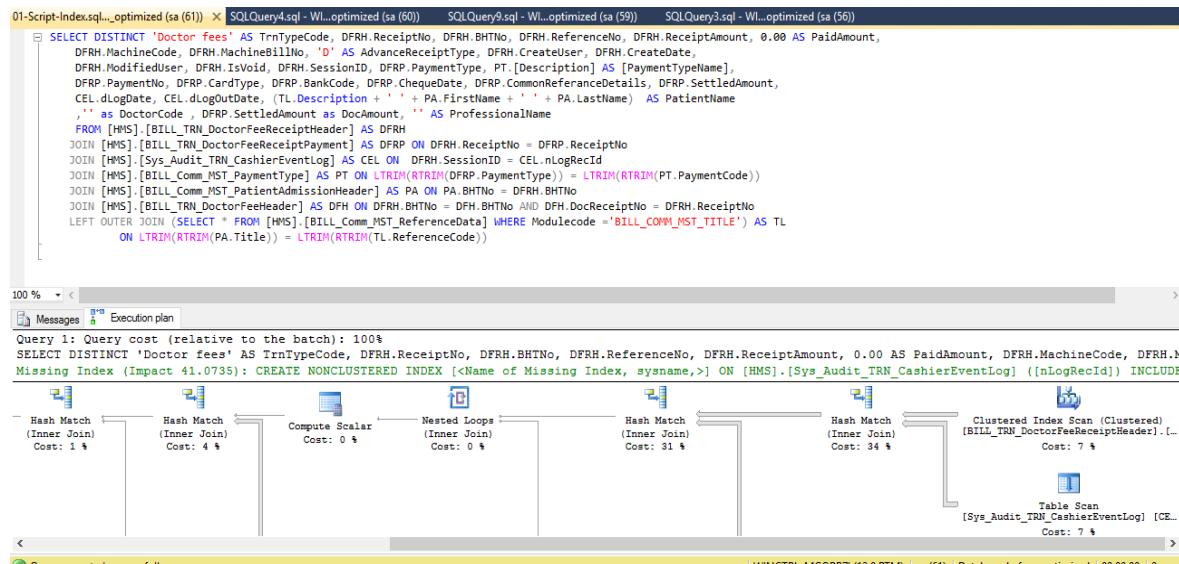


Figure 6.3 - QEP Plan

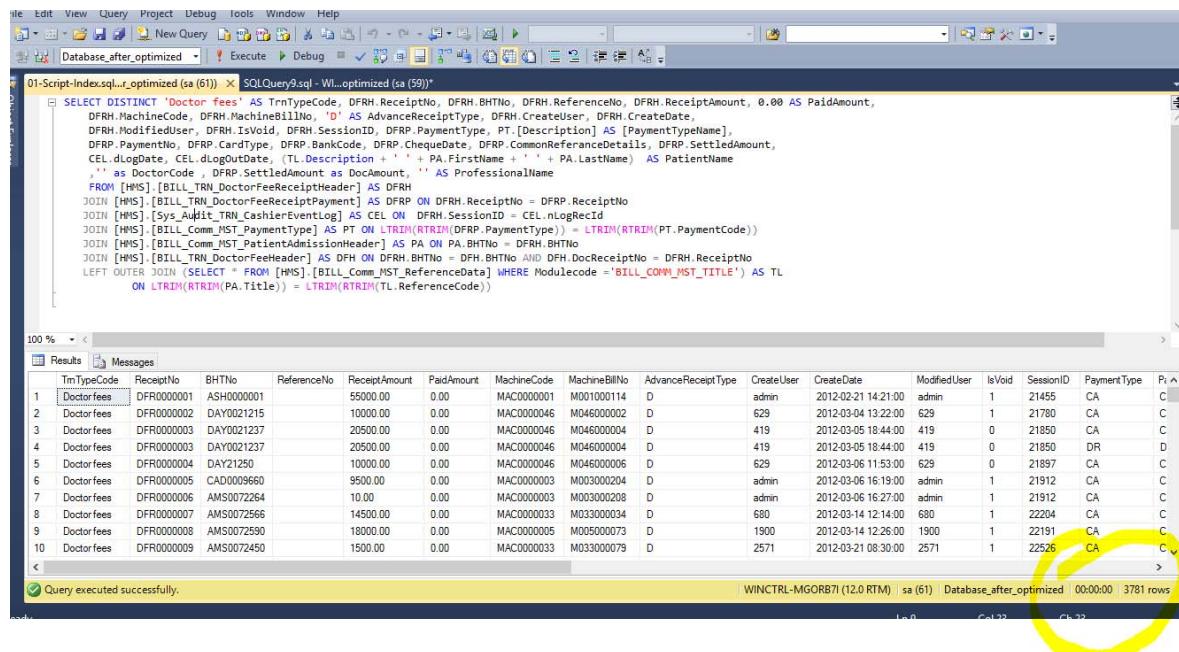


Figure 6.4 – Query Execution Time After Optimized

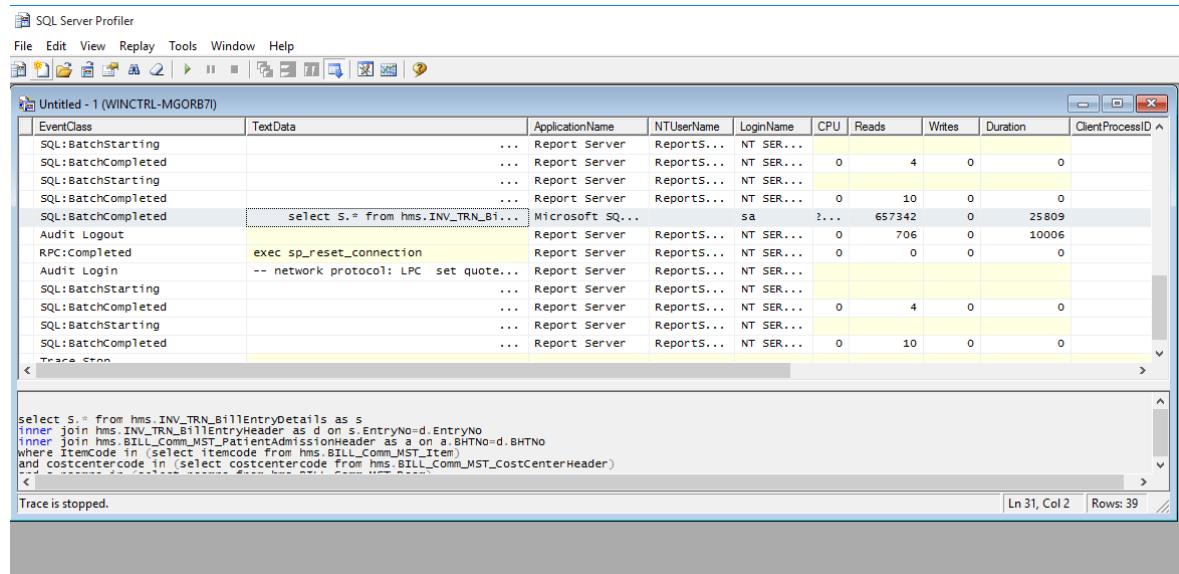


Figure 6.5 – SQL Profiler

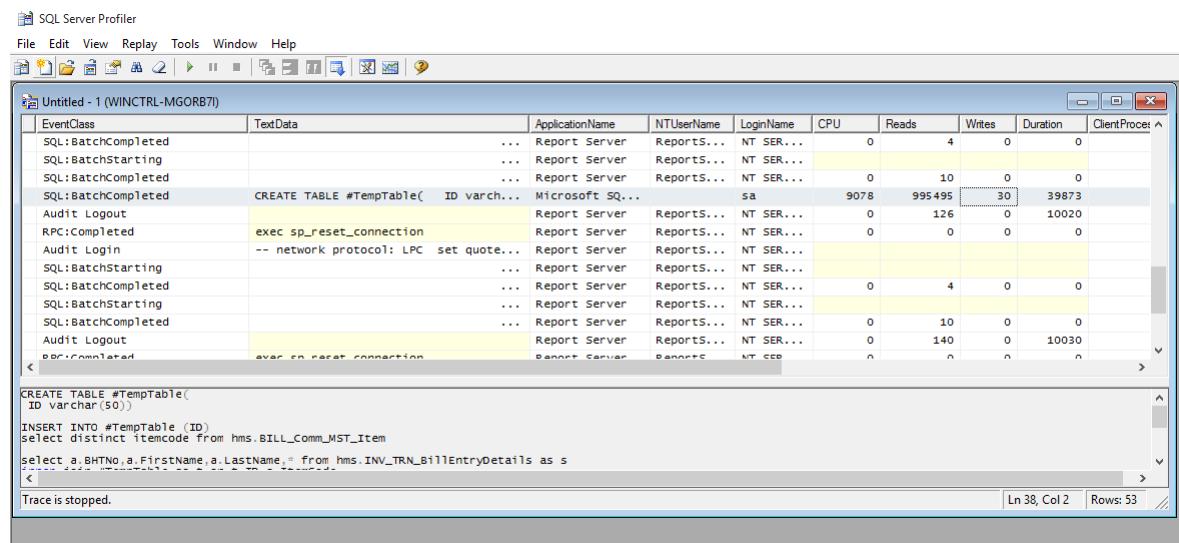


Figure 6.6 – SQL Profiler result

The screenshot shows a SQL Server Management Studio window with two tabs: 'Results' and 'Messages'. The 'Results' tab displays the following output:

```
set statistics time on

SELECT
    s.*
FROM hms.INV_TRN_BillEntryDetails AS s
INNER JOIN hms.INV_TRN_BillEntryHeader AS d
    ON s.EntryNo = d.EntryNo
INNER JOIN hms.BILL_Comm_MST_PatientAdmissionHeader AS a
    ON a.BHTNo = d.BHTNo
WHERE ItemCode IN (SELECT DISTINCT
    itemcode
    FROM hms.BILL_Comm_MST_Item)
    AND costcentercode IN (SELECT
        costcentercode
        FROM hms.BILL_Comm_MST_CostCenterHeader)
    AND a.roomno IN (SELECT
        roomno
        FROM hms.BILL_Comm_MST_RoomHeader)
        WHERE roomno > 0
        GROUP BY roomno
        HAVING COUNT(roomno) > 1
        ORDER BY roomno

(255180 row(s) affected)

SQL Server Execution Times:
    CPU time = 2906 ms,  elapsed time = 56866 ms.
```

Figure 6.7 – SQL Server Execution time for Traditional query

The screenshot shows a SQL Server Management Studio window with two tabs: 'Results' and 'Messages'. The 'Results' tab displays the following output:

```
itemcode
    FROM hms.BILL_Comm_MST_Item
    CREATE NONCLUSTERED INDEX IX_Itemcode ON #TempTable(ID)

SELECT
    s.*
    FROM hms.INV TRN BillEntryDetails AS s
    WHERE itemcode IN (SELECT itemcode
        FROM hms.BILL_Comm_MST_Item
        WHERE itemcode > 0
        GROUP BY itemcode
        HAVING COUNT(itemcode) > 1
        ORDER BY itemcode)

(255180 row(s) affected)

SQL Server Execution Times:
    CPU time = 4141 ms,  elapsed time = 5446 ms.

SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 0 ms.
```

Figure 6.8 – SQL Server Execution time for our new proposed query

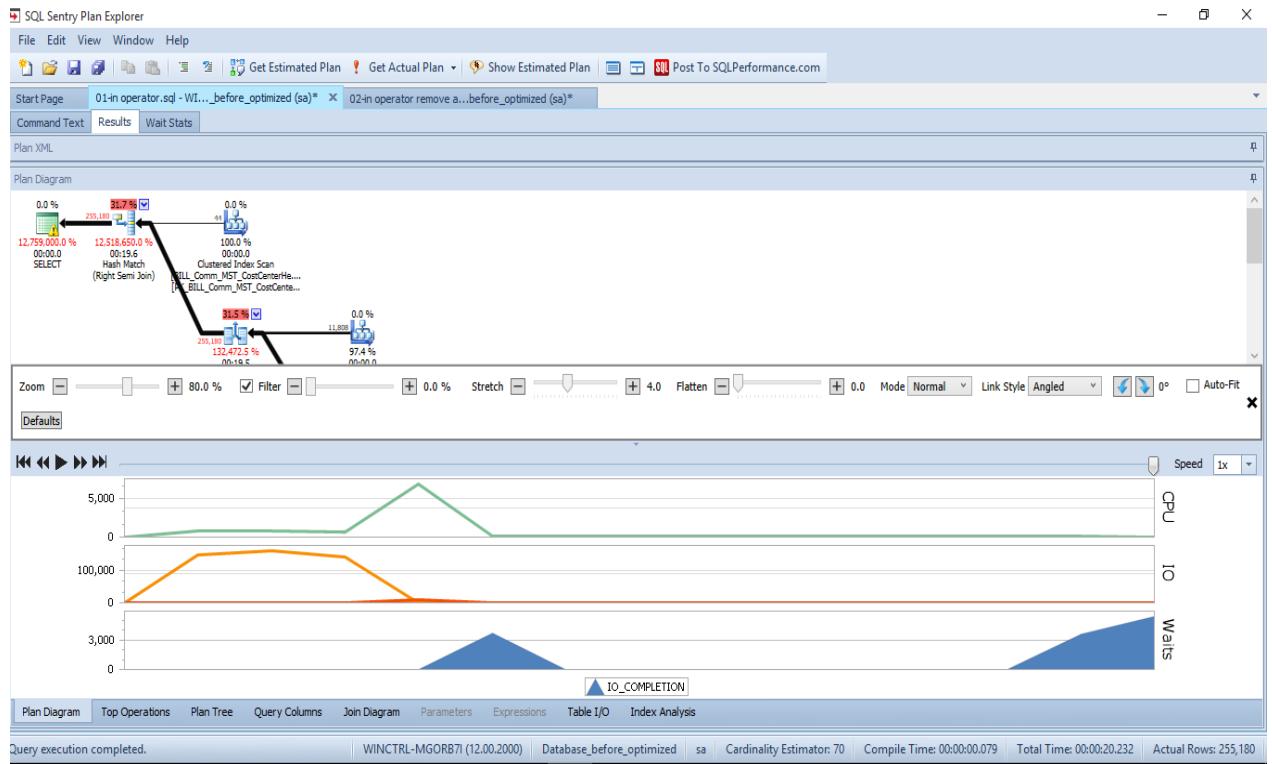


Figure 6.9 - Analyze by using Sentry Plan explore with IN

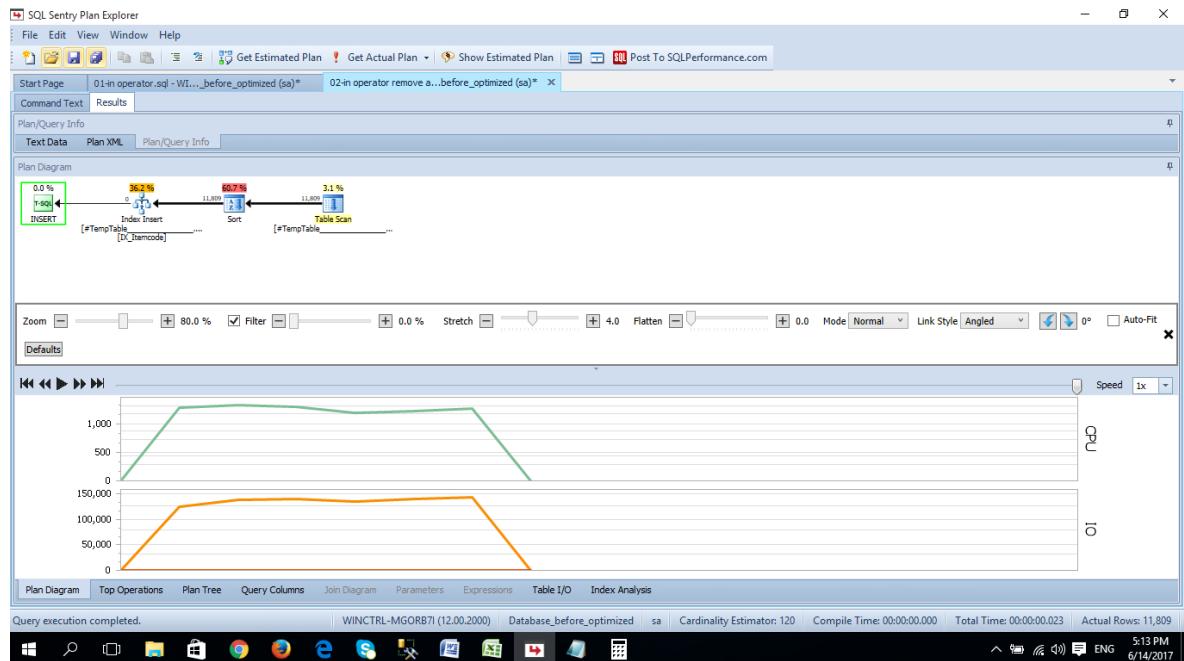


Figure 6.10 - Analyze by using Sentry Plan explore without IN

create it explicitly...ex_Check_DB (sa (5))

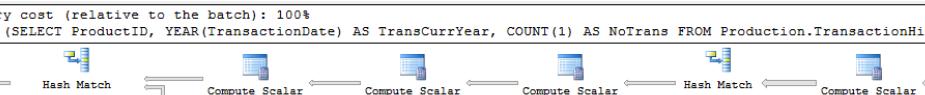
```
FROM BASE Prev2Year  
WHERE CurrYear.ProductID = Prev2Year.ProductID  
AND CurrYear.TransCurrYear = Prev2Year.TransCurrYear - 2) AS Prev2Year  
  
SET STATISTICS time OFF  
  
-- Observations from Query 1:  
100 %  
Messages Execution plan  
  
SET STATS  
Cost: 0 %  
  
Query 2: Query cost (relative to the batch): 100%  
WITH BASE AS (SELECT ProductID, YEAR(TransactionDate) AS TransCurrYear, COUNT(1) AS NoTrans FROM Production.TransactionHistory GROUP...  
  
  
SELECT Cost: 0 % Hash Match (Right Outer Join) Cost: 1 % Compute Scalar Cost: 0 % Compute Scalar Cost: 0 % Compute Scalar Cost: 0 % Hash Match (Aggregate) Cost: 17 % Compute Scalar Cost: 0 % Clus [Trans  
  
Query 3: Query cost (relative to the batch): 0%  
SET STATISTICS time OFF -- Observations from Query 1: --There were over 27 scans with logical reads of 1998 --Although we used a CTE...  
  
SET STATS  
Cost: 0 %
```

Figure 6.11 – Query cost with temp table

```
et $ now use a 1e..._Check_DB (sa (68)) X create it explicitly...ex_Check_DB (sa (68))

set statistics time on
CREATE TABLE #T1
(
    ProductID int
    ,TransCurrYear int
    ,NoTrans int
);
CREATE CLUSTERED INDEX CI_#T1 ON #T1 (TransCurrYear)

TMSCRT TMTD #T1
00 % < >
Messages Execution plan
Query 1: Query cost (relative to the batch): 0%
set statistics time on

T-SQL
SET STATISTICS COST 0
Query 2: Query cost (relative to the batch): 0%
CREATE TABLE #T1 (ProductID int ,TransCurrYear int ,NoTrans int );

T-SQL
CREATE TABLE
Cost: 0 %
Query 3: Query cost (relative to the batch): 0%
CREATE CLUSTERED INDEX CI_#T1 ON #T1 (TransCurrYear)

T-SQL
CREATE INDEX
Cost: 0 %
Query 4: Query cost (relative to the batch): 98%
INSERT INTO #T1 SELECT ProductID, YEAR(TransactionDate) AS TransCurrYear, COUNT(1) AS NoTrans FROM Production.TransactionHistory GRO...
Query executed successfully.
```

Figure 6.12 - Query cost with #temp table

tablevariable.sql -> x_Check_DB (sa (69)) X Let's now use a Te..._Check_DB (sa (68)) create it explicitly...ex_Check_DB (sa (55))

```

set statistics time on
DECLARE @T1 AS TABLE
( ProductID int
, TransCurrYear int
, NoTrans int
)
100 %
Messages Execution plan
Query 1: Query cost (relative to the batch): 0%
set statistics time on
SET STATISTICS TIME ON
Cost: 0 $

Query 2: Query cost (relative to the batch): 99%
DECLARE @T1 AS TABLE (ProductID int ,TransCurrYear int ,NoTrans int , INDEX [IX_TransactionDate] CLUSTERED (ProductID,TransCurrYear)...
```

```

Clustered Index Insert (@T1).[IX_TransactionDate]
Cost: 1 %

Sort Cost: 2 %

Compute Scalar Cost: 0 %

Hash Match (Aggregate) Cost: 51 %

Compute Scalar Cost: 1 %

Clustered Index Scan (Clustered) [TransactionHistory].[PK_TransactionHistory]
Cost: 46 %

< + >
```

```

Query 3: Query cost (relative to the batch): 1%
;With BASE AS ( SELECT * FROM @T1 ) SELECT CurrYear.ProductID, CurrYear.NoTrans AS CurrTransCnt, PrevYear.NoTrans AS PrevTransCnt, P...
```

```

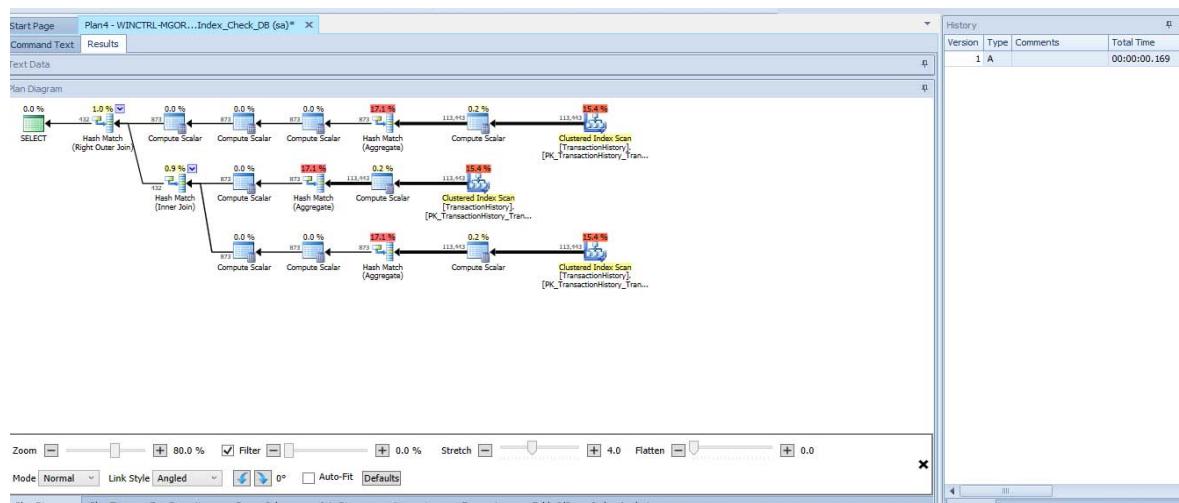
SELECT (Left Outer Join) Cost: 0 $
Nested Loops (Inner Join) Cost: 0 $
Compute Scalar Cost: 0 %
[@T1].[IX_TransactionDate] Cost: 33 %

< + >
```

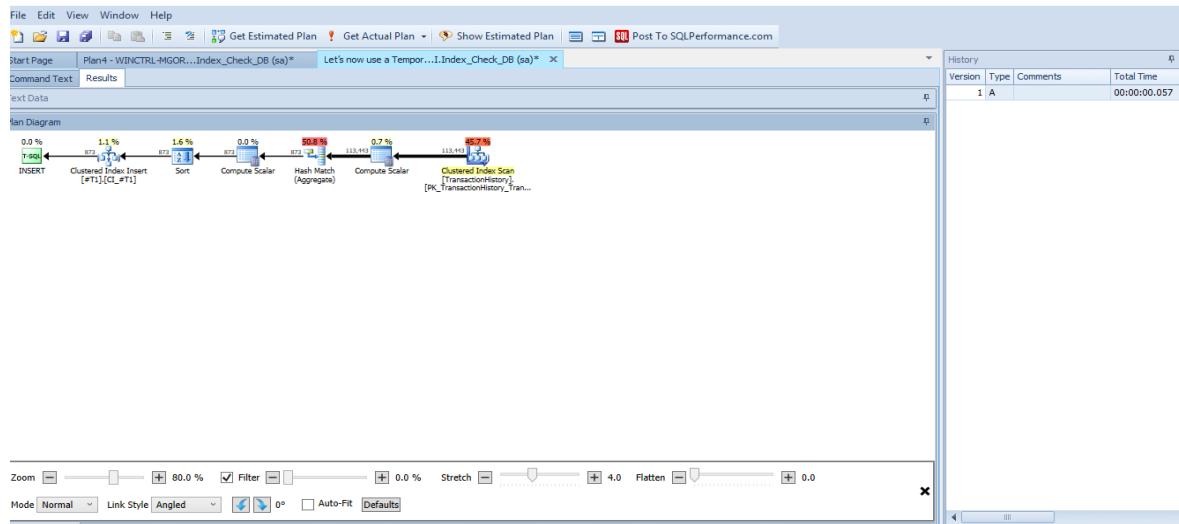
Query executed successfully.

WINCTRL-MGORB7I (12.0 RTM) | sa (69) | Index_Check_DB | 00:00:00 | 0 rows

Figure 6.13 - Query cost with @temp table



6.14 - Sentry plan with #temp table



6.15 - Sentry plan with @temp table

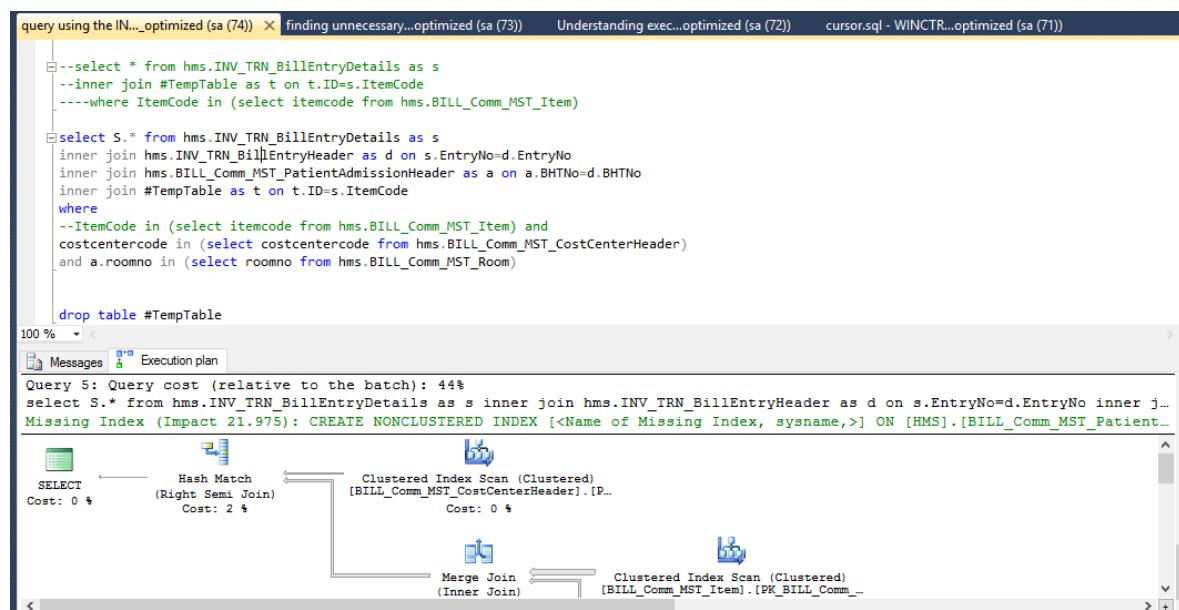


Figure 6.16 – How to find missing index

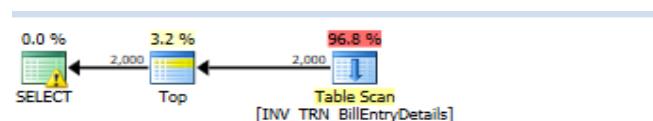


Figure 6.17 – Analyzed best practice IN and Where Clause.

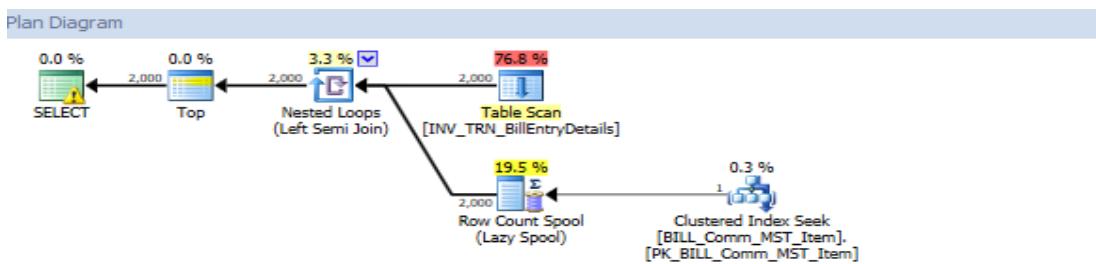


Figure 6.18 – Analyzed bad practice IN and Where Clause

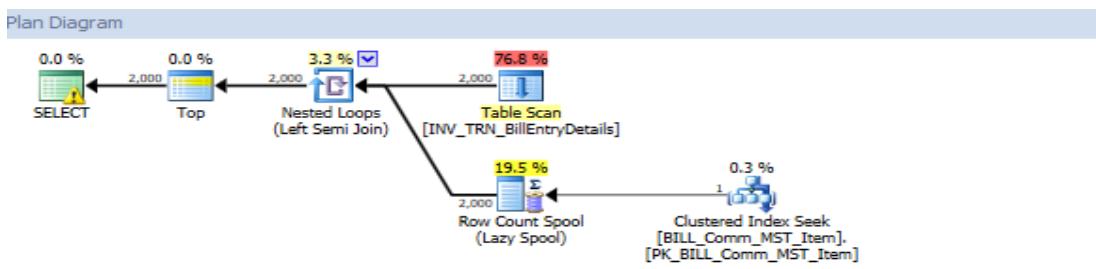


Figure 6.19 – Bad practice for IN and Where

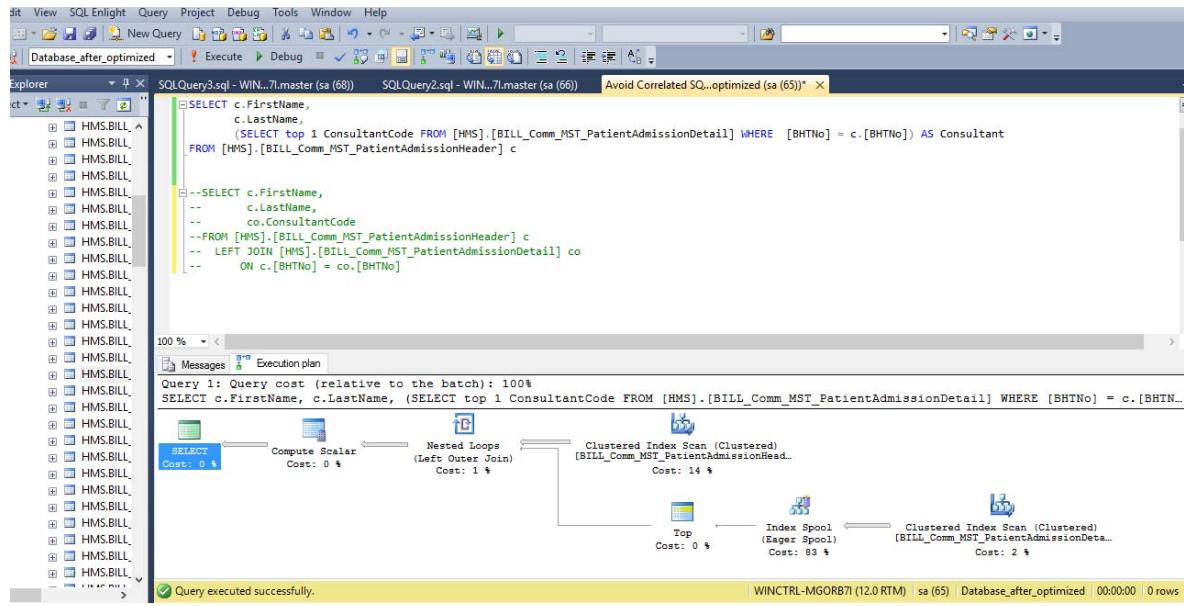


Figure 6.20 – QEP plan and Cost of Correlated SQL subqueries

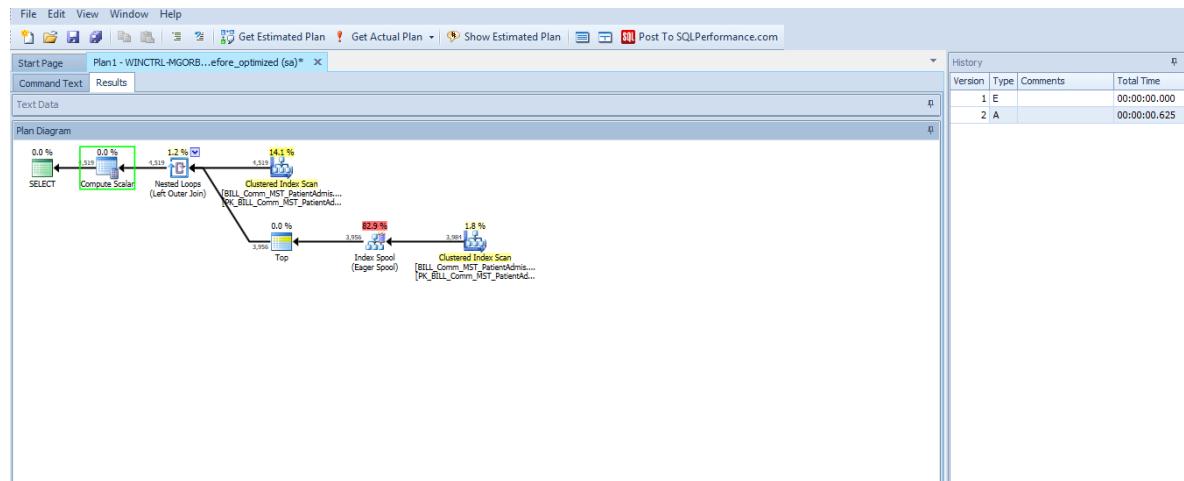


Figure 6.21- QEP plan and Cost of Correlated SQL subqueries in Sentry planner

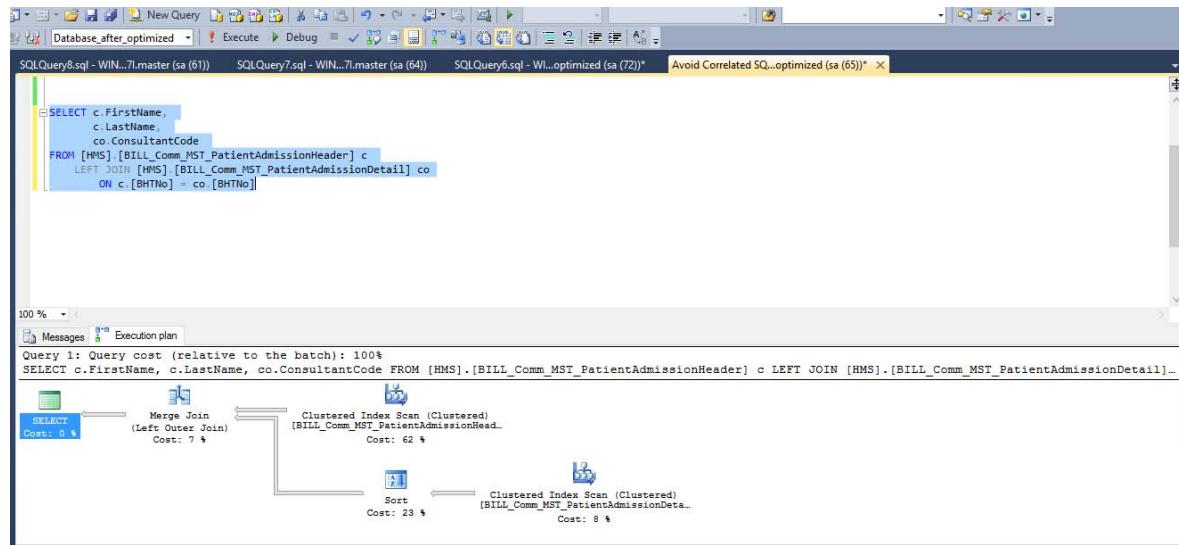


Figure 6.22- Our Query QEP plan and Cost of Correlated SQL subqueries

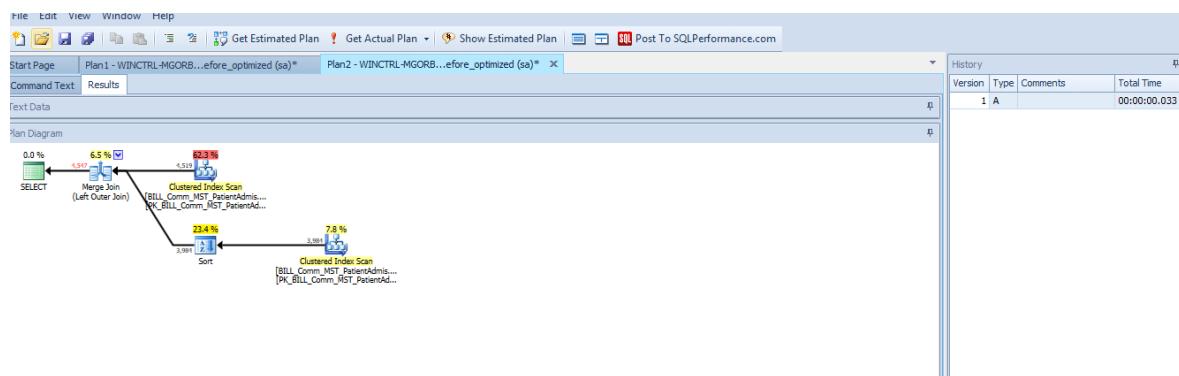


Figure 6.23- Our Query QEP plan and Cost of Correlated SQL subqueries in Sentry planner

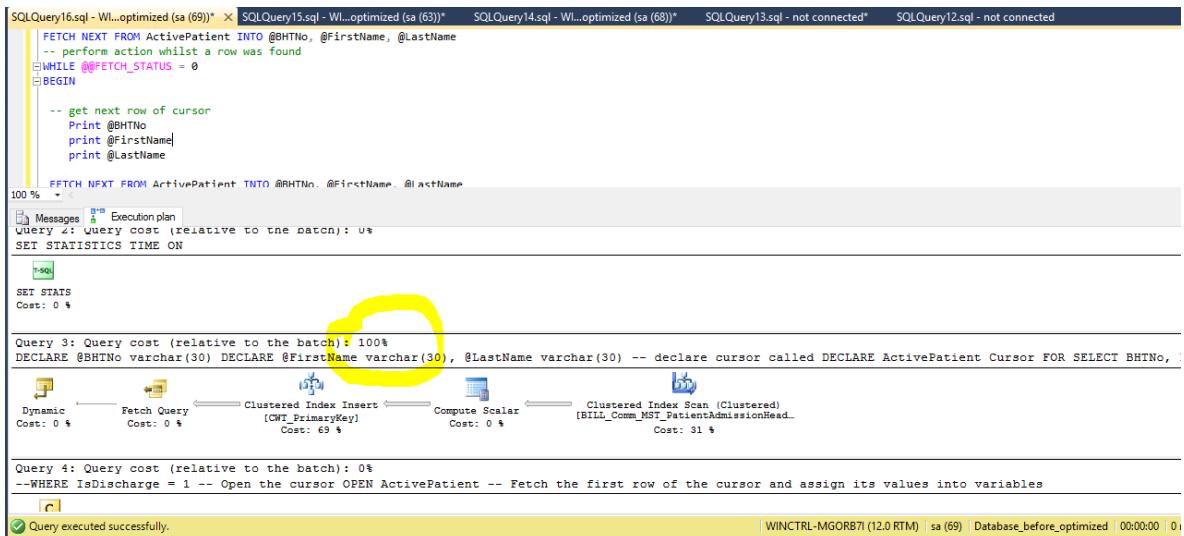


Figure 6.24 – QEP in Cursor

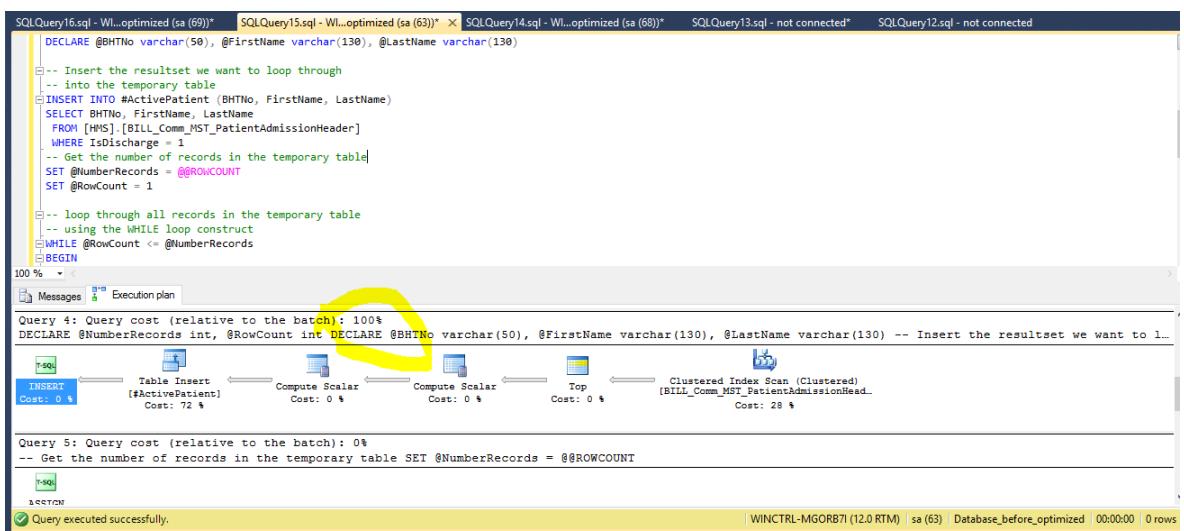
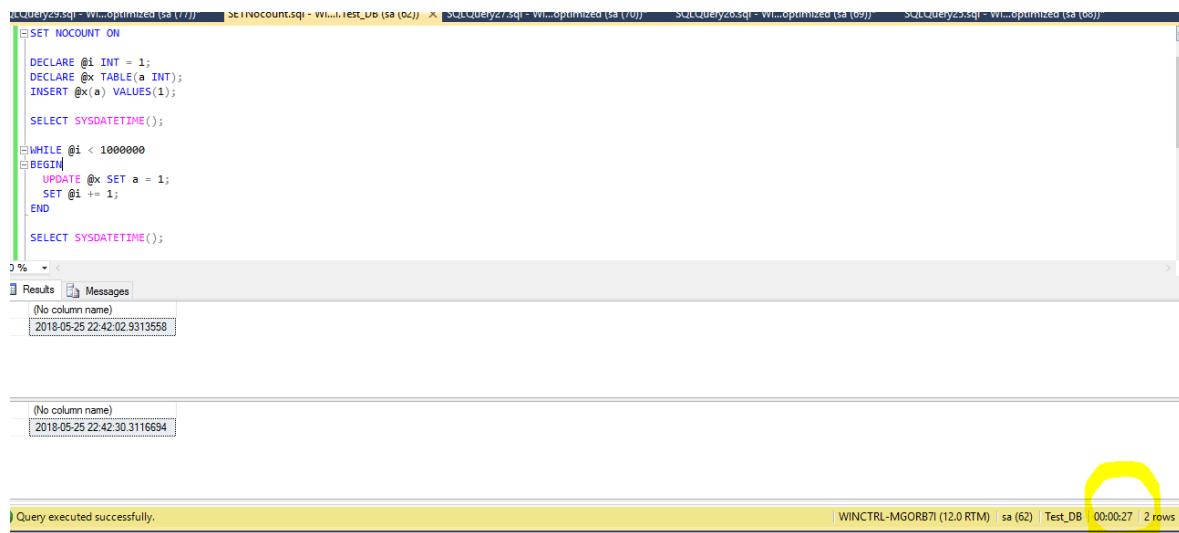


Figure 6.25 – Alternative solution QEP plan and query cost



```

SET NOCOUNT ON;
DECLARE @i INT = 1;
DECLARE @x TABLE(a INT);
INSERT @x(a) VALUES(1);

SELECT SYSDATETIME();

WHILE @i < 1000000
BEGIN
    UPDATE @x SET a = 1;
    SET @i += 1;
END

SELECT SYSDATETIME();

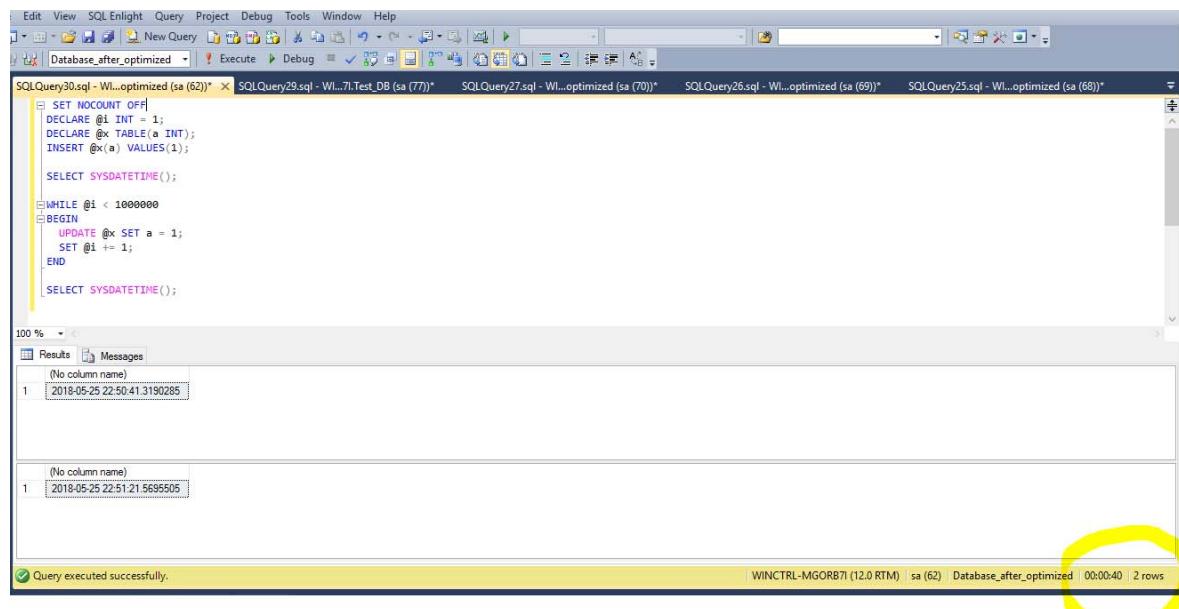
```

Results Messages
 (No column name)
 2018-05-25 22:42:02.931358

(No column name)
 2018-05-25 22:42:30.3116694

Query executed successfully. WINCTRL-MGORB7I (12.0 RTM) | sa (62) | Test_DB | 00:00:27 | 2 rows

Figure 6.26 – Set no count on execution time



```

Edit View SQL Enlight Query Project Debug Tools Window Help
Database_after_optimized * SQLQuery29.sql - WI...!Test_DB (sa (77)) SQLQuery27.sql - WI...optimized (sa (70)) SQLQuery26.sql - WI...optimized (sa (69)) SQLQuery25.sql - WI...optimized (sa (68))
SET NOCOUNT OFF;
DECLARE @i INT = 1;
DECLARE @x TABLE(a INT);
INSERT @x(a) VALUES(1);

SELECT SYSDATETIME();

WHILE @i < 1000000
BEGIN
    UPDATE @x SET a = 1;
    SET @i += 1;
END

SELECT SYSDATETIME();

```

Results Messages
 (No column name)
 1 2018-05-25 22:50:41.3190285

(No column name)
 1 2018-05-25 22:51:21.5695505

Query executed successfully. WINCTRL-MGORB7I (12.0 RTM) | sa (62) | Database_after_optimized | 00:00:40 | 2 rows

Figure 6.27 - 6.26 – Without no count execution time