

OFFICIAL MICROSOFT LEARNING PRODUCT

20462C

**Administering Microsoft® SQL Server®
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.

© 2014 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at <http://www.microsoft.com/about/legal/en/us/IntellectualProperty/Trademarks/EN-US.aspx> are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners

Product Number: 20462C

Part Number (if applicable):

Released: 07/2014

Module 1

Introduction to SQL Server 2014 Database Administration

Contents:

Module Review and Takeaways

2

Module Review and Takeaways

Review Question(s)

Question: When might you use each of the management tools you explored in the lab?

Answer: Most SQL Server administration is performed using SQL Server Management Studio from a client workstation. However, you might choose to install sqlcmd on the server itself to assist in troubleshooting and configuration management in an emergency. You might also use sqlcmd to run scripts from client computers where SSMS is not installed, or in automated batch files. Windows PowerShell is increasingly used to provide a consistent management scripting shell across multiple services, and can be useful when you need to automate tasks that involve a combination of configuration management across Windows, SQL Server, and other applications.

Module 2

Installing and Configuring SQL Server 2014

Contents:

Module Review and Takeaways

2

Module Review and Takeaways

Review Question(s)

Question: What additional considerations do you think there are for installing additional named instances on a server where SQL Server is already installed?

Answer: Multiple instances can co-exist on a single server, but you must consider the impact on server resources. The workloads on each instance will compete for memory, CPU, and storage I/O resources; which can affect application performance.

Module 3

Working with Databases and Storage

Contents:

Lesson 2: Managing Storage for System Databases	2
Lesson 3: Managing Storage for User Databases	4
Lesson 4: Moving Database Files	6
Lesson 5: Configuring the Buffer Pool Extension	8
Module Review and Takeaways	10

Lesson 2

Managing Storage for System Databases

Contents:

Demonstration: Moving tempdb Files

3

Demonstration: Moving tempdb Files

Demonstration Steps

Move tempdb files

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Demofiles\Mod03 folder, run **Setup.cmd** as Administrator.
3. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
4. In Object Explorer, expand **Databases**, expand **System Databases**, and then right-click **tempdb** and click **Properties**.
5. In the **Database Properties** dialog box, on the **Files** page, note the current files and their location. Then click **Cancel**.
6. Open the **MovingTempdb.sql** script file in the **D:\Demofiles\Mod03** folder.
7. View the code in the script, and then click **Execute**. Note the message that is displayed after the code has run.
8. View the contents of **T:** and note that no files have been created in that location, because the SQL Server service has not yet been restarted.
9. In Object Explorer, right-click **MIA-SQL** and click **Restart**. When prompted to allow changes, to restart the service, and to stop the dependent SQL Server Agent service, click **Yes**.
10. View the contents of **T:** and note that the **tempdb.mdf** and **tempdb.ldf** files have been moved to this location.
11. Keep SQL Server Management Studio open for the next demonstration

Lesson 3

Managing Storage for User Databases

Contents:

Resources	5
Demonstration: Creating Databases	5

Resources

Altering User Databases



Additional Reading: For more information about database set options, see *ALTER DATABASE SET Options (Transact-SQL)* in SQL Server Books Online.

Demonstration: Creating Databases

Demonstration Steps

Create a Database by Using SQL Server Management Studio

1. Ensure that you have completed the previous demonstration. If not, start the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines, log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**, and run D:\Demofiles\Mod03\Setup.cmd as Administrator.
2. If SQL Server Management Studio is not open, start it and connect to the **MIA-SQL** database engine using Windows authentication.
3. In Object Explorer, right-click **Databases** and click **New Database**.
4. In the **Database name** box, type **DemoDB1**.
5. In the **Database files** list, note the default logical names, initial size, and autogrowth settings. Then change the **Path** and **File Name** by typing the following values:
 - DemoDB1
 - **Path:** M:\Data\
 - **File Name:** DemoDB1.mdf
 - DemoDB1_log
 - **Path:** L:\Logs\
 - **File Name:** DemoDB1.ldf
6. Click **OK** to create the new database.
7. Expand the **Databases** folder and then right-click **DemoDB1** and click **Properties**.
8. On the **Options** tab, review the database options. Then click **Cancel**.

Create a Database by Using the CREATE DATABASE Statement

1. In SQL Server Management Studio, open the **CreatingDatabases.sql** script file from the D:\Demofiles\Mod03 folder.
2. Select the code under the comment **Create a database** and click **Execute** to create a database named **DemoDB2**.
3. Select the code under the comment **View database info** and click **Execute**. Then view the information that is returned.
4. Keep SQL Server Management Studio open for the next demonstration.

Lesson 4

Moving Database Files

Contents:

Demonstration: Detaching and Attaching a Database

7

Demonstration: Detaching and Attaching a Database

Demonstration Steps

Detach a Database

1. Ensure that you have completed the previous demonstrations in this module, and that you have created a database named **DemoDB2**.
2. In Object Explorer, right-click the **Databases** folder and click **Refresh**; and verify that the **DemoDB2** database is listed.
3. Right-click **DemoDB2**, point to **Tasks**, and click **Detach**. Then in the **Detach Database** dialog box, select **Drop Connections** and **Update Statistics**, and click **OK**.
4. View the M:\Data and L:\Logs folders and verify that the DemoDB2.mdf and DemoDB2.ldf files have not been deleted.

Attach a Database

1. In SQL Server Management Studio, in Object Explorer, in the **Connect** drop-down list, click **Database Engine**. Then connect to the **MIA-SQL\SQL2** database engine using Windows authentication.
2. In Object Explorer, under **MIA-SQL\SQL2**, expand **Databases** and view the databases on this instance.
3. In Object Explorer, under **MIA-SQL\SQL2**, right-click **Databases** and click **Attach**.
4. In the **Attach Databases** dialog box, click **Add**. Then in the **Locate Database Files** dialog box, select the M:\Data\DemoBD2.mdf database file and click **OK**.
5. In the **Attach Databases** dialog box, after you have added the master databases file, note that all of the database files are listed. Then click **OK**.
6. In Object Explorer, under **MIA-SQL\SQL2**, under **Databases**, verify that **DemoDB2** is now listed.

Lesson 5

Configuring the Buffer Pool Extension

Contents:

Demonstration: Configuring the Buffer Pool Extension

9

Demonstration: Configuring the Buffer Pool Extension

Demonstration Steps

Enable the Buffer Pool Extension

1. Ensure that you have completed the previous demonstration. If not, start the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines, log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**, and run D:\Demofiles\Mod03\Setup.cmd as Administrator.
2. If SQL Server Management Studio is not open, start it and connect to the **MIA-SQL** database engine using Windows authentication.
3. Open the script file **ConfiguringBPE.sql** in the D:\Demofiles\Mod03 folder.
4. Review the code under the comment **Enable buffer pool extension**, and note that it creates a Buffer Pool Extension file named MyCache.bpe in S:\. On a production system, this file location would typically be on an SSD device.
5. Use File Explorer to view the contents of the S:\ folder and note that no MyCache.bpe file exists.
6. In SQL Server Management Studio, select the code under the comment **Enable buffer pool extension**, and click **Execute**.

Verify Buffer Pool Extension Configuration

1. View the contents of the S:\ folder and note that the **MyCache.bpe** file now exists.
2. In SQL Server Management Studio, select the code under the comment **View buffer pool extension details**, and click **Execute**. Then note the information about the Buffer Pool Extension that is returned from the dynamic management view.
3. Select the code under the comment **Monitor buffer pool extension**, and click **Execute**. This dynamic management view shows all buffered pages, and the **is_in_bpool_extension** column indicates pages that are stored in the Buffer Pool Extension.

Disable the Buffer Pool Extension

1. In SQL Server Management Studio, select the code under the comment **Disable buffer pool extension**, and click **Execute**.
2. Select the code under the comment **View buffer pool extension details**, and click **Execute**. Then note the information about the Buffer Pool Extension that is returned from the dynamic management view.
3. Use File Explorer to view the contents of the S:\ folder and note that the **MyCache.bpe** file has been deleted.

Module Review and Takeaways

Best Practice

When working with database storage, consider the following best practices:

Plan and test your file layout carefully.

Separate data and log files on the physical level.

Keep the data files of a database at the same size.

Create the database in an appropriate size so it doesn't have to be expanded too often.

Shrink files only if absolutely necessary.

Set a filegroup other than PRIMARY as the default filegroup.

Review Question(s)

Question: Why is it typically sufficient to have one log file in a database?

Answer: Log files are written sequentially. If more than one log file exists, SQL Server writes them in a circular manner, which doesn't provide any advantages according to performance and availability.

Question: Why should only temporary data be stored in the **tempdb** system database?

Answer: Because it is recreated with every new start of the instance.

Module 4

Planning and Implementing a Backup Strategy

Contents:

Lesson 1: Understanding SQL Server Recovery Models	2
Lesson 2: Planning a Backup Strategy	4
Lesson 3: Backing up Databases and Transaction Logs	6
Lesson 4: Using Backup Options	9
Lesson 5: Ensuring Backup Reliability	12
Module Review and Takeaways	14

Lesson 1

Understanding SQL Server Recovery Models

Contents:

Demonstration: Logs and Full Recovery

3

Demonstration: Logs and Full Recovery

Demonstration Steps

Observe Log File Behavior in the Full Recovery Model

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Demofiles\Mod04 folder, run **Setup.cmd** as Administrator.
3. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
4. In Object Explorer, expand **Databases**, right-click the **LogTest** database, and click **Properties**.
5. In the **Database Properties - LogTest** dialog box, on the **Options** page, verify that the **Recovery model** is set to **Full**. Then click **Cancel**.
6. Open the **LogsAndFullRecovery.sql** Transact-SQL file in the **D:\Demofiles\Mod04** folder.
7. Select the code comment **Perform a full database backup** and click **Execute**.
8. Select the code comment **View database file space** and click **Execute**. Note the space used in the **LogTest_log** file (and note that log files have a **type** value of **1**).
9. Select the code comment **Insert data** and click **Execute** to insert 5000 rows.
10. Select the code comment **View log file space** and click **Execute**. Note the space used in the **LogTest_log** file has increased.
11. Select the code comment **Issue a checkpoint** and click **Execute** force SQL Server to perform a checkpoint and flush the modified pages to disk.
12. Select the code comment **View log file space again** and click **Execute**. Note the space used in the **LogTest_log** file has not decreased.
13. Select the code comment **Check log status** and click **Execute**. Note that SQL Server is awaiting a log backup before the log file can be truncated.
14. Select the code comment **Perform a log backup** and click **Execute**.
15. Select the code comment **Verify log file truncation** and click **Execute**. Note the space used in the **LogTest_log** file has decreased because the log has been truncated.
16. Keep SQL Server Management Studio open for the next demonstration.

Lesson 2

Planning a Backup Strategy

Contents:

Resources

5

Resources

Partial Backup Strategies



Additional Reading: For more information about partial backups, see the topic Partial Backups (SQL Server) in SQL Server Books Online.

Lesson 3

Backing up Databases and Transaction Logs

Contents:

Demonstration: Performing Backups

7

Demonstration: Performing Backups

Demonstration Steps

Perform a Full Database Backup

1. Ensure that you have performed the previous demonstration in this module. If not, start the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines, log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**, and in the D:\Demofiles\Mod04 folder, run **Setup.cmd** as Administrator.
2. If SQL Server Management Studio is not already open, start it and connect to the **MIA-SQL** database engine using Windows authentication.
3. In Object Explorer, under **Databases**, right-click **AdventureWorks**, point to **Tasks**, and click **Back Up**.
4. In the **Backup Up Database – AdventureWorks** dialog box, ensure that **Backup type** is set to **Full**, and in the **Destination** section, select each existing file path and click **Remove**. Then click **Add** and in the **Select Backup Destination** dialog box, enter the file name **D:\Demofiles\Mod04\AW.bak** and click **OK**.
5. In the **Backup Up Database – AdventureWorks** dialog box, on the **Media Options** page, note that the default option is to append to an existing media set. In this case, there is no existing media set so a new one will be created, and there are no existing backup sets to overwrite.
6. In the **Backup Up Database – AdventureWorks** dialog box, on the **Backup Options** page, note the default backup name and expiration settings.
7. In the **Backup Up Database – AdventureWorks** dialog box, in the **Script** drop-down list, click **Script Action to a New Query Window**. Then click **OK**.
8. When the backup has completed successfully, click **OK**.
9. In the query pane, view the Transact-SQL BACKUP statement that was used to back up the database.
10. View the D:\Demofiles\Mod04 folder and note the size of the **AW.bak** file.

Perform a Differential Backup

1. In SQL Server Management Studio, open the **UpdatePrices.sql** script file from the D:\Demofiles\Mod04 folder and click **Execute**. This script updates the **Production.Product** table in the **AdventureWorks** database.
2. In Object Explorer, under **Databases**, right-click **AdventureWorks**, point to **Tasks**, and click **Back Up**.
3. In the **Backup Up Database – AdventureWorks** dialog box, in the **Backup type** list, select **Differential**. Then in the **Destination** section, ensure that **D:\Demofiles\Mod04\AW.bak** is the only backup device listed.
4. In the **Backup Up Database – AdventureWorks** dialog box, on the **Media Options** page, verify that the option to append to the existing media set is selected.
5. In the **Backup Up Database – AdventureWorks** dialog box, on the **Backup Options** page, change the **Name** to **AdventureWorks-Diff DatabaseBackup**.
6. In the **Backup Up Database – AdventureWorks** dialog box, in the **Script** drop-down list, click **Script Action to a New Query Window**. Then click **OK**.
7. When the backup has completed successfully, click **OK**.

8. In the query pane, view the Transact-SQL BACKUP statement that was used to back up the database. Note that it includes the WITH DIFFERENTIAL option.
9. View the D:\Demofiles\Mod04 folder and note that size of the **AW.bak** file has increased, but not much – the second backup only includes the extents containing pages that were modified since the full backup.

Perform a Transaction Log Backup

1. In SQL Server Management Studio, switch to the **UpdatePrices.sql** script you opened previously and click **Execute** to update the **Production.Product** table in the **AdventureWorks** database again.
2. In Object Explorer, under **Databases**, right-click **AdventureWorks**, point to **Tasks**, and click **Back Up**.
3. In the **Backup Up Database – AdventureWorks** dialog box, in the **Backup type** list, select **Transaction Log**. Then in the **Destination** section, ensure that **D:\Demofiles\Mod04\AW.bak** is the only backup device listed.
4. In the **Backup Up Database – AdventureWorks** dialog box, on the **Media Options** page, verify that the option to append to the existing media set is selected. Also verify that the option to truncate the transaction log is selected.
5. In the **Backup Up Database – AdventureWorks** dialog box, on the **Backup Options** page, change the **Name** to **AdventureWorks-Transaction Log Backup**.
6. In the **Backup Up Database – AdventureWorks** dialog box, in the **Script** drop-down list, click **Script Action to a New Query Window**. Then click **OK**.
7. When the backup has completed successfully, click **OK**.
8. In the query pane, view the Transact-SQL BACKUP statement that was used to back up the database. Note that this time the statement is BACKUP LOG.
9. View the D:\Demofiles\Mod04 folder and note that size of the **AW.bak** file has increased, but not much – the third backup only includes the transaction log entries for data modifications since the full backup.
10. Keep SQL Server Management Studio open for the next demonstration.

Lesson 4

Using Backup Options

Contents:

Demonstration: Using Backup Compression	10
Demonstration: Using Backup Encryption	10

Demonstration: Using Backup Compression

Demonstration Steps

Use Backup Compression

1. Ensure that you have performed the previous demonstration in this module.
2. In SQL Server Management Studio, in Object Explorer, under **Databases**, right-click **AdventureWorks**, point to **Tasks**, and click **Back Up**.
3. In the **Backup Up Database – AdventureWorks** dialog box, ensure that **Backup type** is set to **Full**, and in the **Destination** section, select the existing file path and click **Remove**. Then click **Add** and in the **Select Backup Destination** dialog box, enter the file name **D:\Demofiles\Mod04\AW_Comp.bak** and click **OK**.
4. In the **Backup Up Database – AdventureWorks** dialog box, on the **Media Options** page, note that the default option is to append to an existing media set. In this case, there is no existing media set so a new one will be created, and there are no existing backup sets to overwrite.
5. In the **Backup Up Database – AdventureWorks** dialog box, on the **Backup Options** page, change the **Name** to **AdventureWorks-Compressed Backup** and in the **Set backup compression** list, select **Compress backup**.
6. In the **Backup Up Database – AdventureWorks** dialog box, in the **Script** drop-down list, click **Script Action to a New Query Window**. Then click **OK**.
7. When the backup has completed successfully, click **OK**.
8. In the query pane, view the Transact-SQL BACKUP statement that was used to back up the database, noting that the COMPRESSION option was specified.
9. View the D:\Demofiles\Mod04 folder and note the size of the **AW_Comp.bak** file. This should be significantly smaller than the **AW.bak** file was after the full database backup in the previous demonstration.
10. Keep SQL Server Management Studio open for the next demonstration.

Demonstration: Using Backup Encryption

Demonstration Steps

Create a Database Master Key

1. Ensure that you have performed the previous demonstration in this module.
2. In SQL Server Management Studio, open the **EncyptionKeys.sql** script file in the D:\Demofiles\Mod04 folder.
3. Select the code under the comment **Create a database master key** and click **Execute**.
4. Select the code under the comment **Back up the database master key** and click **Execute**.

Create a Certificate

1. Select the code under the comment **Create a certificate** and click **Execute**.
2. Select the code under the comment **Back up the certificate and its private key** and click **Execute**.

Encrypt a Database Backup

1. In Object Explorer, under **Databases**, right-click **AdventureWorks**, point to **Tasks**, and click **Back Up**.
2. In the **Backup Up Database – AdventureWorks** dialog box, ensure that **Backup type** is set to **Full**, and in the **Destination** section, select the existing file path and click **Remove**. Then click **Add** and in the **Select Backup Destination** dialog box, enter the file name **D:\Demofiles\Mod04\AW_Encrypt.bak** and click **OK**.
3. In the **Backup Up Database – AdventureWorks** dialog box, on the **Media Options** page, select **Back up to a new media set, and erase all existing backup sets**. Then enter the new media set name **Encrypted Backup**.
4. In the **Backup Up Database – AdventureWorks** dialog box, on the **Backup Options** page, change the **Name** to **AdventureWorks-Encrypted Backup** and in the **Set backup compression** list, select **Compress backup**.
5. In the **Encryption** section, select **Encrypt backup**. Then ensure that the **AES 128** algorithm is selected, and select the **BackupCert** certificate you created previously.
6. In the **Backup Up Database – AdventureWorks** dialog box, in the **Script** drop-down list, click **Script Action to a New Query Window**. Then click **OK**.
7. When the backup has completed successfully, click **OK**.
8. In the query pane, view the Transact-SQL BACKUP statement that was used to back up the database, noting that the ENCRYPTION option was specified.
9. Keep SQL Server Management Studio open for the next demonstration.

Lesson 5

Ensuring Backup Reliability

Contents:

Resources	13
Demonstration: Verifying Backups	13

Resources

Options for Ensuring Backup Integrity



Best Practice: Consider verifying backups on a different system to the one where the backup was performed. This will eliminate the situation where a backup is only readable on the source hardware.

Demonstration: Verifying Backups

Demonstration Steps

View the Backup and Restore Events Report

1. Ensure that you have performed the previous demonstration in this module.
2. In SQL Server Management Studio, in Object Explorer, under **Databases**, right-click **AdventureWorks**, point to **Reports**, point to **Standard Reports**, and click **Backup and Restore Events**.
3. In the **Backup and Restore Events [AdventureWorks]** report, expand **Successful Backup Operations** and view the backup operations that have been performed for this database.
4. In the **Device Type** column, expand each of the **Disk (temporary)** entries to view details of the backup media set files.

Query Backup History Tables

1. In SQL Server Management Studio, open the **VerifyingBackups.sql** script file in the D:\Demofiles\Mod04 folder.
2. Select the code under the comment **View backup history**, and click **Execute**.
3. View the query results.

Verify Backup Media

1. Select the code under the comment **Use RESTORE HEADERONLY**, and click **Execute**.
2. View the query results, which show the backups in the **AW.bak** backup device.
3. Select the code under the comment **Use RESTORE FILELISTONLY**, and click **Execute**.
4. View the query results, which show the database files contained in the backups.
5. Select the code under the comment **Use RESTORE VERIFYONLY**, and click **Execute**.
6. View the message that is returned, which should indicate that the backup is valid.

Module Review and Takeaways

Best Practice

Plan your backup strategy carefully.

Plan the backup strategy in conjunction with the business needs.

Choose the appropriate database recovery model.

Plan your transaction log size based on the transaction log backup frequency.

Consider using differential backups to speed recovery.

Consider compressing backups to reduce storage requirements and backup time.

Review Question(s)

Question: What are the unique features of transaction log restores?

Answer: Point-in-time recovery and the ability to restore up to the point of failure if only data files are corrupt.

Question: When might a full database backup strategy be adequate?

Answer: If it is sufficient, in case of a disaster, to restore the database to the points of full database backup only.

Module 5

Restoring SQL Server 2014 Databases

Contents:

Lesson 2: Restoring Databases	2
Lesson 3: Advanced Restore Scenarios	5
Lesson 4: Point-in-Time Recovery	7
Module Review and Takeaways	9

Lesson 2

Restoring Databases

Contents:

Demonstration: Restoring Databases

3

Demonstration: Restoring Databases

Demonstration Steps

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Demofiles\Mod05 folder, run **Setup.cmd** as Administrator.
3. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
4. In Object Explorer, expand **Databases**, and note that the **AdventureWorks** database is in a **Recovery Pending** state.
5. Click **New Query** and execute the following Transact-SQL code to attempt to bring the database online.

```
ALTER DATABASE AdventureWorks SET ONLINE;
```

6. Note the error message that is displayed. The **AdventureWorks.mdf** data file has been lost, so the database cannot be brought online.
7. Delete the ALTER DATABASE statement, and replace it with the following code to perform a tail-log backup:

```
BACKUP LOG AdventureWorks TO DISK = 'D:\Demofiles\Mod05\AW-TailLog.bak'
WITH NO_TRUNCATE;
```

8. Click **Execute**, and view the resulting message to verify that the backup is successful.

Restore a Database

1. In Object Explorer, right-click the **AdventureWorks** database, point to **Tasks**, point to **Restore**, and click **Database**.
2. In the **Restore Database – AdventureWorks** dialog box, in the **Source** section, select **Device** and click the ellipses (...) button.
3. In the **Select backup devices** dialog box click **Add**, and then in the **Locate backup File – MIA-SQL** dialog box, select D:\Demofiles\Mod05\AW.bak and click **OK**.
4. In the **Select backup devices** dialog box, ensure that **D:\Demofiles\Mod05\AW.bak** is listed, and then click **Add**.
5. In the **Locate backup File – MIA-SQL** dialog box, select D:\Demofiles\Mod05\AW-TailLog.bak and click **OK**.
6. In the **Select backup devices** dialog box, ensure that both **D:\Demofiles\Mod05\AW.bak** and **D:\Demofiles\Mod05\AW-TailLog.bak** are listed and click **OK**.
7. Note that the backup media contains a full backup, a differential backup, and a transaction log backup (these are the planned backups in **AW.bak**); and a copy-only transaction log backup (which is the tail-log backup in **AW-TailLog.bak**). All of these are automatically selected in the **Restore** column.
8. On the **Options** page, ensure that the **Recovery state** is set to **RESTORE WITH RECOVERY**.
9. In the **Script** drop-down list, click **New Query Editor Window**. Then click **OK**.
10. When the database has been restored successfully, click **OK**.

11. View the Transact-SQL code that was used to restore the database, noting that the full backup, the differential backup, and the first transaction log backup were restored using the NORECOVERY option. The restore operation for the tail-log backup used the default RECOVERY option to recover the database.
12. In Object Explorer, verify that the **AdventureWorks** database is now recovered and ready to use.
13. Close SQL Server Management Studio without saving any files.

Lesson 3

Advanced Restore Scenarios

Contents:

Demonstration: Restoring an Encrypted Backup

6

Demonstration: Restoring an Encrypted Backup

Demonstration Steps

Restore an Encrypted Backup

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. If you did not complete the previous demonstration, in the D:\Demofiles\Mod05 folder, run **Setup.cmd** as Administrator.
3. Start SQL Server Management Studio and connect to the **MIA-SQL\SQL2** database engine using Windows authentication.
4. In Object Explorer, under the **MIA-SQL\SQL2** instance, expand **Databases** and view the existing databases on this instance.
5. Open the **Restore Encrypted Backup.sql** script file in the D:\Demofiles\Mod05 folder.
6. Select the code under the comment **Try to restore an encrypted backup** and click **Execute**. Note that this fails because the required certificate is not present.
7. Select the code under the comment **Create a database master key for master** and click **Execute**. This creates a database master key for the master database on **MIA-SQL\SQL2**.
8. Select the code under the comment **Import the backed up certificate** and click **Execute**. This creates a certificate from public and private key backups that were taken from the **MIA-SQL** instance.
9. Select the code under the comment **Restore the encrypted database** and click **Execute**. Note that this time the restore operation succeeds.
10. In Object Explorer, refresh the **Databases** folder and verify that the **AdventureWorksEncrypt** database has been restored.
11. Close SQL Server Management Studio.

Lesson 4

Point-in-Time Recovery

Contents:

Demonstration: Performing a Point-in-Time Recovery

8

Demonstration: Performing a Point-in-Time Recovery

Demonstration Steps

Perform a Point-In-Time Recovery

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
3. In SQL Server Management Studio, open the **Point-in-Time Restore.sql** script file in the D:\Demofiles\Mod05 folder.
4. Select and execute the code under the comment **Create a database and back it up**. This creates a database with a single table, and performs a full backup.
5. Select and execute the code under the comment **enter some data**. This inserts a record into the **Customers** table.
6. Select and execute the code under the comment **get the current time**. This displays the current date and time. Make a note of the current time.
7. Wait until a minute has passed, and then select and execute the code under the comment **get the current time** again to verify that it is now at least a minute since you noted the time.
8. Select and execute the code under the comment **enter some more data**. This inserts a second record into the **Customers** table.
9. Select and execute the code under the comment **backup the transaction log**. This performs a transaction log backup of the database.
10. Close the query window.
11. In Object Explorer, expand **Databases** and verify that **BackupDemo** is listed (if not, right-click the **Databases** folder and click **Refresh**). Then right-click the **BackupDemo** database, point to **Tasks**, point to **Restore**, and click **Database**.
12. In the **Restore Database – BackupDemo** dialog box, click **Timeline**.
13. In the **Backup Timeline: BackupDemo** dialog box, select **Specific date and time** and set the **Time** value to the time you noted earlier (after the first row of data was inserted). Then click **OK**.
14. In the **Restore Database – BackupDemo** dialog box, click **OK**. When notified that the database has been restored successfully, click **OK**.
15. In Object Explorer, expand the **BackupDemo** database and its **Tables** folder. Then right-click **dbo.Customers** and click **Select Top 1000 Rows**. When the results are displayed, verify that the database was restored to the point in time after the first row of data was inserted, but before the second row was inserted.
16. Close SQL Server Management Studio without saving any files.

Module Review and Takeaways

Best Practice

- Don't forget to back up the tail of the log before starting a restore sequence.
- If available, use differential restore to reduce the time taken by the restore process.
- Use file level restore to speed up restores when not all database files are corrupt.
- Perform regular database backups of **master**, **msdb** and **model** system databases.
- Create a disaster recovery plan for your SQL Server and test restoring databases regularly.

Review Question(s)

Question: What are the three phases of the restore process?

Answer: The three phases of a restore include data copy, redo and undo. The data copy phase involves copying all data, log and index pages from the backup media. The redo phase applies the transactions to the data copied from the backup to be rolled forward to the recovery point. The undo phase rolls back any uncommitted transactions and makes the database available to users. After the rollback phase, subsequent backups cannot be restored.

Module 6

Importing and Exporting Data

Contents:

Lesson 2: Importing and Exporting Data	2
Lesson 3: Copying or Moving a Database	6
Module Review and Takeaways	9
Lab Review Questions and Answers	10

Lesson 2

Importing and Exporting Data

Contents:

Demonstration: Using the Import and Export Wizard	3
Demonstration: Using the bcp Utility	3
Demonstration: Using the BULK INSERT Statement	4
Demonstration: Using the OPENROWSET Function	4

Demonstration: Using the Import and Export Wizard

Demonstration Steps

Use Import and Export Wizard to Export Data

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Demofiles\Mod06 folder, run **Setup.cmd** as Administrator.
3. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
4. In Object Explorer, expand **Databases**. Then right-click the **AdventureWorks** database, point to **Tasks**, and click **Export Data**.
5. On the **Welcome to SQL Server Import and Export Wizard** page, click **Next**.
6. On the **Choose a Data Source** page, in the **Data source** drop-down list, select **SQL Server Native Client 11.0**. Then ensure that the **MIA-SQL** server is selected, that **Use Windows Authentication** is selected, and that the **AdventureWorks** database is selected; and click **Next**.
7. On the **Choose a Destination** page, in the **Data source** drop-down list, select **Flat File Destination**. Then in the **File name** box type **D:\Demofiles\Mod06\Currency.csv**, clear the **Column names in the first data row** checkbox, and click **Next**.
8. On the **Specify Table Copy or Query** page, ensure that **Copy data from one or more tables or views** is selected, and click **Next**.
9. On the **Configure Flat File Destination** page, in the **Source table or view** list, select **[Sales].[Currency]**. Then ensure that the **Row delimiter** is **{CR}{LF}** and the **Column delimiter** is **Comma {,}**, and click **Next**.
10. On the **Save and Run Package** page, ensure that **Run immediately** is selected, and click **Next**.
11. On the **Complete the Wizard** page, click **Finish**. Then, when the execution is successful, click **Close**.
12. Start Excel and open the **Currency.csv** file in the D:\Demofiles\Mod06 folder and view the data that has been exported. Then close Excel without saving the file.

Demonstration: Using the bcp Utility

Demonstration Steps

Use bcp to Create a Format File

1. Ensure that you have completed the previous demonstration in this module.
2. Open a command prompt and type the following command to view the bcp syntax help:

```
bcp -?
```

3. In the command prompt window, enter the following command to create a format file:

```
bcp AdventureWorks.Sales.SalesTaxRate format nul -S MIA-SQL -T -c -t , -r \n -x -f D:\Demofiles\Mod06\TaxRateFmt.xml
```

4. Start Notepad and open **TaxRateFmt.xml** in the D:\Demofiles\Mod06 folder. Then view the XML format file and close notepad.

Use bcp to Export Data

1. In the command prompt window, enter the following command to export data from SQL Server.

```
bcp AdventureWorks.Sales.SalesTaxRate out D:\Demofiles\Mod06\SalesTaxRate.csv -S MIA-SQL -T -f D:\Demofiles\Mod06\TaxRateFmt.xml
```

2. Close the command prompt.
3. Start Excel and open the **SalesTaxRate.csv** file in the D:\Demofiles\Mod06 folder and view the data that has been exported. Then close Excel without saving the file.

Demonstration: Using the BULK INSERT Statement

Demonstration Steps

Use the BULK INSERT Statement to Import Data

1. Ensure that you have completed the previous demonstration in this module.
2. In SQL Server Management Studio, in Object Explorer, under **Databases**, expand **Finance**. Then expand the **Tables** folder, right-click **dbo.Currency**, and click **Select Top 1000 Rows**.
3. View the query results, and verify that the **dbo.Currency** table is currently empty.
4. Click **New Query**, and in the new query pane, enter the following Transact-SQL code:

```
BULK INSERT Finance.dbo.Currency
FROM 'D:\Demofiles\Mod06\Currency.csv'
WITH
(
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
);
```

5. Click **Execute** and note the number of rows affected.

Switch to the query pane that retrieves the top 1000 rows from the **dbo.Currency** table and click **Execute** to re-run the SELECT query. Note that the table is now populated with the same number of rows as you noted in the previous step.

Demonstration: Using the OPENROWSET Function

Demonstration Steps

Use the OPENROWSET Function to Import Data

1. Ensure that you have completed the previous demonstrations in this module.
2. In SQL Server Management Studio, in Object Explorer, right-click the **dbo.SalesTaxRate** table in the **Finance** database, and click **Select Top 1000 Rows**.
3. View the query results, and verify that the **dbo.SalesTaxRate** table is currently empty.
4. Click **New Query**, and in the new query pane, enter the following Transact-SQL code:

```
INSERT INTO Finance.dbo.SalestaxRate
SELECT * FROM OPENROWSET (BULK 'D:\Demofiles\Mod06\SalesTaxRate.csv',
    FORMATFILE = 'D:\Demofiles\Mod06\TaxRateFmt.xml') AS rows;
```

5. Click **Execute** and note the number of rows affected.

Switch to the query pane that retrieves the top 1000 rows from the **dbo.SalesTaxRate** table and click **Execute** to re-run the SELECT query. Note that the table is now populated with the same number of rows as you noted in the previous step.

Lesson 3

Copying or Moving a Database

Contents:

Demonstration: Using the Copy Database Wizard	7
Demonstration: Exporting and Importing a Data-tier Application	7

Demonstration: Using the Copy Database Wizard

Demonstration Steps

Use the Copy Database Wizard

1. Ensure that you have completed the previous demonstrations in this module.
2. In SQL Server Management Studio, in Object Explorer, in the **Connect** drop-down list click **Database Engine**. Then connect to the **MIA-SQL\SQL2** instance using Windows authentication.
3. In Object Explorer, under the **MIA-SQL\SQL2** instance, expand **Databases** and verify that the **AdventureWorks** database is not listed.
4. In Object Explorer, under the **MIA-SQL** instance, right-click the **AdventureWorks** database, point to **Tasks**, and click **Copy Database**.
5. On the **Welcome to the Copy Database Wizard** page, click **Next**.
6. On the **Select a Source Server** page, ensure that **MIA-SQL** is selected with the **Use Windows Authentication** option, and click **Next**.
7. On the **Select a Destination Server** page, change the **Destination server** to **MIA-SQL\SQL2** and select the **Use Windows Authentication** option. Then click **Next**.
8. On the **Select the Transfer Method** page, ensure that **Use the detach and attach method** is selected, and click **Next**.
9. On the **Select Databases** page, in the **Copy** column, ensure that the **AdventureWorks** database is selected. Then click **Next**.
10. On the **Configure Destination Database (1 of 1)** page, note the default settings for the database name and file locations. Then click **Next**.
11. On the **Select Server Objects** page, verify that **Logins** is listed in the **Selected related objects** list. Then click **Next**.
12. On the **Configure the Package** page, note the default package name and logging options. Then click **Next**.
13. On the **Scheduling the Package** page, ensure that **Run immediately** is selected, and click **Next**.
14. On the **Completing the Wizard** page, click **Finish**.
15. Wait for the operation to complete. Then click **Close**.
16. In Object Explorer, under the **MIA-SQL\SQL2** instance, right-click **Databases** and click **Refresh**. Then verify that the **AdventureWorks** database has been copied to this instance.

Demonstration: Exporting and Importing a Data-tier Application

Demonstration Steps

Export a Data-Tier Application

1. Ensure that you have completed the previous demonstrations in this module.
2. In SQL Server Manager, in Object Explorer, under the **MIA-SQL** instance, right-click the **Finance** database, point to **Tasks**, and click **Export Data-tier Application**. (Be careful to click **Export Data-tier Application**, and not **Extract Data-tier Application**.)
3. On the **Introduction** page, click **Next**.

4. On the **Export Settings** page, ensure that **Save to local disk** is selected and enter the path **D:\Demofiles\Mod06\Finance.bacpac**. Then click the **Advanced** tab and verify that all tables are selected, and click **Next**.
5. On the **Summary** page, click **Finish**.
6. Wait for the export operation to complete, and then click **Close**.

Import a Data-Tier Application

1. In SQL Server Management Studio, in Object Explorer, under the MIA-**SQL\SQL2** instance, right-click the **Databases** folder and click **Import Data-tier Application**.
2. On the **Introduction** page, click **Next**.
3. On the **Import Settings** page, ensure that **Import from local disk** is selected and enter the path **D:\Demofiles\Mod06\Finance.bacpac**. Then **Next**.
4. On the **Database Settings** page, review the default settings for the database name and file paths, and then click **Next**.
5. On the **Summary** page, click **Finish**.
6. Wait for the import operation to complete, and then click **Close**.
7. On Object Explorer, under the **MIA-SQL\SQL2** instance, if necessary, refresh the **Databases** folder and verify that the **Finance** database has been imported.

Module Review and Takeaways

Best Practice

Choose the right tool for bulk-imports.

Use SSIS for complex transformations.

Use bcp or BULK INSERT for fast imports and exports.

Use OPENROWSET when data needs to be filtered before it gets inserted.

Try to achieve minimal logging to speed up data import.

Review Question(s)

Question: What other factors might you need to consider when importing or exporting data?

Answer: There are many considerations, including the impact on user and application workloads, and security.

Lab Review Questions and Answers

Lab: Importing and Exporting Data

Question and Answers

Lab Review

Question: Why was it not necessary to disable constraints when importing the currency rates?

Answer: The BULK INSERT statement does not check constraints by default.

Question: If the **dbo.JobCandidate** table has included a column for a resume in Microsoft Word document format, which tool or command could you use to import the document into a column in a table?

Answer: The BULK provider in the OPENROWSET command with the SINGLE_BLOB option.

Module 7

Monitoring SQL Server 2014

Contents:

Lesson 1: Introduction to Monitoring SQL Server	2
Lesson 2: Dynamic Management Views and Functions	5
Lesson 3: Performance Monitor	8
Module Review and Takeaways	10
Lab Review Questions and Answers	11

Lesson 1

Introduction to Monitoring SQL Server

Contents:

Demonstration: Using Activity Monitor

3

Demonstration: Using Activity Monitor

Demonstration Steps

View Server Activity in Activity Monitor

1. Ensure that the 20462C-MIA-DC, and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Demofiles\Mod07 folder, run Setup.cmd as Administrator.
3. Start SQL Server management Studio and connect to the **MIA-SQL** database engine instance using Windows authentication.
4. In Object Explorer, right-click the **MIA-SQL** SQL Server instance and click **Activity Monitor**.
5. In Activity Monitor, view the charts in the **Overview** section, which show background activity in the SQL Server instance.
6. Expand the **Processes** section and view the processes currently running in the SQL Server instance.
7. Click the filter icon for the **Application** column header, and filter the data to show only processes for the **Microsoft SQL Server Management Studio** application (you may need to widen the columns to read the headers).
8. Remove the filter to show all applications.
9. Expand the **Resource Waits** section and view the statistics for processes waiting on resources.
10. Expand the **Data File I/O** section and view the details of the database file I/O activity (you may need to wait for a few seconds while the data is collected and displayed).
11. Expand the **Recent Expensive Queries** section and view the list of queries that have consumed query processing resources.

Troubleshoot a Blocked Process

1. With Activity Monitor still open in SQL Server Management Studio, in the D:\Demofiles\Mod07 folder, run **ActivityWorkload.cmd**.
2. In SQL Server Management Studio, in Object Explorer, expand **Databases**, expand **AdventureWorks**, and expand **Tables**. Then right-click **Production.Product** and click **Select Top 1000 Rows**.
3. In status bar at the bottom of the query pane, note that the query continues executing. Another process is preventing it from completing.
4. In the Activity Monitor pane, in the **Processes** section, filter the **Task State** column to show processes that are in a **SUSPENDED** state.
5. In the **Blocked By** column for the suspended process, note the ID of the process that is blocking this one.
6. Remove the filter on the **Task State** column to show all processes, and find the blocking process with the ID you identified in the previous step.
7. Note the value in the **Head Blocker** column for the blocking process. A value of **1** indicates that this process is the first one in a chain that is blocking others.
8. Right-click the blocking process and click **Details**. This displays the Transact-SQL code that is causing the block—in this case, a transaction that has been started but not committed or rolled back.
9. Click **Kill Process**, and when prompted to confirm, click **Yes**. After a few seconds, the Processes list should update to show no blocked processes.

10. Close the Activity Monitor pane, and verify that the query to retrieve the top 1000 rows from **Production.Products** has now completed successfully.
11. Close the command prompt window, but keep SQL Server Management Studio open for the next demonstration.

Lesson 2

Dynamic Management Views and Functions

Contents:

Demonstration: Querying Dynamic Management Views

6

Demonstration: Querying Dynamic Management Views

Demonstration Steps

View SQL Server Service Configuration Settings

1. If you did not complete the previous demonstration in this module, start the 20462C-MIA-DC, and 20462C-MIA-SQL virtual machines, log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**, and in the D:\Demofiles\Mod07 folder, run **Setup.cmd** as Administrator.
2. If SQL Server Management Studio is not already open, start it and connect to the **MIA-SQL** database engine instance using Windows authentication.
3. In SQL Server Management Studio, open the **DMVs.sql** script file in the D:\Demofiles\Mod07 folder.
4. Highlight the Transact-SQL statement under the comment **View service information**, and then click **Execute**.
5. Review the results, noting the values in the **startup_type** column for the SQL Server and SQL Server Agent services.
6. Highlight the Transact-SQL statement under the comment **View registry information**, and then click **Execute**.
7. Review the results. Note the value **Start** in the **value_name** column for the MSSQLSERVER and SQLSERVERAGENT registry keys and the corresponding values in the **value_data** column. These values are the equivalent registry value for the **startup_type** column values returned by the **sys.dm_server_services** dynamic management view.

View Storage Volume Statistics

1. Select the Transact-SQL code under the comment **View volume stats**, noting that it retrieves data from the **sys.sysdatabases** and **sys.master_files** system tables as well as the **sys.dm_os_volume_stats** dynamic management function.
2. Click **Execute** and review the query results, which show the files for all databases in the instance, together with details about the disk volume on which they are hosted.

View Query Statistics

1. Highlight the Transact-SQL statement under the comment **Empty the cache** and then click **Execute**.
2. Highlight the Transact-SQL statement under the comment **get query stats**, noting that this code uses the **sys.dm_exec_query_stats** DMV and the **sys.dm_exec_sql_text** DMF to return details about Transact-SQL queries that have been executed.
3. Click **Execute** and review the results, which show some background system queries, noting the various columns that are returned.
4. Highlight the Transact-SQL statement under the comment **Execute a query**, and then click **Execute**.
5. Re-highlight the Transact-SQL statement under the comment **get query stats**, and then click **Execute** to get the query stats again.
6. In the results, find the row with a **SQLText** column that contains the Transact-SQL statement that you executed and note the statistics returned for the query.
7. Re-highlight the Transact-SQL statement under the comment **Execute a query**, and then click **Execute** to run the query again.
8. Re-highlight the Transact-SQL statement under the comment **get query stats**, and then click **Execute** to get the query stats again.

9. Review the results for the query, noting that the query you executed now has an **execution_count** value of 2.
10. Close SQL Server Management Studio without saving any files.

Lesson 3

Performance Monitor

Contents:

Demonstration: Using Performance Monitor

9

Demonstration: Using Performance Monitor

Demonstration Steps

View Performance Counters

1. If you did not complete the previous demonstration in this module, start the 20462C-MIA-DC, and 20462C-MIA-SQL virtual machines, log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**, and in the D:\Demofiles\Mod07 folder, run **Setup.cmd** as Administrator.
2. Right-click the Start button and click **Computer Management**.
3. In Computer Management, expand **Performance**, expand **Monitoring Tools**, and click **Performance Monitor**.
4. In the toolbar, click the **Add** button (a green +).
5. In the list of objects, expand the **Processor** object, and select only the **% Processor Time** counter. Then in the **Instances of selected object** list, ensure that **_Total** is selected and click **Add**.
6. In the list of objects, expand the **Memory** object and select the **Page Faults/sec** counter. Then click **Add**.
7. In the list of objects, expand the **SQLServer:Locks** object, click the **Average Wait Time (ms)** counter, and then hold the **Ctrl** key and click the **Lock Requests/sec** and **Lock Waits/sec** counters. Then in the **Instances of selected object** list, ensure that **_Total** is selected and click **Add**.
8. In the list of objects, expand the **SQLServer:Plan Cache** object, and select the **Cache Hit Ratio** counter. Then in the **Instances of selected object** list, ensure that **_Total** is selected and click **Add**.
9. In the list of objects, expand the **SQLServer:Transactions** object, and select the **Transactions** counter. Then click **Add**.
10. In the **Add Counters** dialog box, click **OK**. Then observe the counters as they are displayed in Performance Monitor.
11. On the toolbar, click **Freeze Display** and note that the chart is paused. Then click **Unfreeze Display** to resume the chart.
12. On the toolbar, in the **Change Graph Type** list, select **Histogram bar** and view the resulting chart. Then in the **Change Graph Type** list, select **Report** and view the text-based report.
13. On the toolbar, in the **Change Graph Type** list, select **Line** to return to the original line chart view.
14. Click any of the counters in the list below the chart and on the toolbar click **Highlight** so that the selected counter is highlighted in the chart. Press the up and down arrow keys on the keyboard to change the selected counter.
15. In the D:\Demofiles\Mod07 folder, run **PerformanceWorkload1.cmd** and **PerformanceWorkload2.cmd** to generate some activity in the database.
16. In Performance Monitor, observe the effect on the counters as the workloads run.
17. Close both command prompt windows, and observe the effect that ending the workloads has on the Performance Monitor counters.
18. Close Computer Management.

Module Review and Takeaways

Best Practice

When monitoring SQL Server, consider the following best practices:

Identify the system resources that your database workload uses, and determine the key performance metrics that indicate how your database and server are performing.

Record baseline measurements for typical and peak workloads so that you have a basis for comparison when troubleshooting performance problems later.

Identify the DMVs and DMFs that return appropriate performance information for your workloads, and create reusable scripts that you can use to quickly check system performance.

Monitor the overall system periodically and compare the results with the baseline. This can help you detect trends that will eventually result in resource over-utilization before application performance is affected.

Review Question(s)

Question: How are dynamic management views and functions different from system tables?

Answer: Dynamic management views and functions that are virtual objects, providing access to state information. The data they provide is accessed dynamically and is not persisted between server restarts.

Lab Review Questions and Answers

Lab: Monitoring SQL Server 2014

Question and Answers

Lab Review

Question: Based on the results of your monitoring, what aspect of the database solution is most significantly affected by the changes to the workload?

Answer: The number of lock requests, wait time, and I/O statistics should be relatively unchanged between the baseline and the revised workload. The biggest change is the increased requirement for memory, shown by an increase in the number of page faults, a higher average database cache memory figure, and a reduced free memory figure.

The main cause for the increased memory requirements is a query that declares a table variable and inserts details of every order into it.

Module 8

Tracing SQL Server Activity

Contents:

Lesson 1: Tracing SQL Server Workload Activity	2
Lesson 2: Using Traces	5
Module Review and Takeaways	9
Lab Review Questions and Answers	10

Lesson 1

Tracing SQL Server Workload Activity

Contents:

Demonstration: Using SQL Server Profiler	3
Demonstration: Using SQL Trace	4

Demonstration: Using SQL Server Profiler

Demonstration Steps

Use SQL Server Profiler to Create a Trace

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Demofiles\Mod08 folder, run **Setup.cmd** as Administrator.
3. Start SQL Server Management Studio, and connect to the **MIA-SQL** database engine instance using Windows authentication.
4. On the **Tools** menu, click **SQL Server Profiler**.
5. When SQL Server Profile starts, connect to the **MIA-SQL** database engine instance using Windows authentication.
6. In the **Trace Properties** dialog box, on the **General** tab, set the following properties:
 - **Trace name:** Demo Trace
 - **Use the template:** TSQL
 - **Save to file:** D:\Demofiles\Mod08\Demo Trace.trc
7. In the **Trace Properties** dialog box, on the **Events Selection** tab, note the events and columns that were automatically selected from the **TSQL** template.
8. Select **Show all events**, and under **TSQL**, select **SQL:StmntCompleted**. Then clear **Show all events** so that only the selected events, including the one you just selected are shown.
9. Select **Show all columns** and select the **Duration** column for the **SQL:StmntCompleted** event.
10. Click the column header for the **Database Name** column, and in the Edit Filter dialog box, expand **Like**, enter **AdventureWorks**, and click **OK**. Then clear **Show all columns** so that only the selected columns are shown.

Run a Trace and View the Results

1. In the **Trace Properties** dialog box, click **Run**.
2. Observe the trace as it shows some background activity.
3. Switch back to SQL Server Management Studio, open the **Query.sql** script file in the D:\Demofiles\Mod08 folder, and click **Execute**. This script runs a query in the **AdventureWorks** database twenty times.
4. While the query is executing, switch back to SQL Server Profiler and observe the activity.
5. When the query has finished, in SQL Server Profiler, on the **File** menu, click **Stop Trace**.
6. In the trace, select any of the **SQL:StmntCompleted** events and note that the Transact-SQL code is shown in the bottom pane.
7. Keep SQL Server Profiler and SQL Server Management Studio open for the next demonstration.

Demonstration: Using SQL Trace

Demonstration Steps

Export a Trace Definition

1. Ensure that you have completed the previous demonstration in this module.
2. In SQL Server Profiler, with the Demo Trace still open, on the **File** menu, point to **Export**, point to **Script Trace Definition**, and click **For SQL Server 2005 - 2014**.
3. Save the exported trace script as **DemoTrace.sql** in the D:\Demofiles\Mod08 folder, and click **OK** when notified that the script has been saved.
4. Keep SQL Server Profiler open for the next demonstration.

Configure and Run a Trace

1. In SQL Server Management Studio, open the **DemoTrace.sql** script file in the D:\Demofiles\Mod08 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:\Demofiles\Mod08\SQLTraceDemo**.
3. Click **Execute** to start the trace, and note the **TraceID** value that is returned.
4. Switch back to the **Query.sql** tab and click **Execute** to run the workload query.
5. When the query has finished, open **StopTrace.sql** in the D:\Demofiles\Mod08 folder.
6. In the **StopTrace.sql** script, under the comment **Stop the trace**, if necessary, modify the DECLARE statement to specify the **TraceID** value for the trace you started previously.
7. Select the code under the comment **Stop the trace** and click **Execute**. Setting the trace status to **0** stops the trace, and setting it to **2** closes the file and deletes the trace definition on the server.
8. Select the code under the comment **View the trace** and click **Execute**. Then review the traced events.
9. Close the **StopTrace.sql** script and **DemoTrace.sql** tabs without saving any changes so that only the **Query.sql** tab remains open.
10. Keep SQL Server Management Studio open for the next demonstration.

Lesson 2

Using Traces

Contents:

Resources	6
Demonstration: Using the Database Engine Tuning Advisor	6
Demonstration: Correlating a Trace with Performance Data	7
Demonstration: Troubleshooting Deadlocks	8

Resources

Replaying Traces



Additional Reading: For more information about the configuration files, their contents, and their locations, go to *Configure Distributed Replay* at [http://technet.microsoft.com/library/ff878359\(v=sql.120\).aspx](http://technet.microsoft.com/library/ff878359(v=sql.120).aspx).

Demonstration: Using the Database Engine Tuning Advisor

Demonstration Steps

Configure a Tuning Session

1. Ensure that you have completed the previous demonstration in this module.
2. In SQL Server Profiler, on the **Tools** menu, click **Database Engine Tuning Advisor**.
3. When the Database Engine Tuning Advisor starts, connect to the **MIA-SQL** database engine instance using Windows authentication.
4. In the Database Engine Tuning Advisor, In the **Session name** box, type **Tuning Demo**.
5. Under **Workload**, ensure that **File** is selected, and browse to the **D:\Demofiles\Mod08\Demo Trace.trc** file (which is where you saved the trace from SQL Server Profiler in the previous demonstration).
6. In the **Database for workload analysis** drop-down list, select **AdventureWorks**.
7. In the **Select databases and tables to tune** list, select **AdventureWorks** and note that 71 of 71 tables are selected. Then in the drop down list of tables, select only the following tables:
 - o Product
 - o ProductCategory
 - o ProductSubcategory
 - o SalesOrderDetail
 - o SalesOrderHeader
8. On the **Tuning Options** tab, review the default options for recommendations. Then click **Advanced Options**, select **Generate online recommendations where possible**, and click **OK**.

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 **Reports** tab, view the tuning summary and in the **Select report** list, select **Statement detail report**.
4. View the report and compare the **Current Statement Cost** value to the **Recommended Statement Cost** value (cost is an internal value that the SQL Server query processor uses to quantify the work required to process a query).
5. On the **Actions** menu, click **Save Recommendations**, save the recommendations script as **DTA Recommendations.sql** in the D:\Demofiles\Mod08 folder, and click **OK** when notified that the file was saved.

6. Close the Database Engine Tuning Advisor.

Validate Recommendations

1. In SQL Server Management Studio, highlight the SELECT statement in the **Query.sql** script, taking care not to highlight the GO 20 statement that follows it.
2. 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.
3. 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.
4. In SQL Server Management Studio, open the **DTA Recommendations.sql** script you saved from the Database Engine Tuning Advisor in the D:\Demofiles\Mod08 folder. Then click execute to implement the recommended indexes.
5. Switch back to the **Query.sql** tab, and highlight the SELECT statement, taking care once again not to highlight the GO 20 statement that follows it.
6. On the **Query** menu, click **Display Estimated Execution Plan**.
7. 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 that is displayed in a tooltip.

Demonstration: Correlating a Trace with Performance Data

Demonstration Steps

Correlate a Trace with Performance Data

1. Ensure that you have completed the previous demonstration in this module.
2. In the D:\Demofiles\Mod08 folder, double-click **AWCounters.blg** to open the log file in Performance Monitor.
3. View the line chart, noting the times noted along the bottom axis. Then close Performance Monitor.
4. In SQL Server Profiler, open the **AWTrace.trc** file in the D:\Demofiles\Mod08 folder and view the traced events. Noting that the event times in the **StartTime** column match those in the Performance Monitor log.
5. In SQL Server Profiler, on the **File** menu, click **Import Performance Data**. Then open the **AWCounters.blg** log file in the D:\Demofiles\Mod08 folder.
6. In the **Performance Counters Limit Dialog** dialog box, select **\\MIA-SQL** (which selects all of the counters in the log file) and click **OK**.
7. Click the line chart at approximately the **3:15:35 PM** marker. Note that the event in the trace that occurred at that time is selected, and the Transact-SQL statement that was executed is shown in the bottom pane.
8. Keep SQL Server Management Studio open for the next demonstration.

Demonstration: Troubleshooting Deadlocks

Demonstration Steps

Capture a Trace Based on the TSQL_Locks Template

1. Ensure that you have completed the previous demonstrations in this module.
2. In SQL Server Profiler, 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, in the **Trace name** box, type **Locks**. Then in the **Use the template** drop-down list, select **TSQL_Locks**.
4. On the **Events Selection** tab, view the events that are selected in this template. Then click the column header for the **Database Name** column, and in the Edit Filter dialog box, expand **Like**, enter **AdventureWorks**, and click **OK**.
5. Click **Run** to start the trace.
6. While the trace is running, in the D:\Demofiles\Mod08 folder, run **Deadlock.cmd**. This will open two command prompt windows.
7. When both command prompt windows close, in SQL Server profiler, on the **File** menu, click **Stop Trace**.

View a Deadlock Graph

1. In SQL Server Profiler, in the **Locks** trace, find the **Deadlock graph** event and select it.
2. In the bottom pane, view the deadlock graph; which shows that a deadlock occurred and one process was selected as the victim.
3. On the **File** menu, point to **Export**, point to **Extract SQL Server Events**, and click **Extract Deadlock Events**. Then save the deadlock events as **Deadlocks** in the D:\Demofiles\Mod08 folder.
4. Close SQL Server Profiler, and in SQL Server Management Studio, open the **Deadlocks_1.xdl** file in the D:\Demofiles\Mod08 folder. Note that you can view deadlock graph files in SQL Server Management Studio.
5. Hold the mouse pointer over each of the process circles to see the statements that they were executing as a tooltip. Note that the deadlock occurred because one process used a transaction to update records in the **Production.Product** table and then the **Sales.SpecialOffer** table, while the other process tried to update the same records in the opposite order.
6. Close SQL Server Management Studio without saving any files.

Module Review and Takeaways

Best Practice

When tracing activity in SQL Server, consider the following best practices:

Use SQL Server Profiler to perform short traces for debugging and other purposes.

Use SQL Trace for large and long-running traces.

Use SQL Server Profiler to define traces and script them for SQL Trace.

Import trace data into a database table for advanced analysis.

Use Database Engine Tuning Advisor to analyze the database based on the overall workload you want to optimize, rather than focusing on individual queries.

Review Question(s)

Question: In what situations would you use SQL Trace rather than SQL Server Profiler?

Answer: Where you need to minimize the impact on the system being traced.

Question: How would you test a workload after configuration changes?

Answer: Use the replay functionality of SQL Server Profiler or Distributed Replay.

Lab Review Questions and Answers

Lab: Tracing SQL Server Workload Activity

Question and Answers

Lab Review

Question: How do you think the Database Engine Tuning Advisor recommendations you implemented will affect overall performance of the workload?

Answer: The Database Engine Tuning Advisor recommendations you implemented for the workload should have reduced the cost of multiple queries. Although each query cost is reduced by a small amount, the cumulative effect could be significant.

Question: The workload you traced was defined to reflect common reporting functionality and includes only SELECT queries. Alongside this workload, the **InternetSales** database must process INSERT and UPDATE operations submitted by the e-commerce site. How will the recommendations you implemented affect these workloads?

Answer: While indexes generally improve SELECT operations, depending on the specific statements they may degrade the performance of INSERT and UPDATE statements and lead to fragmentation in frequently modified tables. This underlines the importance of using a representative workload for all of the activity you want to optimize when generating recommendations.

Module 9

Managing SQL Server Security

Contents:

Lesson 1: Introduction to SQL Server Security	2
Lesson 2: Managing Server-Level Security	4
Lesson 3: Managing Database-Level Principals	7
Lesson 4: Managing Database Permissions	11
Module Review and Takeaways	13
Lab Review Questions and Answers	14

Lesson 1

Introduction to SQL Server Security

Contents:

Resources

3

Resources

Security Concepts



Best Practice: When planning a security solution, consider the following best practices:

- Provide each principal with only the permissions they actually need.
- Use securable inheritance to minimize the number of implicit permissions that must be set in order to enable the required level of access.
- Use principal containers such as groups to create a layer of abstraction between principals and permissions to access securables. Then use membership of these groups to control access to resources via the permissions you have defined. Changes in personnel should not require changes to permissions.

Lesson 2

Managing Server-Level Security

Contents:

Demonstration: Managing Server-Level Security

5

Demonstration: Managing Server-Level Security

Demonstration Steps

Set the Authentication Mode

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Demofiles\Mod09 folder, run **Setup.cmd** as Administrator.
3. Start SQL Server Management Studio, and connect to the **MIA-SQL** database engine using Windows authentication.
4. In Object Explorer, right-click the **MIA-SQL** instance and click **Properties**.
5. In the **Server Properties – MIA-SQL** dialog box, on the **Security** page, verify that **SQL Server and Windows Authentication mode** is selected. Then click **Cancel**.

Create Logins

1. In Object Explorer, expand **Security**, and expand **Logins** to view the logins that are currently defined on this server instance.
2. Right-click **Logins** and click **New Login**. Then in the **Login – New** dialog box, next to the **Login name** box, click **Search**.
3. In the **Select User or Group** dialog box, click **Object Types**. Then in the **Object Types** dialog box, select only **Users** and **Groups** and click **OK**.
4. In the **Select User or Group** dialog box, click **Locations**. Then in the **Locations** dialog box, expand **Entire Directory**, select **adventureworks.msft** and click **OK**.
5. In the **Select User, Service Account, or Group** dialog box, click **Advanced**. Then click **Find Now**. This produces a list of all users and groups in the Active Directory domain.
6. In the list of domain objects, select **HumanResources_Users** (this is a domain local group that contains multiple global groups, each of which in turn contains users), then click **OK**.
7. In the **Select User, Service Account, or Group** dialog box, ensure that **HumanResources_Users** is listed, and click **OK**.
8. In the **Login – New** dialog box, in the **Default database** drop-down list, select **AdventureWorks**. Then click **OK** and verify that the **ADVENTUREWORKS\HumanResources_Users** login is added to the **Logins** folder in Object Explorer.
9. Right-click **Logins** and click **New Login**. Then in the **Login – New** dialog box, enter the name **Payroll_Application** and select **SQL Server authentication**.
10. Enter and confirm the password **Pa\$\$w0rd**, and then clear the **Enforce password expiration** check box (which automatically clears the **User must change password at next login** check box).
11. In the **Default database** drop-down list, select **AdventureWorks**. Then click **OK** and verify that the **Payroll_Application** login is added to the **Logins** folder in Object Explorer.
12. Open the **CreateLogins.sql** script file in the D:\DemoFiles\Mod09 folder and review the code it contains, which creates a Windows login for the **ADVENTUREWORKS\AnthonyFrizzell** user and the **ADVENTUREWORKS\Database_Managers** local group, and a SQL Server login named **Web_Application**.
13. Click **Execute**. Then, when the script has completed successfully, refresh the **Logins** folder in Object Explorer and verify that the logins have been created.

Manage Server-Level Roles

1. In Object Explorer, expand **Server Roles** and view the server roles that are defined on this instance.
2. Right-click the **serveradmin** fixed server-level role and click **Properties**. Then in the **Server Role Properties – serveradmin** dialog box, click **Add**.
3. In the **Select Server Login or Role** dialog box, click **Browse**, and in the **Browse for Objects** dialog box, select **[ADVENTUREWORKS\Database_Managers]** and click **OK**. Then in the **Select Server Login or Role** dialog box, click **OK**.
4. In the **Server Role Properties – serveradmin** dialog box, ensure that **[ADVENTUREWORKS\Database_Managers]** is listed and click **OK**.
5. Open the **ServerRoles.sql** script file in the D:\DemoFiles\Mod09 folder and review the code it contains, which creates a user-defined server role named **AW_securitymanager** and adds the **ADVENTUREWORKS\AnthonyFrizzell** user to the new role.
6. Click **Execute**. Then, when the script has completed successfully, refresh the **Server Roles** folder in Object Explorer and verify that the role has been created.
7. Right-click the **AW_securitymanager** role and click **Properties**, and verify that **ADVENTUREWORKS\AnthonyFrizzell** is listed as a member. Then click **Cancel**.

Manage Server-Level Permissions

1. Open the **ServerPermissions.sql** script file in the D:\DemoFiles\Mod09 folder and review the code it contains, which grants ALTER ANY LOGIN permission to the **AW_securitymanager** server role.
2. Click **Execute**. Then, when the script has completed successfully, in Object Explorer, right-click the **AW_securitymanager** role and click **Properties**.
3. In the **Server Role Properties - AW_securitymanager** dialog box, on the **General** tab, view the selected securables. Then click **Cancel**.
4. In the **Logins** folder, right-click the **ADVENTUREWORKS\AnthonyFrizzell** login and click **Properties**.
5. In the **Login Properties - ADVENTUREWORKS\AnthonyFrizzell** dialog box, click the **Securables** tab. Then above the **Securables** list, click **Search**, select **All objects of the types** and click **OK**, and select **Logins** and click **OK**.
6. In the **Login Properties - ADVENTUREWORKS\AnthonyFrizzell** dialog box, on the **Securables** tab, in the list of logins, select **Payroll_Application**.
7. In the **Permissions for Payroll_Application** list, on the **Explicit** tab, note that no explicit permissions on this login have been granted to **ADVENTUREWORKS\AnthonyFrizzell**. Then click the **Effective** tab and note that the ALTER permission has been inherited through membership of the **AW_securitymanager** role.
8. In the **Login Properties - ADVENTUREWORKS\AnthonyFrizzell** dialog box, click **Cancel**.
9. Leave SQL Server Management Studio open for the next demonstration.

Lesson 3

Managing Database-Level Principals

Contents:

Demonstration: Managing Database Users and Roles	8
Demonstration: Using an Application Role	8
Demonstration: Using a Contained Database	9

Demonstration: Managing Database Users and Roles

Demonstration Steps

Create Database Users

1. Ensure that you have completed the previous demonstration in this module.
2. In SQL Server Management Studio, in Object Explorer, expand **Databases**, expand the **AdventureWorks** database, and expand its **Security** folder. Then expand the **Users** folder and view the users currently defined in the database.
3. Right-click **Users** and click **New User**. Then, in the **Database user – New** dialog box, enter the user name **Web_Application**, the login name **Web_Application**, and the default schema **Sales**; and click **OK**.
4. Open the **CreateUsers.sql** script file in the D:\DemoFiles\Mod09 folder and review the code it contains, which creates users for the **Payroll_Application**, **ADVENTUREWORKS\HumanResources_Users**, and **ADVENTUREWORKS\AnthonyFrizzell** logins.
5. Click **Execute**. Then, when the script has completed successfully, refresh the **Users** folder in Object Explorer and verify that the users have been created.

Manage Database-Level Roles

1. In Object Explorer, expand the **Roles** folder, and then expand the **Database Roles** folder and view the database roles in the database.
2. Right-click the **db_datareader** role and click **Properties**. This is a fixed database-level role.
3. In the **Database Role Properties – db_datareader** dialog box, click **Add**. In the **Select Database User or Role** dialog box, enter **AnthonyFrizzell** and click **OK**. Then verify that **AnthonyFrizzell** is listed and click **OK**.
4. Right-click **Database Roles** and click **New Database Role**.
5. Enter the role name **hr_reader**, and click **Add**. In the **Select Database User or Role** dialog box, enter **HumanResources_Users**; **Payroll_Application** and click **OK**. Then verify that **HumanResources_Users** and **Payroll_Application** are listed and click **OK**.
6. Open the **DatabaseRoles.sql** script file in the D:\DemoFiles\Mod09 folder and review the code it contains, which creates roles name **hr_writer** and **web_customer**, and adds the **HumanResources_Users** to the **hr_writer** role and **Web_Application** to the **web_customer** role.
7. Click **Execute**. Then, when the script has completed successfully, refresh the **Database Roles** folder in Object Explorer and verify that the roles have been created.
8. Keep SQL Server Management Studio open for the next demonstration.

Demonstration: Using an Application Role

Demonstration Steps

Create an Application Role

1. Ensure that you have completed the previous demonstration in this module.
2. In SQL Server Management Studio, under the **Roles** folder for the **AdventureWorks** database, right-click **Application Roles** and click **New Application Role**.
3. In the **Application Role – New** dialog box, enter the role name **pay_admin**, enter the default schema **HumanResources**, enter and confirm the password **Pa\$\$w0rd**, and click **OK**.

Use an Application Role

1. Open the **ApplicationRole.sql** script file in the D:\DemoFiles\Mod09 folder. The code in this file displays the identity of the current user and login before, during, and after the activation of the **pay_admin** application role.
2. Right-click anywhere in the script window, point to **Connection**, and click **Change Connection**. Then connect to the **MIA-SQL** database engine using SQL Server authentication as **Payroll_Application** with the password **Pa\$\$w0rd**.
3. Click **Execute** and view the results. Note that the **System Identity** does not change (which may be important for auditing reasons), but that the **DB Identity** switched to **pay_admin** while the application role was active.
4. Close the **ApplicationRole.sql** query pane, but keep SQL Server Management Studio open for the next demonstration.

Demonstration: Using a Contained Database

Demonstration Steps

Create a Contained Database

1. Ensure that you have completed the previous demonstrations in this module.
2. In SQL Server Management Studio, open the **ContainedDatabase.sql** script file in the D:\Demofiles\Mod09 folder.
3. Select and execute the code under the comment **Enable contained databases**. This code configures the server options to enable contained databases.
4. Select and execute the code under the comment **Create a contained database**. This code creates a database named **ContainedDB** with a CONTAINMENT setting of PARTIAL.
5. In Object Explorer, under **MIA-SQL**, refresh the **Databases** folder and verify that **ContainedDB** is listed.
6. Right-click **ContainedDB** and click **Properties**. Then, in the **Database Properties – ContainedDB** dialog box, on the **Options** tab, note that the **Containment type** is set to **Partial**, and click **Cancel**.

Create Contained Users

1. In the query window, select and execute the code under the comment **Create contained users**. This code creates two users in the **ContainedDB** database: A SQL Server user with a password, and a Windows user.
2. In Object Explorer, expand the **ContainedDB** database, expand **Security**, and expand **Users**. Note that the two contained users you created are listed.
3. In Object Explorer, under the server-level **Security** folder, refresh the **Logins** folder. Note that there are no logins for the users you created in the contained database.
4. Right-click anywhere in the query window, point to **Connection**, and click **Change Connection**.
5. In the **Connect to Database Engine** dialog box, ensure **MIA-SQL** is selected, in the **Authentication** drop-down list, select **SQL Server Authentication**, enter the login **SalesApp** and the password **Pa\$\$w0rd**, and then click **Options**.
6. On the **Connection Properties** tab, in the **Connect to database** box, ensure **<default>** is selected and click **Connect**. An error occurs because there is no login named **SalesAppUser**. In the **Connect to Database Engine** window, click **OK**.
7. In the **Connect to database** box, type **ContainedDB**. Then click **Connect**. This connection succeeds because the user is defined in the database.

8. Close the **ContainedDatabase.sql** query window, but keep SQL Server Management Studio open for the next demonstration.

Lesson 4

Managing Database Permissions

Contents:

Demonstration: Managing Permissions

11

Demonstration: Managing Permissions

Demonstration Steps

Set Permissions

1. Ensure that you have completed the previous demonstrations in this module.
2. In SQL Server Management Studio, open the **DatabasePermissions.sql** script file in the D:\Demofiles\Mod09 folder.
3. Select the code under the comment **Grant schema permissions** and click **Execute**. This code grants SELECT permission on the **HumanResources** schema to the **hr_reader** database role, and INSERT, UPDATE, and EXECUTE permission on the **HumanResources** schema to the **hr_writer** database role.
4. Select the code under the comment **Grant individual object permissions** and click **Execute**. This code grants EXECUTE permission on the **dbo.uspGetEmployeeManagers** stored procedure to the **hr_reader** database role; INSERT permission on the **Sales.SalesOrderHeader** and **Sales.SalesOrderDetail** tables to the **web_customer** database role; and SELECT permission on the **Production.vProductAndDescription** view to the **web_customer** database role.
5. Select the code under the comment **Override inherited permissions** and click **Execute**. This code grants INSERT and UPDATE permission on the **Sales** schema procedure to the **AnthonyFrizzell** user, grants UPDATE permission on the **HumanResources.EmployeePayHistory** table to the **[Payroll_Application]** user; grants UPDATE permission on the **SalariedFlag** column in the **HumanResources.Employee** table to the **[Payroll_Application]** user; and denies SELECT on the **HumanResources.EmployeePayHistory** table to the **AnthonyFrizzell** user.

View Effective Permissions

1. In Object Explorer, under the **AdventureWorks** database, expand **Tables**, right-click **HumanResources.Employee** and click **Properties**.
2. In the **Table Properties – Employee** dialog box, on the **Permissions** tab, note that the **[Payroll_Application]** user has been explicitly granted **Update** permission.
3. With the **Payroll_Application** user selected, view the permissions in the **Effective** tab, and note that this user has SELECT permission on the table, and UPDATE permission on the **SalariedFlag** column. The SELECT permission has been implicitly granted through membership of the **hr_reader** database role, which has inherited SELECT permission from permissions on the parent schema. The UPDATE permission was granted explicitly.
4. In the **Table Properties – Employee** dialog box, click **Cancel**. Then close SQL Server Management Studio.

Module Review and Takeaways

Best Practice

When implementing security in SQL Server, consider the following best practices:

Minimize the number of SQL Server logins.

Use Windows group logins to simplify ongoing management where possible.

Disable logins rather than dropping them if there is any chance that they will be needed again.

Ensure that expiry dates are applied to logins that are created for temporary purposes.

Use fixed server-level roles to delegate server-level management responsibility, and only create user-defined server-level roles if your specific administrative delegation solution requires them.

Disable the **guest** user in user databases unless you specifically require guest access.

Aim to grant the minimum number of explicit permissions possible to meet the security requirements, and use membership of roles and inheritance to ensure the correct effective permissions.

Ensure every user has only the permission they actually require.

Review Question(s)

Question: Your organization needs to track data access by individual Windows users. Does this mean you cannot base logins on Windows groups?

Answer: No.

Lab Review Questions and Answers

Lab: Managing SQL Server Security

Question and Answers

Lab Review

Question: Compare your solution to the scripts provided in the D:\Labfiles\Lab09\Solution folder. What did you do differently?

Answer: There are multiple ways to meet the requirements, so there may be many differences.

Question: What sort of login would be required for a user in a Windows domain that is not trusted by the domain in which SQL Server is installed?

Answer: A SQL Server login.

Module 10

Auditing Data Access and Encrypting Data

Contents:

Lesson 1: Auditing Data Access in SQL Server	2
Lesson 2: Implementing SQL Server Audit	4
Lesson 3: Encrypting Databases	7
Module Review and Takeaways	9

Lesson 1

Auditing Data Access in SQL Server

Contents:

Demonstration: Using DML Triggers for Auditing

3

Demonstration: Using DML Triggers for Auditing

Demonstration Steps

Create a DML Trigger for Auditing

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Demofiles\Mod10 folder, run **Setup.cmd** as Administrator.
3. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine instance using Windows authentication.
4. Open the **DML Trigger.sql** script file in the D:\Demofiles\Mod10 folder.
5. Select the code under the comment **Create a log table**, and click **Execute**. This creates a table named **AuditRateChange** in the **HumanResources** schema of the **AdventureWorks** database.
6. Select the code under the comment **Create a trigger**, and click **Execute**. This creates a trigger on the **EmployeePayHistory** table that fires on updates. When the Rate column is updated, a row is inserted into the **AuditRateChange** table.
7. Select the code under the comment **Update a rate**, and click **Execute**. This updates a rate in the **EmployeePayHistory** table.
8. Select the code under the comment **View the audit log**, and click **Execute**. This retrieves the logged details from the **AuditRateChange** table.
9. Keep SQL Server Management Studio open for the next demonstration.

Lesson 2


Implementing SQL Server Audit

Contents:

Resources	5
Demonstration: Using SQL Server Audit	5

Resources

Reading Audited Events

 **Additional Reading:** For a full explanation of the structure, see *sys.fn_get_audit_file* (Transact-SQL) in SQL Server Books Online.

Demonstration: Using SQL Server Audit

Demonstration Steps

Create an Audit

1. If you did not complete the previous demonstration, start the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines, log onto 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**, and run **Setup.cmd** in the D:\Demofiles\Mod10 folder as Administrator. Then start SQL Server Management Studio, and connect to **MIA-SQL** using Windows authentication.
2. In SQL Server Management Studio, open the **Audit.sql** script file in the D:\Demofiles\Mod10 folder.
3. Select the code under the comment **Create an audit**, and click **Execute**. This creates an audit that logs events to files in D:\Demofiles\Mod10\Audits.
4. In Object Explorer, expand **Security**, and expand **Audits** (if **Audits** is not expandable, refresh it and try again).
5. Double-click the **AW_Audit** audit you created and view its properties. Then click **Cancel**.

Create a Server Audit Specification

1. In SQL Server Management Studio, in the **Audit.sql** script, select the code under the comment **Create a server audit specification** and click **Execute**. This creates an audit specification for the **AW_Audit** audit that logs failed and successful login attempts.
2. In Object Explorer, refresh the **Server Audit Specifications** folder and expand it. Then double-click **AW_ServerAuditSpec**, view its properties, and click **Cancel**.

Create a Database Audit Specification

1. In SQL Server Management Studio, in the **Audit.sql** script, select the code under the comment **Create a database audit specification** and click **Execute**. This creates an audit specification for the **AW_Audit** audit that logs specific actions by individual principals on the **HumanResources** schema in the **AdventureWorks** database.
2. In Object Explorer, expand **Databases**, expand **AdventureWorks**, expand **Security**, and expand **Database Audit Specifications** (if **Database Audit Specifications** is not expandable, refresh it and try again).
3. Double-click **AW_DatabaseAuditSpec** and view its properties. Then click **Cancel**.

View Audited Events

1. Open a command prompt and enter the following command to run sqlcmd as **ADVENTUREWORKS\ChadCorbitt**. This user is a member of the **ADVENTUREWORKS\Personnel** global group, which in turn is a member of the **ADVENTUREWORKS\HumanResources_Users** domain local group.

```
runas /user:adventureworks\chadcorbitt /noprompt sqlcmd
```

2. When prompted for a password, enter **Pa\$\$w0rd**.

3. In the SQLCMD window, enter the following commands to query the **HumanResources.Employee** table:

```
SELECT LoginID, JobTitle FROM HumanResources.Employee;  
GO
```

4. Close the SQLCMD and command prompt windows.
5. In the D:\Demofiles\Mod10\Audits folder, verify that an audit file has been created.
6. In SQL Server Management Studio, in the **Audit.sql** script, select the code under the comment **View audited events** and click **Execute**. This queries the files in the audit folder and displays the audited events (to simplify this demonstration, events logged for the Student user and the service account for SQL Server have been excluded).
7. Note that all events are logged with the server principal name **ADVENTUREWORKS\ChadCorbitt** despite the fact that this user accesses SQL Server through membership of a Windows group and does not have an individual login.
8. Keep SQL Server Management Studio open for the next demonstration.

Lesson 3

Encrypting Databases

Contents:

Demonstration: Implementing Transparent Data Encryption	8
---	---

Demonstration: Implementing Transparent Data Encryption

Demonstration Steps

Create a Database Master Key

1. If you did not complete the previous demonstration, start the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines, log onto 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**, and run **Setup.cmd** in the D:\Demofiles\Mod10 folder as Administrator. Then start SQL Server Management Studio, and connect to **MIA-SQL** using Windows authentication.
2. In SQL Server Management Studio, open the **TDE.sql** script file in the D:\Demofiles\Mod10 folder.
3. Select the code under the comment **Create DMK** and click **Execute**. This creates a database master key in the **master** database.

Create a Server Certificate

1. In the TDE.sql script, select the code under the comment **Create server certificate** and click **Execute**. This creates a certificate and then backs up the certificate and its private key.

Create a Database Encryption Key

1. In the TDE.sql script, select the code under the comment **Create DEK** and click **Execute**. This creates a database encryption key in the **ConfidentialDB** database.

Enable Database Encryption

1. In the TDE.sql script, select the code under the comment **Enable encryption** and click **Execute**. This enables encryption for the **ConfidentialDB** database, and retrieves database encryption status from the **sys.databases** table in the **master** database.
2. Review the query results, and verify that the **is_encrypted** value for **ConfidentialDB** is **1**.
3. Close SQL Server Management Studio without saving any files.

Module Review and Takeaways

Best Practice

When planning to implement auditing, consider the following best practices:

Choose the option to shut down SQL Server on audit failure. There is usually no point in setting up auditing, and then having situations where events can occur but are not audited. This is particularly important in high-security environments.

Make sure that file audits are placed on drives with large amounts of free disk space and ensure that the available disk space is monitored on a regular basis.

Best Practice

When planning to implement database encryption, consider the following best practices:

Use a complex password to protect the database master key for the **master** database.

Ensure you back up certificates and private keys used to implement TDE, and store the backup files in a secure location.

If you need to implement data encryption on multiple servers in a large organization, consider using an EKM solution to manage encryption keys.

Review Question(s)

Question: What are the three targets for SQL Server audits?

Answer: Files, the Windows application log, and the Windows security log.

Question: You may wish to audit actions by a DBA. How would you know if the DBA stopped the audit while performing covert actions?

Answer: Changes to the audit status are logged.

Module 11

Performing Ongoing Database Maintenance

Contents:

Lesson 1: Ensuring Database Integrity	2
Lesson 2: Maintaining Indexes	4
Lesson 3: Automating Routine Database Maintenance	6
Module Review and Takeaways	9
Lab Review Questions and Answers	10

Lesson 1

Ensuring Database Integrity

Contents:

Demonstration: Using DBCC CHECKDB

3

Demonstration: Using DBCC CHECKDB

Demonstration Steps

Use the DBCC CHECKDB Command

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Demofiles\Mod11 folder, run **Setup.cmd** as Administrator.
3. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine instance using Windows authentication.
4. Open the **DBCCCHECKDB.sql** script file in the D:\Demofiles\Mod11 folder.
5. Select the code under the comment **Run DBCC CHECKDB with default options** and click **Execute**. This checks the integrity of the **AdventureWorks** database and displays detailed informational messages.
6. Select the code under the comment **Run DBCC CHECKDB without informational messages** and click **Execute**. This checks the integrity of the **AdventureWorks** database and only displays messages if errors are found.
7. Select the code under the comment **Run DBCC CHECKDB against CorruptDB** and click **Execute**. This checks the integrity of the **CorruptDB** database and identifies some consistency errors in the **dbo.Orders** table in this database. The last line of output tells you the minimum repair level required.
8. Select the code under the comment **Try to access the Orders table** and click **Execute**. This attempts to query the **dbo.Orders** table in **CorruptDB**, and returns an error because of a logical consistency issue.
9. Select the code under the comment **Access a specific order** and click **Execute**. This succeeds, indicating that only some data pages are affected by the consistency issue.
10. Select the code under the comment **Repair the database** and click **Execute**. Note that this technique is used only as a last resort when no valid backup is available. No guarantee on logical consistency in the database (such as foreign key constraints) is provided.
11. Select the code under the comment **Access the Orders table** and click **Execute**. This succeeds, indicating that the consistency issue has been repaired.
12. Select the code under the comment **Check the internal database structure** and click **Execute**. No error message are displayed, indicating that the database structure is now consistent.
13. Select the code under the comment **Check data loss** and click **Execute**. Note that a number of order details records have no matching order records. The foreign-key constraint between these tables originally enforced a relationship, but some data has been lost.

Lesson 2

Maintaining Indexes

Contents:

Demonstration: Maintaining Indexes

5

Demonstration: Maintaining Indexes

Demonstration Steps

Maintain Indexes

1. If you did not complete the previous demonstration in this module, ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**. Then, in the D:\Demofiles\Mod11 folder, run **Setup.cmd** as Administrator.
2. If SQL Server Management Studio is not already open, start it and connect to the **MIA-SQL** database engine instance using Windows authentication.
3. Open the **MaintainingIndexes.sql** script file in the D:\Demofiles\Mod11 folder.
4. Select the code under the comment **Create a table with a primary key** and click **Execute**. This creates a table with a primary key, which by default creates a clustered index on the primary key field.
5. Select the code under the comment **Insert some data into the table** and click **Execute**. This inserts 10,000 rows into the table.
6. Select the code under the comment **Check fragmentation** and click **Execute**. In the results, note the **avg_fragmentation_in_percent** and **avg_page_space_used_in_percent** values for each index level.
7. Select the code under the comment **Modify the data in the table** and click **Execute**. This updates the table.
8. Select the code under the comment **Re-check fragmentation** and click **Execute**. In the results, note that the **avg_fragmentation_in_percent** and **avg_page_space_used_in_percent** values for each index level have changed as the data pages have become fragmented.
9. Select the code under the comment **Rebuild the table and its indexes** and click **Execute**. This rebuilds the indexes on the table.
10. Select the code under the comment **Check fragmentation again** and click **Execute**. In the results, note that the **avg_fragmentation_in_percent** and **avg_page_space_used_in_percent** values for each index level indicate less fragmentation.

Lesson 3

Automating Routine Database Maintenance

Contents:

Demonstration: Creating a Maintenance Plan

7

Demonstration: Creating a Maintenance Plan

Demonstration Steps

Create a Maintenance Plan

1. If you did not complete the previous demonstration in this module, ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**. Then, in the D:\Demofiles\Mod11 folder, run **Setup.cmd** as Administrator.
2. If SQL Server Management Studio is not already open, start it and connect to the **MIA-SQL** database engine instance using Windows authentication.
3. In Object Explorer, under **MIA-SQL**, expand **Management**, right-click **Maintenance Plans**, and click **Maintenance Plan Wizard**.
4. In the Maintenance Plan Wizard window, click **Next**.
5. In the **Select Plan Properties** window, in the **Name** textbox type **Daily Maintenance**. Note the available scheduling options and click **Change**.
6. In the **New Job Schedule** window, in the **Name** textbox type "Daily". In the **Occurs** drop down list, click **Daily**. In the **Occurs once at** textbox, change the time to 3:00 AM, and click **OK**.
7. In the **Select Plan Properties** window, click **Next**. Then in the **Select Maintenance Tasks** page, select the following tasks and click **Next**.
 - Check Database Integrity
 - Reorganize Index
 - Update Statistics
 - Back up Database (Full)
8. On the **Select Maintenance Task Order** page, click **Next**.
9. On the define **Database Check Integrity Task** page, select the **AdventureWorks** database and click **OK**. Then click **Next**.
10. On the **Define Reorganize Index Task** page, select the **AdventureWorks** database and click **OK**, ensure that **Tables and Views** is selected, and click **Next**.
11. On the **Define Update Statistics Task** page, select the **AdventureWorks** database and click **OK**, ensure that **Tables and Views** is selected, and click **Next**.
12. On the **Define Backup database (Full) Task** page, select the **AdventureWorks** database and click **OK**. Then on the **Destination** tab, ensure that **Create a backup file for every database** is selected, change the **Folder** value to D:\Demofiles\Mod11\Backups\ and click **Next**.
13. On the Select Report Options page, ensure that **Write a report to a text file** is selected, change the **Folder location** to D:\Demofiles\Mod11\ and click **Next**.
14. On the **Complete the Wizard** page, click **Finish**. Then when the operation has completed, click **Close**.
15. In Object Explorer, under **Maintenance Plans**, right-click **Daily Maintenance** and click **Execute**.
16. Wait a minute or so until the maintenance plan succeeds, and in the **Execute Maintenance Plan** dialog box, click **Close**. Then right-click **Daily Maintenance** and click **View History**.
17. In the **Log File Viewer - MIA-SQL** dialog box, expand the **Date** value for the **Daily Maintenance** plan to see the individual tasks.

18. Keep clicking **Refresh** and expanding the tasks until four tasks have been completed. Then click **Close**.
19. In the D:\Demofiles\Mod11 folder, view the **Daily Maintenance_Subplan_1_XXXXX.txt** file that has been created.
20. In the **Backups** folder, verify that a backup of the **AdventureWorks** database has been created.

Module Review and Takeaways

Best Practice

When planning ongoing database maintenance, consider the following best practices.

Run DBCC CHECKDB regularly.

Synchronize DBCC CHECKDB with your backup strategy.

If corruption occurs, consider restoring the database from a backup, and only repair the database as a last resort.

Defragment your indexes when necessary.

Update statistics on a schedule if you don't want it to occur during normal operations.

Use maintenance plans to implement regular tasks.

Review Question(s)

Question: What option should you consider using when running DBCC CHECKDB against large production databases?

Answer: Use the PHYSICAL_ONLY option for regular executions, and perform full checks less frequently.

Lab Review Questions and Answers

Lab: Performing Ongoing Database Maintenance

Question and Answers

Lab Review

Question: After discovering that the **InternetSales** database contained corrupt pages, what would have been a preferable solution to repairing the database with potential data loss?

Answer: Restore the database from the most recent backup, and if possible use a tail-log backup to restore the database to the point of failure.

Question: If you need to execute a maintenance plan with timing that cannot be accommodated by a single schedule, what can you do?

Answer: Add another schedule.

Module 12

Automating SQL Server 2014 Management

Contents:

Lesson 2: Implementing SQL Server Agent Jobs	2
Lesson 3: Managing SQL Server Agent Jobs	5
Lesson 4: Managing Job Step Security Contexts	8
Lesson 5: Managing Jobs on Multiple Servers	10
Module Review and Takeaways	13
Lab Review Questions and Answers	14

Lesson 2

Implementing SQL Server Agent Jobs

Contents:

Demonstration: Creating Jobs

3

Demonstration: Creating Jobs

Demonstration Steps

Create a Job

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Demofiles\Mod12 folder, run **Setup.cmd** as Administrator.
3. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine instance using Windows authentication.
4. In Object Explorer, expand **SQL Server Agent** and **Jobs** to view any existing jobs. Then right-click **Jobs** and click **New Job**.
5. In the **New Job** dialog box, on the **General** page, in the **Name** box, type **Check AdventureWorks DB**.
6. In the **New Job** dialog box, on the **Steps** page, click **New**.
7. In the **New Job Step** dialog box, on the **General** page, in the **Step name** box, type **Make Folder**. Then ensure that **Operating system (CmdExec)** is selected in the **Type** drop-down list and in the **Command** area, type the following command, which calls a batch file to create an empty folder named **AdventureWorks** in the D:\Demofiles\Mod12 folder.

```
D:\Demofiles\Mod12\MakeDir.cmd
```

8. In the **New Job Step** dialog box, click **OK**.
9. In the **New Job** dialog box, on the **Steps** page, click **New**.
10. In the **New Job Step** dialog box, on the **General** page, in the **Step name** box, type **Get DB Info**. Then ensure that **Transact-SQL script (T-SQL)** is selected in the **Type** drop-down list and in the **Command** area, type the following command.

```
EXEC sp_helpdb AdventureWorks;
```

11. In the **New Job Step** dialog box, on the **Advanced** page, in the **Output file** box, type **D:\Demofiles\Mod12\AdventureWorks\DB_Info.txt**. Then click **OK**.
12. In the **New Job** dialog box, on the **Steps** page, click **New**.
13. In the **New Job Step** dialog box, on the **General** page, in the **Step name** box, type **Check DB**. Then ensure that **Transact-SQL script (T-SQL)** is selected in the **Type** drop-down list and in the **Command** area, type the following command.

```
DBCC CHECK ('AdventureWorks');
```

14. In the **New Job Step** dialog box, on the **Advanced** page, in the **Output file** box, type **D:\Demofiles\Mod12\AdventureWorks\CheckDB.txt**. Then click **OK**.
15. In the **New Job** dialog box, on the **Steps** page, verify that the **Start step** is set to **1:Make Folder** and note the **On Success** and **On Failure** actions for the steps in the job.
16. In the **New Job** dialog box, on the **Schedules** page, click **New**.
17. In the **New Job Schedule** dialog box, in the **Name** box, type **Weekly Jobs**; in the **Frequency** area, ensure that only **Sunday** is selected; and in the **Daily frequency** area, ensure that the option to occur once at 12:00 AM is selected. Then click **OK**.

18. In the **New Job** dialog box, click **OK**. Then verify that the job appears in the **Jobs** folder in Object Explorer.

Script a Task to a Job

1. In Object Explorer, expand **Databases**. Then right-click **AdventureWorks**, point to **Tasks**, and click **Back Up**.
2. In the **Back Up Database - AdventureWorks** dialog box, select the existing backup destination and click **Remove**. Then click **Add** and in the **Select Backup Destination** dialog box, in the **File name** box, type **D:\Demofiles\Mod12\Backups\AdventureWorks.bak** and click **OK**.
3. In the **Back Up Database - AdventureWorks** dialog box, in the **Script** drop-down list, select **Script Action to Job**.
4. In the **New Job** dialog box, on the **General** page, note the default name for the job (**Back Up Database - AdventureWorks**). Then on the **Steps** page, note that the job includes one Transact-SQL step named **1**.
5. In the **New Job** dialog box, on the **Schedules** page, click **Pick**. Then in the **Pick Schedule for Job - Back Up Database - AdventureWorks** dialog box, select the **Weekly Jobs** schedule you created previously and click **OK**.
6. In the **New Job** dialog box, click **OK**. Then, in the **Back Up Database - AdventureWorks** dialog box, click **Cancel**.
7. Verify that the job appears in the **Jobs** folder in Object Explorer.

Generate Scripts for Existing Jobs

1. In Object Explorer, right-click the **Check AdventureWorks DB** job, point to **Script Job as**, point to **CREATE To**, and click **New Query Editor Window**. This generates the Transact-SQL code necessary to create the job.
2. In Object Explorer, right-click the **Back Up Database - AdventureWorks** job, point to **Script Job as**, point to **CREATE To**, and click **Clipboard**. Then place the insertion point at the end of the Transact-SQL code in the query editor window and on the **Edit** menu, click **Paste**.
3. Save the Transact-SQL script as **Create Jobs.sql** in the D:\Demofiles\Mod12 folder. Using this technique to generate scripts to create jobs is a common way to ensure that jobs can be recreated if they are accidentally deleted or required on a different server.
4. Keep SQL Server Management Studio open for the next demonstration

Lesson 3

Managing SQL Server Agent Jobs

Contents:

Resources	6
Demonstration: Viewing Job History	6

Resources

Querying SQL Server Agent-related System Tables and Views



Additional Reading: For more information about the system tables which store SQL Server Agent data, see *SQL Server Agent Tables (Transact-SQL)* in SQL Server Books Online.

Demonstration: Viewing Job History

Demonstration Steps

Run Jobs

1. Ensure that you have completed the previous demonstration in this module.
2. In SQL Server Management Studio, in Object Explorer, right-click the **Back Up Database - AdventureWorks** job and click **Start Job at Step**. Then, when the job has completed successfully, click **Close**.
3. In Object Explorer, right-click the **Check AdventureWorks DB** job and click **Start Job at Step**. Then select step 1 and click **Start**. Note that the job fails, and click **Close**.

Troubleshoot a Failed Job

1. In Object Explorer, right-click the **Back Up Database - AdventureWorks** job and click **View History**.
2. In the **Log File Viewer - MIA-SQL** dialog box, expand the date for the most recent instance of the job, and note that all steps succeeded. Then click **Close**.
3. In Object Explorer, right-click the **Check AdventureWorks DB** job and click **View History**.
4. In the **Log File Viewer - MIA-SQL** dialog box, expand the date for the most recent instance of the job, and note that the third step failed.
5. Select the step that failed, and in the pane at the bottom of the dialog box, view the message that was returned. Then click **Close**.
6. In Object Explorer, double-click the **Check AdventureWorks DB** job. Then in the **Job Properties - Check AdventureWorks DB** dialog box, on the **Steps** page, select step 3 (**Check DB**) and click **Edit**.
7. In the **Job Step Properties - Check DB** dialog box, modify the command as follows and click **OK**.

```
DBCC CHECKDB ('AdventureWorks');
```

8. In the **Job Properties - Check AdventureWorks DB** dialog box, click **OK**.
9. In Object Explorer, right-click the **Check AdventureWorks DB** job and click **Start Job at Step**. Then select step 1 and click **Start**.
10. In Object Explorer, double-click **Job Activity Monitor** and note the **Status** of the **Check AdventureWorks DB** job.
11. Click **Refresh** until the **Status** changes to **Idle**, and verify that the **Last Run Outcome** for the job is **Succeeded**. Then click **Close** to close the **Job Activity Monitor**.
12. In the **Start Jobs - MIA-SQL** dialog box (which may be behind SQL Server Management Studio), verify that the job completed with a status of **Success**, and click **Close**.
13. In the D:\Demofiles\Mod12 folder, view the text files generated by the **Check AdventureWorks DB** job in the **AdventureWorks** folder, and verify that a backup file was created in the **Backups** folder by the **Back Up Database - AdventureWorks** job.

14. Keep SQL Server Management Studio open for the next demonstration.

Lesson 4

Managing Job Step Security Contexts

Contents:

Demonstration: Configuring Security Context

9

Demonstration: Configuring Security Context

Demonstration Steps

Create a Credential

1. Ensure that you have completed the previous demonstration in this module.
2. In SQL Server Management Studio, in Object Explorer, under **MIA-SQL**, expand **Security**. Then right-click **Credentials** and click **New Credential**.
3. In the **New Credential** dialog box, enter the following details and click **OK**.
 - **Credential name:** FileAgent
 - **Identity:** MIA-SQL\FileAgent
 - **Password:** Pa\$\$w0rd
 - **Confirm password:** Pa\$\$w0rd
4. **MIA-SQL\FileAgent** is a local Windows user on the MIA-SQL server with minimal privileges.

Create a Proxy Account

1. In Object Explorer, under **SQL Server Agent**, expand **Proxies**.
2. Right-click **Operating System (CmdExec)** and click **New Proxy**.
3. In the **New Proxy Account** dialog box, in the **Proxy name** box type **FileAgentProxy**, and in the **Credential name** box type **FileAgent**. Then ensure that only the **Operating system (CmdExec)** subsystem is selected and click **OK**.

Run a Job Step as a Proxy Account

1. In Object Explorer, under **SQL Server Agent**, in the **Jobs** folder, double-click the **Check AdventureWorks DB** job.
2. In the **Job Properties - Check AdventureWorks DB** dialog box, on the **Steps** page, click step 1 (**Make Folder**) and click **Edit**.
3. In the **Job Step Properties - Make Folder** dialog box, in the **Run as** drop-down list, select **FileAgentProxy**. Then click **OK**.
4. In the **Job Properties - Check AdventureWorks DB** dialog box, click **OK**.
5. Right-click the **Check AdventureWorks DB** job, and click **Start Job at Step**. Then in the **Start Job on MIA-SQL** dialog box, ensure step 1 is selected and click **Start**.
6. When the job has completed successfully, click **Close**.
7. In the D:\Demofiles\Mod12 folder, right-click the **AdventureWorks** folder that was created by your job, and click **Properties**.
8. In the **AdventureWorks properties** dialog box, on the **Security** tab, click **Advanced**.
9. Note that the owner of the folder is **MIA-SQL\FileAgent**. This is the account that was used to create the folder. Then click **Cancel** to close the **Advanced Security Settings for AdventureWorks** dialog box, and click **Cancel** again to close the **AdventureWorks Properties** dialog box.
10. Keep SQL Server Management Studio open for the next demonstration

Lesson 5

Managing Jobs on Multiple Servers

Contents:

Demonstration: Configuring Multi-Server Jobs	11
--	----

Demonstration: Configuring Multi-Server Jobs

Demonstration Steps

Create Master and Target Servers

1. Ensure that you have completed the previous demonstrations in this module.
2. In SQL Server Management Studio, in Object Explorer, under **MIA-SQL**, right-click **SQL Server Agent**, point to **Multi Server Administration**, and then click **Make this a Master**.
3. In the Master Server Wizard – MIA-SQL window, on the **Welcome to the Master Server Wizard** page, click **Next**.
4. On the **Master Server Operator** page, in the **E-mail address** box, type **student@adventureworks.com**, and then click **Next**.
5. On the **Target Servers** page, click **Add Connection**.
6. In the **Connect to Server** dialog box, in the **Server name** box, type **MIA-SQL\SQL2**, and then click **Connect**.
7. On the **Target Servers** page, click **Next**.
8. In the Checking Server Compatibility window, click **Close**.
9. On the **Master Server Login Credentials** page, click **Next**.
10. On the **Complete the Wizard** page, click **Finish**. Then when configuration is complete. Click **Close**.

Create a Job for Target Servers

1. In Object Explorer, expand **SQL Server Agent (MSX)**, and then expand **Jobs**.
2. Right-click **Jobs** and then click **New Job**.
3. In the **New Job** window, in the **Name** box, type **Backup master database**.
4. In the **New Job** window, on the **Steps** page, click **New**.
5. In the **Step name** box, type **Backup master** and ensure that the **Transact-SQL (T-SQL)** type is selected. Then, in the Command box, type the following Transact-SQL, then click **Parse**.

```
BACKUP DATABASE master
TO DISK = 'master.bak';
```

Because no folder path is specified, the command will store the back up in the default backup folder for the SQL Server instance.

6. In the **New Job Step** window, click **OK**.
7. In the **New Job** window, on the **Targets** page, select **Target multiple servers**. Then select **MIA-SQL\SQL2**, and then click **OK**.

Run a Job on a Target Server

1. In Object Explorer, right-click the **Backup master database**, and click **Start Job at Step**. Then, in the **Start Jobs – MIA-SQL** dialog box, click **Close**.
2. In Object Explorer, in the **Connect** drop-down list, click **Database Engine**. Then connect to the **MIA-SQL\SQL2** instance using Windows authentication.
3. In Object Explorer, under **MIA-SQL\SQL2**, expand **SQL Server Agent (TSX: MIA-SQL)** and expand **Jobs**.
4. Right-click the **Backup master database** job on MIA-SQL\SQL2 and click **View History**.

5. Review the job history to verify that it has been executed successfully, and then click **Close**.
6. Close SQL Server Management Studio without saving any files.
7. In File Explorer, browse to the **C:\Program Files\Microsoft SQL Server\MSSQL12.SQL2\MSSQL\Backup** folder, (clicking **Continue** if prompted) and verify it contains a **master.bak** file created in the last few minutes.
8. Close File Explorer.

Module Review and Takeaways

Best Practice

When planning SQL Server Agent jobs, consider the following best practices.

Use SQL Agent jobs to schedule routine tasks.

Create custom categories to group your jobs.

Script your jobs for remote deployment, or in case you need to recreate them.

Use job history to review job and job step outcomes.

Use Job Activity Monitor to real-time monitor jobs.

Apply the “principle of least privilege” when configuring job step execution identities to avoid creating highly-privileged user accounts.

Review Question(s)

Question: What functions do you currently perform manually that could be placed in a job?

Answer: Answers will vary, but they may include backups, data transfers, pre-calculating data for reports, archiving backup files, and so on.

Lab Review Questions and Answers

Lab: Automating SQL Server Management

Question and Answers

Lab Review

Question: Assuming you have administrative control over the local Windows user for the proxy account in this scenario, what considerations would apply to its properties?

Answer: The key consideration is the password expiry setting for the Windows account. If the password expires and must be reset, the credential in SQL Server will need to be altered. A common approach is to configure the Windows account with the **Password does not expire** and **User cannot change password** options.

Module 13

Monitoring SQL Server 2014 with Notifications and Alerts

Contents:

Lesson 1: Monitoring SQL Server Errors	2
Lesson 2: Configuring Database Mail	4
Lesson 3: Configuring Operators, Notifications, and Alerts	7
Module Review and Takeaways	10
Lab Review Questions and Answers	11

Lesson 1

Monitoring SQL Server Errors

Contents:

Demonstration: Viewing the SQL Server Error Log

3

Demonstration: Viewing the SQL Server Error Log

Demonstration Steps

View the SQL Server Error Log

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are running, and log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Demofiles\Mod13 folder, run **Setup.cmd** as Administrator.
3. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine instance using Windows authentication.
4. In Object Explorer, under the **MIA-SQL** instance, expand **Management** and expand **SQL Server Logs**. Then right-click **Current** and click **View SQL Server Log**.
5. Maximize the **Log File Viewer - MIA-SQL** window and view the log entries. Note that when you select a log entry, its details are shown in the bottom pane.
6. In the **Select logs** pane, expand **SQL Server Agent** and select **Current**. Then scroll the main log entries pane to the right until you can see the **Log Type** column and scroll down to find an entry with the log type **SQL Server Agent**.
7. When you have finished viewing the log entries, click **Close**.
8. Minimize SQL Server Management Studio and view the contents of the C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\Log folder. If you are prompted to change your permissions to get access to the folder, click **Continue**. Note that the current SQL Server log is stored here in the file named ERRORLOG, and the current SQL Server Agent log is stored as SQLAGENT.1. The remaining log files contain log entries for other SQL Server components and services.

Cycle the Log File

1. In SQL Server Management Studio, click New Query.
2. In the query window, enter the following Transact-SQL code:

```
EXEC sys.sp_cycle_errorlog;
```

3. Click **Execute**.
4. In Object Explorer, right-click the **Current** SQL Server log and click **View SQL Server Log**.
5. Note that the log has been reinitialized, and then click **Close**.

Lesson 2

Configuring Database Mail

Contents:

Demonstration: Configuring Database Mail

5

Demonstration: Configuring Database Mail

Demonstration Steps

Create a Database Mail Profile

1. If you did not complete the previous demonstration in the module, start the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines, log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**, and in the D:\Demofiles\Mod13 folder, run **Setup.cmd** as Administrator.
2. If SQL Server Management Studio is not already open, start it and connect to the **MIA-SQL** database engine instance using Windows authentication.
3. In Object Explorer, under the **MIA-SQL** instance, under **Management**, right-click **Database Mail**, and click **Configure Database Mail**.
4. In the **Welcome to Database Mail Configuration Wizard** page, click **Next**.
5. In the **Select Configuration Task** page, select the option to set up Database Mail and click **Next**.
6. In the **New Profile** page, in the **Profile name** textbox type **SQL Server Agent Profile**, and click **Add**. Then, in the **Add Account to profile 'SQL Server Agent Profile'** dialog box, click **New Account**.
7. In the **New Database Mail Account** dialog box, enter the following details and click **OK**:
 - **Account name**: AdventureWorks Administrator
 - **E-mail address**: administrator@adventureworks.msft
 - **Display name**: Administrator (AdventureWorks)
 - **Reply e-mail**: administrator@adventureworks.msft.
 - **Server name**: mia-sql.adventureworks.msft
8. In the **New Profile** page, click **Next**.
9. In the **Manage Profile Security** page, select **Public** for the **SQL Server Agent Profile** profile, and set its **Default Profile** setting to **Yes**. Then click **Next**.
10. In the **Configure System Parameters** page, click **Next**. Then, in the **Complete the Wizard** page, click **Finish** and when configuration is complete, click **Close**.

Send a Test E-Mail

1. In Object Explorer, right-click **Database Mail** and click **Sent Test E-Mail**.
2. In the **Send Test E-Mail from MIA-SQL** dialog box, ensure that the **SQL Server Agent Profile** database mail profile is selected, and in the **To** textbox, enter **student@adventureworks.msft**. Then click **Send Test Email**.
3. View the contents of the C:\inetpub\mailroot\Drop folder, and verify that an email message has been created here.
4. Double-click the message to view it in Outlook. When you have read the message, close it and minimize the **Drop** folder window.
5. In the **Database Mail Test E-Mail** dialog box (which may be behind SQL Server Management Studio), click **OK**.

Query Database Mail System Tables

1. In SQL Server Management Studio, click **New Query**.
2. Enter the following Transact-SQL code and click **Execute**.

```
SELECT * FROM msdb.dbo.sysmail_event_log;  
SELECT * FROM msdb.dbo.sysmail_mailitems;
```

3. View the results. The first result shows system events for Database Mail, and the second shows records of e-mail messages that have been sent.
4. Keep SQL Server Management Studio open for the next demonstration

Lesson 3

Configuring Operators, Notifications, and Alerts

Contents:

Demonstration: Configuring SQL Server Agent Operators	8
Demonstration: Configuring SQL Server Agent Alerts	8

Demonstration: Configuring SQL Server Agent Operators

Demonstration Steps

Enable a SQL Server Agent Mail Profile

1. Ensure that you have completed the previous demonstration in this module.
2. In SQL Server Management Studio, in Object Explorer, right-click **SQL Server Agent** and click **Properties**.
3. In the **SQL Server Agent Properties** dialog box, on the **Alert System** page, select **Enable mail profile** and in the **Mail profile** drop-down list, select **SQL Server Agent Profile**. Then click **OK**.

Create an Operator

1. In Object Explorer, under **SQL Server Agent**, right-click **Operators** and click **New Operator**.
2. In the **New Operator** dialog box, in the **Name** box type **Student**, in the **E-mail name** box type **student@adventureworks.msft**, and click **OK**.

Configure a Job to Notify an Operator

1. In Object Explorer, under **SQL Server Agent**, expand **Jobs** and view the existing jobs.
2. Right-click the **Back Up Database - AdventureWorks** job and click **Properties**.
3. In the **Job Properties - Back Up Database - AdventureWorks** dialog box, on the **Notifications** tab, select **E-mail**, select **Student**, and select **When the job completes**. Then click **OK**.
4. Under the **Operators** folder, right-click **Student** and click **Properties**. On the **Notifications** page, select **Jobs**, note the job notifications that have been defined for this operator. Then click **Cancel**.
5. Right-click the **Back Up Database - AdventureWorks** job and click **Start Job at Step**. Then, when the job has completed, click **Close**.
6. Under the **Operators** folder, right-click **Student** and click **Properties**. On the **History** page, note the most recent notification by e-mail attempt. Then click **Cancel**.
7. In the C:\inetpub\mailroot\Drop folder, and verify that a new email message has been created.
8. Double-click the most recent message to view it in Outlook. Then, when you have read the message, close it and minimize the **Drop** window.
9. Keep SQL Server Management Studio open for the next demonstration.

Demonstration: Configuring SQL Server Agent Alerts

Demonstration Steps

Create an Alert

1. In SQL Server Management Studio, in Object Explorer, under **SQL Server Agent**, right-click **Alerts** and click **New Alert**.
2. In the **New Alert** dialog box, on the **General** page, enter the name **Log Full Alert**. In the **Type** drop-down list, note that you can configure alerts on WMI events, performance monitor conditions, and SQL Server events. Then select **SQL Server event alert**, select **Error number**, and enter the number **9002** (which is the error number raised by SQL Server when a database transaction log becomes full).
3. In the **New Alert** dialog box, on the **Response** page, select **Notify operators** and select the **E-mail** checkbox for the **Student** operator.
4. In the **New Alert** dialog box, on the **Options** page, under **Include alert error text in**, select **E-mail**. Then click **OK**.

Test an Alert

1. In SQL Server Management Studio, open the **TestAlert.sql** script file in the D:\Demofiles\Mod13 folder.
2. Click **Execute** and wait while the script fills a table in the **TestAlertDB** database. When the log file for that database is full, error 9002 occurs.
3. In Object Explorer, under the **Alerts** folder, right-click **Log Full Alert** and click **Properties**. Then on the **History** page, note the **Date of last alert** and **Date of last response** values and click **Cancel**.
4. In the C:\inetpub\mailroot\Drop folder, and verify that a new email message has been created.
5. Double-click the most recent message to view it in Outlook. Then, when you have read the message, close it and minimize the **Drop** window.
6. Close SQL Server Management Studio without saving any files.

Module Review and Takeaways

Best Practice

When planning notifications and alerts in SQL Server, consider the following best practices:

Use Database Mail and not SQL Mail (which is deprecated).

Configure different profiles for different usage scenarios.

Provide limited access to the ability to send e-mail messages from the database engine.

Implement a retention policy for Database Mail log and mail auditing.

Create a fail-safe operator.

Define Alerts for severe error messages.

Review Question(s)

Question: You want to designate a colleague in the IT team as an operator, but this colleague does not have a login in the SQL Server instance. Should you create one?

Answer: No. Operators do not need a login, just an e-mail, pager, or NET SEND address at which they can be notified.

Question: You are planning to send notifications from SQL Server, and think it might be easier to use NET SEND notifications instead of e-mail. Why should you not do this?

Answer: Broadcast messages are generally disabled on most modern operating systems, so NET SEND messages would not appear. This option has been deprecated, and will be removed in a future release of SQL Server,

Lab Review Questions and Answers

Lab: Using Notifications and Alerts

Question and Answers

Lab Review

Question: Under what circumstances would e-mail notifications have been sent to the DBA Team operator you created?

Answer: As the fail-safe operator, the DBA Team operator would have been notified if an error prevented SQL Server from accessing the Database Mail configuration tables in the **msdb** database. Had pager notifications been enabled, the fail-safe operator would be used to send alerts at times when no other operator is on duty.

