Assertions and Triggers

Rose-Hulman Institute of Technology

Curt Clifton

Assertions

- Like constraints:
  - Recall: state IN {'IA', 'MN', 'WI', 'MI', 'IL'}
- But can reference all tables
- Defined by:
  - CREATE ASSERTION <name>
    CHECK (<condition>);

Example: Assertion

- In Sells(rest, soda, price), no rest may charge an average of more than $3.
  - CREATE ASSERTION NoRipoffs CHECK (
    NOT EXISTS (
    SELECT rest FROM Sells
    GROUP BY rest
    HAVING AVG(price) > 3
    ));

Example: Assertion

- The minimum price charged for products made by Coca-Cola Co. is $2
- Recall:
  - Soda(name, manf)
  - Sells(rest, soda, price)

Example: Assertion

- The minimum price charged for products made by Coca-Cola Co. is $2
- CREATE ASSERTION NoCheapCoke
  CHECK(
  NOT EXISTS(
  SELECT * FROM Sells, Soda
  WHERE Sells.soda = Soda.name
  AND Soda.manf = 'Coca-Cola Co.'
  AND Sells.price < 2.00
  ))

Timing of Assertion Checks

- Logically, assertions **always** are true
- So when do we have to check them?

Timing of Assertion Checks

- Logically, assertions **always** are true
- So when do we have to check them?
  - Logically, after **any** change
  - Practically, the DBMS could calculate the set of important changes

Triggers: Motivation

- All the power of assertions
- But easier to implement:
  - Column- and row-based checks
  - Programmer specifies when they are activated
- Most DBMS just include triggers, not assertions

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
- Enforce More Complex Data Integrity Than a CHECK Constraint
- Define Custom Error Messages
- Automatically update redundant data
- Compare Before and After States of Data Under Modification

Creating Triggers

- Requires Appropriate Permissions
- Cannot Contain Certain Statements:
  - e.g., DROP DATABASE

```
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

```
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
```

- DISABLE TRIGGER Empl_Delete ON Employees
- ENABLE TRIGGER Empl_Delete ON Employees
- DROP TRIGGER Empl_Delete

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

- Consider:

```
USE Northwind
CREATE TRIGGER OrdDet_Insert
ON [Order Details]
FOR INSERT
AS
UPDATE P SET
UnitsInStock = (P.UnitsInStock - I.Quantity)
FROM Products AS P INNER JOIN Inserted AS I
ON P.ProductID = I.ProductID
```

How an INSERT Trigger Works

```
INSERT [Order Details] VALUES
(10523, 2, 19.00, 5, 0.2)
```

Order Details

OrderID	ProductID	UnitPrice	Quantity	Discount
10522	10	31.00	7	0.2
10523	41	9.65	9	0.15
10524	7	30.00	24	0.0
10523	2	19.00	5	

Products

ProductID	UnitsInStock	...	...
1	15		
2	5		
3	65		
4	20		

Insert statement logged

inserted

10523	2	19.00	5	0.2

How a DELETE Trigger Works

- Consider:

```
USE Northwind
CREATE TRIGGER Category_Delete
ON Categories
FOR DELETE
AS
UPDATE P SET Discontinued = 1
   FROM Products AS P INNER JOIN deleted
AS d
   ON P.CategoryID = d.CategoryID
```

How a DELETE Trigger Works

```
DELETE Categories
WHERE
CategoryID = 4
```

Categories

CategoryID	CategoryName	Description	Picture
1	Beverages	Soft drinks, coffees…	0x15…
2	Condiments	Sweet and savory …	0x15…
3	Confections	Desserts, candies, …	0x15…

Products

ProductID	Discontinued	…	CategoryID
1	0		1
2	1		4
3	0		2
4	0		3

DELETE statement logged

Deleted

4	Dairy Products	Cheeses	0x15…

How an UPDATE Trigger Works

- Consider:

```
USE Northwind
GO
CREATE TRIGGER Employee_Update
ON Employees
FOR UPDATE
AS
IF UPDATE (EmployeeID)
BEGIN
    RAISERROR ('Transaction cannot be processed.\
            ***** Employee ID number cannot be modified.',
            10, 1)
    ROLLBACK TRANSACTION
END
```

How an UPDATE Trigger Works

```
UPDATE Employees
SET EmployeeID = 17
WHERE EmployeeID = 2
```

Employees

EmployeeID	LastName	FirstName	Title	HireDate
1	Davolio	Nancy	Sales Rep.	~~~
2	Fuller	Andrew	Vice Pres.	~~~
3	Leverling	Janet	Sales Rep.	~~~
		Margare	Sales Rep.	~~~

Transaction cannot be processed.
******* Member number cannot be modified**

UPDATE Statement logg

inserted

17	Full

deleted

2	Full

Employees

EmployeeID	LastName	FirstName	Title	HireDate
1	Davolio	Nancy	Sales Rep.	~~~
2	Fuller	Andrew	Vice Pres.	~~~
3	Leverling	Janet	Sales Rep.	~~~
4	Peacock	Margare	Sales Rep.	~~~

INSTEAD OF Triggers

- INSTEAD OF trigger lets us interpret view modifications that wouldn't be allowed
- Example view:
  - CREATE VIEW Synergy(cust,soda,rest)
    AS
    SELECT Likes.customer, Sells.soda, Sells.rest
    FROM Likes, Sells, Frequents
    WHERE Likes.customer = Frequents.customer
    AND Sells.soda = Likes.soda
    AND Sells.rest = Frequents.rest

Interpreting a View Insertion

- INSERT INTO Synergy(cust, soda, rest)
  VALUES ('Molly', 'Sunkist', 'Regal Beagle')
- What does that mean?
- Can use INSTEAD OF trigger to decide

The Trigger

- CREATE TRIGGER SynergyInsert ON Synergy
  INSTEAD OF INSERT
  AS
  DECLARE @c nvarchar(30)
  DECLARE @s nvarchar(30)
  DECLARE @r nvarchar(30)
  SELECT @c=cust, @s=soda, @r=rest
  From Inserted
  INSERT INTO Likes VALUES(@c, @s)
  INSERT INTO Frequents VALUES(@c, @r)
  INSERT INTO Sells VALUES(@r, @s, null)

INSTEAD OF Triggers

- Can use them on views to define action
- Can also use them on regular tables
  - Optionally perform or ignore actions

How Nested Triggers Work

OrDe_Update

Order_Details

OrderID	ProductID	UnitPrice	Quantity	Discount
10522	10	31.00	7	0.2
10523	41	9.65	9	0.15
10524	7	30.00	24	0.0
10525	2	19.00	5	0.2

Placing an order causes the OrDe_Update trigger to execute

InStock_Update

Products

ProductID	UnitsInStock	...	...
1	15		
2	15		
3	65		
4	20		

Executes an UPDATE statement on the Products table

InStock_Update trigger executes

UnitsInStock + UnitsOnOrder
is < ReorderLevel for ProductID 2

Sends message

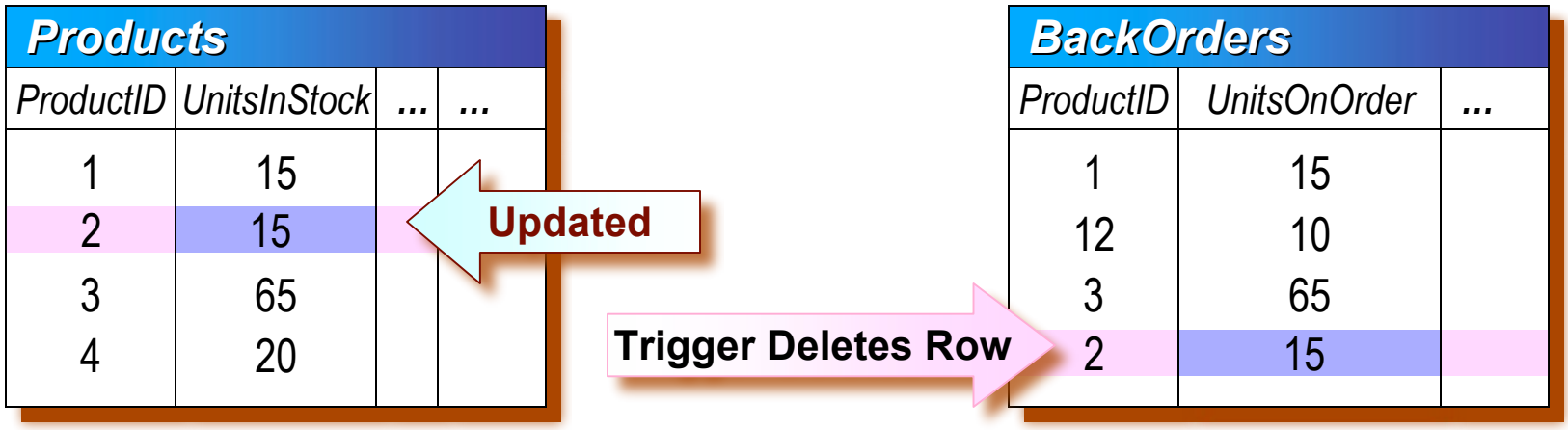
Recursive Triggers

- Activating a Trigger Recursively
  - See ALTER DATABASE command
- 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 that … causes the original trigger to fire again

Examples of Triggers

- Enforcing Data Integrity
- Enforcing Business Rules

```
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
```



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

Trigger code checks the Order Details table

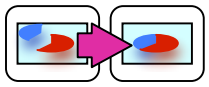
Transaction rolled back

Products

ProductID	UnitsInStock	...	...
1	15		
2	0		
3	65		
4	20		

Order Details

OrderID	ProductID	UnitPrice	Quantity	Discount
10522	10	31.00	7	0.2
10523	2	19.00	9	0.15
10524	41	9.65	24	0.0
10525	7	30.00		



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

Considerations for Using Triggers

- Triggers vs. Constraints
  - Constraints are proactive
  - Triggers reactive (FOR) or proactive (INSTEAD OF)
  - Constraints checked before triggers
- Can have multiple triggers for any action
- Use sp_settriggerorder to designate order
- Views and temporary tables may only have INSTEAD OF triggers

Performance Considerations

- Triggers Work Quickly — 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 Transaction