Microsoft

OFFICIAL MICROSOFT LEARNING PRODUCT

20762B

Developing SQL Databases

Product Number: 20762B

Part Number (if applicable):

Released: 01/2017

Module 1

An Introduction to Database Development

Contents:

Lesson 1

Introduction to the SQL Server Platform

Contents:

Question and Answers

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

Items	
1	Database Engine
2	Data Quality Services Client
3	Enterprise
4	Master Data Services
5	Connectivity
6	Developer
7	Replication
8	Profiler
9	Web
10	Integration Services
11	SQL Server Management Studio
12	Standard
13	SQL Server Data Tools

Category 1		Category 2		Category 3
Component		Tool		Edition

Answer:

Category 1		Category 2		Category 3
Component		Tool		Edition
Database Engine Master Data Services Replication Integration Services		Data Quality Services Client Connectivity Profiler SQL Server Management Studio SQL Server Data Tools		Enterprise Developer Web Standard

Lesson 2

SQL Server Database Development Tasks

Contents:

Question and Answers

Question: Which one of the following tools can you use to create and deploy SSIS packages?

( ) SQL Server Management Studio.

( ) SQL Server Profiler.

( ) Database Engine Tuning Advisor.

( ) SQL Server Data Tools.

( ) SQL Server Configuration Manager.

Answer:

( ) SQL Server Management Studio.

( ) SQL Server Profiler.

( ) Database Engine Tuning Advisor.

(√) SQL Server Data Tools.

( ) SQL Server Configuration Manager.

Demonstration: Using SSMS and SSDT

Demonstration Steps

Use SSMS to Connect to an On-premises Instance of SQL Server

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running and then log on to 20762B-MIA-SQL as **AdventureWorks\Student** with the password **Pa$$w0rd**.
2. Navigate to **D:\Demofiles\Mod01**, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**.
4. On the taskbar, click **Microsoft SQL Server Management Studio**.
5. In the **Connect to Server** dialog box, ensure that **Server type** is set to **Database Engine**.
6. In the **Server name** box, type **MIA-SQL**.

Run a Transact-SQL Script

1. In Object Explorer, expand **Databases**, expand **AdventureWorks**, and then review the database objects.
2. In Object Explorer, right-click **AdventureWorks**, and then click **New Query**.
3. Type the query shown in the snippet below:

```
SELECT * FROM Production.Product ORDER BY ProductID;
```

Note the use of IntelliSense while you are typing this query, and then on the toolbar, click **Execute**. Note how the results are returned.

4. On the **File** menu, click **Save SQLQuery1.sql**.
5. In the **Save File As** dialog box, navigate to **D:\Demofiles\Mod01**, and then click **Save**. Note that this saves the query to a file.
6. On the **Results** tab, right-click the cell for **ProductID 1** (first row and first cell), and then click **Save Results As**.

7. In the **Save Grid Results** dialog box, navigate to the **D:\Demofiles\Mod01** folder.
8. In the **File name** box, type **Demonstration2AResults**, and then click **Save**. Note that this saves the query results to a file.
9. On the **Query** menu, click **Display Estimated Execution Plan**. Note that SSMS can do more than just execute queries.
10. On the **Tools** menu, click **Options**.
11. In the **Options** dialog box, expand **Query Results**, expand **SQL Server**, and then click **General**. Review the available configuration options, and then click **Cancel**.

Open a SQL Server Management Studio Project

1. On the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, open the **D:\Demofiles\Mod01\Demo01.ssmssln** project.
3. In Solution Explorer, note the contents of Solution Explorer, and that by using a project or solution you can save the state of the IDE. This means that any open connections, query windows, or Solution Explorer panes will reopen in the state they were saved in.

Connect to Servers and Databases

1. In Object Explorer, click **Connect** and note the other SQL Server components to which connections can be made.
2. On the **File** menu, point to **New**, and then click **Database Engine Query** to open a new connection.
3. In the **Connect to Database Engine** dialog box, in the **Server name** box, type **MIA-SQL**.
4. In the **Authentication** drop-down list, select **Windows Authentication**, and then click **Connect**.
5. In the **Available Databases** drop-down list on the toolbar, click **tempdb**. Note that this changes the database against which the query is executed.
6. Right-click in the query window, point to **Connection**, and then click **Change Connection**. This will reconnect the query to another instance of SQL Server.
7. In the **Connect to Database Engine** dialog box, click **Cancel**.

Use SSDT to Run a Transact-SQL Script

1. On the taskbar, click **Visual Studio 2015**.
2. On the **Tools** menu, click **Connect to Database**.
3. In the **Choose Data Source** dialog box, in the **Data source** list, click **Microsoft SQL Server**, and then click **Continue**.
4. In the **Add Connection** dialog box, in the **Server name** box, type **MIA-SQL**.
5. In the **Select or enter a database name** drop-down list, click **AdventureWorks**, and then click **OK**.
6. In Server Explorer, expand **Data Connections**.
7. Right-click **mia-sql.AdventureWorks.dbo**, and then click **New Query**.
8. In the SQLQuery1.sql pane, type:

```
SELECT * FROM Production.Product ORDER BY ProductID;
```

9. On the toolbar, click **Execute**.
10. Note that you can view results, just as you can in SSMS.

11. Close Visual Studio 2015 without saving any changes.

Module Review and Takeaways

Question: Which IDE do you think you will use to develop on SQL Server, SSMS or SSDT?

Answer: Answers will vary, depending on which tools students feel most comfortable using. Developers can manage and develop SQL code with both tools.

Module 2

Designing and Implementing Tables

Contents:

Lesson 1

Designing Tables

Contents:

Question and Answers

Question: Would it be reasonable to have columns called, for example, **AddressLine1**, **AddressLine2**, and **AddressLine3** in a normalized design?

> **Answer:** Yes, these columns might represent distinct attributes of an object, such as a customer. As an example, the addresses might represent different lines on a form.

Demonstration: Working with Normalization

Demonstration Steps

1. Ensure that the MSL-TMG1, 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running, and then log on to 20762B-MIA-SQL as **AdventureWorks\Student** with the password **Pa$$w0rd**.
2. In File Explorer, navigate to **D:\Demofiles\Mod02**, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**.
4. On the taskbar, click **Microsoft SQL Server Management Studio**.
5. In the **Connect to Server** dialog box, connect to **MIA-SQL**, using **Windows Authentication**.
6. On the **File** menu, point to **Open**, click **Project/Solution**.
7. In the **Open Project** dialog box, navigate to the **D:\Demofiles\Mod02** folder, click **Demo.ssmssln**, and then click **Open**.
8. In Solution Explorer, expand **Queries**, and then double-click **1 - Normalization.sql**.
9. Select the code under the **Step 1: Set AdventureWorks as the current database** comment, and then click **Execute**.
10. Select the code under the **Step 2: Create a table for denormalizing** comment, and then click **Execute**.
11. Select the code under the **Step 3: Alter the table to conform to third normal form** comment, and then click **Execute**.
12. Select the code under the **Step 4: Drop and recreate the ProductList table** comment, and then click **Execute**.
13. Select the code under the **Step 5: Populate the ProductList table** comment, and then click **Execute**.
14. Close SQL Server Management Studio without saving any changes.

Lesson 2

Data Types

Contents:

Question and Answers

Question: What would be a suitable data type for storing the value of a check box that can be 0 for cleared, 1 for selected, or -1 for disabled?

Answer: smallint (note that **tinyint** cannot be negative).

Lesson 3

Working with Schemas

Contents:

Question and Answers

Question: Which of the following objects cannot be stored in a schema?

( ) Table

( ) Function

( ) Database role

( ) View

( ) Stored procedure

Answer:

( ) Table

( ) Function

(√) Database role

( ) View

( ) Stored procedure

Demonstration: Working with Schemas

Demonstration Steps

1. Ensure that you have completed the previous demonstrations in this module.
2. On the taskbar, click **Microsoft SQL Server Management Studio**.
3. In the **Connect to Server** dialog box, in the **Server** box, type the URL of the Azure server <Server Name>.database.windows.net (where <Server Name> is the name of the server you created).
4. In the **Authentication** list, click **SQL Server Authentication**.
5. In the **User name** box, type **Student**, and in the **Password** box, type **Pa$$w0rd**, and then click **Connect**.
6. On the **File** menu, point to **Open**, click **Project/Solution**.
7. In the **Open Project** dialog box, navigate to the **D:\Demofiles\Mod02** folder, click **Demo.ssmssln**, and then click **Open**.
8. In Solution Explorer, under **Queries**, double-click **2 - Schemas.sql**.
9. In the **Available Databases** list, click **AdventureWorksLT**.
10. Select the code under the **Step 2: Create a Schema** comment, and then click **Execute**.
11. Select the code under the **Step 3: Create a table using the new schema** comment, and then click **Execute**.
12. Select the code under the **Step 4: Drop the schema** comment, and then click **Execute**. Note that the schema cannot be dropped while objects exist in it.
13. Select the code under the **Step 5: Drop and the table and then the schema** comment, and then click **Execute**.
14. Leave SQL Server Management Studio open for the next demonstration.

Lesson 4

Creating and Altering Tables

Contents:

Question and Answers

Question: When creating a computed column, why is it good practice to include the PERSISTED keyword? What are the consequences of excluding PERSISTED when the table has several million records?

> **Answer:** If you include the PERSISTED keyword, the value in the computed column will be retained. Without this, the query engine will calculate the values every time a SELECT statement is run against this column. Small tables with a few hundred rows may not show an obvious loss of performance, but tables with millions of rows could be slow to return results because of the number of calculations that must be performed.

Demonstration: Working with Tables

Demonstration Steps

1. Ensure that you have completed the previous demonstrations in this module.
2. In SQL Server Management Studio, in Solution Explorer, under **Queries**, double-click **3 - Create Tables.sql**.
3. In the **Available Databases** list, click **AdventureWorksLT**.
4. Select the code under the **Step 2: Create a table** comment, and then click **Execute**.
5. Select the code under the **Step 3: Alter the SalesLT.Courier table** comment, and then click **Execute**.
6. Select the code under the **Step 4: Drop the tables** comment, and then click **Execute**.

Leave SQL Server Management Studio open for the next demonstration.

Demonstration: Working with Temporary Tables

Demonstration Steps

1. Ensure that you have completed the previous demonstrations in this module.
2. In SQL Server Management Studio, in Solution Explorer, under **Queries**, double-click **4 - Temporary Tables.sql**.
3. Right-click the query pane, point to **Connection**, and then click **Change Connection**.
4. In the **Connect to Database Engine window** dialog box, in the **Server name** box, type **MIA-SQL**, in the **Authentication** box, select **Windows Authentication**, and then click **Connect**.
5. Select the code under the **Step 1: Create a local temporary table** comment, and then click **Execute**.
6. In Solution Explorer, under **Queries**, double-click **5 - Temporary Tables.sql**.
7. Select the code under the **Step 1: Select and execute the following query** comment, and then click **Execute**. Note that this session cannot access the local temporary table from the other session.
8. Switch to the **4 - Temporary Tables.sql** pane.
9. Select the code under the **Step 3: Create a global temporary table** comment, and then click **Execute**.
10. Switch to the **5 - Temporary Tables.sql** pane.
11. Select the code under the **Step 2: Select and execute the following query** comment, and then click **Execute**. Note that this session can access the global temporary table from the other session.
12. Switch to the **4 - Temporary Tables.sql** pane.

13. Select the code under the **Step 5: Drop the two temporary tables** comment, and then click **Execute**.
14. Leave SQL Server Management Studio open for the next demonstration.

Demonstration: Working with Computed Columns

Demonstration Steps

1. Ensure that you have completed the previous demonstrations in this module.
2. In SQL Server Management Studio, in Solution Explorer, under **Queries**, double-click **6 - Computed Columns.sql**.
3. Right-click the query pane, point to **Connection**, and then click **Change Connection**.
4. In the **Connect to Database Engine window** dialog box, in the **Server name** box, type the URL for the Azure account, in the **Authentication** box, select **SQL Server Authentication**, in the **Login** box, type **Student**, and in the **Password** box, type **Pa$$w0rd**, and then click **Connect**.
5. In the **Available Databases** list, click **AdventureWorksLT**.
6. Select the code under the **Step 2: Create a table with two computed columns** comment, and then click **Execute**.
7. Select the code under the **Step 3: Populate the table with data** comment, and then click **Execute**.
8. Select the code under the **Step 4: Return the results from the SalesLT.SalesOrderDates table** comment, and then click **Execute**.
9. Select the code under the **Step 5: Update a row in the SalesLT.SalesOrderDates table** comment, and then click **Execute**.
10. Select the code under the **Step 6: Create a table with a computed column that is not persisted** comment and then click **Execute**.
11. Select the code under the **Step 7: Populate the table with data** comment, and then click **Execute**.
12. Select the code under the **Step 8 - Return the results from the SalesLT.TotalSales table** comment, and then click **Execute**.
13. Close SQL Server Management Studio without saving any changes.

Module Review and Takeaways

Best Practice

All tables should have primary keys.

Foreign keys should be declared within the database in almost all circumstances. Developers often suggest that the application will ensure referential integrity, but experience shows that this is a poor option. Databases are often accessed by multiple applications, and bugs are also easy to miss when they first start to occur.

Lab Review Questions and Answers

Lab: Designing and Implementing Tables

Question and Answers

Lab Review

Question: When should a column be declared as nullable?

Answer: A column should be declared as nullable when the value can be unknown.

Module 3

Advanced Table Designs

Contents:

Lesson 1

Partitioning Data

Contents:

Question and Answers

Question: What is the maximum number of partitions you can add to a table or index?

( ) 4

( ) 256

( ) 1,000

( ) 15,000

( ) 256,000

Answer:

( ) 4

( ) 256

( ) 1,000

(√) 15,000

( ) 256,000

Demonstration: Creating a Partitioned Table

Demonstration Steps

Creating a Partitioned Table

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are both running, and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Run **D:\Demofiles\Mod03\Setup.cmd** as an administrator.
3. In the **User Account Control** dialog box, click **Yes**, and then if prompted with the question **Do you want to continue with this operation?** type **Y**, then press Enter.
4. After the script has run successfully, and **Press any key to continue** appears, press any key to exit the command window.
5. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine instance using Windows® authentication.
6. Open the **Demo.ssmssln** solution in the **D:\Demofiles\Mod03\Demo** folder.
7. In Solution Explorer, open the **1 - Partitioning.sql** script file.
8. Select and execute the query under **Step 1** to use the master database.
9. Select and execute the query under **Step 2** to create four filegroups, and add a file to each filegroup.
10. Select and execute the query under **Step 3** to switch to the **AdventureWorks** database.
11. Select and execute the query under **Step 4** to create the partition function.
12. Select and execute the query under **Step 5** to create the **OrdersByYear** partition scheme.
13. Select and execute the query under **Step 6** to create the **Sales.SalesOrderHeader_Partitioned** table.
14. Select and execute the query under **Step 7** to copy data into the **Sales.SalesOrderHeader_Partitioned** table.
15. Select and execute the query under **Step 8** to check the rows counts within each of the partitions.

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

Lesson 2

Compressing Data

Contents:

Question and Answers

Question: You have a Customers table with the following columns: Title, FirstName, MiddleInitial, LastName, Address1, Address2, City, PostalCode, Telephone, and Email. Which of the following options will give the best reduction in storage?

( ) Add ROW compression to the table.

( ) Add PAGE compression to the table.

( ) Add Unicode compression to the table.

( ) Create a nonclustered index on the FirstName and LastName columns.

( ) None of the above.

Answer:

( ) Add ROW compression to the table.

(√) Add PAGE compression to the table.

( ) Add Unicode compression to the table.

( ) Create a nonclustered index on the FirstName and LastName columns.

( ) None of the above.

Demonstration: Compressing Data

Demonstration Steps

Compressing Data

1. In Server Management Studio, in Solution Explorer, open the **2 - Compressing Data.sql** script file.
2. Select and execute the query under **Step 1** to use the **AdventureWorks** database.
3. Select and execute the query under **Step 2** to run the **sp_estimate_data_compression_savings** procedure against the **Sales.SalesOrderDetail** table.
4. Select and execute the query under **Step 3** to add row compression to the **Sales.SalesOrderDetail** table.
5. Select and execute the code under **Step 4** to run the **sp_estimate_data_compression_savings** procedure against the **Sales.SalesOrderDetail** table to see if the table can be further compressed.
6. Select and execute the query under **Step 5** to rebuild indexes 1 and 3.
7. Select and execute the query under **Step 6** to run **sp_estimate_data_compression_savings** procedure against the **Sales.SalesOrderDetail** table to show how the size of the table has been reduce.
8. Keep SQL Server Management Studio open for the next demonstration.

Lesson 3

Temporal Tables

Contents:

Question and Answers

Question: Which of the following statements is incorrect?

( ) A temporal table must have two period columns, one for the start time, one for the end time.

( ) If you include the HIDDEN keyword when specifying the period columns on a temporal table, these columns cannot be included in a SELECT query.

( ) The FOR SYSTEM_TIME clause is used to query historical data.

( ) You can add system-versioning to a memory-optimized table.

( ) The history table for a system-versioned memory-optimized table is stored on disk.

Answer:

( ) A temporal table must have two period columns, one for the start time, one for the end time.

(√) If you include the HIDDEN keyword when specifying the period columns on a temporal table, these columns cannot be included in a SELECT query.

( ) The FOR SYSTEM_TIME clause is used to query historical data.

( ) You can add system-versioning to a memory-optimized table.

( ) The history table for a system-versioned memory-optimized table is stored on disk.

Resources

System-Versioned Memory-Optimized Tables

 **Best Practice:** If you want to return just historical data, use the CONTAINED IN subclause for the best performance, as this only uses the history table for querying.

Demonstration: Adding System-Versioning to an Existing Table

Demonstration Steps

Adding System-Versioning to an Existing Table

1. In SQL Server Management Studio, in Solution Explorer, open **the 3 - Temporal Tables.sql** script file.
2. Select and execute the query under **Step 1** to use the AdventureWorks database.
3. Select and execute the query under **Step 2**, then add the two columns, StartDate and EndDate, to the Person.Person table.
4. After you have run the first ALTER TABLE script, select and execute the script to alter the table and add system-versioning.
5. In Object Explorer, expand **Databases**, expand **AdventureWorks2016**, right-click **Tables**, and click then **Refresh**.
6. In the list of tables and point out the **Person.Person** table. The name includes **(System-Versioned)**.
7. Expand the **Person.Person (System-Versioned)** table node to display the history table. Point out the name of the table included **(History)**.
8. Expnad the **Person.Person_History (History)** node, and then expand **Columns**. Point out that the column names are identical to the current table.

9. Select and execute the query under **Step 4** to update the row in the **Person.Person** table for BusinessEntityID 1704.
10. Select and execute the query under **Step 5** to show the history of changes for BusinessEntityID 1704.
11. Close SQL Server Management Studio without saving anything.

Module Review and Takeaways

Best Practice

One of the disadvantages of partitioning is that it can be complicated to set up. Furthermore, this feature is only available in SQL Server Enterprise Edition. However, you can use the Developer Edition to replicate your production systems and test your partitioning scenario before applying it in your live environment. As with any major database changes, it is always recommended that you take a backup before applying these changes.

Review Question(s)

Question: What are the advantages of using system-versioning versus a custom-built application to store data changes?

Answer:

- The functionality of detecting and storing changes is built-in, so it just requires the additional columns, and to be switched on for each table.
- It can be used in all your user databases in the production environment.
- It is very quick to set up.
- It can be added to existing tables with minimal coding and disruption.
- The new FOR SYSTEM_TIME subclauses return historical data within precise timespans.
- It can be turned off at any time.

The biggest advantage of using system-versioned tables rather than coding an application to record data changes is the time saved. This out-of-the-box functionality can be added in a matter of minutes with no additional requirement for complicated coding. Providing storage is not an issue, it can be added to every table as needed, in the minimum amount of time.

Lab Review Questions and Answers

Lab: Using Advanced Table Designs

Question and Answers

Lab Review

Question: Discuss scenarios that you have experienced where you think partitioning would have been beneficial. Have you worked with databases that could have had older data archived? Were the databases large enough to split the partitions across physical drives for better performance, or to quicken the backup process? Furthermore, could any of this data be compressed? Give reasons for your answers.

Answer: Answers will depend on the students' experience.

Module 4

Ensuring Data Integrity Through Constraints

Contents:

Lesson 1

Enforcing Data Integrity

Contents:

Question and Answers

Put the following constraint types in order by numbering each to indicate the order of importance to minimize constraint checking effort.

	Steps
	Specify data type.
	Indicate column null-ability.
	Indicate column default value.
	Indicate a check constraint.
	Write a trigger to control the column contents.

Answer:

	Steps
1	Specify data type.
2	Indicate column null-ability.
3	Indicate column default value.
4	Indicate a check constraint.
5	Write a trigger to control the column contents.

Lesson 2

Implementing Data Domain Integrity

Contents:

Question and Answers

Question: True or false? When you have a check constraint on a column, it is not worth having a NOT NULL constraint because any nulls will be filtered out by the check constraint.

( ) True

( ) False

Answer:

( ) True

(√) False

Demonstration: Data and Domain Integrity

Demonstration Steps

Enforce Data and Domain Integrity

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Run **D:\Demofiles\Mod04\Setup.cmd** as an administrator.
3. In the **User Account Control** dialog box, click **Yes**.
4. On the taskbar, click **Microsoft SQL Server Management Studio**.
5. In the **Connect to Server** dialog box, in the **Server name** box, type **MIA-SQL** and then click **Connect**.
6. On the **File** menu, point to **Open**, click **Project/Solution**.
7. In the **Open Project** dialog box, navigate to **D:\Demofiles\Mod04**, click **Demo04.ssmssln**, and then click **Open**.
8. In Solution Explorer, expand the **Queries** folder, and double-click **21 - Demonstration 2A.sql**.
9. Familiarize yourself with the requirement using the code below **Step 1: Review the requirements for a table design**.
10. Place the pseudo code for your findings for the requirements below **Step 2: Determine the data types, null-ability, default and check constraints that should be put in place**.
11. Highlight the code below **Step 3: Check the outcome with this proposed solution**, and click **Execute**.
12. Highlight the code below **Step 4: Execute statements to test the actions of the integrity constraints**, and click **Execute**.
13. Highlight the code below **Step 5: INSERT rows that test the nullability and constraints**, and click **Execute**. Note the errors.
14. Highlight the code below **Step 6: Query sys.sysconstraints to see the list of constraints**, and click **Execute**.
15. Highlight the code below **Step 7: Explore system catalog views through the INFORMATION_SCHEMA owner**, and click **Execute**.
16. Close SQL Server Management Studio, without saving any changes.

Lesson 3

Implementing Entity and Referential Integrity

Contents:

Question and Answers

Question: You want to set up a cascading referential integrity constraint between two tables that has the minimum impact on queries that are only interested in current data rather than historic data. What would you use?

( ) ON DELETE CASCADE

( ) ON DELETE RESTRICT

( ) ON DELETE SET DEFAULT

( ) ON DELETE SET NULL

Answer:

( ) ON DELETE CASCADE

( ) ON DELETE RESTRICT

( ) ON DELETE SET DEFAULT

(√) ON DELETE SET NULL

Demonstration: Sequences Demonstration

Demonstration Steps

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Run **D:\Demofiles\Mod04\Setup.cmd** as an administrator.
3. In the **User Account Control** dialog box, click **Yes**.
4. On the taskbar, click **Microsoft SQL Server Management Studio**.
5. In the **Connect to Server** dialog box, in the **Server name** box, type **MIA-SQL** and then click **Connect**.
6. On the **File** menu, point to **Open**, click **Project/Solution**.
7. In the **Open Project** dialog box, navigate to **D:\Demofiles\Mod04**, click **Demo04.ssmssln**, and then click **Open**.
8. In Solution Explorer, double-click the **31 - Demonstration 3A.sql** script file.
9. Highlight the code below **Step 1: Open a new query window to the tempdb database**, and click **Execute**.
10. Highlight the code below **Step 2: Create the dbo.Opportunity table**, and click **Execute**.
11. Highlight the code below **Step 3: Populate the table with two rows**, and click **Execute**.
12. Highlight the code below **Step 4: Check the identity values added**, and click **Execute**.
13. Highlight the code below **Step 5: Try to insert a specific value for OpportunityID**, and click **Execute**. Note the error.
14. Highlight the code below **Step 6: Add a row without a value for LikelyClosingDate**, and click **Execute**.
15. Highlight the code below **Step 7: Query the table to see the value in the LikelyClosingDate column**, and click **Execute**.
16. Highlight the code below **Step 8: Create 3 Tables with separate identity columns**, and click **Execute**.

17. Highlight the code below **Step 9: Insert some rows into each table**, and click **Execute**.
18. Highlight the code below **Step 10: Query the 3 tables in a single view and note the overlapping ID values**, then click **Execute**.
19. Highlight the code below **Step 11: Drop the tables**, and click **Execute**.
20. Highlight the code below **Step 12: Create a sequence to use with all 3 tables**, and click **Execute**.
21. Highlight the code below **Step 13: Recreate the tables using the sequence for default values**, and click **execute**.
22. Highlight the code below **Step 14: Reinsert the same data**, and click **Execute**.
23. Highlight the code below **Step 15: Note the values now appearing in the view**, and click **execute**.
24. Highlight the code below **Step 16: Note that sequence values can be created on the fly**, and click **Execute**.
25. Highlight the code below **Step 17: Re-execute the same code and note that the sequence values**, and click **Execute**.
26. Highlight the code below **Step 18: Note that when the same entry is used multiple times in a SELECT statement, that the same value is used**, and click **Execute**.
27. Highlight the code below **Step 19: Fetch a range of sequence values**, and click **Execute**.
28. Close SQL Server Management Studio, without saving any changes.

Demonstration: Entity and Referential Integrity

Demonstration Steps

Define entity integrity for a table, define referential integrity for a table, and define cascading referential integrity actions for the constraint.

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running, and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Run **D:\Demofiles\Mod04\Setup.cmd** as an administrator.
3. In the **User Account Control** dialog box, click **Yes**.
4. On the taskbar, click **Microsoft SQL Server Management Studio**.
5. In the **Connect to Server** dialog box, in the **Server name** box, type **MIA-SQL** and then click **Connect**.
6. On the **File** menu, point to **Open**, click **Project/Solution**.
7. In the **Open Project** dialog box, navigate to **D:\Demofiles\Mod04**, click **Demo04.ssmssln**, and then click **Open**.
8. In Solution Explorer, double-click the **32 - Demonstration 3B.sql** script file.
9. Highlight the code below **Step 1: Open a new query window to tempdb**, and click **Execute**.
10. Highlight the code below **Step 2: Create the Customer and CustomerOrder tables**, and click **Execute**.
11. Highlight the code below **Step 3: Select the list of customers**, and click **Execute**.
12. Highlight the code below **Step 4: Try to insert a CustomerOrder row for an invalid customer**, and click **Execute**. Note the error message.

13. Highlight the code below **Step 5: Try to remove a customer that has an order**, and click **Execute**. Note the error message.
14. Highlight the code below **Step 6: Replace it with a named constraint with cascade**, and click **Execute**.
15. Highlight the code below **Step 7: Select the list of customer orders, try a delete again**, and click **Execute**.
16. Highlight the code below **Step 8: Note how the cascade option caused the orders**, and click **Execute**.
17. Highlight the code below **Step 9: Try to drop the referenced table and note the error**, then click **Execute**. Note the error message.
18. Close SQL Server Management Studio, without saving any changes.

Module Review and Takeaways

Best Practice

When you create a constraint on a column, if you do not specify a name for the constraint, SQL Server will generate a unique name for it. However, you should always name constraints to adhere to your naming conventions. This makes the constraints easier to identify when they appear in error messages. They are also easier to alter because you don't have to remember any arbitrary numbers that SQL Server uses to make the names unique when it generates constraint names automatically.

Review Question(s)

Question: Would you consider that you need to CHECK constraints if an application is already checking the input data?

> **Answer:** Yes. Even if an application checks that data conforms in the user interface or back-end code, error conditions may arise that cause fields to become corrupt or null. Also, procedures for archiving, backing up, and triggers might attempt to copy bad data into the table, which can then cause an application to fail. Multiple applications may be accessing the same data—some of these do not perform the checks that are expected.

Question: What are some scenarios in which you might want to temporarily disable constraint checking?

> **Answer:** Constraint checking can affect performance, so you might want to disable it when performing large inserts, such as in a restore, or copying large numbers of records for an archive. In addition, you might know that duplicate or invalid data exists in your source or destination and have a plan to deal with cleaning up the data afterwards—such as by using a script or other procedure.

Lab Review Questions and Answers

Lab: Ensuring Data Integrity Through Constraints

Question and Answers

Lab Review

Question: Why implement CHECK constraints if an application is already checking the input data?

Answer: Even if an application checks that data conforms in the user interface or back-end code, error conditions may arise that cause fields to become corrupt or null. Also, procedures for archiving, backing up, and triggers might attempt to copy bad data into the table, which can then cause an application to fail. Multiple applications may be accessing the same data.

Question: What are some scenarios in which you might want to temporarily disable constraint checking?

Answer: Constraint checking can affect performance, so you might want to disable it when performing large inserts, such as in a restore procedure or copying large numbers of records for an archive. You might also know that duplicate or invalid data exists in your source or destination. You should have a plan to deal with cleaning up the data afterwards, such as by using a script or other procedure.

Question: True or false? A PRIMARY KEY and a UNIQUE constraint are doing the same thing using different code words.

( ) True

( ) False

Answer:

( ) True

(√) False

Module 5

Introduction to Indexes

Contents:

Lesson 1

Core Indexing Concepts

Contents:

Question and Answers

Categorize each item into the appropriate property of an index. Indicate your answer by writing the category number to the right of each item.

Items	
1	A factor of the number of rows returned, compared to the total number of rows in the index.
2	How unique the data is, compared to the other data in the index.
3	The number of unique levels between the root node and leaf nodes in the index.
4	The most important consideration when designing an index.

Category 1		Category 2		Category 3
Selectivity		Density		Depth

Answer:

Category 1		Category 2		Category 3
Selectivity		Density		Depth
A factor of the number of rows returned, compared to the total number of rows in the index. The most important consideration when designing an index.		How unique the data is, compared to the other data in the index.		The number of unique levels between the root node and leaf nodes in the index.

Demonstration: Viewing Index Fragmentation

Demonstration Steps

Identify Fragmented Indexes

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Navigate to **D:\Demofiles\Mod05**, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**.
4. On the taskbar, click **Microsoft SQL Server Management Studio**.
5. In the **Connect to Server** dialog box, in the **Server name** box, type **MIA-SQL**, and then click **Connect**.
6. On the **File** menu, point to **Open**, click **Project/Solution**.
7. In the **Open Project** dialog box, navigate to **D:\Demofiles\Mod05**, click **Demo05.ssmssln**, and then click **Open**.
8. In Solution Explorer, expand **Queries**, and then double-click **Demonstration 1.sql**.
9. Select the code under the **Step 1: Open a query window to the AdventureWorks database** comment, and then click **Execute**.
10. Select the code under the **Step 2: Query the index physical stats DMV** comment, and then click **Execute**.
11. Note the avg_fragmentation_in_percent returned.
12. Select the code under the **Step 4: Note that there are choices on the level of detail returned** comment, and then click **Execute**.

View the Fragmentation of an Index in SSMS

1. In Object Explorer, expand **Databases**, expand **AdventureWorks**, expand **Tables**, expand **Production.Product**, and then expand **Indexes**.
2. Right-click **AK_Product_Name (Unique, Non-Clustered)**, and then click **Properties**.
3. In the **Index Properties - AK_Product_Name** dialog box, in the **Select a page** pane, click **Fragmentation**.
4. Note the **Total fragmentation** is **75%**, and that this matches the results from the query executed in the previous task step 11.
5. In the **Index Properties - AK_Product_Name** dialog box, click **Cancel**.
6. Keep SQL Server Management Studio open for the next demonstration.

Lesson 2

Data Types and Indexes

Contents:

Question and Answers

Put the following data types in order of the smallest to the maximum possible size.

	Steps
	BIT
	date
	bigint
	datetimeoffset
	uniqueidentifier
	char
	text

Answer:

	Steps
1	BIT
2	date
3	bigint
4	datetimeoffset
5	uniqueidentifier
6	char
7	text

Lesson 3

Heaps, Clustered, and Nonclustered Indexes

Contents:

Question and Answers

Categorize each attribute of an index. Indicate your answer by writing the attribute number to the right of each index.

Items	
1	Data is stored in the table wherever there is space.
2	Data is stored by a key in a specified order.
3	Will be defined on top of a heap or rowstore.
4	Most efficient operation is an insert.
5	Can improve the performance of updates, deletes and selects.
6	Can improve the performance of selects.
7	Best used when scanning for data.
8	Best used when seeking for data.

Category 1		Category 2		Category 3
Heap		Clustered		Nonclustered

Answer:

Category 1		Category 2		Category 3
Heap		Clustered		Nonclustered
Data is stored in the table wherever there is space. Most efficient operation is an insert. Best used when scanning for data.		Data is stored by a key in a specified order. Can improve the performance of updates, deletes and selects. Best used when seeking for data.		Will be defined on top of a heap or rowstore. Can improve the performance of selects.

Demonstration: Working with Clustered and Nonclustered Indexes

Demonstration Steps

1. In SQL Server Management Studio, in Solution Explorer, double-click **Demonstration 2.sql**.
2. Select the code under the **Step 1: Open a new query window against the tempdb database** comment, and then click **Execute**.
3. Select the code under the **Step 2: Create a table with a primary key specified** comment, and then click **Execute**.
4. Select the code under the **Step 3: Query sys.indexes to view the structure** comment, and then click **Execute**.
5. In Object Explorer, expand **Databases**, expand **System Databases**, expand **tempdb**, expand **Tables**, expand **dbo.PhoneLog**, and then expand **Indexes**.
6. Note that a clustered index was automatically created on the table.
7. Select the code under the **Step 4: Insert some data into the table** comment, and then click **Execute**.
8. Select the code under the **Step 5: Check the level of fragmentation via sys.dm_db_index_physical_stats** comment, and then click **Execute**.
9. Scroll to the right, and note the **avg_fragmentation_in_percent** and **avg_page_space_used_in_percent** returned.
10. Select the code under the **Step 7: Modify the data in the table - this will increase data and cause page fragmentation** comment, and then click **Execute**.
11. Select the code under the **Step 8: Check the level of fragmentation via sys.dm_db_index_physical_stats** comment, and then click **Execute**.
12. Scroll to the right, and note the **avg_fragmentation_in_percent** and **avg_page_space_used_in_percent** returned.
13. Select the code under the **Step 10: Rebuild the table and its indexes** comment, and then click **Execute**.

14. Select the code under the **Step 11: Check the level of fragmentation via sys.dm_db_index_physical_stats** comment, and then click **Execute**.
15. Scroll to the right, and note the **avg_fragmentation_in_percent** and **avg_page_space_used_in_percent** returned.
16. On the **Query** menu, click **Include Actual Execution Plan**.
17. Select the code under the **Step 13: Run a query showing the execution plan** comment, and then click **Execute**.
18. Review the execution plan.
19. Select the code under the **Step 14: Create a covering index, point out the columns included** comment, and then click **Execute**.
20. Select the code under the **Step 15: Run the query showing the execution plan (CTR+M) – it now uses the new index** comment, and then click **Execute**.
21. Review the execution plan.
22. Select the code under the **Step 16: Drop the table** comment, and then click **Execute**.
23. Keep SQL Server Management Studio open for the next demonstration.

Lesson 4

Single Column and Composite Indexes

Contents:

Demonstration: Viewing Index Statistics

Demonstration Steps

Use Transact-SQL to View Statistics

1. In SQL Server Management Studio, in Solution Explorer, double-click **Demonstration 3.sql**.
2. Select the code under the comment **Step 1: Run the Transact-SQL up to the end step 1**, and then click **Execute**.
3. Walk through the important columns in the results.
4. On the **Query** menu, click **Include Actual Execution Plan**.
5. Select the code under the comment **Step 2: Check the freshness of the statistics, CTRL-M to switch on the Execution Plan**, and then click **Execute**.

Use SQL Server Management Studio to View Statistics

1. In Object Explorer, expand **Databases**, expand **AdventureWorks**, expand **Tables**, expand **HumanResources.Employee**, and then expand **Statistics**.
2. Right-click **AK_Employee_LoginID**, and then click **Properties**.
3. In the **Statistics Properties - AK_Employee_LoginID** dialog box, in the **Select a page** section, click **Details**.

Inspect the Statistics Settings for a Database

1. In Object Explorer, under **Databases**, right-click **AdventureWorks**, and then click **Properties**.
2. In the **Database Properties - AdventureWorks** dialog box, in the **Select a page** section, click **Options**.
3. In the **Other options** list, scroll to the top, under the **Automatic** heading, note that the **Auto Create Statistics** and **Auto Update Statistics** are set to **True**.
4. In the **Database Properties - AdventureWorks** dialog box, click **Cancel**.
5. Close SSMS without saving any changes.

Module Review and Takeaways

Best Practice

Choose columns with a high level of selectivity for indexes.

Rebuild highly fragmented indexes.

Use nonclustered indexes to improve the performance of particular queries, but balance their use with the overhead that they introduce.

Use actual execution plans to obtain missing index hints.

Module 6

Designing Optimized Index Strategies

Contents:

Lesson 1

Index Strategies

Contents:

Question and Answers

Question: Your sales table holds data for customers across the world. You want to improve the performance of a query run by the French office that calculates the total sales, only for customers based in France. Which index would be most appropriate, and why?

( ) A covering index.

( ) A clustered index.

( ) A filtered index.

( ) A nonclustered index on a heap.

( ) Add region as a nonkey column using the INCLUDE clause.

Answer:

( ) A covering index.

( ) A clustered index.

(√) A filtered index.

( ) A nonclustered index on a heap.

( ) Add region as a nonkey column using the INCLUDE clause.

Lesson 2

Managing Indexes

Contents:

Question and Answers

Question: When would you set FILL FACTOR for an index?

( ) When the index contains an IDENTITY column.

( ) When a table is used frequently for data inserts, and is heavily used in queries.

( ) When you want to force the query optimizer to join tables in a certain way.

( ) When you are running short of disk space.

( ) When you want to be sure that statistics are up to date.

Answer:

( ) When the index contains an IDENTITY column.

(√) When a table is used frequently for data inserts, and is heavily used in queries.

( ) When you want to force the query optimizer to join tables in a certain way.

( ) When you are running short of disk space.

( ) When you want to be sure that statistics are up to date.

Resources

Implementing Fill Factor and Padding

Best Practice: Amending the fill factor will cause the index to be rebuilt. Make the change at a time when the database is not being used heavily, or when it is not being used at all.

Managing Statistics

Best Practice: Do not update statistics more than necessary, because cached query execution plans will be marked for recompilation. Excessive updating of statistics may result in unnecessary CPU time being spent recompiling query execution plans.

Using Query Hints

Common Issues and Troubleshooting Tips

Common Issue	Troubleshooting Tip
Adding a query hint generates an error 8622. This means that the query optimizer cannot create a query plan.	Remove the query hint and rewrite the query without a query hint. Consider other ways to improve the performance of the query, such as an index, or creating an indexed view.

Best Practice: Use query hints judiciously. Unless there is a good reason to use a query hint, the query optimizer will find the best query plan.

Lesson 3

Execution Plans

Contents:

Question and Answers

Question: Why would you use the execution plan view?

Answer: The estimated and actual execution plans can help in a number of ways. For example:

- Understanding why a query is performing badly (or well).
- Understanding whether adding an index might help a query.

Resources

Common Execution Plan Elements

 **Best Practice:** The query execution plan is a guide to help you to understand how queries are being executed. Do not, however, try to manipulate how the query optimizer handles a query. When table statistics are accurate, and appropriate indexes are available, the query optimizer will almost always find the fastest way of executing the query.

Lesson 4

The Database Engine Tuning Advisor

Contents:

Question and Answers

Question: The Database Engine Turning Advisor is best used in which situations?

( ) When you need to add a large number of records to a table.

( ) When you want to ensure indexes are rebuilt regularly.

( ) When you want to identify missing indexes.

( ) When you need to create an XML file.

( ) Every time you run a Transact-SQL script.

Answer:

( ) When you need to add a large number of records to a table.

( ) When you want to ensure indexes are rebuilt regularly.

(√) When you want to identify missing indexes.

( ) When you need to create an XML file.

( ) Every time you run a Transact-SQL script.

Lesson 5

Query Store

Contents:

Question and Answers

Question: True or false? Due to limitations of using the cloud, Azure SQL Database does not contain all the Query Store functionality that is available in the SQL Server on-premises product.

( ) True

( ) False

Answer:

( ) True

(√) False

Demonstration: Using Query Store with Azure SQL Database

Demonstration Steps

1. Ensure the MSL-TMG1, 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running, and log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Open SSMS and connect to the server you created earlier—for example, **20762ce20160405.database.windows.net**, using SQL Server Authentication.
3. In the **Login** box, type **Student**, in the **Password** box, type **Pa$$w0rd**, and then click **Connect**.
4. In Object Explorer, expand **Databases** right-click **AdventureWorksLT**, and then click **Properties**.
5. In the **Database Properties - AdventureWorksLT** dialog box, click the **Query Store** page, and in the **General** section, ensure the **Operation Mode (Requested)** is set to **Read Write**. Point out the Query Store settings to students. When you are finished, click **OK**.
6. In Object Explorer, expand the **AdventureWorksLT** node to see that a folder called **Query Store** has been created.
7. Expand the **Query Store** node to show the four views.
8. On the **File** menu, point to **Open**, and then click **File**.
9. In the **Open File** dialog box, navigate to **D:\Demofiles\Mod06**, and open **QueryStore_Demo.sql**.
10. Select the code under the comment **Create a covering index on the TempProduct table**, and then click **Execute**. First, the query creates a covering index on the **TempProduct** table, and then uses three columns from this table—point out that the text columns have been included as nonkey columns.
11. Select the code under the comment **Clear the Query Store**, and then click **Execute**.
12. Select the code under the comment **Work load 1**, and then click **Execute**.
13. Repeat the previous step another five times, waiting a few seconds between each execution.
14. Select the code under the comment **Work load 2**, and then click **Execute**.
15. Repeat the previous step another three times, waiting a few seconds between each execution.
16. Select the code under the comment **Regress work load 1**, and then click **Execute**.
17. Select the code under the comment **Work load 1**, and then click **Execute**.
18. Repeat the previous step another five times, waiting a few seconds between each execution.
19. In Object Explorer, open the **Top Resource Consuming Queries** window, and see the difference between the two execution plans for Workload 1.

20. Demonstrate how you can change the metric using the drop-down box. Note the force plan button.
21. On the **File** menu, point to **Open**, and then click **File**.
22. In the **Open File** dialog box, navigate to **D:\Demofiles\Mod06**, select **QueryStore_Demo_CatalogViews.sql**, and then click **Open**.
23. Select the code under the comment **Create a covering index on the TempProduct table**, and then click **Execute**.
24. Select the code under the comment **Clear the Query Store**, and then click **Execute**.
25. Select the code under the comment **Work load 1**, and then click **Execute**.
26. Repeat the previous step.
27. Select the code under the comment **Work load 2**, and then click **Execute**.
28. Repeat the previous step another two times, waiting a few seconds between each execution.
29. Select the code under the comment **Regress work load 1**, and then click **Execute**.
30. Select the code under the comment **Work load 1**, and then click **Execute**.
31. Select the code under the comment **Examine sys.query_store_query_text and sys.query_context_settings**, and then click **Execute**.
32. Select the code under the comment **Examine sys.query_store_query**, and then click **Execute**.
33. Select the code under the comment **Examine sys.query_store_plan**, and then click **Execute**.
34. Select the code under the comment **Examine sys.query_store_runtime_stats_interval**, and then click **Execute**.
35. Select the code under the comment **Examine runtime statistics**, and then click **Execute**.
36. Close SSMS without saving any changes.

Module Review and Takeaways

Best Practice

Understand how queries are being executed using the estimated and actual execution plans, in addition to using Query Store. When you need to optimize a query, you will then be well prepared and have a good understanding of how SQL Server executes your Transact-SQL script.

Lab Review Questions and Answers

Lab: Optimizing Indexes

Question and Answers

Lab Review

Question: Which Query Store features will be most beneficial to your SQL Server environment?

Answer: Answers will vary and might include:

- Being able to force the query optimizer to use a particular query plan.
- Being able to identify when a query is using a different query plan.
- Identifying the most expensive queries in terms of CPU, memory, or duration.
- Having a graphical view of the cost of queries.
- Having query execution information stored on disk.

Question: In which situation might a heap be a better choice than a table with a clustered index?

Answer: Answers will vary and might include (i) never, (ii) writing to a log, or (iii) very small tables.

Effective use of heaps is rare and, for the most part, should only be used by experienced developers who fully understand the implications of creating a heap. Most tables perform better with a clustered index, and other nonclustered indexes designed according to the query workload.

Tables that are used intensively for inserts are candidates for a heap. These might include a log table, and tables that require very fast writes.

Question: Why is it sometimes quicker to retrieve records from a heap than a clustered index with a simple SELECT statement?

Answer: The clustered index does not cover the query, so the database engine must still retrieve columns from the table. In this situation, the heap is quicker to retrieve unordered records with no filtering or joins. In all other situations, the clustered index is faster.

Module 7

Columnstore Indexes

Contents:

Lesson 1

Introduction to Columnstore Indexes

Contents:

Question and Answers

Categorize each index property into the appropriate index type. Indicate your answer by writing the category number to the right of each property.

Items	
1	Perform the best when seeking for specific data.
2	A high degree of compression is possible, due to data being of the same category.
3	Can greatly improve the performance of database queries.
4	Implemented as a b-tree index structure.
5	Perform best when aggregating data.
6	Can be stored in memory optimized tables.

Category 1		Category 2		Category 3
Rowstore Index		Columnstore Index		Applies to both types of index

Answer:

Category 1		Category 2		Category 3
Rowstore Index		Columnstore Index		Applies to both types of index
Perform the best when seeking for specific data. Implemented as a b-tree index structure.		A high degree of compression is possible, due to data being of the same category. Perform best when aggregating data.		Can greatly improve the performance of database queries. Can be stored in memory optimized tables.

Demonstration: The Benefits of Using Columnstore Indexes

Demonstration Steps

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Run **D:\Demofiles\Mod07\Setup.cmd** as an administrator to revert any changes.
3. In the **User Account Control** dialog box, click **Yes**. When the script completes, press any key to close the window.
4. On the taskbar, click **Microsoft SQL Server Management Studio**.
5. In the **Connect to Server** window, in the **Server name** box, type **MIA-SQL**. Ensure **Windows Authentication** is selected in the **Authentication** box, and then click **Connect**.
6. On the **File** menu, point to **Open**, click **File**.
7. In the **Open File** dialog box, navigate to **D:\Demofiles\Mod07\Demo\Demo.sql** script file, and then click **Open**.
8. Select the code under the **Step 1** comment, and then click **Execute**.
9. Select the code under the **Step 2** comment, and then click **Execute**.
10. Select the code under the **Step 3** comment, and then click **Execute**.
11. Select the code under the **Step 1** comment, and then click **Execute**.
12. Close Microsoft SQL Server Management Studio without saving changes.

Lesson 2

Creating Columnstore Indexes

Contents:

Question and Answers

Question: How will you create your indexes in a database—with SSMS or Transact-SQL?

Answer: There are advantages to both approaches.

If you are using Transact-SQL, the code can be saved and reused against other tables or databases.

If you use SSMS, you don't have to remember the Transact-SQL syntax—the options available are visible. Once created, you can script the index to use Transact-SQL elsewhere.

Resources

Creating a Clustered Columnstore Index

Common Issues and Troubleshooting Tips

Common Issue	Troubleshooting Tip
Unable to create columnstore index on an Azure SQL Database.	Ensure you are using at least V12 of an Azure SQL Database. The pricing tier of the database also has to be a minimum of Premium.

Demonstration: Creating Columnstore Indexes Using SQL Server Management Studio

Demonstration Steps

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. On the taskbar, click **Microsoft SQL Server Management Studio**.
3. In the **Connect to Server** window, in the **Server name** box, type **MIA-SQL**. Ensure that **Windows Authentication** is selected in the **Authentication** box, and then click **Connect**.
4. In Object Explorer, expand **Databases**, expand **AdventureWorksDW**, expand **Tables**, and then expand **dbo.AdventureWorksDWBuildVersion**.
5. Right-click **Indexes**, point to **New Index**, and then click **Clustered Columnstore Index**.
6. In the **New Index** dialog box, click **OK** to create the index.
7. In Object Explorer, expand **Indexes** to show the new clustered index.
8. In Object Explorer, expand **dbo.FactResellersSales**.
9. Right-click **Indexes**, point to **New Index**, and then click **Non-Clustered Columnstore Index**.
10. In the **Columnstore columns** table, click **Add**.
11. Select the **SalesOrderNumber**, **UnitPrice**, and **ExtendedAmount** check boxes, and then click **OK**.
12. In the **New Index** dialog box, click **OK**.
13. In Object Explorer, expand **Indexes** to show the created nonclustered index.
14. Close Microsoft SQL Server Management Studio without saving.

Lesson 3

Working with Columnstore Indexes

Contents:

Question and Answers

Question: When would you consider converting a rowstore table, containing dimension data in a data warehouse, to a columnstore table?

( ) When mission critical analytical queries join one or more fact tables to the dimension table, and those fact tables are columnstore tables.

( ) When the data contained in the dimension table has a high degree of randomness and uniqueness.

( ) When the dimension table has very few rows.

( ) When the dimension table has many millions of rows, with columns containing small variations in data.

( ) It is never appropriate to convert a dimension table to a columnstore table.

Answer:

( ) When mission critical analytical queries join one or more fact tables to the dimension table, and those fact tables are columnstore tables.

( ) When the data contained in the dimension table has a high degree of randomness and uniqueness.

( ) When the dimension table has very few rows.

(√) When the dimension table has many millions of rows, with columns containing small variations in data.

( ) It is never appropriate to convert a dimension table to a columnstore table.

Module Review and Takeaways

Best Practice

Introduced in SQL Server 2012, columnstore indexes are used in large data warehouse solutions by many organizations. This module highlighted the benefits of using these indexes on large datasets; the improvements made to columnstore indexes in SQL Server 2016; and the considerations needed to use columnstore indexes effectively in your solutions.

Lab Review Questions and Answers

Lab: Using Columnstore Indexes

Question and Answers

Lab Review

Question: Why do you think the disk space savings were so large for the disk based clustered columnstore index?

Answer: There are advantages to both approaches.

If you are using Transact-SQL, the code can be saved and reused against other tables or databases.

If you use SSMS, you don't have to remember the Transact-SQL syntax—the options available are visible. Once created, you can script the index to use Transact-SQL elsewhere.

Module 8

Designing and Implementing Views

Contents:

Lesson 1

Introduction to Views

Contents:

Question and Answers

Question: What is the purpose of a view?

( ) To list all the DMVs available in SQL Server.

( ) To present relevant information to users and hide complexity.

( ) To remove the need for any security in a SQL Server database.

( ) To encrypt data in certain tables.

( ) To make tables faster to update.

Answer:

( ) To list all the DMVs available in SQL Server.

(√) To present relevant information to users and hide complexity.

( ) To remove the need for any security in a SQL Server database.

( ) To encrypt data in certain tables.

( ) To make tables faster to update.

Demonstration: Querying Catalog Views and DMVs

Demonstration Steps

Query System Views and Dynamic Management Views

1. Ensure that the MSL-TMG1, 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running, and then log on to 20762B-MIA-SQL as **AdventureWorks\Student** with the password **Pa$$w0rd**.
2. Start SQL Server Management Studio.
3. In the **Connect to Server** dialog box, in the **Server name** box, type the name of the Azure server you created before the course started; for example, ***<servername>*.database.windows.net**.
4. In the **Authentication** list, click **SQL Server Authentication**.
5. In the **Login** box, type **Student**, in the **Password** box, type **Pa$$w0rd**, and then click **Connect**.
6. On the **File** menu, point to **Open**, and then click **File**.
7. In the **Open File** dialog box, navigate to **D:\Demofiles\Mod08**, click **Mod_08_Demo_1A.sql**, and then click **Open**.
8. On the toolbar, in the **Available Databases** list, click **AdventureWorksLT**.
9. Select the code under the **Step 2 - Query sys.views** comment, and then click **Execute**.
10. Select the code under the **Step 3 - Query sys.tables** comment, and then click **Execute**.
11. Select the code under the **Step 4 - Query sys.objects** comment, and then click **Execute**.
12. Select the code under the **Step 5 - Query information_schema.tables** comment, and then click **Execute**.
13. In Object Explorer, expand **Databases**, expand **AdventureWorksLT**, expand **Views**, and then expand **System Views**. Note the system views and user-defined views.
14. Select the code under the **Step 6 - Query sys.dm_exec_connections** comment, and then click **Execute**.
15. Select the code under the **Step 7 - Query sys.dm_exec_sessions** comment, and then click **Execute**.

16. Select the code under the **Step Step 8 - Query sys.dm_exec_requests** comment, and then click **Execute**.
17. Select the code under the **Step 9 - Query sys.dm_exec_query_stats** comment, and then click **Execute**.
18. Select the code under the **Step 10 - Modify the query to add a TOP(20) and an ORDER BY** comment, and then click **Execute**.
19. Leave SSMS open for the next demonstration.

Lesson 2

Creating and Managing Views

Contents:

Question and Answers

Question: What does the WITH CHECK option do?

( ) Checks that the data in the view does not contain mistakes.

( ) Checks that the view definition is well formed.

( ) Checks that there is no corruption in the underlying tables.

( ) Checks that inserted data conforms to the view definition.

( ) Checks that inserted data complies with table constraints.

Answer:

( ) Checks that the data in the view does not contain mistakes.

( ) Checks that the view definition is well formed.

( ) Checks that there is no corruption in the underlying tables.

(√) Checks that inserted data conforms to the view definition.

( ) Checks that inserted data complies with table constraints.

Resources

Create a View

 **Best Practice:** It is good practice to prefix the name of your view with vw; for example, vwEmployeeList. Although database developers differ in their naming conventions, most would agree that it is beneficial to be able to see clearly which objects are tables, and which are views.

Drop a View

 **Best Practice:** Keep database documentation up to date, including the purpose for each view you create, and where they are used. This will help to identify views that are no longer required. These views can then be dropped from the database. Keeping old views that have no use makes database administration more complex, and adds unnecessary work, particularly at upgrade time.

Demonstration: Creating, Altering, and Dropping a View

Demonstration Steps

1. In SSMS, on the **File** menu, point to **Open**, and then click **File**.
2. In the **Open File** dialog box, navigate to **D:\Demofiles\Mod08**, click **Mod_08_Demo_2A.sql**, and then click **Open**.
3. On the toolbar, in the **Available Databases** list, click **AdventureWorksLT**.
4. Select the code under the **Step 2 - Create a new view** comment, and then click **Execute**.
5. Select the code under the **Step 3 - Query the view** comment, and then click **Execute**.
6. Select the code under the **Step 4 - Query the view and order the results** comment, and then click **Execute**.

7. Select the code under the **Step 5 - Query the view definition via OBJECT_DEFINITION** comment, and then click **Execute**.
8. Select the code under the **Step 6 - Alter the view to use WITH ENCRYPTION** comment, and then click **Execute**.
9. Select the code under the **Step 7 - Requery the view definition via OBJECT_DEFINITION** comment, and then click **Execute**.
10. Note that the query definition is no longer accessible because the view is encrypted.
11. Select the code under the **Step 8 - Drop the view** comment, and then click **Execute**.
12. Close SSMS without saving any changes.

Lesson 3

Performance Considerations for Views

Contents:

Question and Answers

Question: Can you think of queries in your SQL Server environment that use nested views? What advantages and disadvantages are there with using nested views?

Answer: Answers will vary.

Advantages of using nested views include:

- You can use a view that someone else has created and tested.
- A complex query can be split into understandable chunks.

Disadvantages of using nested views include:

- Performance can suffer when too many views are nested.
- Ownership chains can be broken when tables and views are owned by different people.
- Maintenance can be difficult as the database evolves, and DBAs or developers leave.

Resources

Indexed Views

 **Best Practice:** Indexed views are useful in decision support systems that are regularly queried, but updated infrequently. A data warehouse or data mart might use indexed views because much of the data is aggregated for reporting.

Module Review and Takeaways

Review Question(s)

Question: When you create a new view, what does SQL Server store in the database?

Answer: SQL Server stores the view definition, in addition to some metadata about the view. SQL Server does not store the records returned through the view unless it is an indexed view.

Lab Review Questions and Answers

Lab: Designing and Implementing Views

Question and Answers

Lab Review

Question: What are three requirements for a view to be updateable?

Answer: You could choose three from the following:

- The columns are from one table only.
- The view directly references the base table columns.
- The view does not include an aggregate function: AVG, COUNT, SUM, MIN, MAX, GROUPING, STDEV, STDEVP, VAR, or VARP.
- The view does not include DISTINCT, GROUP BY, HAVING clauses.
- The view does not include computed columns that are formed by using any other columns.
- TOP is not used in the view definition.

Question: What is a standard, nonindexed view?

Answer: A standard view returns data from one or more tables, or views. The view definition is stored, and the records are returned (or materialized) at run time.

Module 9

Designing and Implementing Stored Procedures

Contents:

Lesson 1

Introduction to Stored Procedures

Contents:

Question and Answers

Question: The system stored procedure prefix (sp_) and the extended stored procedure prefix (xp_) have become a little muddled over time. What does this say about the use of prefixes when naming objects like stored procedures?

> **Answer:** Prefixes that attempt to indicate the function of an object are not recommended. A well thought out and implemented naming convention is a much better way of naming stored procedures.

Demonstration: Working with System Stored Procedures and Extended Stored Procedures

Demonstration Steps

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running, and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Navigate to the folder **D:\Demofiles\Mod09** and execute **Setup.cmd** as an administrator.
3. In the **User Account Control** dialog box, click **Yes**.
4. Start SQL Server Management Studio and connect to the **MIA-SQL** instance using Windows authentication.
5. In SQL Server Management Studio, open the file **D:\Demofiles\Mod09\Module09.ssmssln**.
6. In Solution Explorer, in the **Queries** folder, double-click the **11 - Demonstration1A.sql** script file.
7. Highlight the text under the comment **Step 1 - Switch to the AdventureWorks database**, and click **Execute**.
8. Highlight the text under the comment **Step 2 - Execute the sp_configure system stored procedure**, and click **Execute**.
9. Highlight the text under the comment **Step 3 - Execute the xp_dirtree extended system stored procedure**, and click **Execute**.
10. Keep SQL Server Management Studio open for the next demo.

Lesson 2

Working with Stored Procedures

Contents:

Question and Answers

Question: Obfuscating the body of a stored procedure is best avoided, but when might you want to use this functionality?

( ) When transferring the stored procedure between servers.

( ) When emailing the stored procedure code to a colleague.

( ) When the stored procedure takes input parameters that should not be disclosed.

( ) When the stored procedure contains intellectual property that needs protecting.

Answer:

( ) When transferring the stored procedure between servers.

( ) When emailing the stored procedure code to a colleague.

( ) When the stored procedure takes input parameters that should not be disclosed.

(√) When the stored procedure contains intellectual property that needs protecting.

Demonstration: Stored Procedures

Demonstration Steps

1. In Solution Explorer, in the **Queries** folder, double-click the **21 - Demonstration2A.sql** script file.
2. Highlight the code under the comment **Step 1 - Switch to the AdventureWorks database**, and click **Execute**.
3. Highlight the code under the comment **Step 2 - Create the GetBlueProducts stored procedure**, and click **Execute**.
4. Highlight the code under the comment **Step 3 - Execute the GetBlueProducts stored procedure**, and click **Execute**.
5. Highlight the code under the comment **Step 4 - Create the GetBlueProductsAndModels stored procedure**, and click **Execute**.
6. Highlight the code under the comment **Step 5 - Execute the GetBlueProductsAndModels stored procedure which returns multiple rowsets**, and click **Execute**.
7. Highlight the code under the comment **Step 6 - Alter the procedure because the 2nd query does not show only blue products**, and click **Execute**.
8. Highlight the code under the comment **Step 7 - And re-execute the GetBlueProductsAndModels stored procedure**, and click **Execute**.
9. Highlight the code under the comment **Step 8 - Query sys.procedures to see the list of procedures**, and click **Execute**.
10. Keep SQL Server Management Studio open for the next demo.

Lesson 3

Implementing Parameterized Stored Procedures

Contents:

Question and Answers

Question: What is the main advantage of creating parameterized stored procedures over nonparameterized stored procedures?

Answer: Parameterized stored procedures enable code reuse. One parameterized stored procedure can potentially replace many nonparameterized stored procedures.

Lesson 4

Controlling Execution Context

Contents:

Question and Answers

Question: What permission is needed to EXECUTE AS another login or user?

( ) sysadmin

( ) IMPERSONATE

( ) TAKE OWNERSHIP

Answer:

( ) sysadmin

(√) IMPERSONATE

( ) TAKE OWNERSHIP

Demonstration: Viewing Execution Context

Demonstration Steps

1. In Solution Explorer, expand the **Queries** folder, and then double-click the **31 - Demonstration 3A.sql** script file.
2. Highlight the code under the comment **Step 1 - Open a new query window to the tempdb database**, and click **Execute**.
3. Highlight the code under the comment **Step 2 - Create a stored procedure that queries sys.login_token and sys.user_token**, and click **Execute**.
4. Highlight the code under the comment **Step 3 - Execute the stored procedure and review the rowsets returned**, and click **Execute**.
5. Highlight the code under the comment **Step 4 - Use the EXECUTE AS statement to change context**, and click **Execute**.
6. Highlight the code under the comment **Step 5 - Try to execute the procedure. Why does it not it work?** Click **Execute** and note the error message.
7. Highlight the code under the comment **Step 6 - Revert to the previous security context**, and click **Execute**.
8. Highlight the code under the comment **Step 7 - Grant permission to SecureUser to execute the procedure**, and click **Execute**.
9. Highlight the code under the comment **Step 8 - Now try again and note the output**, and click **Execute**.
10. Highlight the code under the comment **Step 9 - Alter the procedure to execute as owner**, and click **Execute**.
11. Highlight the code under the comment **Step 10 - Execute as SecureUser again and note the difference**, and click **Execute**.
12. Highlight the code under the comment **Step 11 - Drop the procedure**, and click **Execute**.
13. Close SQL Server Management Studio without saving any changes.

Module Review and Takeaways

Best Practice

Include the SET NOCOUNT ON statement in your stored procedures immediately after the AS keyword. This improves performance.

While it is not mandatory to enclose Transact-SQL statements within a BEGIN END block in a stored procedure, it is good practice and can help make stored procedures more readable.

Reference objects in stored procedures using a two- or three-part naming convention. This reduces the processing that the database engine needs to perform.

Avoid using SELECT * within a stored procedure even if you need all columns from a table. Specifying the column names explicitly reduces the chance of issues, should columns be added to a source table.

Module 10

Designing and Implementing User-Defined Functions

Contents:

Lesson 1

Overview of Functions

Contents:

Question and Answers

Question: Which of these is a ranking function?

( ) OPENROWSET

( ) ROWCOUNT_BIG

( ) GROUPING_ID

( ) ROW_NUMBER

( ) OPENXML

Answer:

( ) OPENROWSET

( ) ROWCOUNT_BIG

( ) GROUPING_ID

(√) ROW_NUMBER

( ) OPENXML

Lesson 2

Designing and Implementing Scalar Functions

Contents:

Question and Answers

Question: Which of these data types can be returned by a scalar function used in managed code?

( ) rowversion

( ) table

( ) cursor

( ) integer

Answer:

(√) rowversion

( ) table

( ) cursor

(√) integer

Question: True or false? A deterministic function is one that may return different results for the same set of input values each time it is called, even if the database remains in the same state.

( ) True

( ) False

Answer:

( ) True

(√) False

Demonstration: Working with Scalar Functions

Demonstration Steps

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running, and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Run **D:\Demofiles\Mod10\Setup.cmd** as an administrator.
3. In the **User Access Control** dialog box, click **Yes**.
4. Wait for **setup.cmd** to complete successfully.
5. On the taskbar, click **Microsoft SQL Server Management Studio**.
6. In the **Connect to Server** dialog box, in **Server** name, type **MIA-SQL** and then click **Connect**.
7. On the **File** menu, point to **Open**, and then click **Project/Solution**.
8. In the **Open Project** dialog box, navigate to **D:\Demofiles\Mod10\Demo10.ssmssqlproj**, and then click **Open**.
9. In Solution Explorer, expand the **Queries** folder, and then double-click **21 - Demonstration 2A.sql**.
10. Select the code under **Step A** to use the **tempdb** database, and then click **Execute**.
11. Select the code under **Step B** to create a function that calculates the end date of the previous month, and then click **Execute**.
12. Select the code under **Step C** to query the function, and then click **Execute**.
13. Select the code under **Step D** to establish if the function is deterministic, and then click **Execute**.

14. Select the code under **Step E** to drop the function, and then click **Execute**.
15. Create a function under **Step F** using the EOMONTH function (the code should resemble the following), and then click **Execute**.

```
CREATE FUNCTION dbo.EndOfPreviousMonth (@DateToTest date)
RETURNS date
AS BEGIN
  RETURN EOMONTH ( dateadd(month, -1, @DateToTest ));
END;
GO
```

16. Select the code under **Step G** to query the new function, and then click **Execute**.
17. Select the code under **Step H** to drop the function, and then click **Execute**.
18. Keep SQL Server 2016 Management Studio open with the **Demo10.ssmssqlpro** solution loaded for the next demo.

Lesson 3

Designing and Implementing Table-Valued Functions

Contents:

Question and Answers

Question: True or false? Both scalar functions and TVFs can return a table that might contain many rows of data, each with many columns—TVFs are a simpler way of returning data in tables.

( ) True

( ) False

Answer:

( ) True

(√) False

Question: You have learned that TVFs return tables. What are the two types of TVF and how do they differ?

Answer:

An inline TVF returns a table defined by one **SELECT** statement.

A multistatement TVF is defined by multiple **SELECT** statements.

Unlike scalar functions, TVFs return a table that can contain many rows of data, each with many columns.

Table-Valued Functions

There are two types of TVFs you can deploy.

Inline TVFs

These return an output table that is defined by a **RETURN** statement consisting of a single **SELECT** statement.

Multistatement TVFs

If the logic of the function is too complex to include in a single **SELECT** statement, you need to implement the function as a multistatement TVF. Multistatement TVFs construct a table within the body of the function, and then return the table. They also need to define the schema of the table to be returned.

You can use both types of TVF as the equivalent of parameterized views.

Demonstration: Implementing Table-Valued Functions

Demonstration Steps

1. In Solution Explorer, expand the **Queries** folder, and then double-click **31 - Demonstration 3A.sql**.
2. Select the code under **Step A** to use the AdventureWorks database, and then click **Execute**.
3. Select the code under **Step B** to create a table-valued function, and then click **Execute**.
4. Select the code under **Step C** to query the function, and then click **Execute**.
5. Select the code under **Step D** to use CROSS APPLY to call the function, and then click **Execute**.
6. Select the code under **Step E** to drop the function, and then click **Execute**.
7. Close SQL Server Management Studio without saving any changes.

Lesson 4

Considerations for Implementing Functions

Contents:

Question and Answers

Question: True or false? The underuse of scalar functions is a common cause of performance problems in SQL Server systems.

( ) True

( ) False

Answer:

( ) True

(√) False

Question: True or false? Before you can create a function that executes as another user, you need to have IMPERSONATE permission on that user, or be part of the **dbo** role.

( ) True

( ) False

Answer:

(√) True

( ) False

Question: Which of these is not considered when creating UDFs?

( ) Function Naming

( ) Function Code Size

( ) Function Log Size

( ) Function Exception Handling

( ) Function Performance

Answer:

( ) Function Naming

( ) Function Code Size

(√) Function Log Size

( ) Function Exception Handling

( ) Function Performance

Demonstration: Controlling the Execution Context

Demonstration Steps

Alter the Execution Context of a Function

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running, and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Run **D:\Demofiles\Mod10\Setup.cmd** as an administrator.
3. In the **User Access Control** dialog box, click **Yes**.
4. On the taskbar, click **Microsoft SQL Server 2016 Management Studio**.
5. In the **Connect to Server** dialog box, in **Server name** box, type **MIA-SQL**, and then click **Connect**.

6. On the **File** menu, point to **Open**, click **Project/Solution**.
7. In the **Open Project** dialog box, navigate to **D:\Demofiles\Mod10\Demo10.ssmssqlproj**, and then click **Open**.
8. In Solution Explorer, expand the **Queries** folder, and then double-click **41 - Demonstration 4A.sql**.
9. Select the code under **Step A** to use the master database, and then click **Execute**.
10. Select the code under **Step B** to create a test login, and then click **Execute**.
11. Select the code under **Step C** to use the **AdventureWorks** database and create a user, and then click **Execute**.
12. Select the code under **Step D** to create a function with default execution context, and then click **Execute**.
13. Select the code under **Step E** to try to add WITH EXECUTE AS, and then click **Execute**.
14. Select the code under **Step F** to recreate the function as a multistatement table-valued function, and then click **Execute**.
15. Select the code under **Step G** to select from the function, and then click **Execute**.
16. Select the code under **Step H** to drop the objects, and then click **Execute**.
17. Close SQL Server Management Studio without saving any changes.

Lesson 5

Alternatives to Functions

Contents:

Question and Answers

Question: What is wrong with this TVF code fragment?

SELECT * FROM (EXEC dbo.GetCriticalPathNodes);

( ) Incorrect syntax for a TVF.

( ) You cannot select from a stored procedure in a TVF.

( ) dbo.GetCriticalPathNodes does not exist.

( ) The statement needs more parentheses.

( ) None of these—this code is good.

Answer:

( ) Incorrect syntax for a TVF.

(√) You cannot select from a stored procedure in a TVF.

( ) dbo.GetCriticalPathNodes does not exist.

( ) The statement needs more parentheses.

( ) None of these—this code is good.

Question: True or false? You can update views, inline TFVs, and multistatement TVFs.

( ) True

( ) False

Answer:

( ) True

(√) False

Module Review and Takeaways

Best Practice

When working with functions, consider the following best practices:

Avoid calling multistatement TVFs for each row of a query. In many cases, you can dramatically improve performance by extracting the code from the query into the surrounding query.

Use the WITH EXECUTE AS clause to override the security context of code that needs to perform actions that the user who is executing the code does not have.

Review Question(s)

Question: When you are using the EXECUTE AS clause, what privileges should you grant to the login or user that is being impersonated?

> **Answer:** For the login or user that is being impersonated, specify a login or user that has the least privileges needed to perform the operations that are required in the session. For example, do not specify a login name with server-level permissions if only database-level permissions are required. Also, do not specify a database owner account unless those permissions are required.

Question: When you are using the EXECUTE AS clause, what privileges should you grant to the login or user who is creating the code?

> **Answer:** You should grant IMPERSONATE permission for the login or user who is creating the code.

Module 11

Responding to Data Manipulation Via Triggers

Contents:

Lesson 1

Designing DML Triggers

Contents:

Question and Answers

Question: Which types of statements fire DML triggers?

( ) CREATE, ALTER, DROP

( ) MERGE, ALTER, DELETE

( ) MERGE, CREATE, INSERT

( ) CREATE, UPDATE, DELETE

( ) DELET, INSERT, UPDATE

Answer:

( ) CREATE, ALTER, DROP

( ) MERGE, ALTER, DELETE

( ) MERGE, CREATE, INSERT

( ) CREATE, UPDATE, DELETE

(√) DELET, INSERT, UPDATE

Question: What reasons can you think of for deploying AFTER triggers?

Answer: Common reasons for implementing AFTER triggers are:

- Providing auditing of the changes that were made.
- Implementing complex rules involving the relationship between tables.
- Implementing default values or calculated values within rows.

Resources

Considerations for Triggers

 **Best Practice:** To prevent triggers from firing under escalated privileges, you should first understand what triggers you have in the database and server instance. You can use the **sys.triggers** and **sys.server_triggers** views to find out. Secondly, use the DISABLE_TRIGGER statement to disable triggers that use escalated privileges.

Lesson 2

Implementing DML Triggers

Contents:

Question and Answers

Question: Is the following statement true or false?

"When rows are deleted from a table by using a DELETE statement, any AFTER DELETE triggers are fired when the deletion is completed. DELETE ROWS is an administrative option that removes all rows from a table."

( ) True

( ) False

Answer:

( ) True

(√) False

Question:

Analyze this create trigger code and indicate the four errors. You can assume the table and columns have been created.

Outline some code you could use to test the trigger.

```
CREATE TRIGGER TR_SellingPrice_InsertUpdate
IN dbo.SellingPrice
AFTER INPUT, UPDATE AS BEGIN
  SET NOCOUNT OPEN;
  INSERT sp
  SET sp.ExtendedAmount = sp.SubTotal
                        + sp.TaxAmount
                        + sp.FreightAmount
  FROM dbo.SellingPrice AS sp
  INNER JOIN inserted AS i
  ON sp.SellingPriceID = i.SellingPriceId;
END;
GO
```

Answer:

The following code sample indicates the errors in the preceding code:

```
CREATE TRIGGER TR_SellingPrice_InsertUpdate
ON dbo.SellingPrice
AFTER INSERT, UPDATE AS BEGIN
  SET NOCOUNT ON;
  UPDATE sp
  SET sp.ExtendedAmount = sp.SubTotal
                        + sp.TaxAmount
                        + sp.FreightAmount
  FROM dbo.SellingPrice AS sp
  INNER JOIN inserted AS i
  ON sp.SellingPriceID = i.SellingPriceId;
END;
GO
```

The following example code will test the INSERT trigger. You should first INSERT some data, and then perform a SELECT to view the results:

```
INSERT INTO dbo.SellingPrice
  (SubTotal, TaxAmount, FreightAmount)
  VALUES (12.3, 1.23, 10), (5, 1, 2);
GO

SELECT * FROM dbo.SellingPrice;
GO
```

Demonstration: Working with AFTER INSERT Triggers

Demonstration Steps

Create an AFTER INSERT Trigger

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running, and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Run **D:\Demofiles\Mod11\Setup.cmd** as an administrator.
3. In the **User Account Control** dialog box, click **Yes**.
4. On the taskbar, click **Microsoft SQL Server Management Studio**.
5. In the **Connect to Server** dialog box, in the **Server name** box, type **MIA-SQL**, and then click **Connect**.
6. On the **File** menu, point to **Open**, click **Project/Solution**.
7. In the **Open Project** dialog box, navigate to **D:\Demofiles\Mod11\Demo11**, click **Demo11.ssmssqlproj**, and then click **Open**.
8. In Solution Explorer, expand the **Queries** folder, and then double-click the **21 - Demonstration 2A.sql** script file to open it.
9. Select the code under the **Step A** comment, and then click **Execute**.
10. Select the code under the **Step B** comment, and then click **Execute**.
11. Select the code under the **Step C** comment, and then click **Execute**.
12. Select the code under the **Step D** comment, and then click **Execute**.
13. Select the code under the **Step E** comment, and then click **Execute**. Note the error message.
14. Select the code under the **Step F** comment, and then click **Execute**.
15. Do not close SQL Server Management Studio.

Demonstration: Working with AFTER DELETE Triggers

Demonstration Steps

Create and Test AFTER DELETE Triggers

1. In Solution Explorer, in the **Queries** folder, double-click the **22 - Demonstration 2B.sql** script file to open it.
2. Select the code under the **Step A** comment, and then click **Execute**.
3. Select the code under the **Step B** comment, and then click **Execute**.

4. Select the code under the **Step C** comment, and then click **Execute**.
5. Select the code under the **Step D** comment, and then click **Execute**. Note the error message.
6. Select the code under the **Step E** comment, and then click **Execute**.
7. Do not close SQL Server Management Studio.

Demonstration: Working with AFTER UPDATE Triggers

Demonstration Steps

Create and Test AFTER UPDATE Triggers

1. In Solution Explorer, in the **Queries** folder, double-click **23 - Demonstration 2C.sql** to open it.
2. Select the code under the **Step A** comment, and then click **Execute**.
3. Select the code under the **Step B** comment, and then click **Execute**.
4. Select the code under the **Step C** comment, and then click **Execute**.
5. Select the code under the **Step D** comment, and then click **Execute**.
6. Select the code under the **Step E** comment, and then click **Execute**.
7. Select the code under the **Step F** comment, and then click **Execute**.
8. Select the code under the **Step G** comment, and then click **Execute**.
9. Select the code under the **Step H** comment, and then click **Execute**.
10. Select the code under the **Step I** comment, and then click **Execute**.
11. Select the code under the **Step J** comment, and then click **Execute**.
12. Select the code under the **Step K** comment, and then click **Execute**. Note that no triggers are returned.
13. Do not close SQL Server Management Studio.

Lesson 3

Advanced Trigger Concepts

Contents:

Question and Answers

Question: What are the four missing terms from this statement indicated by <XXX>?

<XXX>

These triggers are most commonly used to enable views that are based on multiple base tables to be updatable. You can define <XXX> triggers on views that have one or more base tables, where they can extend the types of updates that a view can support.

This trigger executes instead of the original triggering action. <XXX> triggers increase the variety of types of updates that you can perform against a view. Each table or view is limited to one <xxx> trigger for each triggering action (<XXX>).

You can specify an <XXX> trigger on both tables and views. You cannot create an <XXX> trigger on views that have the <XXX> clause defined. You can perform operations on the base tables within the trigger. This avoids the trigger being called again. For example, you could perform a set of checks before inserting data, and then perform the insert on the base table.

( ) How Nested Triggers Work; INSTEAD OF; DELETE; CHECK OPTION

( ) Updatable Views; AFTER INSERT; INSERT, UPDATE, or DELETE; NO ROWCOUNT

( ) Updatable Views; INSTEAD OF; DML or DDL; CHECK OPTION

( ) Updatable Views; RATHER THAN; UPDATE; CHECK OPTION

( ) Updatable Views; INSTEAD OF; INSERT, UPDATE, or DELETE; CHECK OPTION

Answer:

( ) How Nested Triggers Work; INSTEAD OF; DELETE; CHECK OPTION

( ) Updatable Views; AFTER INSERT; INSERT, UPDATE, or DELETE; NO ROWCOUNT

( ) Updatable Views; INSTEAD OF; DML or DDL; CHECK OPTION

( ) Updatable Views; RATHER THAN; UPDATE; CHECK OPTION

(√) Updatable Views; INSTEAD OF; INSERT, UPDATE, or DELETE; CHECK OPTION

Demonstration: Working with INSTEAD OF Triggers

Demonstration Steps

Create and Test an INSTEAD OF DELETE Trigger

1. In Solution Explorer, in the **Queries** folder, double-click **31 - Demonstration 3A.sql** in the **Solution Explorer** script file to open it.
2. Select the code under the **Step A** comment, and then click **Execute**.
3. Select the code under the **Step B** comment, and then click **Execute**.
4. Select the code under the **Step C** comment, and then click **Execute**.
5. Select the code under the **Step D** comment, and then click **Execute**.
6. Select the code under the **Step E** comment, and then click **Execute**.
7. Select the code under the **Step F** comment, and then click **Execute**.
8. Select the code under the **Step G** comment, and then click **Execute**.
9. Select the code under the **Step H** comment, and then click **Execute**.
10. Select the code under the **Step I** comment, and then click **Execute**.

11. Select the code under the **Step J** comment, and then click **Execute**.
12. Select the code under the **Step K** comment, and then click **Execute**.
13. Select the code under the **Step L** comment, and then click **Execute**. Note the error message.
14. Select the code under the **Step M** comment, and then click **Execute**. Note the error message.
15. Select the code under the **Step N** comment, and then click **Execute**.
16. Select the code under the **Step O** comment, and then click **Execute**.
17. Select the code under the **Step P** comment, and then click **Execute**.
18. Select the code under the **Step R** comment, and then click **Execute**.
19. Select the code under the **Step S** comment, and then click **Execute**.
20. Select the code under the **Step U** comment, and then click **Execute**.
21. Select the code under the **Step V** comment, and then click **Execute**.
22. Do not close SQL Server Management Studio.

Demonstration: Replacing Triggers with Computed Columns

Demonstration Steps

Replace a Trigger with a Computed Column

1. In Solution Explorer, in the **Queries** folder, double-click **32 - Demonstration 3B.sql** in the **Solution Explorer** script file to open it.
2. Select the code under the **Step A** comment, and then click **Execute**.
3. Select the code under the **Step B** comment, and then click **Execute**.
4. Select the code under the **Step C** comment, and then click **Execute**.
5. Select the code under the **Step D** comment, and then click **Execute**.
6. Select the code under the **Step E** comment, and then click **Execute**.
7. Select the code under the **Step F** comment, and then click **Execute**.
8. Select the code under the **Step G** comment, and then click **Execute**.
9. Close SQL Server Management Studio, without saving any changes.

Module Review and Takeaways

Best Practice

In many business scenarios, it makes sense to mark records as deleted with a status column and use a trigger or stored procedure to update an audit trail table. The changes can then be audited, the data is not lost, and the IT staff can perform purges or archival of the deleted records.

Avoid using triggers in situations where constraints could be used instead.

Review Question(s)

Question: How do constraints and triggers differ regarding timing of execution?

> **Answer:** Constraints fire before the data manipulation has occurred. AFTER triggers fire after the data manipulation has occurred. INSTEAD OF triggers fire instead of the data manipulation.

Module 12

Using In-Memory Tables

Contents:

Lesson 1

Memory-Optimized Tables

Contents:

Question and Answers

Question: You are creating an index for a date column in a memory-optimized table. What is likely to be the most suitable type of index? Explain your reasons.

> **Answer:** A nonclustered index would be most suitable.
>
> Query predicates that are applied to date columns often use inequality operators such as **<**,**>**, and **BETWEEN**. Queries of this type are not benefited by hash indexes. Therefore, unless you are sure that you will never need to use an inequality operator to query this column, you should use a nonclustered index.

Demonstration: Using Memory-Optimized Tables

Demonstration Steps

Create a Database with a Filegroup for Memory-Optimized Data

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running, and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the **D:\Demofiles\Mod12** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**.
4. Start SQL Server Management Studio, and then connect to the **MIA-SQL** database engine instance by using Windows authentication.
5. In Object Explorer, under **MIA-SQL**, right-click **Databases**, and then click **New Database**.
6. In the **New Database** dialog box, in the **Database name** box, type **MemDemo**.
7. On the **Filegroups** page, in the **MEMORY OPTIMIZED DATA** section, click **Add Filegroup**.
8. In the **Name** box, type **MemFG**. Note that the filegroups in this section are used to contain FILESTREAM files because memory-optimized tables are persisted as streams.
9. On the **General** page, click **Add** to add a database file. Then add a new file that has the following properties:
   - **Logical Name**: MemData
   - **File Type**: FILESTREAM Data
   - **Filegroup**: MemFG
10. In the **Script** drop-down list, click **Script Action to New Query Window**.
11. In the **New Database** dialog box, click **Cancel** to view the script file that has been generated.
12. Review the script, noting the syntax that has been used to create a filegroup for memory-optimized data. You can use similar syntax to add a filegroup to an existing database.

Use Memory-Optimized Tables

1. On the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the Open Project dialog box, navigate to **D:\Demofiles\Mod12\Demo12.ssmssln**, and then click **Open**.
3. In Solution Explorer, expand **Queries**, and then double-click **11 - Demonstration 1A.sql**.
4. Select the code under **Step 1 - Create a memory-optimized table**, and then click **Execute**.
5. Select the code under **Step 2 - Create a disk-based table**, and then click **Execute**.
6. Select the code under **Step 3 - Insert 500,000 rows into DiskTable**, and then click **Execute**.

This code uses a transaction to insert rows into the disk-based table.

7. When code execution is complete, look at the lower right of the query editor status bar, and note how long it has taken.
8. Select the code under **Step 4 - Verify DiskTable contents**, and then click **Execute**.
9. Confirm that the table now contains 500,000 rows.
10. Select the code under **Step 5 - Insert 500,000 rows into MemoryTable**, and then click **Execute**.

    This code uses a transaction to insert rows into the memory-optimized table.
11. When code execution is complete, look at the lower right of the query editor status bar and note how long it has taken. It should be significantly lower than the time that it takes to insert data into the disk-based table.
12. Select the code under **Step 6 - Verify MemoryTable contents**, and then click **Execute**.
13. Confirm that the table now contains 500,000 rows.
14. Select the code under **Step 7 - Delete rows from DiskTable**, and then click **Execute**.
15. Note how long it has taken for this code to execute.
16. Select the code under **Step 8 - Delete rows from MemoryTable**, and then click **Execute**.
17. Note how long it has taken for this code to execute. It should be significantly lower than the time that it takes to delete rows from the disk-based table.
18. Select the code under **Step 9 - View memory-optimized table stats**, and then click **Execute**.
19. Close SQL Server Management Studio, without saving any changes.

Lesson 2

Natively Compiled Stored Procedures

Contents:

Question and Answers

Question: You are executing a native stored procedure that inserts a row into a memory-optimized table for customer data, and then inserts a row into a memory-optimized table for sales data. The row is inserted into the customer table successfully, but the statement to insert the row into the sales table fails, causing the procedure to return an error.

True or false? When you check the tables by running a SELECT query, the row that was successfully inserted will show in the customer table.

( ) True

( ) False

Answer:

( ) True

(√) False

Demonstration: Creating a Natively Compiled Stored Procedure

Demonstration Steps

Create a Natively Compiled Stored Procedure

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running, and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Ensure that you have run the previous demonstration.
3. Start SQL Server Management Studio, and then connect to the **MIA-SQL** database engine instance by using Windows authentication.
4. On the **File** menu, click **Open**, click **Project/Solution**.
5. In the **Open Project** dialog box, navigate to **D:\Demofiles\Mod12\Demo12.ssmssln**, and then click **Open**.
6. In Solution Explorer, expand **Queries**, and then double-click **21 - Demonstration 2A.sql**.
7. Select the code under **Step 1 - Use the MemDemo database**, and then click **Execute**.
8. Select the code under **Step 2 - Create a native stored proc**, and then click **Execute**.
9. Select the code under **Step 3 - Use the native stored proc**, and then click **Execute**.
10. Note how long it has taken for the stored procedure to execute. This should be significantly lower than the time that it takes to insert data into the memory-optimized table by using a Transact-SQL INSERT statement.
11. Select the code under **Step 4 - Verify MemoryTable contents**, and then click **Execute**.
12. Confirm that the table now contains 500,000 rows.
13. Close SQL Server Management Studio without saving any changes.

Module Review and Takeaways

Question: Which of the following statements is true?

( ) Interpreted stored procedures cannot be applied to memory-optimized tables.

( ) Native stored procedures cannot be applied to disk-based tables.

( ) Native stored procedures can be applied to memory-optimized tables and to disk-based tables.

( ) Interpreted stored procedures can contain atomic blocks.

( ) Native stored procedures compile the code the first time the stored procedure is executed.

Answer:

( ) Interpreted stored procedures cannot be applied to memory-optimized tables.

(√) Native stored procedures cannot be applied to disk-based tables.

( ) Native stored procedures can be applied to memory-optimized tables and to disk-based tables.

( ) Interpreted stored procedures can contain atomic blocks.

( ) Native stored procedures compile the code the first time the stored procedure is executed.

Module 13

Implementing Managed Code in SQL Server

Contents:

Lesson 1

Introduction to CLR Integration in SQL Server

Contents:

Question and Answers

Question: Why might you include managed code in your SQL Server dataset?

( ) To create a new data type that is used widely in your database.

( ) To replace some Transact-SQL code that is running slowly.

( ) To create a trigger that alters code in another application.

( ) To back up SQL Server at a certain time on Monday morning.

Answer:

(√) To create a new data type that is used widely in your database.

( ) To replace some Transact-SQL code that is running slowly.

( ) To create a trigger that alters code in another application.

( ) To back up SQL Server at a certain time on Monday morning.

Lesson 2

Implementing and Publishing CLR Assemblies

Contents:

Question and Answers

Question: Do you use managed code in your SQL Server databases? Who maintains the code when it needs amending?

Answer: Answers will vary.

Some DBAs may have come from a programming background and be comfortable with integrating managed code into a database. For DBAs without programming knowledge: do they have an in-house development department? Or do they outsource development work?

Put the following steps in order by numbering each to indicate the correct order:

	Steps
	Check which version of the .NET Framework is installed on the machine hosting your SQL Server.
	Open Visual Studio and check that SSDT is installed.
	Create a new project.
	Add a new item to the project.
	Amend the template with your new code.
	Build the solution.
	Publish the solution.
	Open SSMS and check the new function appears.
	Create a Transact-SQL query using your new managed code function.

Answer:

	Steps
1	Check which version of the .NET Framework is installed on the machine hosting your SQL Server.
2	Open Visual Studio and check that SSDT is installed.
3	Create a new project.
4	Add a new item to the project.
5	Amend the template with your new code.
6	Build the solution.
7	Publish the solution.
8	Open SSMS and check the new function appears.
9	Create a Transact-SQL query using your new managed code function.

Demonstration: Creating a User-Defined Function

Demonstration Steps

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are both running, and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Navigate to **D:\Demofiles\Mod13**, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.
4. On the Start screen, type **SQL Server Data Tools 2015**, and then click **SQL Server Data Tools 2015**.
5. On the **File** menu, point to **New**, and then click **Project**.
6. In the **New Project** dialog box, expand **Templates**, and then click **SQL Server**.
7. In the middle pane, click **SQL Server Database Project**, in the top pane, in the **.NET Framework** list, click **.NET Framework 4.6**.
8. In the **Name** box, type **ClrDemo**, in the **Location** box, type **D:\Demofiles\Mod13**, and then click **OK**.
9. In Solution Explorer, right-click **ClrDemo**, point to **Add**, and then click **New Item**.
10. In the **Add New Item** dialog box, in the **Installed** list, under **SQL Server**, click **SQL CLR C#**, in the middle pane, click **SQL CLR C# User Defined Function**, in the **Name** box, type **HelloWorld.cs**, and then click **Add**.
11. Locate the **return** statement, which is immediately below the comment **// Put your code here**.
12. Amend the function to read:

```
return new SqlString("Hello World!");
```

13. On the **File** menu, click **Save All**.
14. In Solution Explorer, right-click **ClrDemo**, and then click **Properties**.
15. On the **Project Settings** page, in the **Target platform** list, note that **SQL Server 2016** is selected.
16. On the **SQLCLR** page, in the **Permission level** list, note that **SAFE** is selected.
17. On the **File** menu, click **Save All**.
18. On the **Build** menu, click **Build Solution**.
19. In Solution Explorer, right-click **ClrDemo**, and then click **Publish**.
20. In the **Publish Database** dialog box, click **Edit**.
21. In the **Connect** dialog box, on the **Browse** tab, expand **Local**, and then click **MIA-SQL**.
22. In the **Database Name** list, click **AdventureWorks2014**, and then click **Test Connection**.
23. In the **Connect** message box, click **OK**.
24. In the **Connect** dialog box, click **OK**.
25. In the **Publish Database** dialog box, click **Publish**.
26. After a few minutes, a message will be displayed to say it has published successfully.
27. On the taskbar, click **Microsoft SQL Server Management Studio**.
28. In the **Connect to Server** dialog box, in the **Server name** box, type **MIA-SQL**, and then click **Connect**.

29. In Object Explorer, expand **Databases**, expand **AdventureWorks2014**, expand **Programmability**, expand **Functions**, and then expand **Scalar-valued Functions**. Note that **dbo.HelloWorld** appears in the list.
30. In Object Explorer, right-click **AdventureWorks2014**, and click **New Query**.
31. In the new query window, type the following code, and then click **Execute**.

```
SELECT dbo.HelloWorld();
```

32. Close SSMS without saving any changes.
33. Close Visual Studio.

Module Review and Takeaways

Question: This module has reviewed the pros and cons of using managed code within a SQL Server database. You have integrated some prewritten C# functions into a database and tested them in some queries.

How might you use managed code in your own SQL Server environment? How do you assess the pros and cons for your specific situation?

Answer: Answers will vary.

Possible pros:

Being able to implement a new aggregate or user-defined data type.

Possible cons:

Finding developers to write and maintain the code.

Debugging problems.

Integrating managed code into the database when SQL Server is upgraded.

Lab Review Questions and Answers

Lab: Implementing Managed Code in SQL Server

Question and Answers

Lab Review

Question: After publishing managed code to a database, what do you think the issues are with using it?

Answer: Answers will vary.

Possible issues are:

- Ensuring database developers know the function is available.
- Amending the code as business requirements change.
- Debugging the code in case of problems.

Module 14

Storing and Querying XML Data in SQL Server

Contents:

Lesson 1

Introduction to XML and XML Schemas

Contents:

Question and Answers

Question: Do you currently work with applications that use XML? If your application does use XML, have you considered storing and processing that XML data on SQL Server?

> **Answer:** Applications that make use of REST, SOAP and Service Oriented Architecture are commonplace in the IT industry. SQL Server supports these technical approaches by storing and enabling the manipulation of XML data.
>
> Later in this module, you will see situations where it's considered best practice to use XML, and also some situations where it may not.

Resources

XML Namespaces

 **Best Practice:** Industry best practice is to include namespace attributes in the top level node to reduce unnecessary duplication throughout the document.

Demonstration: Introduction to XML and XML Schemas

Demonstration Steps

Structure XML and Structure XML Schemas

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running, and then log on to 20762B-MIA-SQL as **AdventureWorks\Student** with the password **Pa$$w0rd**.
2. Navigate to **D:\Demofiles\Mod14**, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and then wait until the script finishes.
4. On the taskbar, click **Microsoft SQL Server Management Studio**.
5. In the **Connect to Server** dialog box, in **Server name** box, type **MIA-SQL** and then click **Connect**.
6. On the **File** menu, point to **Open**, and then click **Project/Solution**.
7. In the **Open Project** dialog box, navigate to **D:\Demofiles\Mod14**, click **Demo14.ssmssln**, and then click **Open**.
8. In Solution Explorer, under **Miscellaneous**, and then double-click **XML_Sample_1.xml**.
9. Note the warning line under the second **<Production.Product>** tag. Position the cursor on the tag to display the warning.
10. Note that the XML editor in SSMS understands XML and formats it appropriately.
11. Note that the **Color** attribute is missing from elements where the data is NULL.
12. In Solution Explorer, under **Miscellaneous**, double-click **XML_Sample_2.xml**.
13. Note that this document contains a root element, so there is no longer a warning on the second **<Production.Product>** tag.
14. In Solution Explorer, under **Miscellaneous**, double-click **XML_Sample_3.xml**.
15. Note that this file contains an XML schema followed by the XML data.
16. Leave SSMS open for the next demonstration.

Lesson 2

Storing XML Data and Schemas in SQL Server

Contents:

Question and Answers

Question: Which of the following is not true about the xml data type?

( ) It can store up to 2 GB of data.

( ) It can be stored with, or without, an associated XSD.

( ) It can only store well-formed XML documents.

( ) The CONTENT keyword is the default value for typed XML.

( ) It can be used for variables, parameters, and columns.

Answer:

( ) It can store up to 2 GB of data.

( ) It can be stored with, or without, an associated XSD.

(√) It can only store well-formed XML documents.

( ) The CONTENT keyword is the default value for typed XML.

( ) It can be used for variables, parameters, and columns.

Demonstration: Working with Typed vs. Untyped XML

Demonstration Steps

Work with Typed and Untyped XML

1. Ensure that you have completed the previous demonstration.
2. In SSMS, in Solution Explorer, expand **Queries**, and then double-click **Demonstration 2.sql**.
3. Select the code under **Step 1**, and then click **Execute**.
4. Select the code under **Step 2**, and then click **Execute** to create an XML schema collection.
5. Select the code under **Step 3**, and then click **Execute** to create a table with a column that uses the collection.
6. Select the code under **Step 4**, and then click **Execute** to try to insert malformed XML. Note that the INSERT statement fails.
7. Select the code under **Step 5**, and then click **Execute** to try to insert well-formed XML that does not conform to the schema. Note that this INSERT statement fails too.
8. Select the code under **Step 6**, and then click **Execute** to insert a single row fragment.
9. Select the code under **Step 7**, and then click **Execute** to insert a multirow fragment.
10. Select the code under **Step 8**, and then click **Execute** to view the added XML data in the table.
11. Leave SSMS open for the next demonstration.

Lesson 3

Implementing the XML Data Type

Contents:

Question and Answers

Categorize each statement against the appropriate index. Indicate your answer by writing the category number to the right of each statement.

Items	
1	Requires a clustered primary index to already exist on the table.
2	Requires a primary XML index to already exist.
3	Only one type.
4	Three different types.
5	Will provide the most performance improvement.
6	Could help improve XQueries on Values.
7	Avoids the need to parse the whole XML.
8	Could help improve XQueries on Properties.
9	Could help improve XQueries checking the existence of a node.

Category 1		Category 2
PRIMARY Index		SECONDARY Index

Answer:

Category 1		Category 2
PRIMARY Index		SECONDARY Index
Requires a clustered primary index to already exist on the table. Only one type. Will provide the most performance improvement. Avoids the need to parse the whole XML.		Requires a primary XML index to already exist. Three different types. Could help improve XQueries on Values. Could help improve XQueries on Properties. Could help improve XQueries checking the existence of a node.

Demonstration: Implementing XML Indexes

Demonstration Steps

1. Ensure that you have completed the previous demonstration.
2. In SSMS, in Solution Explorer, double-click **Demonstration 3.sql**.
3. Select the code under **Step 1**, and then click **Execute**.
4. Select the code under **Step 2**, and then click **Execute** to create a primary XML index.
5. Select the code under **Step 3**, and then click **Execute** to create a secondary VALUE index.
6. Select the code under **Step 4**, and then click **Execute** to query the **sys.xml_indexes** system view.
7. Select the code under **Step 5**, and then click **Execute** to drop and recreate the table without a primary key.
8. Select the code under **Step 6**, and then click **Execute** to try to add the primary **xml index** again. Note that this will fail.
9. Leave SSMS open for the next demonstration.

Lesson 4

Using the Transact-SQL FOR XML Statement

Contents:

Question and Answers

Question: Which of the following is not true about RAW mode queries?

( ) By default, they produce XML fragments.

( ) They produce nested XML based on SELECT heuristics.

( ) A root node can be added with the ROOT option.

( ) XSINIL will result in NULL columns being added to the results.

Answer:

( ) By default, they produce XML fragments.

(√) They produce nested XML based on SELECT heuristics.

( ) A root node can be added with the ROOT option.

( ) XSINIL will result in NULL columns being added to the results.

Demonstration: FOR XML Queries

Demonstration Steps

1. Ensure that you have completed the previous demonstration.
2. In SSMS, in Solution Explorer, double-click **Demonstration 4.sql**.
3. Select the code under **Step 1**, and then click **Execute**.
4. Select the code under **Step 2**, click **Execute** to execute RAW mode queries, and then review the results.
5. Select the code under **Step 3**, click **Execute** to execute an AUTO mode query, and then review the results.
6. Select the code under **Step 4**, click **Execute** to execute an EXPLICIT mode query, and then review the results.
7. Select the code under **Step 5**, click **Execute** to execute PATH mode queries, and then review the results.
8. Select the code under **Step 6**, click **Execute** to execute a query using TYPE, and then review the results.
9. Select the code under **Step 7**, click **Execute** to run the same query without using the TYPE keyword, and then compare the results with those obtained in the previous step.
10. Leave SSMS open for the next demonstration.

Lesson 5

Getting Started with XQuery

Contents:

Question and Answers

Categorize each item against its correct XQuery method. Indicate your answer by writing the method number to the right of each item.

Items	
1	Returns a scalar SQL Type.
2	Has insert, delete and replace value of as options.
3	Returns 1, 0 or NULL.
4	Need to specify [1] to select a single XML element.
5	Normally used in an UPDATE statement.
6	Should be used in preference to a value() method.
7	Can be used in the SELECT or WHERE clause.
8	Normally used in the WHERE clause.

Category 1		Category 2		Category 3
value() Method		modify() Method		exist() Method

Answer:

Category 1		Category 2		Category 3
value() Method		modify() Method		exist() Method
Returns a scalar SQL Type. Need to specify [1] to select a single XML element. Can be used in the SELECT or WHERE clause.		Has insert, delete and replace value of as options. Normally used in an UPDATE statement.		Returns 1, 0 or NULL. Should be used in preference to a value() method. Normally used in the WHERE clause.

Demonstration: XQuery Methods in a DDL Trigger

Demonstration Steps

1. Ensure that you have completed the previous demonstration.
2. In SSMS, in Solution Explorer, double-click **Demonstration 5.sql**.
3. Select the code under **Step 1**, and then click **Execute**.
4. Select the code under **Step 2**, and then click **Execute** to create the trigger.
5. Select the code under **Step 3**, and then click **Execute** to test the trigger.
6. Select the code under **Step 4**, and then click **Execute** to drop the trigger.
7. Select the code under **Step 5**, and then click **Execute** to create a trigger to enforce naming conventions.
8. Select the code under **Step 6**, and then click **Execute** to test the trigger. Note that the code to create a stored procedure named **sp_GetVersion** fails, due to the trigger.
9. Select the code under **Step 7**, and then click **Execute** to create a trigger to enforce tables to have primary keys.
10. Select the code under **Step 8**, and then click **Execute** to test the trigger. Note that the CREATE TABLE statement will fail because there is no primary key defined.
11. Select the code under **Step 9**, and then click **Execute** to clean up the database.
12. Leave SSMS open for the next demonstration.

Lesson 6

Shredding XML

Contents:

Question and Answers

Question: Which of the following statements about the OPENXML statement is false?

( ) It can only be used to shred XML.

( ) You must call the sp_xml_preparedocument to create an in-memory node tree before using it.

( ) You should call the sp_xml_removedocument after using it.

( ) In most situations, it will perform worse than the xml nodes() method.

Answer:

(√) It can only be used to shred XML.

( ) You must call the sp_xml_preparedocument to create an in-memory node tree before using it.

( ) You should call the sp_xml_removedocument after using it.

( ) In most situations, it will perform worse than the xml nodes() method.

Demonstration: Shredding XML

Demonstration Steps

1. Ensure that you have completed the previous demonstration.
2. In SSMS, in Solution Explorer, double-click **Demonstration 6.sql**.
3. Select the code under **Step 1**, and then click **Execute**.
4. Select the code under **Step 2**, and then click **Execute** to select the contents of the **dbo.DatabaseLog** table.
5. In the results pane, in the **XmlEvent** column, click the first entry to view the format of the XML. Note that this is the EVENTDATA structure returned by the DDL and LOGON triggers.
6. Switch back to the **Demonstration 6.sql** pane, select the code under **Step 4**, and then click **Execute**. Compare the first row in the results with the data shown in the XmlEvent1.xml pane.
7. Select the code under **Step 5**, and then click **Execute**.
8. Select the code under **Step 6**, and then click **Execute** to show the same results obtained by using OPENXML.
9. Close SSMS without saving any changes.

Module Review and Takeaways

Best Practice

This module has considered a variety of different aspects of using XML data within SQL Server, including storing XML data, querying XML data, performance and XML indexes, and shredding XML data.

Review Question(s)

Question: Which XML query mode did you use for implementing the WebStock.GetAvailableModelsAsXML stored procedure?

> **Answer:** The XML query mode that was used for implementing the WebStock.GetAvailableModelsAsXML stored procedure was RAW mode.
>
> AUTO and PATH would also have been valid choices.

Module 15

Storing and Querying Spatial Data in SQL Server

Contents:

Lesson 1

Introduction to Spatial Data

Contents:

Question and Answers

Question: Which existing SQL Server data type could you use to store, but not directly process, raster data?

( ) varchar

( ) varbinary

( ) int

( ) string

Answer:

( ) varchar

(√) varbinary

( ) int

( ) string

Demonstration: Spatial Reference Systems

Demonstration Steps

1. Ensure that the 20762B-MIA-DC and 20762B-MIA-SQL virtual machines are running, and then log on to 20762B-MIA-SQL as **AdventureWorks\Student** with the password **Pa$$w0rd**.
2. Run **D:\Demofiles\Mod15\Setup.cmd** as an administrator.
3. In the **User Account Control** dialog box, click **Yes**. When the script completes, press any key.
4. Start SQL Server Manager Studio, and connect to the **MIA-SQL** instance using Windows authentication.
5. On the **File** menu, point to **Open**, and then click **Project/Solution**.
6. In the **Open Project** dialog box, navigate to **D:\Demofiles\Mod15**, click **20762_15.ssmssln**, and then click **Open**..
7. In Solution Explorer, expand **Queries**, and then double-click the **11 - Demonstration 1A.sql** script.
8. Highlight the Transact-SQL under the comment **Step 1 - Switch to the tempdb database**, and click **Execute**.
9. Highlight the Transact-SQL under the comment **Step 2 - Query the sys.spatial_reference_systems system view**, and click **Execute**.
10. Highlight the Transact-SQL under the comment **Step 3 - Drill into the value for srid 4326**, and click **Execute**.
11. Highlight the Transact-SQL under the comment **Step 4 - Query the available measurement systems**, and click **Execute**.
12. On the **File** menu, click **Close**.

Lesson 2

Working with SQL Server Spatial Data Types

Contents:

Question and Answers

Question: You have used a web service to calculate the coordinates of an address. What is this process commonly called, and what services are available?

> **Answer:** The process of calculating the coordinates of an address is geocoding. Bing provides a geocoding service.

Demonstration: Spatial Data Types

Demonstration Steps

1. In SQL Server Manager, in Solution Explorer, under **Queries**, double-click the **21 - Demonstration 2A.sql** script file.
2. Highlight the Transact-SQL under the comment **Step 1 - Switch to the AdventureWorks database**, and click **Execute**.
3. Highlight the Transact-SQL under the comment **Step 2 - Draw a shape using geometry**, and click **Execute**.
4. Click the **Spatial results** tab.
5. Highlight the Transact-SQL under the comment **Step 3 - Draw two shapes**, and click **Execute**.
6. Click the **Spatial results** tab.
7. Highlight the Transact-SQL under the comment **Step 4 - Show what happens if you perform a UNION rather than a UNION ALL. This will fail, as spatial types are not comparable**, and click **Execute**. Note the error message.
8. Highlight the Transact-SQL under the comment **Step 5 - Join the two shapes together**, and click **Execute**.
9. Click the **Spatial results** tab.
10. Highlight the Transact-SQL under the comment **Step 6 - How far is it from New York to Los Angeles in meters?** and click **Execute**.
11. Highlight the Transact-SQL under the comment **Step 7 - Draw the Pentagon**, and click **Execute**.
12. Click the **Spatial results** tab.
13. Highlight the Transact-SQL under the comment **Step 8 - Call the ToString method to observe the use of the Z and M values that are stored but not processed**, and click **Execute**.
14. Highlight the Transact-SQL under the comment **Step 9 - Use GML for input**, and click **Execute**.
15. Click the **Spatial results** tab.
16. Highlight the Transact-SQL under the comment **Step 10 - Output GML from a location (start and end points of the Panama Canal only – not the full shape)**, and click **Execute**.
17. Click the **Spatial results** tab.
18. Highlight the Transact-SQL under the comment **Step 11 - Show how collections can include different types of objects**, and click **Execute**.
19. Click the **Spatial results** tab.
20. On the **File** menu, click **Close**.

Lesson 3

Using Spatial Data in Applications

Contents:

Question and Answers

Question: Where might spatial data prove useful in your organization?

> **Answer:** There is no correct answer to this question. Answers will vary, depending on the experience of students and the industries in which they work.

Demonstration: Spatial Data in Applications

Demonstration Steps

1. In SQL Server Manager, in Solution Explorer, under **Queries**, double-click the **31 - Demonstration 3A.sql** script file.
2. Highlight the Transact-SQL under the comment **Step 1 - Open a new query window to the AdventureWorks database**, and click **Execute**.
3. Highlight the Transact-SQL under the comment **Step 2 - Which salesperson is closest to New York?** and click **Execute**.
4. Highlight the Transact-SQL under the comment **Step 3 - Which two salespeople live the closest together?** and click **Execute**.
5. Note: this will take a few minutes to run.
6. Close SQL Server Management Studio without saving any changes.

Module Review and Takeaways

Best Practice

This module described spatial data and how this data can be implemented within SQL Server.

Common Issues and Troubleshooting Tips

Common Issue	Troubleshooting Tip
Choose the correct data type for your needs.	Remember that the geometry data type holds only X and Y co-ordinates. The geography data type holds coordinates as latitude and longitude.
Consider the performance of your application.	Holding spatial data can impact performance, depending on how much data is held. Ensure you test to understand how your application will perform under load.
Consider security options for tables that hold geography and geometry data types.	Always Encrypted, the new encryption option introduced in SQL Server 2016, does not support certain data types, including geography and geometry.

Module 16

Storing and Querying BLOBs and Text Documents in SQL Server

Contents:

Lesson 1

Considerations for BLOB Data

Contents:

Question and Answers

Question: You want to store résumés, which are Word documents, in the Human Resources database. You want to store each document in a column of the Employees table. A typical résumé is around 50 KB in size and you want to maximize performance, and ensure referential and transactional integrity. Which of the following approaches should you use?

( ) Store documents as BLOBs within the database.

( ) Store documents on a separate file server. In the database, create a varchar() column that links an employee record to the correct résumé.

( ) Use the FILESTREAM feature.

( ) Use a FileTable.

Answer:

( ) Store documents as BLOBs within the database.

( ) Store documents on a separate file server. In the database, create a varchar() column that links an employee record to the correct résumé.

(√) Use the FILESTREAM feature.

( ) Use a FileTable.

Demonstration: BLOBs in the Adventure Works Database

Demonstration Steps

Investigate BLOB Storage in the Adventure Works Database

1. On the taskbar, click **Microsoft SQL Server Management Studio**.
2. In the **Connect to Server** dialog box, in **Server name**, type **MIA-SQL**, and then click **Connect**.
3. On the **File** menu, point to **Open**, and then click **Project/Solution**.
4. In the **Open Project** dialog box, navigate to **D:\Demofiles\Mod16\Demo**, click **demo.ssmssln**, and then click **Open**.
5. In Solution Explorer, double-click the **1 - AdventureWorks BLOBs.sql** script file.
6. Select the code under the **Step 1** comment, and then click **Execute**.
7. Select the code under the **Step 2** comment, and then click **Execute**.
8. Select the code under the **Step 3** comment, and then click **Execute**.
9. Select the code under the **Step 4** comment, and then click **Execute**. Note that no results are returned.
10. Close Microsoft SQL Server Management Studio, without saving any changes.

Lesson 2

Working with FILESTREAM

Contents:

Question and Answers

You have no FILESTREAM or FileTable prerequisites configured. You want to create a FileTable for BLOB storage. Put the following steps in order by numbering each to indicate the correct order.

	Steps
	Enable FILESTREAM at the instance level.
	Create a FILESTREAM filegroup at the database level.
	Configure nontransactional access at the database level.
	Configure a directory for FileTables at the database level.
	Start creating FileTables.

Answer:

	Steps
1	Enable FILESTREAM at the instance level.
2	Create a FILESTREAM filegroup at the database level.
3	Configure nontransactional access at the database level.
4	Configure a directory for FileTables at the database level.
5	Start creating FileTables.

Demonstration: Configuring FILESTREAM and FileTables

Demonstration Steps

Enable FILESTREAM at the Instance Level

1. On the Start screen, type **SQL Server 2016 Configuration Manager**, and then click **SQL Server 2016 Configuration Manager**.
2. In the **User Account Control** dialog box, click **Yes**.
3. In the left pane, click **SQL Server Services**.
4. In the right pane, right-click **SQL Server (MSSQLSERVER)**, and then click **Properties**.
5. In the **SQL Server (MSSQLSERVER) Properties** dialog box, on the **FILESTREAM** tab, check that the **Enable FILESTREAM for Transact-SQL access** check box is selected, and then click **OK**.

Configure FILESTREAM and FileTables in the AdventureWorks Database

1. On the taskbar, click **Microsoft SQL Server Management Studio**.
2. In the **Connect to Server** dialog box, in **Server name**, type **MIA-SQL**, and then click **Connect**.
3. On the **File** menu, point to **Open**, and then click **Project/Solution**.
4. In the **Open Project** dialog box, navigate to **D:\Demofiles\Mod16\Demo**, click **demo.ssmssln**, and then click **Open**.
5. In Solution Explorer, double-click the **2 - Configuring FILESTREAM and FileTables.sql** script file.
6. Select the code under the **Step 1** comment, and then click **Execute**.

7. Select the code under the **Step 2** comment, and then click **Execute**.
8. Select the code under the **Step 3** comment, and then click **Execute**.
9. Select the code under the **Step 4** comment, and then click **Execute**.
10. Select the code under the **Step 5** comment, and then click **Execute**.
11. Select the code under the **Step 6** comment, and then click **Execute**.
12. Select the code under the **Step 7** comment, and then click **Execute**.
13. Select the code under the **Step 8** comment, and then click **Execute**.
14. Select the code under the **Step 9** comment, and then click **Execute**.
15. Keep Microsoft SQL Server Management Studio open for the next demonstration.

Lesson 3

Using Full-Text Search

Contents:

Question and Answers

Question: You have a full-text index set up on the HumanResources.Employees table that includes the Resume column. You want to locate employees who have management skills. You want to search the Resume column for the word "manage" and you want résumés with the words "manager", "managed", and "managing" to be included in the results. What kind of search should you use?

( ) A simple term search.

( ) A generation term search.

( ) A proximity term search.

( ) A thesaurus search.

( ) A weighted term search.

Answer:

( ) A simple term search.

(√) A generation term search.

( ) A proximity term search.

( ) A thesaurus search.

( ) A weighted term search.

Demonstration: Configuring and Using Full-Text Search

Demonstration Steps

Create and Use a Full-Text Index

1. In Solution Explorer, double-click the **3 - Configuring and Using Full-Text Search.sql** script file.
2. Select the code under the **Step 1** comment, and then click **Execute**.
3. Select the code under the **Step 2** comment, and then click **Execute**.
4. Select the code under the **Step 3** comment, and then click **Execute**.
5. Select the code under the **Step 4** comment, and then click **Execute**.
6. Select the code under the **Step 5** comment, and then click **Execute**.
7. Select the code under the **Step 6** comment, and then click **Execute**.
8. Close Microsoft SQL Server Management Studio, without saving any changes.

Lab Review Questions and Answers

Lab: Storing and Querying BLOBs and Text Documents in SQL Server

Question and Answers

Lab Review

Question: How did the results of the simple term query you executed in Exercise 3, Task 2 differ from the results of the generation terms query?

Answer: There were more results for the generational query.

Question: What did you notice about the results of the third query you ran against the full-text index?

Answer: The Description column included results that contained forms of the word "Bike" but not results that included the word "Bike" itself. For example, results that contained the word "Bikes" were included. This illustrates the difference between a simple term query and a generation term query.

Module 17

SQL Server Concurrency

Contents:

Lesson 1

Concurrency and Transactions

Contents:

Question and Answers

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

( ) Pessimistic concurrency

( ) Optimistic concurrency

Answer:

( ) Pessimistic concurrency

(√) Optimistic concurrency

Demonstration: Analyzing Concurrency Problems

Demonstration Steps

Preparation

1. Ensure that the MSL-TMG1, 20762B-MIA-DC, and 20762B-MIA-SQL virtual machines are running, and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Run **Setup.cmd** in the **D:\Demofiles\Mod17** folder as Administrator.
3. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.
4. Start SQL Server Management Studio and connect to your Azure instance of the **AdventureWorksLT** database engine instance using SQL Server authentication.
5. In the **Login** box, type **Student**, in the **Password** box, type **Pa$$w0rd**, and then click **Connect**.
6. Open the **Demo1.ssmssln** solution in the **D:\Demofiles\Mod17\Demo1** folder.
7. Open the **Demo 1a.sql** and the **Demo 1b.sql** script files; open these files in different query windows, because you will be switching between them.

Dirty Reads

1. In the **Demo 1a.sql** script file, under the comment that begins **Step 1**, select the code, and then click **Execute** to check the current settings for SNAPSHOT isolation.
2. Under the comment that begins **Step 2**, select the code, and then click **Execute** to view the current state of the row used in this demonstration.
3. In the **Demo 1b.sql** script file, under the comment that begins **Query 1**, select the code, and then click **Execute**.
4. In the **Demo 1a.sql** file, under the comment that begins **Step 3**, select the code, and then click **Execute** to demonstrate READ UNCOMMITTED isolation.
5. Under the comment that begins **Step 4**, select the code, and then click **Execute** to demonstrate READ COMMITTED isolation. The query will wait until you complete the next step.
6. In the **Demo 1b.sql** script file, under the comment that begins **Query 2**, select the code, and then click **Execute**.

Non-repeatable Reads

1. In the **Demo 1a.sql** script file, under the comment that begins **Step 5**, select the first five lines of code, and then click **Execute**.
2. In the **Demo 1b.sql** script file, under the comment that begins **Query 3**, select the code, and then click **Execute**.
3. In the **Demo 1a.sql** script file, under the comment that begins **Step 5**, select the final four lines of code, and then click **Execute** to demonstrate a non-repeatable read.
4. Under the comment that begins **Step 6**, select the first six lines of code, and then click **Execute**.
5. In the **Demo 1b.sql** script file, under the comment that begins **Query 4**, select the code, and then click **Execute**. Note that this query will not return until you complete the next step, because REPEATABLE READ holds a lock on the affected row.

Phantom Read

1. In the script **Demo 1a.sql** that begins **Step 7**, select the first six lines of code, and then click **Execute**.
2. In the **Demo 1b.sql** script file, under the comment that begins **Query 5**, select the code, and then click **Execute**.

Serializable and Phantom Read

1. In the script **Demo 1a.sql** under the comment that begins **Step 8**, select the first six lines of code, and then click **Execute**.
2. In the **Demo 1b.sql** script file under the comment that begins **Query 5**, select the code, and then click **Execute**. Note that this query will not return until you complete the next step, since SERIALIZABLE holds a lock on the affected table.

Snapshot Isolation

1. In **Demo 1a.sql**, under the comment that begins **Step 9**, select the first eight lines of code, and then click **Execute**. This can take up to 30 minutes to complete.
2. Select the last three lines of code under **Step 9**, and then click **Execute**. Note the locks.
3. In the **Demo 1b.sql** script file, under the comment that begins **Step 6**, select the code, and then click **Execute**. Note that the query does not complete.
4. In the **Demo 1a.sql** script file, under the comment that begins **Query 10**, select the code, and then click **Execute**. Note the locks.
5. In the **Demo 1a.sql** script file, under the comment that begins **Step 11**, select the code, and then click **Execute.** Note that the query in Demo 1b aborted, demonstrating the behavior of SNAPSHOT isolation.
6. Close SQL Server Management Studio without saving any changes.

Lesson 2

Locking Internals

Contents:

Question and Answers

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

( ) Exclusive (X)

( ) Shared (S)

( ) Intent shared (IS)

( ) Intent exclusive (IX)

( ) Intent update (IU)

Answer:

( ) Exclusive (X)

( ) Shared (S)

( ) Intent shared (IS)

(√) Intent exclusive (IX)

( ) Intent update (IU)

Resources

Locking Hints

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

Demonstration: Applying Locking Hints

Demonstration Steps

1. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine instance using Windows authentication.
2. Open the **Demo2.ssmssln** solution in the **D:\Demofiles\Mod17\Demo2** folder.
3. Open the **Demo 2a - lock hints 1.sql** and **Demo 2b - lock hints 2.sql** script files. Ensure that both scripts use the **AdventureWorks** database.
4. In the **Demo 2a - lock hints 1.sql** script file, under the comment that begins **Step 3**, select the code, and then click **Execute** to show the current isolation level.
5. Under the comment that begins **Step 4**, select the first three lines of code, and then click **Execute** to demonstrate the locks held by a transaction using READ UNCOMMITTED isolation.
6. Under the comment that begins **Step 4**, select the remaining five lines of code, and then click **Execute**.
7. Under the comment that begins **Step 5**, select the first three lines of code, and then click **Execute** to demonstrate the locks held by a transaction using REPEATABLE READ isolation.
8. Under the comment that begins **Step 5**, select the remaining five lines of code, and then click **Execute**.

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

Module Review and Takeaways

Review Question(s)

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

( ) SERIALIZABLE

( ) READ UNCOMMITTED

( ) REPEATABLE READ

( ) READ COMMITTED

Answer:

( ) SERIALIZABLE

( ) READ UNCOMMITTED

( ) REPEATABLE READ

(√) READ COMMITTED

Lab Review Questions and Answers

Lab: Concurrency and Transactions

Question and Answers

Lab Review

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

( ) Database: Shared (S)
Table: Exclusive (X)

( ) Database: Shared (S)
Table: Intent Exclusive (IX)
Partition: Exclusive (X)

( ) Database: Shared (S)
Table: Exclusive (X)
Partition: Exclusive (X)

Answer:

( ) Database: Shared (S)
Table: Exclusive (X)

(√) Database: Shared (S)
Table: Intent Exclusive (IX)
Partition: Exclusive (X)

( ) Database: Shared (S)
Table: Exclusive (X)
Partition: Exclusive (X)

Module 18

Performance and Monitoring

Contents:

Lesson 1

Extended Events

Contents:

Question and Answers

Question: Which of the following statements about Extended Events is incorrect?

( ) Extended Events can be viewed using Watch Live Data without starting a session.

( ) Extended Events sessions can be created using Transact-SQL commands.

( ) Extended Events is limited in what it can do and will soon be deprecated. Use SQL Trace whenever possible.

( ) Watch Live Data provides a real-time view of query execution statistics.

Answer:

( ) Extended Events can be viewed using Watch Live Data without starting a session.

( ) Extended Events sessions can be created using Transact-SQL commands.

(√) Extended Events is limited in what it can do and will soon be deprecated. Use SQL Trace whenever possible.

( ) Watch Live Data provides a real-time view of query execution statistics.

Demonstration: Creating an Extended Events Session

Demonstration Steps

1. Start the 20762B-MIA-DC, and 20762B-MIA-SQL virtual machines, and then log on to 20762B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the **D:\Demofiles\Mod18** folder, run **Setup.cmd** as Administrator.
3. In the **User Account Control** dialog box, click **Yes** and wait for the script to finish.
4. Start **SQL Server Management Studio** and connect to the **MIA-SQL** database engine instance using Windows authentication.
5. On the **File** menu, point to **Open**, and then click **Project/Solution**.
6. In the **Open Project** dialog box, navigate to the **D:\Demofiles\Mod18** folder, click **Demo.ssmssln**, and then click **Open**.
7. In Solution Explorer, double-click **Demo 1 - create xe session.sql**.
8. Select code under the comment that begins **Step 1**, and then click **Execute** to create an Extended Events session.
9. Select code under the comment that begins **Step 2**, and then click **Execute** to verify that the session metadata is visible.
10. Select code under the comment that begins **Step 3**, and then click **Execute** to start the session and execute some queries.
11. Select code under the comment that begins **Step 4**, and then click **Execute** to query the session data.
12. Select code under the comment that begins **Step 5**, and then click **Execute** to refine the session data query.
13. In Object Explorer, under **MIA-SQL**, expand **Management**, expand **Extended Events**, expand **Sessions**, expand **SqlStatementCompleted**, and then double-click **package0.ring_buffer**.
14. In the **Data** column, click the XML value, and note that this is the same data that is returned by the query under the comment that begins Step 4 (note that additional statements will have been captured because you ran the code earlier).

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

Lesson 2

Working with Extended Events

Contents:

Question and Answers

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

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

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

Answer:

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

Demonstration: Tracking Session-Level Waits

Demonstration Steps

1. In SSMS, in Solution Explorer, double-click **Demo 2 - track waits by session.sql**.
2. In Object Explorer, expand **Management**, expand **Extended Events**, right-click **Sessions**, and then click **New Session**.
3. In the **New Session** dialog box, on the **General** page, in the **Session name** box, type **Waits by Session**.
4. On the **Events** page, in the **Event library** box, type **wait**, and then, in the list below, double-click **wait_info** to add it to the **Selected events** list.
5. Click **Configure** to display the **Event configuration options** list.
6. In the **Event configuration options** list, on the **Global Fields (Actions)** tab, select the **session_id** check box.
7. On the **Filter (Predicate)** tab, click **Click here to add a clause**.
8. In the **Field** list, click **sqlserver.session_id**, in the **Operator** list, click **>**, and then in the **Value** box, type **50**. This filter will exclude most system sessions from the session.
9. On the **Data Storage** page, click **Click here to add a target**.
10. In the **Type** list, click **event_file**, in the **File name on server** box, type **D:\Demofiles\Mod18\waitbysession**, in the first **Maximum file** size box, type **5**, in the second **Maximum file** size box, click **MB**, and then click **OK**.
11. In Object Explorer, under **Sessions**, right-click **Waits by Session**, and then click **Start Session**.
12. In File Explorer, in the **D:\Demofiles\Mod18** folder, right-click **start_load_1.ps1**, and then click **Run with PowerShell**. If a message is displayed asking you to confirm a change in execution policy, type **Y**, and then press Enter. Leave the workload to run for a minute or so before proceeding.
13. In SSMS, in the **Demo 2 - track waits by session.sql** query pane, select the code under the comment that begins **Step 14**, click **Execute**, and then review the results.
14. Select the code under the comment that begins **Step 15**, and then click **Execute** to stop and drop the session, and to stop the workload.
15. In File Explorer, in the **D:\Demofiles\Mod18** folder, note that one (or more) files with a name matching **waitbysession*.xel** have been created.
16. Close File Explorer, and then close Windows PowerShell®.
17. Keep SQL Server Management Studio open for the next demonstration.

Lesson 3

Live Query Statistics

Contents:

Question and Answers

Question: Which of the following statements will enable Live Query Statistics for all sessions in Activity Monitor?

( ) ALTER DATABASE SET STATISTICS ON

( ) SET LIVE QUERY STATISTICS ON

( ) ENABLE LIVE QUERY STATISTICS

( ) SET STATISTICS PROFILE OFF

( ) SET STATISTICS XML ON

Answer:

( ) ALTER DATABASE SET STATISTICS ON

( ) SET LIVE QUERY STATISTICS ON

( ) ENABLE LIVE QUERY STATISTICS

( ) SET STATISTICS PROFILE OFF

(√) SET STATISTICS XML ON

Demonstration: Enable Live Query Statistics for a Session

Demonstration Steps

1. In SSMS, in Solution Explorer, double-click **Demo 3 - live query statistics.sql** script file.
2. Highlight the script under the **Step 1** description, and then click **Execute**.
3. Highlight the script under the **Step 2** description, and then click **Execute**.
4. On the **Query** menu, click **Include Live Query Statistics**.
5. Highlight the script under the **Step 4** description, and then click **Execute**.
6. On the **Query** menu, click **Include Live Query Statistics**.
7. Close SQL Server Management Studio, without saving any changes.

Lesson 4

Optimize Database File Configuration

Contents:

Question and Answers

Question: Which of the following statements is correct?

( ) Each SQL Server user database has one tempdb.

( ) tempdb is a shared resource between all databases on a SQL Server instance.

( ) The number of tempdb databases depends on the number of logical processors on the server.

( ) You can have any number of user databases on an instance, but the exact number is hidden.

( ) tempdb is new in SQL Server 2016.

Answer:

( ) Each SQL Server user database has one tempdb.

(√) tempdb is a shared resource between all databases on a SQL Server instance.

( ) The number of tempdb databases depends on the number of logical processors on the server.

( ) You can have any number of user databases on an instance, but the exact number is hidden.

( ) tempdb is new in SQL Server 2016.

Resources

Improving Performance with tempdb

 **Best Practice:** Use fast storage for tempdb. Unlike a log file that writes records to disk sequentially, tempdb has varied reads and writes, and benefits from fast storage media.

Configuring tempdb

 **Best Practice:** Ensure tempdb files are large enough for normal workload without relying on autogrowth. For best performance, autogrowth should handle the unexpected, not the everyday.

 **Best Practice:** When configuring a SQL Server instance for failover clustering, ensure the directories are valid for all cluster nodes. If the tempdb directory(s)—including log directory—are not on the failover target node, SQL Server will not fail over.

Lesson 5

Metrics

Contents:

Resources

Generating Baseline Metrics

 **Best Practice:** Capture relevant data and keep it only for as long as it is useful—typically three to four months.

Module Review and Takeaways

Review Question(s)

Question: Do you use baselines to help you manage your SQL Server systems? If so, what tools do you use to create baselines? If not, what are your reasons for not creating baselines?

Answer: Answers will vary.

Lab Review Questions and Answers

Lab: Monitoring, Tracing, and Baselining

Question and Answers

Lab Review

Question: What advantages do you see in using Extended Events to monitor your SQL Server databases?

Answer: Answers will vary.