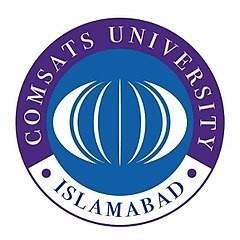
`

**COMSATS University Islamabad, Lahore Campus**

**Block–B, Department of Computer Engineering**

**COMSATS University Islamabad, Lahore Campus 1.5KM Defence Road, Off Raiwind Road, Lahore**

Assignment

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Course Title: | Database Systems | | | | Course Code: | CSC270 | Credit Hours: | 4(3,***1***) |
| Course Instructor/s: | Modassir Ishfaq | | | | Programme Name: | BCE | | |
| Semester: | 4th | Batch: | Fa23 | Section: | A&B | Date: |  | |
| **Time Allowed:** | **Within 3 days** | | | | **Maximum Marks:** | | **10** | |
| Student’s Name: | Muhammad Ahmad | | | | Reg. No. | FA23-BCE-113 | | |
|  | | | | | | | | |

Question:

You are managing the backend of an online store. Customers place orders, and the system must deduct inventory and record the transaction safely. The following tables are in use:

*CREATE TABLE Products ( ProductID INT PRIMARY KEY,*

*ProductName VARCHAR(100), Stock INT,*

*Price DECIMAL(10,2), flag BOOL*

*);*

*CREATE TABLE Customer ( customerID INT PRIMARY KEY, CustomerName VARCHAR(100), Balance INT,*

*DateofJoining DATETIME*

*);*

*CREATE TABLE Orders (*

*OrderID INT IDENTITY(1,1) PRIMARY KEY,*

*ProductID INT, CustomerID INT, Quantity INT,*

*TotalAmount DECIMAL(12,2),*

*OrderDate DATETIME DEFAULT GETDATE()*

*);*

You want to audit all changes (INSERT, UPDATE, DELETE) on orders table and record them in an AuditLog\_orders table with information such as Operation type (INSERT / UPDATE / DELETE), User ID, orderID, Timestamp of change. In case of deletion, the data must be logged in a separate table. In case of update, it must log the attribute, its old value and new value in another separate table. Therse separate tables must have a link with the auditlog ID.

-- Create AuditLog\_orders table to audit all changes (INSERT, UPDATE, DELETE) on Orders table and record them with operation type, User ID, orderID, and Timestamp

CREATE TABLE AuditLog\_orders (

AuditID INT IDENTITY(1,1) PRIMARY KEY,

OperationType VARCHAR(10), -- Operation type (INSERT / UPDATE / DELETE)

UserID INT, -

OrderID INT,

Timestamp DATETIME DEFAULT GETDATE()change

);

-- Create a separate table to record deleted order data, linked with the auditlog ID

CREATE TABLE DeletedOrders (

DeletedOrderID INT IDENTITY(1,1) PRIMARY KEY,

AuditID INT, -- Links to AuditLog\_orders

OrderID INT,

ProductID INT,

CustomerID INT,

Quantity INT,

TotalAmount DECIMAL(12,2),

OrderDate DATETIME,

FOREIGN KEY (AuditID) REFERENCES AuditLog\_orders(AuditID)

);

-- Create a separate table to log updated attributes, their old value and new value, linked with the auditlog ID

CREATE TABLE OrderUpdateLog (

UpdateLogID INT IDENTITY(1,1) PRIMARY KEY,

AuditID INT, -- Links to AuditLog\_orders

AttributeName VARCHAR(50),

OldValue NVARCHAR(255),

NewValue NVARCHAR(255),

FOREIGN KEY (AuditID) REFERENCES AuditLog\_orders(AuditID)

);

-- Trigger to audit all INSERT changes on Orders table and record in AuditLog\_orders

CREATE TRIGGER trg\_Orders\_Insert

ON Orders

AFTER INSERT

AS

BEGIN

INSERT INTO AuditLog\_orders (OperationType, UserID, OrderID, Timestamp)

SELECT 'INSERT', SYSTEM\_USER, OrderID, GETDATE()

FROM inserted; -- inserted is a system table which hold the last inserted dara

END;

-- Trigger to audit all DELETE changes on Orders table and record in AuditLog\_orders, and record deleted data in a separate table

CREATE TRIGGER trg\_Orders\_Delete

ON Orders

AFTER DELETE

AS

BEGIN

INSERT INTO AuditLog\_orders (OperationType, UserID, OrderID, Timestamp)

SELECT 'DELETE', SYSTEM\_USER, OrderID, GETDATE()

FROM deleted; -- inserted is a system table which hold the last deleted data

INSERT INTO DeletedOrders (AuditID, OrderID, ProductID, CustomerID, Quantity, TotalAmount, OrderDate)

SELECT SCOPE\_IDENTITY(), OrderID, ProductID, CustomerID, Quantity, TotalAmount, OrderDate

FROM deleted;

END;

-- Trigger to audit all UPDATE changes on Orders table and record in AuditLog\_orders, and log the attribute, its old value and new value in a separate table

CREATE TRIGGER trg\_Orders\_Update

ON Orders

AFTER UPDATE

AS

BEGIN

INSERT INTO AuditLog\_orders (OperationType, UserID, OrderID, Timestamp)

SELECT 'UPDATE', SYSTEM\_USER, OrderID, GETDATE()

FROM inserted;

DECLARE @AuditID INT = SCOPE\_IDENTITY();

INSERT INTO OrderUpdateLog (AuditID, AttributeName, OldValue, NewValue)

SELECT @AuditID, 'ProductID', CAST(d.ProductID AS NVARCHAR(255)), CAST(i.ProductID AS NVARCHAR(255))

FROM deleted d

INNER JOIN inserted i ON d.OrderID = i.OrderID

WHERE d.ProductID != i.ProductID;

INSERT INTO OrderUpdateLog (AuditID, AttributeName, OldValue, NewValue)

SELECT @AuditID, 'CustomerID', CAST(d.CustomerID AS NVARCHAR(255)), CAST(i.CustomerID AS NVARCHAR(255))

FROM deleted d

INNER JOIN inserted i ON d.OrderID = i.OrderID

WHERE d.CustomerID != i.CustomerID;

INSERT INTO OrderUpdateLog (AuditID, AttributeName, OldValue, NewValue)

SELECT @AuditID, 'Quantity', CAST(d.Quantity AS NVARCHAR(255)), CAST(i.Quantity AS NVARCHAR(255))

FROM deleted d

INNER JOIN inserted i ON d.OrderID = i.OrderID

WHERE d.Quantity != i.Quantity;

INSERT INTO OrderUpdateLog (AuditID, AttributeName, OldValue, NewValue)

SELECT @AuditID, 'TotalAmount', CAST(d.TotalAmount AS NVARCHAR(255)), CAST(i.TotalAmount AS NVARCHAR(255))

FROM deleted d

INNER JOIN inserted i ON d.OrderID = i.OrderID

WHERE d.TotalAmount != i.TotalAmount;

INSERT INTO OrderUpdateLog (AuditID, AttributeName, OldValue, NewValue)

SELECT @AuditID, 'OrderDate', CAST(d.OrderDate AS NVARCHAR(255)), CAST(i.OrderDate AS NVARCHAR(255))

FROM deleted d

INNER JOIN inserted i ON d.OrderID = i.OrderID

WHERE d.OrderDate != i.OrderDate;

END;

After performing various operations the condition of tables

