Creating Programmatic SQL Database Objects

Lab 2 – Responding to Data Manipulation by Using Triggers

Overview

You are required to audit any changes to data in a table that contains sensitive balance data. You have decided to implement this by using DML triggers because the SQL Server Audit mechanism does not provide directly for the requirements in this case.

Before starting this lab, you should view **Module 2 – Creating Triggers** in the course *Creating Programmatic SQL Database Objects*. Then, if you have not already done so, follow the instructions in the **Getting Started** document for this course to set up the lab environment.

If you find some of the challenges difficult, don't worry – you can find suggested solutions for all of the challenges in the **Lab Solution** folder for this module.

What You'll Need

To complete the labs, you will need the following:

- An Azure SQL Database instance with the AdventureWorksLT sample database. Review the Getting Started document for information about how to provision this.
- The lab files for this course

Challenge 1: Create an Update Trigger

You are required to audit any changes to data in a table that contains sensitive balance data. You have decided to implement this by using DML triggers because the SQL Server Audit mechanism does not provide directly for the requirements in this case.

Review the Reports.GetProductColors Stored Procedure Specification

Review the following design requirements for your stored procedure:

The **Production.Product** table includes a column called **ListPrice**. Whenever an update is made to the table, if either the existing balance or the new balance is greater than **1,000 US dollars**, an entry must be written to the **Production.ProductAudit** audit table.

The **Production.ProductAudit** table is used to hold changes to high-value products. The data to be inserted in each column is shown in the following table:

Column	Data type	Value to insert
AuditID	int	IDENTITY
ProductID	int	ProductID
UpdateTime	datetime2	SYSDATETIME()
ModifyingUser	varchar(30)	ORIGINAL_LOGIN()
OriginalListPrice	decimal(18,2)	ListPrice before update
NewListPrice	decimal(18,2)	ListPrice after update

Inserts or deletes on the table do not have to be audited. Details of the current user can be taken from the ORIGINAL_LOGIN() function.

Create the ProductAudit Table

Design and implement the ProductAudit table in accordance with the design specifications.

Create the TR_ProductListPrice_Update Trigger

Design and implement the trigger in accordance with the design specifications.

Test the Behavior of the Trigger

Update products with a ProductID between 749 and 753 to have a ListPrice of 3978.00.

Query the ProductAudit table to validate the trigger.

Review the Reports.GetProductsAndModels Stored Procedure Specification

1. Review the following design requirements for your stored procedure:

Stored Procedure:	Reports.GetProductsAndModels
Input Parameters:	None
Output Parameters:	None
Output Columns:	ProductID, Name, ProductNumber, SellStartDate, SellEndDate and Color (from SalesLT.Product), ProductModelID (from SalesLT.ProductModel), Description (from SalesLT.ProductDescription).
Output Order:	ProductID, ProductModelID
Notes:	For descriptions, return the Description column from the SalesLT.ProductDescription table.

Create the Reports.GetProductsandModels Stored Procedure

1. Design, implement, and execute the stored procedure in accordance with the design specifications.

Challenge 2: Create Parameterized Stored Procedures

In this exercise, you will create a stored procedure to support one of the new reports.

Review the Reports.GetProductsByColor Stored Procedure specification

1. Review the following design requirements for your stored procedure:

Stored Procedure	Reports.GetProductsByColor
Input parameters	@Color (same data type as the Color column in the Production.Product table)
Output parameters	None
Output columns	ProductID, Name, ListPrice (returned as a column named Price), Color, and Size (from Production.Product)
Output order	Name
Notes	The procedure should return products that have no Color if the parameter is NULL.

Create the Reports.GetProductsByColor Stored Procedure

1. Design and create the Reports.GetProductsByColor stored procedure.
2. Execute the Reports.GetProductsByColor stored procedure with a color of 'Blue'.
3. Execute the Reports.GetProductsByColor stored procedure with a color of NULL.