Module 11: Implementing Triggers
Overview

- Introduction
- Defining
  - Create, drop, alter triggers
- How Triggers Work
- Examples
- Performance Considerations
  - Analyze performance issues related to triggers
Introduction to Triggers

- What Is a Trigger?
- Uses
- Considerations for Using Triggers
What Is a Trigger?

- Associated with a Table
- Invoked Automatically
- Cannot Be Called Directly
- Is Part of a Transaction
  - Along with the statement that calls the trigger
  - Can ROLLBACK transactions (use with care)
Uses of Triggers

- Cascade Changes Through Related Tables in a Database
  - A delete or update trigger can cascade changes to related tables: Soda name change to change in soda name in Sells table

- Enforce More Complex Data Integrity Than a CHECK Constraint
  - Change prices in case of price rip-offs.

- Define Custom Error Messages

- Maintain Denormalized Data
  - Automatically update redundant data.

- Compare Before and After States of Data Under Modification
Considerations for Using Triggers

- Triggers Are Reactive; Constraints Are Proactive
- Constraints Are Checked First
- Tables Can Have Multiple Triggers for Any Action
- Table Owners Can Designate the First and Last Trigger to Fire
- You Must Have Permission to Perform All Statements That Define Triggers
- Table Owners Cannot Create AFTER Triggers on Views or Temporary Tables
◆ Defining Triggers

- Creating Triggers
- Altering and Dropping Triggers
Creating Triggers

- Requires Appropriate Permissions
- Cannot Contain Certain Statements

Use Northwind
GO
CREATE TRIGGER Empl_Delete ON Employees
FOR DELETE
AS
IF (SELECT COUNT(*) FROM Deleted) > 1
BEGIN
    RAISERROR('You cannot delete more than one employee at a time.', 16, 1)
    ROLLBACK TRANSACTION
END
Altering and Dropping Triggers

- **Altering a Trigger**
  - Changes the definition without dropping the trigger
  - Can disable or enable a trigger

```sql
USE Northwind
GO
ALTER TRIGGER Empl_Delete ON Employees
FOR DELETE
AS
IF (SELECT COUNT(*) FROM Deleted) > 6
BEGIN
    RAISERROR('You cannot delete more than six employees at a time.', 16, 1)
    ROLLBACK TRANSACTION
END
```

- **Dropping a Trigger**
How Triggers Work

- How an INSERT Trigger Works
- How a DELETE Trigger Works
- How an UPDATE Trigger Works
- How an INSTEAD OF Trigger Works
- How Nested Triggers Work
- Recursive Triggers
How an INSERT Trigger Works

1. INSERT Statement to a Table with an INSERT Trigger Defined
2. INSERT Statement Logged
3. Trigger Actions Executed
How a DELETE Trigger Works

1. DELETE Statement to a Table with a DELETE Statement Defined
2. DELETE Statement Logged
3. Trigger Actions Executed
How an UPDATE Trigger Works

1. UPDATE Statement to a Table with an UPDATE Trigger Defined

2. UPDATE Statement Logged as INSERT and DELETE Statements

3. Trigger Actions Executed
How an INSTEAD OF Trigger Works

1. INSTEAD OF Trigger Can Be on a Table or View

2. The Action That Initiates the Trigger Does NOT Occur

3. Allows Updates to Views Not Previously Updateable
How Nested Triggers Work

Placing an order causes the OrDe_Update trigger to execute

Executes an UPDATE statement on the Products table

InStock_Update trigger executes

Sends message

OrDe_Update

InStock_Update

UnitsInStock + UnitsOnOrder is < ReorderLevel for ProductID 2
Recursive Triggers

- Activating a Trigger Recursively
- Types of Recursive Triggers
  - *Direct recursion* occurs when a trigger fires and performs an action that causes the same trigger to fire again
  - *Indirect recursion* occurs when a trigger fires and performs an action that causes a trigger on another table to fire
- Determining Whether to Use Recursive Triggers
Examples of Triggers

- Enforcing Data Integrity
- Enforcing Business Rules
Enforcing Data Integrity

CREATE TRIGGER BackOrderList_Delete
    ON Products FOR UPDATE
    AS
    IF (SELECT BO.ProductID FROM BackOrders AS BO JOIN Inserted AS I ON BO.ProductID = I.Product_ID ) > 0
    BEGIN
        DELETE BO FROM BackOrders AS BO
        INNER JOIN Inserted AS I
        ON BO.ProductID = I.ProductID
    END

<table>
<thead>
<tr>
<th>Products</th>
<th>BackOrders</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductID</td>
<td>UnitsInStock</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>65</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Updated</td>
</tr>
</tbody>
</table>
Enforcing Business Rules

**Products with Outstanding Orders Cannot Be Deleted**

IF (Select Count (*)
    FROM [Order Details] INNER JOIN deleted
    ON [Order Details].ProductID = deleted.ProductID
) > 0
ROLLBACK TRANSACTION

DELETE statement executed on Product table

<table>
<thead>
<tr>
<th>ProductID</th>
<th>UnitsInStock</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>65</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
</tr>
</tbody>
</table>

Trigger code checks the Order Details table

<table>
<thead>
<tr>
<th>OrderID</th>
<th>ProductID</th>
<th>UnitPrice</th>
<th>Quantity</th>
<th>Discount</th>
</tr>
</thead>
<tbody>
<tr>
<td>10522</td>
<td>10</td>
<td>31.00</td>
<td>7</td>
<td>0.2</td>
</tr>
<tr>
<td>10523</td>
<td>2</td>
<td>19.00</td>
<td>9</td>
<td>0.15</td>
</tr>
<tr>
<td>10524</td>
<td>41</td>
<td>9.65</td>
<td>24</td>
<td>0.0</td>
</tr>
<tr>
<td>10525</td>
<td>7</td>
<td>30.00</td>
<td>9</td>
<td>0.0</td>
</tr>
</tbody>
</table>

'Transaction cannot be processed'
'This product has order history'

Transaction rolled back
Performance Considerations

- Triggers Work Quickly Because the Inserted and Deleted Tables Are in Cache

- Execution Time Is Determined by:
  - Number of tables that are referenced
  - Number of rows that are affected

- Actions Contained in Triggers Implicitly Are Part of a Transaction
Recommended Practices

- Use Triggers Only When Necessary
- Keep Trigger Definition Statements as Simple as Possible
- Include Recursion Termination Check Statements in Recursive Trigger Definitions
- Minimize Use of ROLLBACK Statements in Triggers
Review

- Introduction
- Defining
  - Create, drop, alter triggers
- How Triggers Work
- Examples
- Performance Considerations
  - Analyze performance issues related to triggers