TRIGGER

Nội dung

- Giới thiệu về Trigger
- Các công dụng của Trigger
- Insert Trigger
- Update Trigger
- Delete Trigger
- Instead of Trigger

Giới thiệu

- Là một loại Stored Procedure đặc biệt
- Được thực hiện một cách tự động khi có sự kiện hiệu chỉnh dữ liệu xảy ra (Update, Insert, Delete)
- Không thể gọi thực hiện trực tiếp
- SQL Server xem trigger và lời gọi trigger là một giao tác. Do đó không cần chỉ định Begin Transaction nhưng có thể dùng Rollback bên trong một trigger

Công dụng

- Thực hiện các thay đổi dây chuyền trong các bảng có liên quan với nhau của CSDL
- Đảm bảo được nhiều ràng buộc toàn vẹn hơn so với việc dùng các CONSTRAINT để kiểm tra các ràng buộc
- Có thể định nghĩa các thông báo lỗi riêng của người dùng
- So sánh được các trạng thái trước và sau khi có sự thay đổi trên bảng

Inserted table và deleted table

- Khi một trigger được kích hoạt thì dữ liệu mới được insert hay mới được thay đổi sẽ được chứa trong inserted table còn dữ liệu mới delete được chứa trong deleted table
- Inserted table và deleted table là 2 bảng tạm, được chứa trong bộ nhớ và chỉ có giá trị bên trong trigger
- Thông tin từ 2 bảng này được dùng để kiểm tra dữ liệu mới thay đổi có hợp lệ không

Syntax

```
CREATE TRIGGER trigger_name
ON {table_name | view_name}
[WITH dml_trigger_option [,…]]
{FOR | AFTER | INSTEAD OF}
{ [INSERT] [,] [UPDATE] [,] [DELETE]}
 {AS  sql_statement  }
```

Syntax

- FOR/AFTER trigger
  - Chạy sau các hành động kiểm tra dữ liệu (rule, constraint):
  - Thực hiện sau khi hành động INSERT, UPDATE hay DELETE đã được thực hiện
  - Chỉ định nghĩa được AFTER trigger trên Table
- INSTEAD OF trigger
  - Thay thế cho hành động kích hoạt trigger (do vậy sẽ chạy trước các hành động kiểm tra dữ liệu (rule, constraint)
  - Có thể định nghĩa được trên một view (với một hoặc nhiều table cơ sở)
  - Không dùng được trên View có WITH CHECK OPTION

Ví dụ tạo mới trigger

```
CREATE TRIGGER EMPLOYEE_DELETE
ON EMPLOYEE FOR DELETE
AS
IF      EXISTS (SELECT D.EMP_NO
                  FROM works_on W JOIN deleted D
                 ON W.emp_no = D.emp_no)
BEGIN
        RAISERROR ('CANNOT DELETE!',10,1)
        ROLLBACK
END
```

https://docs.microsoft.com/en-us/sql/t-sql/language-elements/raiserror-transact-sql#syntax

ALTER trigger

```
ALTER TRIGGER EMPLOYEE_DELETE
ON EMPLOYEE FOR DELETE
AS
IF      EXISTS (SELECT D.EMP_NO
                  FROM works_on W JOIN deleted D
                 ON W.emp_no = D.emp_no)
BEGIN
        RAISERROR ('KHONG THE XOA MAU TIN NAY!',10,1)
        ROLLBACK
END
```

DROP trigger

- DROP TRIGGER Trigger_Name

Enable hoặc disable một trigger

```
ALTER TABLE Tablename
ENABLE|DISABLE TRIGGER {ALL|TriggerName}
```

Insert Trigger

Một câu lệnh INSERT vào một Table có định nghĩa INSERT Trigger

```
INSERT loan VALUES
(603, 4, 11, 123, GETDATE(), (GETDATE() + 30))
```

loan

isbn	copy_no	title_no	mem_no	outdate	duedate
1	1	1001	1001	02/13/91	02/27/91
603	4	11	123	02/15/91	03/17/91
4	2	1004	1002	02/14/91	02/28/91
3	1	1002	1003	02/14/91	02/28/91

Câu lệnh INSERT được ghi nhận (Logged)

inserted

603	4	11	123	02/15/91	03/17/91

Insert Trigger

```
CREATE TRIGGER loan_insert
ON loan
FOR INSERT
AS
UPDATE c SET on_loan = 'Y'
FROM copy c INNER JOIN inserted I
ON c.isbn = isbn and c.copy_no = i.copy_no
```

loan

isbn	copy_no	title_no	mem_no	outdate	duedate
1	1	1001	1001	02/13/91	02/27/91
603	4	11	123	02/15/91	03/17/91
4	2	1004	1002	02/14/91	0[illegible]/28/91
3	1	1002	1003	02/14/91	02[illegible]

copy

isbn	copy_no	title_no	on_loan
1	1	1001	Y
603	4	11	Y
4	2	1004	N
3	1	1002	N

Delete Trigger

Một câu lệnh Delete vào một Table có định nghĩa Delete Trigger

```
DELETE loan
WHERE isbn = 4
AND copy_no = 1
```

loan

isbn	*copy_no*	*title_no*	*mem_no*	*outdate*	*duedate*
1	1	1001	1001	02/13/91	02/27/91
4	2	1004	1002	02/14/91	02/28/91
3	1	1002	1003	02/14/91	02/28/91

Câu lệnh Delete được ghi nhận (Logged)

Deleted

4	1	1004	1001	02/13/91	02/27/91

Delete Trigger

```
USE library
CREATE TRIGGER loan_delete
    ON loan
    FOR DELETE
AS
    UPDATE c SET on_loan = 'N'
      FROM copy c INNER JOIN deleted d
      ON c.isbn = d.isbn AND c.copy_no = d.copy.no
```

copy

isbn	copy_no	title_no	on_loan
1	1	1001	Y
4	1	1004	N
4	2	1004	N
3	1	1002	N

Update Trigger

Một câu lệnh Update vào một Table có định nghĩa Update Trigger

```
UPDATE member
SET member_no = 10021
WHERE member_no = 1234
```

member

member_no	*lastname*	*firstname*	*middleinitial*	*photograph*
10020	Anderson	Andrew	A	~~~
1234	Barr	Andrew	R	~~~
10022	Barr	Bill	NULL	~~~
10023	Anderson	Bill	B	

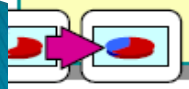
Câu lệnh Update được ghi nhận như là hai câu Insert và Delete

inserted

10021	Barr	Andrew	R	~~~

deleted

1234	Barr	Andrew	R	~~~



Update Trigger

Hàm update(col) : Kiểm tra cột nào có thay đổi

```
USE library
GO
CREATE TRIGGER member_update
 ON member
 FOR UPDATE
AS
IF UPDATE (member_no)
BEGIN
  RAISERROR ('Transaction cannot be processed.\
  ***** Member number cannot be modified.', 10, 1)
  ROLLBACK TRANSACTION
```

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

member

member_no	lastname	firstname	middleinitial	photograph
10020	Anderson	Andrew	A	~~~
1234	Barr	Andrew	R	~~~
10022	Barr	Bill	NULL	~~~
10023	Anderson	Bill	B	

Instead of Trigger

- Ví dụ

```
create trigger depart_insert_instead
on department instead of insert
as
        print 'Cant not insert!'
```

Các ứng dụng của AFTER trigger

- Tạo vết kiểm tra các hoạt động trong các bảng
- Thực thi các quy tắc nghiệp vụ
- Thực thi các ràng buộc toàn vẹn

Tạo vết kiểm tra các hoạt động trong các bảng

- Tạo bảng audit_budget dùng lưu vết hoạt động bảng project

```
USE sample;
GO
CREATE TABLE audit_budget   (project_no CHAR(4) NULL,
                                    user_name CHAR(16) NULL,
                                    date DATETIME NULL,
                                    budget_old FLOAT NULL,
                                    budget_new FLOAT NULL);
GO
```

Tạo vết kiểm tra các hoạt động trong bảng

- Tạo TRIGGER modify_budget

```
CREATE TRIGGER modify_budget    ON project
AFTER UPDATE
AS IF UPDATE(budget)
        BEGIN
                DECLARE @budget_old FLOAT
                DECLARE @budget_new FLOAT
                DECLARE @project_number CHAR(4)
                SELECT @budget_old = (SELECT budget FROM deleted)
                SELECT @budget_new = (SELECT budget FROM inserted)
                SELECT @project_number = (SELECT project_no FROM
                deleted)
                INSERT INTO audit_budget VALUES
                (@project_number,USER_NAME(),GETDATE(),@budget_old,
                @budget_new)
        END
```

Thực thi các quy tắc nghiệp vụ

```
CREATE TRIGGER total_budget     ON project
AFTER UPDATE
AS IF UPDATE (budget)
BEGIN
        DECLARE @sum_old1 FLOAT
        DECLARE @sum_old2 FLOAT
        DECLARE @sum_new FLOAT
        SELECT @sum_new = (SELECT SUM(budget) FROM inserted)
        SELECT @sum_old1 = (SELECT SUM(p.budget)
                                FROM project p
                                WHERE p.project_no NOT IN (SELECT
                                d.project_no FROM deleted d))
        SELECT @sum_old2 = (SELECT SUM(budget) FROM deleted)
        IF @sum_new > (@sum_old1 + @sum_old2)  *1.5
                BEGIN
                        PRINT 'No modification of budgets'
                        ROLLBACK TRANSACTION
                END
                ELSE
                        PRINT 'The modification of budgets executed'

END
```

Thực thi các ràng buộc toàn vẹn

```
CREATE TRIGGER workson_integrity    ON works_on
AFTER INSERT, UPDATE
AS IF UPDATE(emp_no)
            BEGIN
            IF (SELECT employee.emp_no
                FROM employee, inserted
                WHERE employee.emp_no = inserted.emp_no) IS NULL
                        BEGIN
                                ROLLBACK TRANSACTION
                                PRINT 'No insertion/modification of the row'
                        END
                        ELSE
                                PRINT 'The row inserted/modified'
            END
```

Thực thi các ràng buộc toàn vẹn

```
CREATE TRIGGER refint_workson2    ON employee
AFTER DELETE, UPDATE
AS IF UPDATE (emp_no)
            BEGIN
                        IF (SELECT COUNT(*)  FROM WORKS_ON, deleted
                              WHERE works_on.emp_no = deleted.emp_no) > 0
                           BEGIN
                                    ROLLBACK TRANSACTION
                                    PRINT 'No modification/deletion of the row'
                           END
                                    ELSE PRINT 'The row is deleted/modified'
                        END
```

Bài tập

Sử dụng database Sample.

1. Viết trigger **Project_delete** không cho phép xóa một dự án khi có nhân viên đang tham gia vào dự án
2. Giả sử bảng Department có thêm thuộc tính **emp_total** là tổng số nhân viên của phòng ban. Viết trigger **employee_insert** tự động tăng số lượng nhân viên của phòng ban khi thêm mới một nhân viên