

OFFICIAL MICROSOFT LEARNING PRODUCT

10987C

Performance Tuning and Optimizing SQL Databases

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2017 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at <https://www.microsoft.com/en-us/legal/intellectualproperty/trademarks/en-us.aspx> are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners

Product Number: 10987C

Part Number (if applicable):

Released: 11/2017

Module 1

SQL Server Architecture, Scheduling, and Waits

Contents:

Lesson 1: SQL Server Components and SQLOS	2
Lesson 2: Windows Scheduling vs. SQL Server Scheduling	4
Lesson 3: Waits and Queues	6
Module Review and Takeaways	9
Lab Review Questions and Answers	11

Lesson 1

SQL Server Components and SQLOS

Contents:

Question and Answers	3
Demonstration: Monitoring Engine Behavior	3

Question and Answers

Question: Which database engine component is responsible for thread management?

- () The SQL Server Network Interface
- () The query optimizer
- () The relational engine
- () The SQL Server Operating System (SQLOS)
- () The storage engine

Answer:

- () The SQL Server Network Interface
- () The query optimizer
- () The relational engine
- (v) The SQL Server Operating System (SQLOS)
- () The storage engine

Demonstration: Monitoring Engine Behavior

Demonstration Steps

1. Ensure that the **10987C-MIA-DC** and **10987C-MIA-SQL** virtual machines are both running, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Run **Setup.cmd** in the D:\Demofiles\Mod01 folder as Administrator. In the **User Account Control** dialog box, click **Yes**, and wait for the script to complete.
3. Start SQL Server Management Studio, and then connect to the **MIA-SQL** database engine instance by using Windows authentication.
4. Open the **Project.ssmsslnproj** solution in the D:\Demofiles\Mod01\Project folder.
5. In Solution Explorer, expand, **Queries**, and then open the **Demo1 - Monitor_Engine.sql** script file.
6. In the D:\Demofiles\Mod01 folder, right-click **start_load_1.ps1**, and then click **Run with PowerShell**. If the **Execution Policy change** message appears, type **y**, and then press Enter.
Leave the script running, and continue with the demo.
7. In SQL Server Management Studio, highlight the script below **Step 2**, and then click **Execute**.
8. Highlight the script below **Step 3**, and then click **Execute**.
9. Highlight the script below **Step 4**, and then click **Execute**.
10. Highlight the script below **Step 5**, and then click **Execute**.
11. Highlight the script below **Step 6**, and then click **Execute**.
12. Highlight the script below **Step 7**, and then click **Execute**.
13. Keep SQL Server Management Studio open for the next demonstration.

Lesson 2

Windows Scheduling vs. SQL Server Scheduling

Contents:

Demonstration: Analyzing the Life Cycle of a Thread

5

Demonstration: Analyzing the Life Cycle of a Thread

Demonstration Steps

1. Ensure that you have completed the previous demonstration in this module. Alternatively, start the **MT17B-WS2016-NAT**, **10987C-MIA-DC**, and **10987C-MIA-SQL** virtual machines, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Run **Setup.cmd** in the D:\Demofiles\Mod01 folder as Administrator. In the **User Account Control** dialog box, click **Yes**, and wait for the script to complete.
3. If it is not already running, start SQL Server Management Studio and connect to the **MIA-SQL** database engine instance by using Windows authentication.
4. If it is not already open, open the **Project.ssmsslproj** solution in the D:\Demofiles\Mod01\Project folder.
5. In Solution Explorer, open the **Demo2i - Create hanging transaction.sql** script file, and click **Execute**.
6. Open the **Demo2ii - Start blocked transaction.sql** script file, and click **Execute**.
7. Open the **Demo2iii - DMV queries.sql** script file.
8. Substitute the values of **update_session_id** and **select_session_id** collected in the last two steps into the **VALUES** clause below **step 3**, click **Execute**.
9. Highlight the script below **Step 4**, and then click **Execute**.
10. Highlight the script below **Step 5**, and then click **Execute**.
11. Highlight the script below **Step 6**, and then click **Execute**.
12. Highlight the script below **Step 7**, and then click **Execute**.
13. Highlight the script below **Step 8**, and then click **Execute**.
14. Highlight the script below **Step 9**, and then click **Execute**.
15. On the **Demo2i - Create hanging transaction.sql** query, uncomment and execute the ROLLBACK command at the end of the file.
16. On the **Demo2ii - Start blocked transaction.sql**, note that the query is no longer blocked and results have been returned.
17. At the end of the demonstration, close SQL Server Management Studio without saving changes.

Lesson 3

Waits and Queues

Contents:

Question and Answers	7
Demonstration: Monitoring Common Wait Types	7

Question and Answers

Question: Why is signal_wait_time_ms an important metric to review during performance issues?

- () If signal wait time is a high percentage of total wait time, this may indicate memory pressure.
- () If signal wait time is a high percentage of total wait time, this may indicate I/O pressure.
- () If signal wait time is a high percentage of total wait time, this may indicate CPU pressure.
- () If signal wait time is a high percentage of total wait time, this may indicate a bottleneck in tempdb.

Answer:

- () If signal wait time is a high percentage of total wait time, this may indicate memory pressure.
- () If signal wait time is a high percentage of total wait time, this may indicate I/O pressure.
- (√) If signal wait time is a high percentage of total wait time, this may indicate CPU pressure.
- () If signal wait time is a high percentage of total wait time, this may indicate a bottleneck in tempdb.

Demonstration: Monitoring Common Wait Types

Demonstration Steps

1. Ensure that you have completed the previous demonstration in this module. Alternatively, start the **MT17B-WS2016-NAT**, **10987C-MIA-DC**, and **10987C-MIA-SQL** virtual machines, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Run **Setup.cmd** in the D:\Demofiles\Mod01 folder as Administrator. In the **User Account Control** dialog box, click **Yes**, and wait for the script to complete.
3. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine instance by using Windows authentication.
4. Open the **Project.ssmsslnproj** solution in the D:\Demofiles\Mod01\Project folder.
5. In Solution Explorer, expand **Queries**, open the **Demo2i - Create hanging transaction.sql** script file, and then click **Execute**.
6. Open the **Demo2ii - Start blocked transaction.sql** script file, and click **Execute**. Note the select_session_id.
7. Open the **Demo3 - instructions.sql** script file.
8. Highlight the script below **Step 3**, and then click **Execute**.
9. Add the **select_session_id** value collected in step 6 into the clause below **step 4**, and then click **Execute**.
10. On the **Demo2i - Create hanging transaction.sql** query, uncomment and execute the **ROLLBACK** command at the end of the file.
11. In **Demo3 - instructions.sql**, highlight the script below **Step 4**, and then click **Execute**. Note that this time, the LCK_M_S wait is included in the results because the session has finished waiting.
12. Highlight the script below **Step 7**, and then click **Execute**.
13. Highlight the script below **Step 8**, and then click **Execute**.
14. On the Start menu, type **Resource Monitor**, and then click **Resource Monitor**.

15. In Resource Monitor, on the **Disk** tab, notice that the **Disk Queue** graph for the D: drive (where the database files are located) is near zero.
16. In File Explorer, browse to **D:\Demofiles\Mod01**, right-click **start_load_2.ps1**, and then click **Run with PowerShell** to start the load. The load will run for approximately five minutes before stopping, and then press Enter.
17. Wait 30 seconds for the load to start.
18. In SQL Server Management Studio, select the query under the comment **Step 11**, and click **Execute**. Repeat this step several times over the course of a minute whilst the load is running.
19. In Resource Monitor, observe that the **Disk Queue** graph for the D: drive is elevated above zero.
20. At the end of the demonstration, close Resource Monitor, close SSMS without saving changes, and then close PowerShell.

Module Review and Takeaways

Best Practice

It is good practice to keep a baseline of wait statistics data; this makes it much easier to determine when a trend has changed.

Review Question(s)

Put the following steps of the query life cycle in order by numbering each to indicate the correct order.

	Steps
	Command received from client application.
	Command is parsed into a parse tree.
	Parse tree is bound to database objects.
	Plan hash is generated.
	Check for existing query plan.
	Query Optimizer selects a suitable plan.
	Query plan is executed.
	Results are returned to the client application.

Answer:

	Steps
1	Command received from client application.
2	Command is parsed into a parse tree.
3	Parse tree is bound to database objects.
4	Plan hash is generated.
5	Check for existing query plan.
6	Query Optimizer selects a suitable plan.
7	Query plan is executed.
8	Results are returned to the client application.

Tools

Several SQL Server MVPs have published useful tools for tracking waiting tasks and wait statistics:

Adam Machanic's **sp_WhoIsActive**.

Glenn Berry's performance monitoring scripts.

Jonathan Kehayias and Erin Stellato at sqlskills.com have written an unofficial update to the *SQL Server 2005 Waits and Queues* document, which is called *SQL Server Performance Tuning Using Wait Statistics: A Beginner's Guide*. It is available as a PDF file from sqlskills.com.

Lab Review Questions and Answers

Lab: SQL Server Architecture, Scheduling, and Waits

Question and Answers

Lab Review

Question: The output from **sys.dm_os_schedulers** (accessed in the last task in Exercise 1) shows some schedulers as HIDDEN ONLINE. What is meant by hidden schedulers?

Answer: Hidden schedulers are used internally by SQL Server for its internal operations. This is to ensure that internal operations are not affected when visible schedulers are under pressure.

Module 2

SQL Server I/O

Contents:

Lesson 1: Core Concepts of I/O	2
Lesson 2: Storage Solutions	4
Lesson 3: I/O Setup and Testing	6
Module Review and Takeaways	8
Lab Review Questions and Answers	10

Lesson 1

Core Concepts of I/O

Contents:

Question and Answers

3

Question and Answers

Question: Which I/O performance measure reflects the responsiveness of a storage device to an I/O request?

- ☐ Throughput
- ☐ Latency factor
- ☐ IOPS

Answer:

- ☐ Throughput
- ☒ Latency factor
- ☐ IOPS

Lesson 2

Storage Solutions

Contents:

Question and Answers

5

Question and Answers

Question: Which storage solution does your organization most commonly use? What are its benefits? What are its limitations?

Answer: Your answer will be determined by your experiences.

Lesson 3

I/O Setup and Testing

Contents:

Question and Answers	7
Demonstration: Benchmarking I/O	7

Question and Answers

Question: Which Diskspd parameter controls the percentage of reads and writes?

- () -d
- () -c
- () -r
- () -t
- () -w

Answer:

- () -d
- () -c
- () -r
- () -t
- (√) -w

Demonstration: Benchmarking I/O

Demonstration Steps

1. Ensure that the **10987C-MIA-DC** and **10987C-MIA-SQL** virtual machines are running, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In Windows Explorer, navigate to the folder D:\Demofiles\Mod02, and then run **Setup.cmd** as Administrator.
3. In the **User Account Control** dialog box, click **Yes**.
4. Double-click the **demo.PerfmonCfg** file. Performance Monitor will start with several counters included, but hidden on the histogram.
5. Return to D:\Demofiles\Mod02 in Windows Explorer, right-click **start_load_1.ps1**, and then click **Run with PowerShell**. If a message is displayed asking you to confirm a change in execution policy, type **Y**. This script starts a workload that ramps up over 60 seconds, and runs for five minutes.
6. While the load is running, return to Performance Monitor and display the performance counters on the histogram by selecting the box next to their names in the list at the bottom of the window. Values relate to benchmark measures as follows:
 - IOPS:
 - Physical Disk: Avg. Disk Bytes/Read
 - Physical Disk: Avg. Disk Bytes/Write
 - Throughput:
 - Physical Disk: Disk Read Bytes/Sec
 - Physical Disk: Disk Write Bytes/Sec
 - Latency factor:
 - Physical Disk: Avg. Disk sec/Read
 - Physical Disk: Avg. Disk sec/Write
7. Close Performance Monitor, Windows Explorer, and PowerShell.

Module Review and Takeaways

Best Practice

In this module, you have learned how to discuss and analyze storage systems in the context of SQL Server installations. You have gained an understanding of various storage system types, and the ways in which they may be used, and learned about tools and techniques to assess storage system performance.

Review Question(s)

Place each Performance Monitor performance counter into the category of the performance measure to which it relates. Indicate your answer by writing the category number to the right of each item.

Items	
1	Physical Disk: Disk Reads/Sec
2	Physical Disk: Disk Read Bytes/Sec
3	Physical Disk: Avg. Disk Sec/Read
4	Physical Disk: Disk Writes/Sec
5	Physical Disk: Disk Write Bytes/Sec
6	Physical Disk: Avg. Disk Sec/Write
7	Logical Disk: Disk Reads/Sec
8	Logical Disk: Disk Read Bytes/Sec
9	Logical Disk: Avg. Disk Sec/Read
10	Logical Disk: Disk Writes/Sec
11	Logical Disk: Disk Write Bytes/Sec
12	Logical Disk: Avg. Disk Sec/Write

Category 1		Category 2		Category 3
IOPS		Throughput		Latency Factor

Answer:

Category 1		Category 2		Category 3
IOPS		Throughput		Latency Factor
Physical Disk: Disk Reads/Sec Physical Disk: Disk Writes/Sec Logical Disk: Disk Reads/Sec Logical Disk: Disk Writes/Sec		Physical Disk: Disk Read Bytes/Sec Physical Disk: Disk Write Bytes/Sec Logical Disk: Disk Read Bytes/Sec Logical Disk: Disk Write Bytes/Sec		Physical Disk: Avg. Disk Sec/Read Physical Disk: Avg. Disk Sec/Write Logical Disk: Avg. Disk Sec/Read Logical Disk: Avg. Disk Sec/Write

Lab Review Questions and Answers

Lab: Testing Storage Performance

Question and Answers

Lab Review

Question: What do the results from Diskspd show?

Answer:

- CPU activity during the test, for each CPU.
- Total I/O, read I/O, and write I/O statistics for each thread.
- Total speed, read speed, and write speed by percentile.

Module 3

Database Structures

Contents:

Lesson 1: Database Structure Internals	2
Lesson 2: Data File Internals	5
Lesson 3: tempdb Internals	7

Lesson 1

Database Structure Internals

Contents:

Question and Answers	3
Demonstration: Analyzing Database Structures	3

Question and Answers

Question: Which of the following page types track changes to pages since the last full backup?

- () Global Allocation Map
- () Index Allocation Map
- () Differential Change Map
- () Shared Global Allocation Map
- () Page Free Space

Answer:

- () Global Allocation Map
- () Index Allocation Map
- (v) Differential Change Map
- () Shared Global Allocation Map
- () Page Free Space

Demonstration: Analyzing Database Structures

Demonstration Steps

1. Start the **10987C-MIA-DC** and **10987C-MIA-SQL** virtual machines, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Demofiles\Mod03** folder, run **Setup.cmd** as Administrator.
3. In the **User Account Control** dialog box, click **Yes**, wait for the script to finish, and then press Enter.
4. Open SQL Server Management Studio, connect to **MIA-SQL** database engine with Windows Authentication.
5. In SQL Server Management Studio, open the **PageStructure.sql** Transact-SQL file in the **D:\Demofiles\Mod03** folder.
6. Highlight the Transact-SQL code under the comment **Create table and insert test data**, and click **Execute** to create a test table and populate it with data.
7. Highlight the Transact-SQL code under the comment **Find Page information**, and click **Execute**, and from the query results note the value in the column **allocated_page_page_id**.
8. Highlight the Transact-SQL code under the comment **Enable trace flag**, and click **Execute** to turn on trace flag 3604.
9. Modify the Transact-SQL code under the comment **View page allocation**, replacing **XXX** with the **allocated_page_page_id** value from step 7.
10. Highlight the Transact-SQL code under the comment **View page allocation**, and click **Execute**, and examine the page information, noting that slot 0, slot 1, and slot 2 are of the type **PRIMARY_RECORD** because they contain row data.
11. Highlight the Transact-SQL code under the comment **Update data**, and click **Execute** to update the data stored on the page.
12. Highlight the Transact-SQL code under the comment **Find Page information**, and click **Execute**, noting that an extra row is returned because data for the table now consumes more than one page.

13. Modify the Transact-SQL code under the comment **View updated page allocation**, replacing **XXX** with the **allocated_page_page_id** value from step 7.
14. Highlight the Transact-SQL code under the comment **View updated page allocation**, and click **Execute**. Examine the page information, noting that slot 0 is of the type FORWARDING_STUB because some row data is stored in a separate page.
15. Close SQL Server Management Studio without saving changes.

Lesson 2

Data File Internals

Contents:

Question and Answers	6
Resources	6
Demonstration: Shrinking Databases	6

Question and Answers

Question: When configuring auto grow for SQL Server data files, should you specify a percentage growth or a fixed size growth—and why?

Answer: Generally, it is better to use a fixed size growth rather than a percentage growth, because using a percentage growth means each subsequent auto growth is larger, resulting in increased IO. Auto growth should not be used as a contingency, and not for day-to-day management of data files.

Resources

Auto Grow and Auto Shrink



Best Practice: You should consider creating data and log files with sufficient space for the expected volume of data in the database. This will avoid the frequent auto growth of files.

Demonstration: Shrinking Databases

Demonstration Steps

1. Open SQL Server Management Studio, connect to **MIA-SQL** database engine with Windows Authentication.
2. In SQL Server Management Studio, open the **shrinkdb.sql** Transact-SQL file in the **D:\Demofiles\Mod03** folder.
3. Highlight the Transact-SQL code under the comment **Check database free space**, and click **Execute**, note that the database has more than 350 MB of unallocated space.
4. Highlight the Transact-SQL code under the comment **Check index fragmentation**, and click **Execute**. Note the values in the **avg_fragmentation_in_percent** column.
5. Highlight the Transact-SQL code under the comment **Shrink database data file**, and click **Execute**.
6. Highlight the Transact-SQL code under the comment **Check database free space**, and click **Execute**. Note that the unallocated space has reduced significantly.
7. Highlight the Transact-SQL code under the comment **Check index fragmentation**, and click **Execute**. Note that some values in the **avg_fragmentation_in_percent** column are much higher than they were before shrinking the database file.
8. Close SQL Server Management Studio without saving changes.

Lesson 3

tempdb Internals

Contents:

Question and Answers	8
Demonstration: Monitoring tempdb Usage	8

Question and Answers

Question: **tempdb** has run out of space causing your SQL Server instance to crash. How might you recover from this scenario?

Answer: Because **tempdb** is recreated at server startup, you could recover from this scenario by restarting the SQL Server instance. After SQL Server restarts, it is important that the reason for **tempdb** running out of space is analyzed and **tempdb** is resized if appropriate.

Demonstration: Monitoring tempdb Usage

Demonstration Steps

1. In SQL Server Management Studio, connect to the **MIA-SQL** database engine with Windows Authentication.
2. In SQL Server Management Studio, open the **monitortempdb.sql** Transact-SQL file in the **D:\Demofiles\Mod03** folder.
3. Highlight the Transact-SQL code under the comment **Amount of space in each tempdb file (Free and Used)**, and click **Execute**. Note the amount of space in each **tempdb** file.
4. Highlight the Transact-SQL code under the comment **Amount of free space in each tempdb file**, and click **Execute**. The amount of free space in each **tempdb** file is shown.
5. Highlight the Transact-SQL code under the comment **Amount of space used by the version Store**, and click **Execute**. Note the amount of **tempdb** space used by the version store.
6. Highlight the Transact-SQL code under the comment **Number of pages and the amount of space in MB used by internal objects**, and click **Execute**. The amount of **tempdb** space used by internal objects is returned.
7. Highlight the Transact-SQL code under the comment **Amount of space used by user objects in tempdb**, and click **Execute**. The amount of space used by user objects in **tempdb** is shown.
8. Close SQL Server Management Studio without saving changes.

Module 4

SQL Server Memory

Contents:

Lesson 2: SQL Server Memory	2
Lesson 3: In-Memory OLTP	4
Module Review and Takeaways	6

Lesson 2

SQL Server Memory

Contents:

Question and Answers	3
Demonstration: Analyzing Memory Allocations	3

Question and Answers

Question: Which of the following is an advantage of using the SQL Server locked memory model over the conventional memory model?

- () The locked memory model uses less RAM.
- () The locked memory model will not allow pages allocated to SQL Server to be paged to disk.
- () The locked memory model writes least used pages to disk, freeing up system memory.
- () The locked memory model can run on a system with lower available memory.

Answer:

- () The locked memory model uses less RAM.
- (√) The locked memory model will not allow pages allocated to SQL Server to be paged to disk.
- () The locked memory model writes least used pages to disk, freeing up system memory.
- () The locked memory model can run on a system with lower available memory.

Demonstration: Analyzing Memory Allocations

Demonstration Steps

1. Ensure that the **10987C-MIA-DC** and **10987C-MIA-SQL** virtual machines are both running, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Navigate to the folder **D:\Demofiles\Mod04\Demo01** and run the file **Setup.cmd** as Administrator.
3. In the **User Account Control** dialog box, click **Yes**, and then wait until the script completes.
4. Click **Start** menu, type **SQL Server 2017 Configuration Manager**, and press Enter.
5. In the **User Account Control** dialog, click **Yes**.
6. In SQL Server Configuration Manager, click **SQL Server Services** node, and restart the **SQL Server (MSSQLSERVER)** service.
7. In SQL Server Management Studio, connect to the **MIA-SQL** database instance.
8. Open a new query window and click **File, Open, File**. Navigate to the **D:\DemoFiles\Mod04\Demo01** folder and double-click **MonitorMemory.sql** to open it. Execute the script.
9. Click the **Start** menu, open **Resource Monitor**, and on the memory tab note the values for **Working Set** and **Commit**.
10. In the Results pane in SQL Server Management Studio, compare the values under the columns **physical_memory_in_use_kb** and **virtual_address_space_committed_kb** with the values under the columns **Working Set(KB)** and **Commit(KB)** for the process **sqlservr.exe** in the Resource Monitor.
11. Close Resource Monitor.

Lesson 3

In-Memory OLTP

Contents:

Question and Answers	5
Demonstration: Creating Memory-Optimized Tables	5

Question and Answers

Question: One of your organization's key transactional systems is suffering from poor performance due to the volume of transactions on an important table. You decide to convert this table to a memory-optimized table. Which type of memory-optimized table mode would be most suitable?

- ☐ Non-durable memory-optimized table.
- ☐ Durable memory-optimized table with delayed durable transactions.
- ☐ Memory-optimized tables are not suitable for transactional systems.
- ☐ Durable memory-optimized table.

Answer:

- ☐ Non-durable memory-optimized table.
- ☐ Durable memory-optimized table with delayed durable transactions.
- ☐ Memory-optimized tables are not suitable for transactional systems.
- ☒ Durable memory-optimized table.

Demonstration: Creating Memory-Optimized Tables

Demonstration Steps

1. Ensure that the 10987C-MIA-DC and 10987C-MIA-SQL virtual machines are both running, and then log on to 10987C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Start SQL Server Management Studio and connect to the **MIA-SQL** instance database engine using Windows authentication.
3. Open the file **CreateMemoryOptTables.sql** located in the D:\Demofiles\Mod04\Demo02 folder.
4. Review and execute the Transact-SQL code, noting that it creates and populates two memory-optimized tables, one durable and one non-durable.
5. Open the file **SelectFromMemoryOptTables.sql** located in D:\Demofiles\Mod04\Demo02.
6. Execute the code preceding the comment **Restart SQL Server Services** and note both tables return results.
7. In Object Explorer, right-click **MIA-SQL** and then click **Restart**. In the **User Account Control** dialog box, click **Yes** to allow the restart to proceed.
8. In the **Microsoft SQL Server Management Studio** dialog box, click **Yes** to restart the service, and then in the second **Microsoft SQL Server Management Studio** dialog box, click **Yes** to stop the SQL Server Agent service.
9. When the SQL Server service restarts, Execute the two SELECT statements again, noting that both tables still exist, but the non-durable table no longer contains data.
10. Close SQL Server Management Studio without saving changes.

Module Review and Takeaways

Best Practice

Avoid setting the Min Server Memory and Max Server Memory in SQL Server to identical values and effectively fixing the amount of memory SQL Server uses. Allowing SQL Server to manage memory dynamically will give the best overall system performance.

Module 5

SQL Server Concurrency

Contents:

Lesson 1: Concurrency and Transactions	2
Lesson 2: Locking Internals	6
Module Review and Takeaways	9
Lab Review Questions and Answers	10

Lesson 1

Concurrency and Transactions

Contents:

Question and Answers	3
Demonstration: Analyzing Concurrency Problems	3

Question and Answers

Question: User A starts to update a customer record, and while the transaction is still in progress, User B tries to update the same record. User A's update completes successfully, but User B's update fails with an error message: "This customer's record has been updated by another user". Which concurrency model is the system using?

- () Pessimistic concurrency
- () Optimistic concurrency

Answer:

- () Pessimistic concurrency
- (√) Optimistic concurrency

Demonstration: Analyzing Concurrency Problems

Demonstration Steps

1. Ensure that the **MT17B-WS2016-NAT**, **10987C-MIA-DC**, and **10987C-MIA-SQL** virtual machines are running, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Run **Setup.cmd** in the **D:\Demofiles\Mod05** folder as Administrator. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.
3. Start SQL Server Management Studio and connect to your Azure instance of the **AdventureWorksLT** database engine instance using SQL Server authentication. If the **Microsoft SQL Server Management Studio** dialog box appears, click **OK**.
4. Open the **Demo1.ssmssln** solution in the **D:\Demofiles\Mod05\Demo1** folder.
5. Open the **Demo 1a - concurrency 1.sql** script file and the **Demo 1b - concurrency 2.sql** script file; open these files in different query windows, because you will be switching between them.
6. In both script files, click **ADVENTUREWORKSLT** in the **Available databases** list.
7. In the **Demo 1a - concurrency 1.sql** script file, under the comment that begins **Step 3**, select the code, and then click **Execute** to check the current settings for SNAPSHOT isolation.
8. Under the comment that begins **Step 4**, select the code, and then click **Execute** to view the current state of the row used in this demonstration.
9. In the **Demo 1b - concurrency 2.sql** script file, under the comment that begins **Query 1**, select the code, and then click **Execute**.
10. In the **Demo 1a - concurrency 1.sql** file, under the comment that begins **Step 5**, select the code, and then click **Execute** to demonstrate READ UNCOMMITTED isolation.
11. Under the comment that begins **Step 6**, select the code, and then click **Execute** to demonstrate READ COMMITTED isolation. The query will wait until you complete the next step.
12. In the **Demo 1b - concurrency 2.sql** script file, under the comment that begins **Query 2**, select the code, and then click **Execute**.
13. In the **Demo 1a - concurrency 1.sql** script file, note that the query under **Step 6** (which was already running) has now returned a result.
14. Under the comment that begins **Step 7**, select the first five lines of code, and then click **Execute**.

15. In the **Demo 1b - concurrency 2.sql** script file, under the comment that begins **Query 3**, select the code, and then click **Execute**.
16. In the **Demo 1a - concurrency 1.sql** script file, under the comment that begins **Step 7**, select the final four lines of code, and then click **Execute** to demonstrate a non-repeatable read.
17. Under the comment that begins **Step 8**, select the first six lines of code, and then click **Execute**.
18. In the **Demo 1b - concurrency 2.sql** script file, under the comment that begins **Query 4**, select the code, and then click **Execute**. Note that this query will not return until you complete the next step, because REPEATABLE READ holds a lock on the affected row.
19. In the **Demo 1a - concurrency 1.sql** script file, under the comment that begins **Step 8**, select the final four lines of code, and then click **Execute** to demonstrate that REPEATABLE READ isolation prevents a non-repeatable read.
20. Under the comment that begins **Step 9**, select the first six lines of code, and then click **Execute**.
21. In the **Demo 1b - concurrency 2.sql** script file, under the comment that begins **Query 5**, select the code, and then click **Execute**.
22. In the **Demo 1a - concurrency 1.sql** script file, under the comment that begins **Step 9**, select the final four lines of code, and then click **Execute** to demonstrate that READ COMMITTED isolation allows a phantom read.
23. Under the comment that begins **Step 10**, select the first six lines of code, and then click **Execute**.
24. In the **Demo 1b - concurrency 2.sql** script file under the comment that begins **Query 5**, select the code, and then click **Execute**. Note that this query will not return until you complete the next step, since SERIALIZABLE holds a lock on the affected table.
25. In the **Demo 1a - concurrency 1.sql** script file, under the comment that begins **Step 10**, select the final four lines of code, and then click **Execute** to demonstrate that SERIALIZABLE isolation prevents a phantom read.
26. Under the comment that begins **Step 11**, select the first two lines of code, and then click **Execute** to reset the value of the target data row.
27. In the **Demo 1b - concurrency 2.sql** script file, under the comment that begins **Query 6**, select the code, and then click **Execute**.
28. In the **Demo 1a - concurrency 1.sql** script file, under the comment that begins **Step 11**, select the six lines of code under the comment that begins **run the following statements**, and then click **Execute**. Note that the committed value of the row is returned.
29. In the **Demo 1b - concurrency 2.sql** file, under the comment that begins **Query 7**, select the code, and then click **Execute**.
30. In the **Demo 1a - concurrency 1.sql** script file, under the comment that begins **Step 11**, select the final four lines of code, and then click **Execute** to demonstrate the behavior of READ_COMMITTED_SNAPSHOT ON.
31. Under the comment that begins **Step 12**, select the first two lines of code, and then click **Execute** to reset the value of the target data row.
32. In the **Demo 1b - concurrency 2.sql** script file, under the comment that begins **Query 6**, select the code, and then click **Execute**.
33. In the **Demo 1a - concurrency 1.sql** script file, under the comment that begins **Step 12**, select the six lines of code under the comment that begins **run the following statements**, and then click **Execute**. Note that the committed value of the row is returned.

34. In the **Demo 1b - concurrency 2.sql** script file, under the comment that begins **Query 7**, select the code, and then click **Execute**.
35. In the **Demo 1a - concurrency 1.sql** script file, under the comment that begins **Step 12**, select the final four lines of code, and then click **Execute** to demonstrate the behavior of SNAPSHOT isolation.
36. Under the comment that begins **Step 13**, select the first two lines of code, and then click **Execute** to reset the value of the target data row.
37. In the **Demo 1b - concurrency 2.sql** script file, under the comment that begins **Query 6**, select the code, and then click **Execute**.
38. In the **Demo 1a - concurrency 1.sql** script file, under the comment that begins **Step 13**, select the six lines of code under the comment that begins **run the following statements**, and then click **Execute**. Note that this query will not return until you complete the next step.
39. In the **Demo 1b - concurrency 2.sql** script file, under the comment that begins **Query 7**, select the code, and then click **Execute**.
40. In the **Demo 1a - concurrency 1.sql** query window, notice that an update conflict in SNAPSHOT isolation mode has been reported.
41. Under the comment that begins **Step 13**, select the final line of code, and then click **Execute** to demonstrate that the open transaction was rolled back when the error was raised.
42. Close SQL Server Management Studio without saving any changes.

Lesson 2

Locking Internals

Contents:

Question and Answers	7
Resources	7
Demonstration: Applying Locking Hints	7

Question and Answers

Question: If a process is attempting to acquire an exclusive row lock, what lock mode will it attempt to acquire on the data page and table that contain the row?

- ☐ Exclusive (X)
- ☐ Shared (S)
- ☐ Intent shared (IS)
- ☐ Intent exclusive (IX)
- ☐ Intent update (IU)

Answer:

- ☐ Exclusive (X)
- ☐ Shared (S)
- ☐ Intent shared (IS)
- ☒ Intent exclusive (IX)
- ☐ Intent update (IU)

Resources

Locking Hints



Best Practice: In general, it is best to avoid locking hints and allow the SQL Server Query Optimizer to select an appropriate locking strategy. Be sure to regularly review any locking hints you use; confirm that they are still appropriate.

Demonstration: Applying Locking Hints

Demonstration Steps

1. Ensure that the **MT17B-WS2016-NAT**, **10987C-MIA-DC**, and **10987C-MIA-SQL** virtual machines are running, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine instance using Windows authentication.
3. Open the **Demo2.ssmssln** solution in the D:\Demofiles\Mod05\Demo2 folder.
4. Open the **Demo 2a - lock hints 1.sql** script file and the **Demo 2b - lock hints 2.sql** script file. Ensure that both scripts use the **AdventureWorks** database.
5. In the **Demo 2a - lock hints 1.sql** script file, under the comment that begins **Step 3**, select the code, and then click **Execute** to show the current isolation level.
6. Under the comment that begins **Step 4**, select the code, and then click **Execute** to demonstrate the locks held by a transaction using READ UNCOMMITTED isolation.
7. Under the comment that begins **Step 5**, select the first three lines of code, and then click **Execute** to demonstrate the locks held by a transaction using REPEATABLE READ isolation.
8. Under the comment that begins **Step 5**, select the remaining five lines of code, and then click **Execute**.

9. Under the comment that begins **Step 6**, select the first three lines of code, and then click **Execute** to demonstrate the locks held by a transaction using REPEATABLE READ isolation and a READCOMMITTED locking hint.
10. Under the comment that begins **Step 6**, select the remaining five lines of code, and then click **Execute**.
11. Under the comment that begins **Step 7**, select the first three lines of code, and then click **Execute** to demonstrate the locks held by a transaction using READ COMMITTED isolation and a TABLOCKX locking hint.
12. Under the comment that begins **Step 7**, select the remaining five lines of code, and then click **Execute**.
13. Under the comment that begins **Step 8**, select the first three lines of code, and then click **Execute** to demonstrate the locks held by a transaction using REPEATABLE READ isolation and a TABLOCKX locking hint.
14. Under the comment that begins **Step 8**, select the remaining five lines of code, and then click **Execute**.
15. In the **Demo 2b - concurrency 2.sql** script file, under the comment that begins **Query 1**, select the code, and then click **Execute**.
16. In the **Demo 2a - lock hints 1.sql** script file, under the comment that begins **Step 9**, select the code, and then click **Execute** to demonstrate that the statement waits.
17. Allow the query to wait for a few seconds, and then on the **Query** menu, click **Cancel Executing Query**.
18. Under the comment that begins **Step 10**, select the code, and then click **Execute** to demonstrate the behavior of the READPAST hint.
19. In the **Demo 1b - concurrency 2.sql** script file, under the comment that begins **Query 2**, select the code, and then click **Execute** to close the open transaction.
20. Close SSMS without saving any changes.

Module Review and Takeaways

Review Question(s)

Question: A transaction is running with the SERIALIZABLE transaction isolation level. The transaction includes a SELECT statement with a single table in the FROM clause; the table is referenced with the READCOMMITTED table hint. Which transaction isolation level applies to the SELECT statement?

- ☐ SERIALIZABLE
- ☐ READ UNCOMMITTED
- ☐ REPEATABLE READ
- ☐ READ COMMITTED

Answer:

- ☐ SERIALIZABLE
- ☐ READ UNCOMMITTED
- ☐ REPEATABLE READ
- ☒ READ COMMITTED

Lab Review Questions and Answers

Lab: Concurrency and Transactions

Question and Answers

Lab Review

Question: When partition level locking is enabled, what combination of locks will be held by an UPDATE statement that updates all the rows in a single partition? Assume that the partition contains more than 1 million rows.

() Database: Shared (S)

Table: Exclusive (X)

() Database: Shared (S)

Table: Intent Exclusive (IX)

Partition: Exclusive (X)

() Database: Shared (S)

Table: Exclusive (X)

Partition: Exclusive (X)

Answer:

() Database: Shared (S)

Table: Exclusive (X)

(√) Database: Shared (S)

Table: Intent Exclusive (IX)

Partition: Exclusive (X)

() Database: Shared (S)

Table: Exclusive (X)

Partition: Exclusive (X)

Module 6

Statistics and Index Internals

Contents:

Lesson 1: Statistics Internals and Cardinality Estimation	2
Lesson 2: Index Internals	5
Lesson 3: Columnstore Indexes	7
Module Review and Takeaways	10
Lab Review Questions and Answers	12

Lesson 1

Statistics Internals and Cardinality Estimation

Contents:

Question and Answers	3
Resources	3
Demonstration: Analyzing Cardinality Estimation	3

Question and Answers

Question: Which of the following cannot be returned by the DBCC SHOW_STATISTICS command?

- () AUTO_UPDATE_STATISTICS setting
- () Statistics stream
- () Statistics header
- () Density vector
- () Histogram

Answer:

- (√) AUTO_UPDATE_STATISTICS setting
- () Statistics stream
- () Statistics header
- () Density vector
- () Histogram

Resources

Cost-Based Optimization



Additional Reading: For more information on the internals of the query optimizer, see Module 7 of this course: *Query Execution and Query Plan Analysis*.

Demonstration: Analyzing Cardinality Estimation

Demonstration Steps

1. Ensure that the **MT17B-WS2016-NAT**, **10987C-MIA-DC**, and **10987C-MIA-SQL** virtual machines are running, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In file explorer, navigate to the **D:\Demofiles\Mod06** folder, right-click **Setup.cmd** and click **Run as Administrator**.
3. Click **Yes** when prompted at the **User Control** dialog box.
4. Start **SQL Server Management Studio** and connect to the **MIA-SQL** database engine instance using Windows authentication.
5. Open the **Demo.ssmssl** solution in the **D:\Demofiles\Mod06\Demo** folder.
6. Open the **Demo 1 - cardinality.sql** script file.
7. Execute the code under the comment that begins **Step 1** to use the **AdventureWorks** database.
8. Execute the code under the comment that begins **Step 2** to display the statistics objects linked to **Person.Person**.
9. Execute the code under the comment that begins **Step 3** to examine the **PK_Person_BusinessEntityID** statistics object.
10. Execute the code under the comment that begins **Step 4** to examine the **IX_Person_LastName_FirstName_MiddleName** statistics object.

11. Highlight the code under the comment that begins **Step 5** and press Ctrl+L to examine the estimated execution plan for a simple query matching a ROW_HI_KEY value.
12. Execute the code under the comment that begins **Step 6** to examine the **IX_Person_LastName_FirstName_MiddleName** statistics object again.
13. Highlight the code under the comment that begins **Step 6a** and press Ctrl+L to examine the estimated execution plan for a simple query within a step range.
14. Execute the code under the comment that begins **Step 7** to examine the header of the **IX_Person_LastName_FirstName_MiddleName** statistics object.
15. Execute the code under the comment that begins **Step 7a** to examine the density vector of the **IX_Person_LastName_FirstName_MiddleName** statistics object.
16. Highlight the code under the comment that begins **Step 7b** and press Ctrl+L to examine the estimated execution plan for a parameterized query.
17. Execute the code under the comment that begins **Step 7c** to demonstrate how the estimated number of rows from the plan in the previous step is calculated.
18. Highlight the code under the comment that begins **Step 7d** and press Ctrl+L to demonstrate that the calculation is not affected by the parameter value.
19. Highlight the code under the comment that begins **Step 8** and press Ctrl+L to examine the estimated execution plan when a function is used in the predicate.
20. Highlight the code under the comment that begins **Step 8a** and press Ctrl+L to examine the estimated execution plan when two columns from the same table are compared. Note that this is the same as the plan in the previous step.
21. Highlight the code under the comment that begins **Step 9** and press Ctrl+L to examine the estimated execution plan for a query with multiple predicates.
22. Execute the code under the comment that begins **Step 9a** to show that a new statistics object has been created by AUTO_CREATE_STATISTICS.
23. Execute the code under the comment that begins **Step 9b** to view the histogram for the new statistics object.
24. Execute the code under the comment that begins **Step 9c** to demonstrate the cardinality calculation used to generate the estimated row count for the query under the heading **Step 9**.
25. Leave SSMS open for the next demonstration.

Lesson 2

Index Internals**Contents:**

Question and Answers	6
Demonstration: Picking the Right Index Key	6

Question and Answers

Question: What does the abbreviation “HoBT” stand for?

- () Heap or bulk table
- () Heap or b-tree
- () Head of block table
- () Highly organized block tree

Answer:

- () Heap or bulk table
- (√) Heap or b-tree
- () Head of block table
- () Highly organized block tree

Demonstration: Picking the Right Index Key

Demonstration Steps

1. Ensure that the **MT17B-WS2016-NAT**, **10987C-MIA-DC**, and **10987C-MIA-SQL** virtual machines are running, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **Demo.ssmssl** solution, in Solution Explorer, open the **Demo 2 - indexes.sql** script file.
3. Execute the code under the comment that begins **Step 1** to use the **AdventureWorks** database.
4. Execute the code under the comment that begins **Step 2** to create a table with a GUID clustered primary key.
5. Execute the code under the comment that begins **Step 3** to insert 10,000 rows into the new table.
6. Execute the code under the comment that begins **Step 4** to view index fragmentation.
7. Execute the code under the comment that begins **Step 5** to view the page sequence of the new table.
8. Execute the code under the comment that begins **Step 6** to create a table with an integer IDENTITY primary key.
9. Execute the code under the comment that begins **Step 7** to insert 10,000 rows into the new table.
10. Execute the code under the comment that begins **Step 8** to view index fragmentation.
11. Execute the code under the comment that begins **Step 9** to view the page sequence of the new table; fragmentation is much lower.
12. Leave SSMS open for the next demonstration.

Lesson 3

Columnstore Indexes**Contents:**

Question and Answers	8
Resources	8
Demonstration: Implementing a Columnstore Index	8

Question and Answers

Question: On which of the following table types could you not create a nonclustered columnstore index?

- () A heap
- () A table with a row-based clustered index
- () An in-memory optimized table without any indexes
- () A table with a columnstore clustered index
- () A temporary table

Answer:

- () A heap
- () A table with a row-based clustered index
- () An in-memory optimized table without any indexes
- (√) A table with a columnstore clustered index
- () A temporary table

Resources

What Is a Columnstore Index?



Best Practice: Columnstore indexes are most effective on tables with many millions of rows, where the column values have low selectivity and can be grouped into sets that can be effectively compressed. Columnstore indexes are not suitable for tables with many highly-selective or unique data values.

Demonstration: Implementing a Columnstore Index

Demonstration Steps

1. Ensure that the **MT17B-WS2016-NAT**, **10987C-MIA-DC**, and **10987C-MIA-SQL** virtual machines are running, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. If it is not already running, start **SQL Server Management Studio** and connect to the **MIA-SQL** database engine instance using SQL Server authentication.
3. If it is not already open, open the **Demo.ssmssl** solution in the **D:\Demofiles\Mod06\Demo** folder. If the Solution Explorer pane is not visible, on the **View** menu, click **Solution Explorer**.
4. Open the **Demo 3 - columnstore.sql** script file.
5. Execute the code under the comment that begins **Step 1** to use the **AdventureWorks** database.
6. Execute the code under the comment that begins **Step 2** to create a table with a clustered columnstore index.
7. Execute the code under the comment that begins **Step 3** to add 10 rows to the table.
8. Execute the code under the comment that begins **Step 4** to examine the table rowgroups.
9. Execute the code under the comment that begins **Step 5** to insert 1.1 million rows in blocks of 1,000 rows. This will close the deltastore.
10. Execute the code under the comment that begins **Step 6** to examine the table rowgroups again.

11. Execute the code under the comment that begins **Step 7** to bulk insert 2 million rows into the table in a single step. This will bypass the deltastore.
12. Execute the code under the comment that begins **Step 8** to examine the rowgroups again.
13. Execute the code under the comment that begins **Step 9** to examine the rowgroup segments.
14. Execute the code under the comment that begins **Step 10** to delete 10 rows from the table.
15. Execute the code under the comment that begins **Step 11** to examine the deltastore and deleted bitmap.
16. Execute the code under the comment that begins **Step 12** to drop the table.
17. Close SSMS without saving changes to any files.

Module Review and Takeaways

Categorize each factor by the index type that best addresses it. Indicate your answer by writing the category number to the right of each item.

Items	
1	Highly selective data
2	Data with low selectivity
3	Many queries for single values
4	Many aggregate and range queries
5	Replication will be used
6	Large data volume (millions or billions of rows)

Category 1		Category 2
Row-based index		Columnstore index

Answer:

Category 1	Category 2
Row-based index	Columnstore index
Highly selective data Many queries for single values Replication will be used	Data with low selectivity Many aggregate and range queries Large data volume (millions or billions of rows)

Lab Review Questions and Answers

Lab: Statistics and Index Internals

Question and Answers

Lab Review

Question: When AUTO_UPDATE_STATISTICS is on, approximately what percentage of a table's data must change to prompt a statistics update?

- ☐ 5 percent
- ☐ 10 percent
- ☐ 15 percent
- ☐ 20 percent
- ☐ 25 percent

Answer:

- ☐ 5 percent
- ☐ 10 percent
- ☐ 15 percent
- ☒ 20 percent
- ☐ 25 percent

Module 7

Query Execution and Query Plan Analysis

Contents:

Lesson 1: Query Execution and Query Optimizer Internals	2
Lesson 2: Query Execution Plans	5
Lesson 3: Analyzing Query Execution Plans	9
Lesson 4: Adaptive Query Processing	12
Module Review and Takeaways	14
Lab Review Questions and Answers	15

Lesson 1

Query Execution and Query Optimizer Internals

Contents:

Question and Answers	3
Resources	3
Demonstration: Analyzing Query Optimizer Internals	3

Question and Answers

Question: At which of the following optimization phases will a plan be selected for a simple query, such as `SELECT * FROM Sales.Customer`?

- ☐ Simplification
- ☐ Trivial plan
- ☐ Full optimization search 0
- ☐ Full optimization search 1
- ☐ Full optimization search 2

Answer:

- ☐ Simplification
- ☒ Trivial plan
- ☐ Full optimization search 0
- ☐ Full optimization search 1
- ☐ Full optimization search 2

Resources

The Query Optimizer



Additional Reading: For more information about database object statistics, see Module 6 of this course, *Statistics and Index Internals*.

Demonstration: Analyzing Query Optimizer Internals

Demonstration Steps

1. Ensure that the **10987C-MIA-DC** and **10987C-MIA-SQL** virtual machines are running and log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Run **Setup.cmd** in the **D:\Demofiles\Mod07** folder as Administrator.
3. In the **User Account Control** dialog box, click **Yes**, wait until the script completes, and then press any key.
4. Start **SQL Server Management Studio** and connect to the **MIA-SQL** database engine instance using Windows authentication.
5. Open the **Demo.ssmssl** solution in the **D:\Demofiles\Mod07** folder.
6. In Solution Explorer, double-click the **Demo 1 - query optimizer.sql** script file.
7. Select the query under the comment that begins **Module 7 - Demo 1**, and click **Execute**.
8. Select the query under the comment that begins **Step 1** and on the **Query** menu, click **Display Estimated Execution Plan**. Note that no reference to **Sales.SalesOrderHeader** appears in the plan.
9. Select the query under the comment that begins **Step 2** and on the **Query** menu, click **Display Estimated Execution Plan**. The plan is simplified because a check constraint prevents any data from matching the filter.

10. On the **Query** menu, click **Include Actual Execution Plan**, and then select the query under the comment that begins **Step 3**, and click **Execute**.
11. On the **Execution plan** tab, click the **SELECT** operator, and then in the Properties pane, note that **Optimization Level** is **TRIVIAL**.
12. Select the query under the comment that begins **Step 4**, and click **Execute**. This returns a result set showing all the transformation rules used in producing the query plan.
13. Keep SQL Server Management Studio open for the next demonstration.

Lesson 2

Query Execution Plans

Contents:

Question and Answers	6
Resources	7
Demonstration: Capturing Query Execution Plans	7

Question and Answers

Place each SET option into the appropriate category. Indicate your answer by writing the category number to the right of each item.

Items	
1	SHOWPLAN_XML
2	STATISTICS XML
3	SHOWPLAN_TEXT
4	STATISTICS Profile
5	SHOWPLAN_ALL

Category 1	Category 2
Estimated Execution Plan	Actual Execution Plan

Answer:

Category 1	Category 2
Estimated Execution Plan	Actual Execution Plan
SHOWPLAN_XML SHOWPLAN_TEXT SHOWPLAN_ALL	STATISTICS XML STATISTICS Profile

Resources**Capturing Execution Plans**

Additional Reading: These methods are covered in more detail in Module 8 of this course, *Plan Caching and Recompilation*.

Demonstration: Capturing Query Execution Plans**Demonstration Steps**

1. In SQL Server Management Studio, in Solution Explorer, double-click the **Demo 2 - Capture Execution Plan.sql** script file.
2. Select the query under the comment that begins **Module 7 – Demo 2**, and click **Execute**.
3. Select the query under the comment that begins **Step 1** and on the **Query** menu, click **Display Estimated Execution Plan**. Click on each operator to examine its properties in the Properties pane.
4. Click the query pane, and then on the **Query** menu, click **Include Actual Execution Plan**.
5. Select the query under the comment that begins **Step 2**, and click **Execute**.
6. On the **Execution plan** tab, examine the execution plan. Click on each operator to examine its properties.
7. Right-click on the execution plan and click **Show Execution Plan XML** to review the XML plan.
8. In the **Demo 2 - Capture Execution Plan** query window, right-click the actual execution plan generated in the previous step, and select **Save Execution Plan As**. Save the plan as **D:\Demofiles\Mod07\demo2.sqlplan**.
9. On the **File** menu, point to **Open**, and then click **File**. In the **Open File** dialog box, select **D:\Demofiles\Mod07\demo2.sqlplan**, and then click **Open**. The plan will open in a new SSMS pane. Review the plan, and note that it was opened in a graphical format.

10. In the **Demo 2 - Capture Execution Plan** query window, on the **Query** menu, click **Include Actual Execution Plan**.
11. On the **Query** menu, click **Include Live Query Statistics**. Select the query under the comment that begins **Step 4**, and click **Execute**. Watch the live query statistics. This query will take some time to complete; you can stop it before it finishes when you are ready to move on.
12. On the **Query** menu, click **Cancel Executing Query**.
13. On the **Query** menu, click **Include Live Query Statistics** to disable Live Query Statistics.
14. Select the query under the comment that begins **Step 5**, and click **Execute** to demonstrate an XML estimated execution plan.
15. In the Results pane, click the XML result; it will open in a new SSMS pane as a graphical plan. Right-click the plan and click **Show Execution Plan XML** to review the XML.
16. In the **Demo 2 - Capture Execution Plan** query window, select the query under the comment that begins **Step 6**, and click **Execute** to demonstrate an XML actual execution plan.
17. In the Results pane, click the XML result; it will open in a new SSMS pane as a graphical plan. Right-click the plan and click **Show Execution Plan XML** to review the XML.
18. In the **Demo 2 - Capture Execution Plan** query window, select the query under the comment that begins **Step 7**, and click **Execute** to review a text estimated execution plan generated by SHOWPLAN.
19. Select the query under the comment that begins **Step 8**, and click **Execute** to review a text estimated execution plan generated by SHOWPLAN_ALL.
20. Select the query under the comment that begins **Step 9**, and click **Execute** to review a text actual execution plan.
21. Leave SSMS open for the next demonstration.

Lesson 3

Analyzing Query Execution Plans

Contents:

Question and Answers	10
Resources	10
Demonstration: Working with Query Execution Plans	10

Question and Answers

Question: Which query execution plan join operator requires that both input data streams are sorted?

- ☐ Merge Join
- ☐ Hash Match
- ☐ Nested Loops

Answer:

- ☒ Merge Join
- ☐ Hash Match
- ☐ Nested Loops

Resources

Parallel Query Execution Plans



Additional Reading: For more information about how SQL Server operates on multiprocessor systems, see Module 1 of this course: *SQL Server Architecture, Scheduling, and Waits*.

Demonstration: Working with Query Execution Plans

Demonstration Steps

1. In SQL Server Management Studio, in Solution Explorer, double-click the **Demo 3 - Working with plans.sql** script file.
2. Select the query under the comment that begins **Module 7 – Demo 3**, and click **Execute**.
3. Select the query under the comment that begins **Step 1** and, on the **Query** menu, click **Display Estimated Execution Plan**. Note the warning indicator.
4. Right-click the plan generated in the previous step, and then click **Compare Showplan**.
5. In the **Open** dialog box, click **D:\Demofiles\Mod07\demo2.sqlplan**, and then click **Open** to display two plans together.
6. In the **Demo 3 - Working with plans.sql** query window, select the query under the comment that begins **Step 3**, and on the **Query** menu, click **Display Estimated Execution Plan**. Note that both statements have the same estimated query plan.
7. On the **Query** menu, click **Include Actual Execution Plan**. Select the query under the comment that begins **Step 3**, and click **Execute**.
8. On the **Execution plan** tab, note that the actual execution plans are the same. Hover over the **Clustered Index Seek** of the Orders table (no statistics) icon and note the warning. Also note the discrepancy between the actual and estimated number of rows for the Clustered Index Seek of the Orders table.
9. Select the query under the comment that begins **Step 4**, and click **Execute** to generate a statistics object.

10. Select the query under the comment that begins **Step 3**, and on the **Query** menu, click **Display Estimated Execution Plan**. Notice that the missing statistics warning is no longer shown. If the warning is still there, execute the query again.
11. Select the query under the comment that begins **Step 5**, and click **Execute** and examine the actual execution plan.
12. On the **Execution plan** tab, note the **Estimated Subtree Cost** from the SELECT operator, and the missing index suggestion.
13. Select the query under the comment that begins **Step 6**, and click **Execute** to create a covering index for the query.
14. Select the query under the comment that begins **Step 7**, and click **Execute** and examine the actual execution plan.
15. On the **Execution plan** tab, note that the **Estimated Subtree Cost** from the SELECT operator has reduced.
16. Close SQL Server Management Studio without saving any changes.

Lesson 4

Adaptive Query Processing

Contents:

Question and Answers

13

Question and Answers

Question: Which is a benefit of adaptive query processing?

- () Better optimized queries for SQL Server 2016 and later
- () Faster queries that could use hash joins or merge joins
- () Queries that have incorrect memory grants stored in the execution plan
- () Queries that do not make sufficient use of indexes

Answer: Adaptive query processing benefits queries that have incorrect memory grants stored in the execution plan.

Module Review and Takeaways

Best Practice

Use execution plans to find high-cost operations which may be limiting the performance of your queries.

Review Question(s)

Question: In which direction is the flow of data in a graphical query execution plan?

- ☐ Left to right
- ☐ Top to bottom
- ☐ Bottom to top
- ☐ Right to left

Answer:

- ☐ Left to right
- ☐ Top to bottom
- ☐ Bottom to top
- ☒ Right to left

Lab Review Questions and Answers

Lab: Query Execution and Query Plan Analysis

Question and Answers

Lab Review

Question: How might you determine that a performance problem is due to a query execution plan, not a server-level resource problem (such as I/O, CPU or memory)?

Answer: You could check the wait statistics and Windows performance counters. If these show no sign of unexpected I/O, CPU, or memory pressure, but individual queries still perform poorly, it is most likely that you have a performance problem related to query execution plans.

Module 8

Plan Caching and Recompilation

Contents:

Lesson 1: Plan Cache Internals	2
Lesson 2: Troubleshooting with the Plan Cache	5
Lesson 3: Automatic Tuning	8
Lesson 4: Query Store	10
Module Review and Takeaways	13
Lab Review Questions and Answers	14

Lesson 1

Plan Cache Internals

Contents:

Question and Answers	3
Resources	3
Demonstration: Analyzing the Query Plan Cache	3

Question and Answers

Question: In which plan cache store are query execution plans for ad hoc Transact-SQL statements cached?

- () Object Plans
- () Bound Trees
- () Extended Stored Procedures
- () SQL Plans

Answer:

- () Object Plans
- () Bound Trees
- () Extended Stored Procedures
- (v) SQL Plans

Resources



Additional Reading: For more details of the process used by the query optimizer to compile a query execution plan for a Transact-SQL statement, see Module 7 of this course, *Query Execution and Query Plan Analysis*.

Demonstration: Analyzing the Query Plan Cache

Demonstration Steps

1. Ensure that the **MT17B-WS2016-NAT**, **10987C-MIA-DC**, and **10987C-MIA-SQL** virtual machines are running, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Run **Setup.cmd** in the **D:\Demofiles\Mod08** folder as Administrator. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.
3. Start **SQL Server Management Studio** (SSMS) and connect to the **MIA-SQL** database engine instance using Windows authentication.
4. Open the **Demo.ssmssl** solution in the **D:\Demofiles\Mod08** folder.
5. In Solution Explorer, expand the queries folder and open the **Demo 1 - plan cache.sql** script file.
6. Execute the query under the comment that begins **Step 1** to view high level information about the plan cache stores.
7. Execute the query under the comment that begins **Step 2** to show the hash bucket count in each plan cache store.
8. Execute the query under the comment that begins **Step 3** to show the attributes that make up a plan cache key.
9. Execute the query under the comment that begins **Step 4** to show cost information about query plans in the **SQL Plans** and **Object Plans** plan cache stores.
10. Execute the query under the comment that begins **Step 5** to illustrate clearing the plan cache using **DBCC FREEPROCCACHE**. Emphasize that this should only be done on nonproduction systems.

11. Execute the query under the comment that begins **Step 6** to execute a stored procedure to add a plan to the cache.
12. Execute the query under the comment that begins **Step 7** to examine the cached plan for **uspGetOrderTrackingBySalesOrderID**. Click on the XML value returned in the **query_plan** column to show a graphical query plan.
13. In the **Demo 1 - plan cache.sql** pane, execute the query under the comment that begins **Step 8** to change set options for this session.
14. Execute the query under the comment that begins **Step 9** to execute the stored procedure again.
15. Execute the query under the comment that begins **Step 7** to examine the cached plan for **uspGetOrderTrackingBySalesOrderID** again.
16. Execute the query under the comment that begins **Step 11** to recompile the stored procedure.
17. Execute the query under the comment that begins **Step 7** to examine the cached plan for **uspGetOrderTrackingBySalesOrderID** again—there will be no plans in the cache.
18. Execute the query under the comment that begins **Step 13** to execute a Transact-SQL statement that will be auto-parameterized.
19. Execute the query under the comment that begins **Step 14** to examine the cached plan for the statement you just ran. Click on the XML value returned in the **query_plan** column to show a graphical query plan.

Notice that the graphical plan looks unusual—it has only a SELECT operator.
20. Right-click the graphical query plan and click **Show Execution Plan XML**. Notice from the value of the **ParameterizedText** attribute that the query plan has been parameterized. Notice from the value of the **ParameterizedPlanHandle** attribute that this plan references another plan handle. Copy the value of the **ParameterizedPlanHandle** attribute.
21. In the **Demo 1 - plan cache.sql** pane, edit the query under the comment that begins **Step 15** to replace **<plan handle here>** with the value of the plan handle that you copied in the previous step. Execute the query, and click on the XML value returned in the **query_plan** column to show a graphical query plan. Notice that this is the full plan.
22. Keep SSMS open for the next demonstration

Lesson 2

Troubleshooting with the Plan Cache

Contents:

Question and Answers	6
Resources	7
Demonstration: Troubleshooting Ad Hoc Plan Caching	7

Question and Answers

Place each DMV or DMF into the appropriate category, based on the information it returns. Indicate your answer by writing the category number to the right of each item.

Items	
1	sys.dm_exec_cached_plans
2	sys.dm_exec_query_text
3	sys.dm_exec_query_stats
4	sys.dm_exec_query_plan
5	sys.dm_exec_procedure_stats
6	sys.dm_exec_text_query_plan

Category 1		Category 2		Category 3
Query Execution Plan		Query Text		Execution Statistics

Answer:

Category 1		Category 2		Category 3
Query Execution Plan		Query Text		Execution Statistics
sys.dm_exec_cached_plans sys.dm_exec_query_plan sys.dm_exec_text_query_plan		sys.dm_exec_query_text		sys.dm_exec_query_stats sys.dm_exec_procedure_stats

Resources

Query Execution Plan Recompilation



Additional Reading: For more details of the process used by the query optimizer to compile a query execution plan for a Transact-SQL statement, see Module 7 of this course, *Query Execution and Query Plan Analysis*.

Demonstration: Troubleshooting Ad Hoc Plan Caching

Demonstration Steps

1. In SSMS, expand **Queries**, and then open the **Demo 2 - ad-hoc workloads.sql** script file.
2. Execute the statement under the comment that begins **Step 1** to clear the plan cache.
3. Execute the statement under the comment that begins **Step 2** to execute three similar queries. Notice that the only difference between the first and second SELECT queries is an additional space before "43667" in the WHERE clause.
4. Execute the query under the comment that begins **Step 3** to examine the plan cache. Notice that the **query_hash** value is the same for all the plans.
5. Execute the query under the comment that begins **Step 4** to check the current value of the **optimize for ad-hoc workloads** setting. It should be off.
6. Execute the query under the comment that begins **Step 5** to turn on the **optimize for ad-hoc workloads** setting.
7. Execute the query under the comment that begins **Step 1**.
8. Execute the query under the comment that begins **Step 7** to examine the plan cache. Notice that **query_plan** is NULL; only a stub has been added to the plan cache.
9. Execute the query under the comment that begins **Step 8** to rerun one of the SELECT statements from step 2. It is important that you highlight all the text here up to and including GO (but no further) so that the query text exactly matches the first execution.
10. Execute the query under the comment that begins **Step 9** to examine the plan cache. Notice that one of the cached **query_plan** values is no longer NULL. The stub has been removed and a full plan added to the plan cache.
11. Execute the query under the comment that begins **Step 10** to return the **optimize for ad-hoc workloads** setting to its default value.
12. Leave SSMS open for the next demonstration.

Lesson 3

Automatic Tuning

Contents:

Question and Answers

9

Question and Answers

Question: In your organization, how much time is spent trying to fix poor performance after query plans have changed?

Answer: Answers will vary

Lesson 4

Query Store

Contents:

Question and Answers	11
Demonstration: Working with the Query Store	11

Question and Answers

Question: Which Query Store report contains Transact-SQL statements that show a trend of reduced performance over time?

- () Overall Resource Consumption
- () Tracked Queries
- () Top Resource Consuming Queries
- () Regressed Queries

Answer:

- () Overall Resource Consumption
- () Tracked Queries
- () Top Resource Consuming Queries
- (v) Regressed Queries

Demonstration: Working with the Query Store

Demonstration Steps

1. In SSMS, open the **Demo 3 - query store.sql** script file.
2. In Object Explorer, expand **Databases**, and then expand **TSQL** to show that it has no **Query Store** node.
3. Execute the code under the comment that begins **Step 2** to switch on and configure the Query Store.
4. To start a workload, in File Explorer, right-click **D:\Demofiles\Mod08\start_load_1.ps1**, and then click **Run with PowerShell**. If a message is displayed asking you to confirm a change in execution policy, type **Y**, and then press Enter. Once the workload has started, continue with the demo.
5. Switch to SQL Server Management Studio.
6. In Object Explorer, right-click **TSQL**, and then click **Properties**. On the **Query Store** page, show that the **Operation Mode (Actual)** option is now set to **Read Write**, then click **Cancel**.
7. In Object Explorer, refresh the list of objects under the **TSQL** database to display the new Query Store node. Expand the **Query Store** node to show the queries being tracked.
8. Execute the code under the comment that begins **Step 6** to expand Query Store storage.
9. In Object Explorer, under **Query Store**, double-click **Overall Resource Consumption**. In the report pane, click **Configure** (top right). In the **Time Interval** section, in the first drop-down box, select **Last hour**, and then click **OK**. You should see some bars at the right-hand side of each graph caused by the workload.
10. In Object Explorer, double-click **Top Resource Consuming Queries**. In the **Metric** drop-down box (top left), select **Execution Count**. The tallest bar in the list should be for the workload query with text starting "SELECT so.custid, so.orderdate, so.orderid, so.shipaddress...". If it is not the tallest bar, locate the bar for this query. Note the query id (visible either on the x-axis of the chart, or in the tooltip shown by hovering over the relevant bar on the chart).
11. In Object Explorer, double-click **Tracked Queries**. In the **Tracking query** box, type the query id you identified in the previous step and then press Enter. This shows the query plan history for the query.

12. In the **Demo 3 - query store.sql** pane, execute the code under the comment that begins **Step 10** to create a temp table and double the number of rows in the **Sales.Orders** table (this should prompt a statistics update and a new query plan).
13. Wait for approximately 60 seconds.
14. After 60 seconds have passed, in the **Tracked Queries** pane, click **Refresh**. You should see that a new query plan has been compiled. If you do not see a new plan, rerun the (INSERT, SELECT, FROM) statements from the previous step and check again.
15. In Object Explorer, double-click **Regressed Queries**. In the report, click **Configure** (top right). In the **Time Interval** section, in the **Recent** drop-down, select **Last 5 minutes**, and then press Enter. The SELECT statement with text starting "SELECT so.custid, so.orderdate, so.orderid, so.shipaddress..." should appear in the report.
16. In the **Tracked Queries** pane, select one of the query plans (the dots shown on the graph), and then click **Force Plan**. In the **Confirmation** dialog box, click **Yes**.
17. In the **Top Resource Consumers** report pane, click **Refresh**. Notice that executions using the forced plan have a tick in the **Tracked Queries** scatter graph.
18. Return to the **Tracked Queries** report. Click the ticked dot (representing the forced plan), then click **Unforce Plan**. In the Confirmation window, click **Yes**.
19. In the **Demo 3 - query store.sql** pane, execute the code under the comment that begins **Step 16** to stop the workload.
20. Close SSMS without saving changes. Press ENTER in the PowerShell® workload window.

Module Review and Takeaways

Question: Which Query Store configuration option determines the frequency with which the Query Store writes data to disk?

- ☐ OPERATION_MODE
- ☐ INTERVAL_LENGTH_MINUTES
- ☐ DATA_FLUSH_INTERVAL_SECONDS
- ☐ CLEANUP_POLICY

Answer:

- ☐ OPERATION_MODE
- ☐ INTERVAL_LENGTH_MINUTES
- ☒ DATA_FLUSH_INTERVAL_SECONDS
- ☐ CLEANUP_POLICY

Lab Review Questions and Answers

Lab: Plan Caching and Recompilation

Question and Answers

Lab Review

Question: Under what circumstances might you consider forcing a query plan on a production SQL Server instance?

Answer: Generally, unless you have found an edge case where no alternative is available, you should let the query optimizer select an optimal query plan.

Module 9

Extended Events

Contents:

Lesson 1: Extended Events Core Concepts	2
Lesson 2: Working with Extended Events	5
Module Review and Takeaways	9
Lab Review Questions and Answers	10

Lesson 1

Extended Events Core Concepts

Contents:

Question and Answers	3
Demonstration: Creating an Extended Events Session	3

Question and Answers

Question: Which system DMV provides the list of events configured in an active Extended Events session?

- () sys.dm_xe_session_targets
- () sys.dm_xe_session_events
- () sys.dm_xe_sessions
- () sys.dm_xe_session_event_actions

Answer:

- () sys.dm_xe_session_targets
- (√) sys.dm_xe_session_events
- () sys.dm_xe_sessions
- () sys.dm_xe_session_event_actions

Demonstration: Creating an Extended Events Session

Demonstration Steps

1. Start the **MT17B-WS2016-NAT**, **10987C-MIA-DC**, and **10987C-MIA-SQL** virtual machines are running, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Demofiles\Mod09** folder, run **Setup.cmd** as Administrator.
3. Click **Yes** in the User Account Control window and wait for the script to finish.
4. Start **SQL Server Management Studio** and connect to the **MIA-SQL** database engine instance using Windows authentication.
5. On the **File** menu, point to **Open**, and then click **Project/Solution**. In the **Open Project** dialog box, navigate to the **D:\Demofiles\Mod09** folder, click **Demo.ssmssl**, and then click **Open**.
6. In Solution Explorer, double-click **Demo 1 – create xe session.sql**.
7. Select code under the comment that begins -- **Step 1**, and then click **Execute** to create an Extended Events session.
8. Select code under the comment that begins -- **Step 2**, and then click **Execute** to verify that the session metadata is visible.
9. Select code under the comment that begins -- **Step 3**, and then click **Execute** to start the session and execute some queries.
10. Select code under the comment that begins -- **Step 4**, and then click **Execute** to query the session data.
11. Select code under the comment that begins -- **Step 5**, and then click **Execute** to refine the session data query.
12. In Object Explorer, under **MIA-SQL (SQL Server 13.0.1000 - ADVENTUREWORKS\Student)**, expand **Management**, expand **Extended Events**, and then expand **Sessions**.
13. Expand **SqlStatementCompleted**, and then double-click **package0.ring_buffer**.
14. In the **Data** column, click the XML value, and note that this is the same data that is returned by the query under the comment that begins **Step 4** (note that additional statements will have been captured because you ran the code earlier).

15. In Object Explorer, right-click **SqlStatementCompleted**, and then click **Watch Live Data**.
16. In the **Demo 1 – create xe sessions.sql** query pane, select the code under the comment that begins **-- Step 7**, and then click **Execute** to execute some SQL statements.
17. Return to the **MIA-SQL – SqlStatementCompleted: Live Data** pane. Wait for the events to be captured and displayed; this can take a few seconds. Other SQL statements from background processes might be captured by the session.
18. In the **Demo 1 – create xe sessions.sql** query pane, select the code under the comment that begins **-- Step 8**, and then click **Execute** to stop the session.
19. In Object Explorer, right-click **SqlStatementCompleted**, and then click **Properties**. Review the settings on the **General**, **Events**, **Data Storage** and **Advanced** pages, if necessary referring back to the session definition under the comment that begins **-- Step 1**.
20. In the **Session Properties** dialog box, click **Cancel**.
21. Select the code under the comment that begins **-- Step 10**, and then click **Execute** to drop the session.
22. Keep SQL Server Management Studio open for the next demonstration.

Lesson 2

Working with Extended Events**Contents:**

Question and Answers	6
Demonstration: Tracking Session-Level Waits	7

Question and Answers

Place each Extended Events target type into the appropriate category. Indicate your answer by writing the category number to the right of each item.

Items	
1	Ring buffer target
2	Event file target
3	Histogram target
4	Event tracking for Windows target
5	Event pairing target
6	Event counter target

Category 1		Category 2
Written to Memory Buffers		Written to File on Disk

Answer:

Category 1	Category 2
Written to Memory Buffers	Written to File on Disk
Ring buffer target Histogram target Event pairing target Event counter target	Event file target Event tracking for Windows target

Demonstration: Tracking Session-Level Waits**Demonstration Steps**

1. In SSMS, in Solution Explorer, double-click **Demo 2 - track waits by session.sql**.
2. In Object Explorer, expand **Management**, expand **Extended Events**, right-click **Sessions**, and then click **New Session**.
3. In the **New Session** dialog box, on the **General** page, in the **Session name** box, type **Waits by Session**.
4. On the **Events** page, in the **Event library** box, type **wait**, and then, in the list below, double-click **wait_info** to add it to the **Selected events** list.
5. Click **Configure** to display the **Event configuration options** list.
6. In the **Event configuration options** list, on the **Global Fields (Actions)** tab, select the **session_id** check box.
7. On the **Filter (Predicate)** tab, click **Click here to add a clause**. In the **Field** list, click **sqlserver.session_id**, in the **Operator** list, click **>**, and then in the **Value** box, type **50**. This filter will exclude most system sessions from the session.
8. On the **Data Storage** page, click **Click here to add a target**. In the **Type** list, click **event_file**, in the **File name on server** box, type **D:\Demofiles\Mod09\waitbysession**, in the first **Maximum file size** box, type **5**, in the second **Maximum file size** box, click **MB**, and then click **OK**.
9. In Object Explorer, under **Sessions**, right-click **Waits by Session**, and then click **Start Session**.
10. In File Explorer, in the **D:\Demofiles\Mod09** folder, right-click **start_load_1.ps1**, and then click **Run with PowerShell**. If a message is displayed asking you to confirm a change in execution policy, type **Y**, and then press **ENTER**. Leave the workload to run for a minute or so before proceeding.
11. In SSMS, in the **Demo 2 - track waits by session.sql** pane, select the code under the comment that begins **--Step 14**, click **Execute**, and then review the results.

12. Select the code under the comment that begins -- **Step 15**, and then click **Execute** to stop and drop the session, and to stop the workload.
13. In File Explorer, in the **D:\Demofiles\Mod09** folder, note that one (or more) files with a name matching **waitbysession*.xel** have been created.
14. Close File Explorer, close SSMS without saving changes, and then in the Windows PowerShell® window, press **ENTER** twice to close the window.

Module Review and Takeaways

Question: Which of the following sources does **not** contain detailed information about Extended Events event definitions?

- () SQL Server Management Studio Extended Events GUI.
- () The DMV sys.dm_xe_objects.
- () Microsoft Docs.

Answer:

- () SQL Server Management Studio Extended Events GUI.
- () The DMV sys.dm_xe_objects.
- (v) Microsoft Docs.

Lab Review Questions and Answers

Lab: Extended Events

Question and Answers

Lab Review

Question: If an Extended Events session has no targets defined, how would you view the data generated by the session?

Answer: In SSMS, right-click the session name and click Watch Live Data.

Module 10

Monitoring, Tracing, and Baselines

Contents:

Lesson 1: Monitoring and Tracing	2
Lesson 2: Baselining and Benchmarking	4

Lesson 1

Monitoring and Tracing

Contents:

Question and Answers	3
Demonstration: Analyzing Performance Monitor Data	3

Question and Answers

Question: Which tool or feature might you use instead of SQL Server Profiler to find resource-intensive queries on a SQL Server instance?

- () SQL Trace
- () Performance Monitor
- () Extended Events
- () SQL Server Agent

Answer:

- () SQL Trace
- () Performance Monitor
- (v) Extended Events
- () SQL Server Agent

Demonstration: Analyzing Performance Monitor Data

Demonstration Steps

1. Ensure the **10987C-MIA-DC** and **10987C-MIA-SQL** virtual machines are running, and then log on to **10987C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Demofiles\Mod10** folder, right-click **Setup.cmd**, and then click **Run as Administrator**.
3. In the **User Account Control** dialog box, click **Yes**, wait for the script to finish, and then press Enter.
4. Start **SQL Server Management Studio**, and then connect to the **MIA-SQL** database engine instance by using Windows authentication.
5. In SQL Server Management Studio, in the **D:\Demofiles\Mod10\Demo01** folder, open the **CollectPerfMonData.sql** file, and then execute the script to create the **CollectPerfMonData** job.
6. In Windows Explorer, navigate to the **D:\Demofiles\Mod10\Demo01** folder, right-click **RunWorkload.cmd** and then click **Run as Administrator**.
7. In the **User Account Control** dialog box, click **Yes**, and wait for the script to finish.
8. In SQL Server Management Studio, in the **D:\Demofiles\Mod10\Demo01** folder, open the **AnalysisQueries.sql** file.
9. Select the code under **Step 1 - CPU Usage Trend**, click **Execute**, and then observe the **CPU usage %** counter value that is collected during the workload execution.
10. Select the code under **Step 2 - Memory usage trend**, click **Execute**, and then observe how the **Total Server Memory (KB)** increases over time, and then remains consistent.
11. Select the code under **Step 3 - Transaction throughput**, click **Execute**, and then observe the trend in the **Transactions/sec** counter value.
12. Select the code under **Step 4 - Transaction reads per second**, click **Execute**, and then observe the trend in the **Page reads/sec** counter value.
13. Select the code under **Step 5 - Cleanup job and database**, and then click **Execute** to clean up the database and agent job.
14. Close SQL Server Management Studio without saving any changes.

Lesson 2

Baselining and Benchmarking

Contents:

Question and Answers	5
Demonstration: Collecting Performance Data using DMVs	5

Question and Answers

Question: True or false? Performance Monitor is useful for capturing baseline data, such as wait statistics, SQLOS information, and cached query plans that cannot be obtained through other sources.

☐ True

☐ False

Answer:

☐ True

☒ False

Demonstration: Collecting Performance Data using DMVs

Demonstration Steps

1. Start SQL Server Management Studio, and then connect to the **MIA_SQL** database engine instance by using Windows authentication.
2. In the **D:\Demofiles\Mod10\Demo02** folder, open the **DMVDataCollection.sql** file.
3. Select the code under **Step 1 - Collect physical index stats**, and then click **Execute**. Note the index details, particularly the **avg_fragmentation_in_percent** column.
4. Select the code under **Step 2 - Return current executing queries**, and then click **Execute**. Note that the query returns all queries that are currently executing on the instance.
5. Select the code under **Step 3 - Return I/O usage**, and then click **Execute**. Note that the query returns I/O stats for all database files.
6. Close SQL Server Management Studio without saving changes.

