

Triggers and Stored Procedures

5 January 2010
Lecture 12

Topics for Today

- Triggers and Active Databases
- Stored Procedures
- Remote invocation

Source: RG 5.8-5.9, 6.5

Triggers

- Trigger: a procedure that is invoked automatically in the data base based on some predefined conditions
- Three parts:
 - Event – The change which activates
 - Condition –Test to perform when the event
 - Action – What to do if the condition is true
- A database with triggers is an **active database**

Trigger Types

- **Statement level triggers** which run for each statement (command) which matches the condition for the event
- **Row-level triggers** that run once for each row which is new or modified

Example:

```
UPDATE Sailors SET rating = rating + 1 WHERE age = 20
```

If five (5) sailors match the condition:

- A statement level trigger will run once
- A row level trigger will run five times

Trigger: PL/SQL Examples

- Count the number of under 18 students added per INSERT statement

```
CREATE TRIGGER init_count
BEFORE INSERT ON Students
DECLARE count INTEGER;
BEGIN
count := 0;
END
```

Event

Action

```
CREATE TRIGGER incr_count
AFTER INSERT ON Students
WHEN (new.age < 18)
FOR EACH ROW
BEGIN /* Action */
count := count + 1;
END
```

Condition ('new' is the just inserted tuple)

Perform the action for each row that matches

Trigger: SQL Example

- For each INSERT statement, record the number of under 18 students entered in the StatisticsTable

```
CREATE TRIGGER set_count
AFTER INSERT ON Students
REFERENCING NEW TABLE AS InsertedTuples
FOR EACH STATEMENT
INSERT INTO StatisticsTable (ModifiedTable,
ModificationType, Count)
SELECT 'Students', 'Insert', COUNT(*)
FROM InsertedTuples I
WHERE I.age < 18
```

Creates a virtual table with just the newly entered rows

Default

WHEN would go here

Trigger: SQL Example

- When a bank account goes below zero, we allow "minus" by making an overdraft into a loan

```
CREATE TRIGGER overdraft_trigger
AFTER UPDATE ON Account
REFERENCING NEW ROW AS nrow
FOR EACH ROW
WHEN nrow.balance < 0
BEGIN ATOMIC
  INSERT INTO Borrower VALUES
  (SELECT customer_name, account_number
   FROM Depositor D
   WHERE nrow.account_number = D.account_number);
  INSERT INTO Loan VALUES (nrow.account_number,
  nrow.branch_name, 0 - nrow.balance);
  UPDATE Account A SET A.balance = 0
  WHERE A.account_number = nrow.account_number;
END
```

January 5, 2010

ISE 322: Database Systems

7

Trigger: MS SQL Server Example

- For each INSERT statement, record the number of under 18 students entered in the StatisticsTable

```
CREATE TRIGGER set_count
ON Sailors AFTER INSERT
AS
INSERT INTO StatisticsTable (modifiedTable,
modificationType, count)
SELECT 'Sailors', 'Insert', COUNT(*)
FROM inserted I
WHERE I.age < 18
```

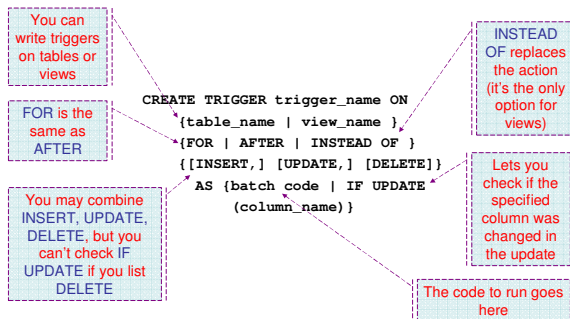
Special table with the new rows

January 5, 2010

ISE 322: Database Systems

8

Trigger Format: MS SQL Server



January 5, 2010

ISE 322: Database Systems

9

Trigger: MS SQL Server Example

- Record in the statistics table whenever we raise the ratings of sailors

```
CREATE TRIGGER update_sailors
ON Sailors AFTER UPDATE
AS IF UPDATE(rating)
INSERT INTO StatisticsTable (modifiedTable,
modificationType, count)
SELECT 'Sailors', 'Update', COUNT(*)
FROM inserted I, deleted D
WHERE D.sid = I.sid and I.rating > D.rating
```

What happens if we run UPDATE Sailors SET rating = rating - 1 ?

Only if rating is modified

Records the old rows

January 5, 2010

ISE 322: Database Systems

10

Using Triggers

- One trigger can lead to another
 - Chains
 - Can trigger itself
- The order that they run in is may not be fixed
- MS SQL allows you to set the order of execution for triggers on the same table
 - By default it's the order they were created
 - You can set the actual order using the built in function:

```
EXEC sp_settriggerorder
@triggername='trigger_name',
@order='first', @stmttype='update'
```

LAST | NONE

Action

January 5, 2010

ISE 322: Database Systems

11

Constraints vs Triggers

- Many things with triggers can be done with foreign keys and constraints
- When to use triggers?
 - If a constraint can be fully done with foreign keys cascading/update, unique, use them
 - If the constraint is only on one table, CHECK
 - If the constraint is too complex and inter-table, use ASSERTIONS
- Use triggers if you need to *do something*, not just allow/forbid/update
 - Complex calculations
 - Editing operations
 - Logging

January 5, 2010

ISE 322: Database Systems

12

Other uses of Triggers

- Check state for updates which are interesting:
 - Ex. Announce when you have 100 sales by a salesman
 - Ex. Notify when a stock of widgets is low
- Inform users or administrators of events
 - Announce when the day's data got entered into the database
- Databases often use triggers for internal monitoring, auditing, and enforcing policy rules

January 5, 2010

ISE 322: Database Systems

13

Enforcing Policy Rules

Audit raises in ratings greater than 2

```

CREATE TRIGGER check_rating_raise
ON Sailors AFTER UPDATE
AS IF UPDATE (rating)
BEGIN

    DECLARE @old_rat INT
    DECLARE @new_rat INT
    DECLARE @sid INT

    SELECT @old_rat = (SELECT rating FROM deleted)
    SELECT @new_rat = (SELECT rating FROM inserted)
    SELECT @sid = (SELECT sid FROM deleted)

    IF (@new_rat - @old_rat) > 2
        INSERT INTO Auditing (sid, oldrating,
            newrating, system_user_name,
            datetime_changed)
        VALUES (@sid, @old_rat, @new_rat,
            USER_NAME(), GETDATE())
    END
    
```

CREATE TABLE Auditing
(sid INTEGER,
oldrating INTEGER,
newrating INTEGER,
system_user_name char(30),
datetime_changed datetime,
PRIMARY KEY (sid,
datetime_changed))

Now we can only update one rating at a time

January 5, 2010

ISE 322: Database Systems

14

Enforcing Policy Rules

Audit raises in ratings greater than 2

```

CREATE TRIGGER check_rating_raise
ON Sailors AFTER UPDATE
AS IF UPDATE (rating)
BEGIN

    DECLARE @old_rat INT
    DECLARE @new_rat INT
    DECLARE @sid INT

    SELECT @old_rat = (SELECT rating FROM deleted)
    SELECT @new_rat = (SELECT rating FROM inserted)
    SELECT @sid = (SELECT sid FROM deleted)

    IF (@new_rat - @old_rat) > 2
        INSERT INTO Auditing (system_user_name,
            sid, oldrating, newrating, datetime_changed)
        VALUES (@sid, @old_rat, @new_rat,
            USER_NAME(), GETDATE())
    END
    
```

Another Option:

```

CREATE TRIGGER check_rating_raise2
ON Sailors AFTER UPDATE
AS IF UPDATE (rating)
BEGIN

    INSERT INTO Auditing (sid, oldrating,
        newrating, system_user_name,
        datetime_changed)

        SELECT d.sid, d.rating, i.rating,
            USER_NAME(), GETDATE()
        FROM inserted i, deleted d
        WHERE i.sid = d.sid
        AND i.rating > d.rating + 2
    END
    
```

January 5, 2010

ISE 322: Database Systems

15

Enforcing Policy Rules

Forbid any raises in ratings greater than 3

```

CREATE TRIGGER checkRaises
ON Sailors AFTER UPDATE
AS IF UPDATE(rating)
BEGIN
    DECLARE @old_rating INT
    DECLARE @new_rating INT
    SELECT @old_rating = (SELECT rating FROM deleted)
    SELECT @new_rating = (SELECT rating FROM inserted)
    IF @new_rating - @old_rating > 3
        BEGIN
            PRINT 'Rating Raise Exceeds Policy Limits'
            ROLLBACK TRANSACTION
        END
    ELSE
        BEGIN
            PRINT 'Rating Raise Approved'
        END
    END;
    
```

January 5, 2010

ISE 322: Database Systems

16

So Far

- Triggers and Active Databases
- Stored Procedures
- Remote invocation

January 5, 2010

ISE 322: Database Systems

17

Stored Procedures

- Include logic in the database, functions or procedures
- Why use them?
 - They are precompiled (no typos, may run faster)
 - Accessible to users
 - Can take inputs and outputs
 - Extra layer of indirection

January 5, 2010

ISE 322: Database Systems

18

Stored Procedures

- The return type of a stored procedure (or query sent via remote connection) is a **cursor** (can represent a temporary table stored in memory) or can be a **single value**
- Parameter types (for SQL:1999)
 - IN
 - OUT
 - INOUT

January 5, 2010

ISE 322: Database Systems

19

Example: SQL Stored Procedures

- Shows the number of orders made

```
CREATE PROCEDURE ShowNamesOfOrders ()
SELECT C.cis, C.cname, COUNT(*)
FROM Customers C, Orders O
WHERE C.cid = O.cid
GROUP BY C.cid, C.cname
```
- Update a book inventory when books are added

```
CREATE PROCEDURE AddInventory (
IN book_isbn VARCHAR(10),
IN addedQty INTEGER)
UPDATE Books B
SET B.qty_in_stock = B.qty_in_stock + addedQty
WHERE B.isbn = book_isbn
```

January 5, 2010

ISE 322: Database Systems

20

Stored Procedures: MS SQL Server

- Give every Sailor a raise in rating

```
CREATE PROCEDURE rating_raise (@raise_amount int)
AS
UPDATE Sailors
SET rating = rating + @raise_amount;
```

Parameters
begin with @

- You can run the procedure using

```
EXEC rating_raise 1
EXEC rating_raise @raise_amount = 1
```

January 5, 2010

ISE 322: Database Systems

21

Stored Procedures in MS SQL Server

- General format

Put a value here for a 'default' parameter value in case one is not provided

```
CREATE PROCEDURE [owner.]procedure_name [(@parameter1
datatype1 [=DEFAULT], [(@parameter2 datatype2 [= DEFAULT], ...
.))]
AS {INSERT | UPDATE | DELETE} table_name
{code to execute for the procedure}
```

January 5, 2010

ISE 322: Database Systems

22

Stored Procedure: MS SQL Server

- Give a single sailor a raise in rating. Also print the sailor's rating before and after the change

```
CREATE PROCEDURE individual_raise (@sid int, @raise_amount int)
AS
SELECT * FROM Sailors WHERE sid = @sid
UPDATE Sailors SET rating = rating + @raise_amount
WHERE sid = @sid
SELECT * FROM Sailors WHERE sid = @sid;
```

- Run it with:

```
EXEC individual_raise 14, 1
EXEC individual_raise @sid = 14, @raise_amount = 1
```

January 5, 2010

ISE 322: Database Systems

23

Stored Procedure: MS SQL Server

- Give a single sailor a raise in rating and age. Also print the sailor's rating and age before and after the change. The default raise is 1.

```
CREATE PROCEDURE age_rating (@sid int, @raise_amount int = 1,
@age_amount int)
AS
SELECT * FROM Sailors WHERE sid = @sid
UPDATE Sailors
SET rating = rating + @raise_amount, age = age + @age_amount
WHERE sid = @sid
SELECT * FROM Sailors WHERE sid = @sid;
```

- Run it with:

```
EXEC age_rating 14, 1, 1
EXEC age_rating @sid = 14, @raise_amount = 1, @age_amount = 1
EXEC age_rating @sid = 14, @age_amount = 1
```

January 5, 2010

ISE 322: Database Systems

24

Remote Invocation

- Can be invoked via remote queries
 - Build a connection (connection string)
 - Create the list of the parameters
 - Send the name of the stored procedure and the parameters
 - Run the stored procedure
 - Examine the results (DataSet or DataTable in C#)

Conclusion

- Triggers and Active Databases
- Stored Procedures
- Remote invocation