Lab Answer Key: Module 8: Tracing SQL Server Activity

Lab: Tracing SQL Server Workload Activity

Exercise 1: Capturing a Trace in SQL Server Profiler

Task 1: Prepare the Lab Environment

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are both running, and then log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Labfiles\Lab08\Starter folder, right-click **Setup.cmd** and then click **Run as administrator**.
3. Click **Yes** when prompted to confirm that you want to run the command file, and then wait for the script to finish.

Task 2: Create a SQL Server Profiler Trace

1. On the Start screen, type **Profiler** and then start **SQL Server 2014 Profiler**.
2. In SQL Server Profile, on the **File** menu, click **New Trace**. Then connect to the **MIA-SQL** database engine instance using Windows authentication.
3. In the **Trace Properties** dialog box, on the **General** tab, set the following properties:

   - **Trace name**: InternetSales Workload
   - **Use the template**: TSQL
   - **Save to file**: D:\Labfiles\Lab08\Starter\InternetSales Workload.trc

4. In the **Trace Properties** dialog box, on the **Events Selection** tab, note the events and

Ce document est la propriété de Jerome BURESI (j.buresi.indus@arianespace.com) Toute copie non autorisée est interdite !

columns that were automatically selected from the **TSQL** template.

5. Select **Show all events**, and under **TSQL**, select **SQL:StmtCompleted**. Then clear **Show all events** so that only the selected events, including the one you just selected are shown.

6. Select **Show all columns** and select the **Duration** column for the **SQL:StmtCompleted** event.

7. Click the column header for the **Database Name** column, and in the **Edit Filter** dialog box, expand **Like**, enter **InternetSales**, and click **OK**. Then clear **Show all columns** so that only the selected columns are shown.

Task 3: Capture Workload Events

1. In the **Trace Properties** dialog box, click **Run**.

2. In the D:\Labfiles\Lab08\Starter folder, right-click **Workload.ps1** and click **Run with PowerShell**. This starts a workload in the **InternetSales** database that lasts for approximately three minutes.

3. While the workload is running, switch back to SQL Server Profiler and observe the activity.

4. When the workload has finished, in SQL Server Profiler, on the **File** menu, click **Stop Trace**.

5. In the trace, select any of the **SQL:StmntCompleted** events and note that the Transact-SQL code is shown in the bottom pane.

Result: After this exercise, you should have captured a workload using SQL Server Profiler.

Exercise 2: Generating Database Tuning Recommendations

Task 1: Create a Tuning Session

Ce document est la propriété de Jerome BURESI (j.buresi.indus@arianespace.com) Toute copie non autorisée est interdite !

1. In SQL Server Profiler, on the **Tools** menu, click **Database Engine Tuning Advisor**.
2. When the Database Engine Tuning Advisor starts, connect to the **MIA-SQL** database engine instance using Windows authentication.
3. In the Database Engine Tuning Advisor, In the **Session name** box, type **Tune InternetSales**.
4. Under **Workload**, ensure that **File** is selected, and browse to the **D:\Labfiles\Lab08\Starter\InternetSales Workload.trc** file (which is where you saved the trace from SQL Server Profiler in the previous exercise).
5. In the **Database for workload analysis** drop-down list, select **InternetSales**.
6. In the **Select databases and tables to tune** list, select **InternetSales** and note that all of its tables are selected. Then in the drop down list of tables, clear the checkbox for the **CurrencyRate** table.
7. On the **Tuning Options** tab, review the default options for recommendations. Then click **Advanced Options**, select **Generate online recommendations where possible**, and click **OK**.

Task 2: Generate Recommendations

1. In the Database Engine Tuning Advisor, on the toolbar, click **Start Analysis**.
2. When the analysis is complete, on the **Recommendations** tab, review the index recommendations that the DTA has generated.
3. On the **Actions** menu, click **Save Recommendations**, save the recommendations script as **DTA Recommendations.sql** in the D:\Labfiles\Lab08\Starter folder, and click **OK** when notified that the file was saved.

Task 3: Validate the Recommendations

1. In the Database Engine Tuning Advisor, on the **Reports** tab for the **Tune InternetSales** session, view the tuning summary and in the **Select report** list, select **Event frequency**

Ce document est la propriété de Jerome BURESI. (j.buresi.indus@arianespace.com) Toute copie non autorisée est interdite !

report.

2. View the report and identify the most frequently used query in the workload.

3. In the **Select report** list, select **Statement detail report**.

4. View the report and compare the **Current Statement Cost** and **Recommended Statement Cost** values for the query you identified as being most frequently used.

5. Select the **Statement String** cell for the most frequently used statement, and then right-click the selected cell and click **Copy**.

6. Minimize the Database Engine Tuning Advisor and start SQL Server Management Studio. When prompted, connect to the **MIA-SQL** database engine instance using Windows authentication.

7. In SQL Server Management Studio, click **New Query** and paste the statement you copied previously into the query window.

8. In the **Available Databases** drop-down list, select **InternetSales**. Then on the **Query** menu, click **Display Estimated Execution Plan**. This displays a breakdown of the tasks that the query processor will perform to process the query.

9. Note that the query processor suggests that there is at least one missing index that would improve query performance. Then hold the mouse over the SELECT icon at the left side of the query plan diagram and view the **Estimated Subtree Cost** value that is displayed in a tooltip.

10. In SQL Server Management Studio, open the **DTA Recommendations.sql** script you saved from the Database Engine Tuning Advisor in the D:\Labfiles\Lab08\Starter folder. Then click execute to implement the recommended indexes.

11. Close the **DTA Recommendations.sql** tab, and return to the query and its estimated execution plan.

12. On the **Query** menu, click **Display Estimated Execution Plan** again, and note that the query processor no longer suggests that there is a missing index. Then hold the mouse over the SELECT icon at the left side of the query plan diagram and view the **Estimated Subtree Cost** value.

Ce document est la propriété de Jerome BURESI (j.buresi.indus@arianespace.com) Toute copie non autorisée est interdite !

Result: After this exercise, you should have analyzed the trace in the Database Engine Tuning Advisor, and reviewed the recommendations.

Exercise 3: Using SQL Trace

Task 1: Export a SQL Trace Script

1. In SQL Server Profiler, with the **InternetSales Workload** trace still open, on the **File** menu, point to **Export**, point to **Script Trace Definition**, and click **For SQL Server 2005 - 2014**.
2. Save the exported trace script as **InternetSales Trace.sql** in the D:\Labfiles\Lab08\Starter folder, and click **OK** when notified that the script has been saved.

Task 2: Run the Trace

1. In SQL Server Management Studio, open the **InternetSales Trace.sql** script file in the D:\Labfiles\Lab08\Starter folder (which you exported from SQL Server Profiler in the previous task).
2. View the Transact-SQL code, and in the line that begins **exec @rc = sp_trace_create**, replace **InsertFileNameHere** with **D:\Labfiles\Lab08\Starter\InternetSales**.
3. Click **Execute** to start the trace, and note the **TraceID** value that is returned.
4. In the D:\Labfiles\Lab08\Starter folder, right-click **Workload.ps1** and click **Run with PowerShell**. This starts a workload in the **InternetSales** database that lasts for approximately three minutes.
5. While the workload is running, switch back to SQL Server Management Studio and click **New Query**, then in the new query window, enter the following code, replacing *TraceID* with the **TraceID** value you noted when you started the trace. Do not execute the code yet.

```
DECLARE @TraceID int = TraceID;
```

Ce document est la propriété de Jerome BURESI (j.buresi.indus@arianespace.com) Toute copie non autorisée est interdite !

```
EXEC sp_trace_setstatus @TraceID, 0;
EXEC sp_trace_setstatus @TraceID, 2;
GO
```

6. When the Windows PowerShell window closes (indicating that the workload has finished), in SQL Server Management Studio, click **Execute** to stop the trace.

Task 3: View the Trace Results

1. In SQL Server Management Studio, in the query pane, below the existing code, add the following Transact-SQL statement, which retrieves the text data, start time, and duration for each **SQL:StmntComplete** event in the trace file.

```
SELECT TextData, StartTime, Duration
FROM fn_trace_gettable('D:\Labfiles\Lab08\Starter
\InternetSales.trc', default)
WHERE EventClass = 41;
```

2. Select the code you added in the previous step and click **Execute**. Then view the results.
3. Close SQL Server Management Studio, SQL Server profiler, and the Database Engine Tuning Advisor without saving any files.

Result: After this exercise, you should have captured a trace using SQL Trace.

Ce document est la propriété de Jérôme BURESI (j.buresi-dus@arianespace.com) Toute copie non autorisée est interdite !