SQL Server 2012 – Database Development

Lesson 5: Implementing Triggers

Capgemini

Lesson Objectives

- In this lesson, you will learn:
  - Introduction to Triggers
  - Defining Triggers
  - How Triggers Work

Introduction

- A trigger is a mechanism that is invoked when a particular action occurs on a particular table
- Each trigger has three general parts:
  - A name
  - The action
  - The execution
- The action of a trigger can be either a DML statement (INSERT, UPDATE, or DELETE) or a DDL statement
- Therefore, there are two trigger forms: DML triggers and DDL triggers
- The execution part of a trigger usually contains a stored procedure or a batch

Advantages of triggers

- Cascade Changes Through Related Tables in a Database
- Enforce More Complex Data Integrity check than constraints
- Implement complex business rules
- Triggers can be used to create business rules for an application
- Procedural integrity constraints are handled by triggers

Features

- Triggers Are Reactive; Constraints Are Proactive
- Constraints Are Checked First
- Tables Can Have Multiple Triggers for Any Action
- Table owners only can create triggers
- Triggers can be written for DDL and DML operations on the table
- SQL Triggers are AFTER triggers

After Trigger

- AFTER triggers are executed after the action of the INSERT,UPDATE, or DELETE statement is performed
- Specifying AFTER is the same as specifying FOR, which is the only option available in earlier versions of SQL Server
- You can define AFTER triggers only on tables
- When a trigger is defined as After, the trigger fires after the modification has passed all constraints
- Multiple After Triggers can be defined for one action like INSERT, UPDATE or DELETE
- If you have multiple trigger created for the same action, you can specify the order in which they can get fired
- You can achieve this with use of sp_settriggerorder system stored procedure
- In SQL Server triggers are by default After Trigger

Tables used by DML trigger

- Tables used by trigger
  - inserted table: It always stores new values while execution of trigger
  - deleted table: Stores old values while execution of trigger

- In after trigger the statement which fires trigger gets logged which will give old and new values through inserted and deleted tables.

Trigger Name	Inserted	Deleted
INSERT	Newly inserted record	None
DELETE	Non	Deleted record
UPDATE	Record with New values	Record with Old values

Creating Triggers

```
CREATE TRIGGER Empl_Delete ON Employees
FOR DELETE
AS
IF exists (select 'a' from loan,deleted  where
    loan.EmpCode = deleted.EmpCOde)
BEGIN
   RAISERROR(
     'You cannot delete   employee having loan', 16, 1)
   ROLLBACK TRANSACTION
END
```

Instead of Trigger

- A trigger with an INSTEAD OF clause replaces the corresponding triggering action
- It is executed after the corresponding inserted and deleted tables are created, but before any integrity constraint or any other action is performed
- INSTEAD OF triggers can be created on tables as well as on views
- There are certain requirements on column values that are supplied by an INSTEAD OF trigger:
  - Values cannot be specified for computed columns
  - Values cannot be specified for columns with the TIMESTAMP data type
  - Values cannot be specified for columns with an IDENTITY property, unless the IDENTITY_INSERT option is set to ON

Instead of Trigger - Example

```
CREATE TRIGGER delEmployee
ON [HumanResources].[Employee]
INSTEAD OF DELETE
AS
BEGIN
        DECLARE @DeleteCount int;
        SELECT @DeleteCount = COUNT(*) FROM deleted
        IF @DeleteCount > 0
                        BEGIN
                        RAISERROR
                        ('Employees cannot be deleted. They can only be marked as not
                        current.', -- Message
                        10, -- Severity.
                        1); -- State.
-- Roll back any active or uncommittable transactions
        IF @@TRANCOUNT > 0
                        BEGIN
                        ROLLBACK TRANSACTION;
                        END
        END
END
```

Altering and Dropping a trigger

- Altering a Trigger
  - Changes the definition without dropping the trigger
  - Can disable or enable a trigger
  - ALTER Trigger <trggername>
- Dropping a Trigger

  DROP Trigger <trggername>
- When a table is dropped Triggers are also automatically dropped

Demo

- Working with Triggers

Summary

In this lesson, you have learnt:

- How to define Triggers
- How the Triggers works

Review Question

- Question 1: A trigger with an INSTEAD OF clause replaces the corresponding triggering action
  - True
  - False