TRIGGER

Nội dung

- Khái niệm
- DML Trigger
- DDL Trigger

Khái niệm

- Trigger là một đối tượng dữ liệu đặc biệt tương tự nội thủ tục mà thực thi trong quá trình thao tác dữ liệu (DML trigger) hoặc trong các hành động với mô hình dữ liệu như tạo bảng, … (DDL trigger).
- DML trigger tác động lên một bảng trong quá trình Insert, Update, Delete
- DDL trigger xảy ra khi các hành động tác động lên database hoặc server: Create, alter, drop, login

DML TRIGGER

Bẫy sự kiện (trigger)

- Trigger được phát sinh sau những hành vi thêm mới hay thay đổi, xóa trên bảng.
  - Có thể hủy các cập nhật trên dữ liệu
- Trigger được phát sinh để thay thế những hành vi thêm, đổi, xóa.
- Trigger lưu giữ tách rời giá trị mới được đưa vào và giá trị cũ được xóa bỏ.
  - Dùng bảng tạm **Inserted** và **Deleted**

Trigger (tt)

- Là một loại sp đặc biệt gắn với 1 table / view
- Tự động thi hành khi có một sự kiện xảy ra với bảng, view như: insert, update, delete 1 / 1 số các row
- Có 2 loại trigger
  - After trigger
  - Instead of trigger
- Trigger ko trả về giá trị, không có tham số

Trigger (tt)

- Trigger thường dùng trong các trường hợp:
  - Ràng buộc toàn vẹn dữ liệu
  - Kiểm soát dữ liệu hiện tại khi có thay đổi đến giá trị trong mẩu tin của bảng
  - Kiểm tra dữ liệu nhập vào phù hợp với mối liên hệ dữ liệu giữa các bảng với nhau
  - Kiểm chứng khi xóa mẩu tin trong bảng
  - Lưu ý: cần tạo khóa cho bảng khi dùng trigger

Tạo trigger

- CREATE TRIGGER name ON table | view

[FOR | AFTER | INSTEAD OF]

[INSERT, UPDATE, DELETE]

AS

BEGIN ... END

- Xóa và thay đổi

  Alter | Drop trigger tên_trigger

Trigger (tt)

```
CREATE TRIGGER tg4 ON [dbo].[T1]
FOR UPDATE
AS
begin
    declare @id_del int
    select @id_del = id1 from deleted
    raiserror( 'ban da xoa ban ghi co  id = %d', 16, 1,
    @id_del)
end
```

Trigger (tt)

Sự kiện lồng nhau được tối đa 32

- Thay đổi thông số cho phép lồng nhau

```
alter database tendatabase
    set  recursive_triggers  { on | off }
```

- Thiết lập giới hạn lồng nhau

```
exec sp_configure 'Nested Triggers' n
```

Trigger (tt)

- Tạo trigger cho bảng DSHS (mahs, tenhs, lop) thỏa mãn điều kiện một lớp không quá 40 người.

```
Create trigger lop_40hs on dshs for insert,
update As
Begin
declare @malop nvarchar(4), @ts int
select @malop = lop from inserted
select @ts = count(mahs) from DSHS
where lop=@malop
if (@ts > 40)
rollback transaction
end
```

Trigger (tt)

- Tạo trigger cho bảng DSHS, mỗi khi thực hiện hành động xóa học sinh thì không xóa bản ghi mà cập nhật trường ghi chú là chuyển trường

```
create trigger xoahs on dshs instead of delete As
Begin
declare @mahs nvarchar(5)
select @mahs=mahs from deleted
update dshs set GHICHU=N'Chuyển trường' where
mahs=@mahs
end
```

Bảng Inserted và Deleted

- Bảng Inserted: Lưu các thông tin được bổ sung vào CSDL
- Bảng Deleted: Lưu các thông tin loại bỏ khỏi CSDL

Hành động	**Bảng Inserted chứa**	**Bảng Deleted chứa**
Insert	Các bản ghi được thêm	Rỗng
Update	Các bản ghi sau khi sửa	Các bản ghi trước khi được sửa
Delete	Rỗng	Các bạn ghi đã xóa

After trigger và Instead of Trigger

- After trigger: Các câu lệnh trong trigger thực hiện khi có hành động của sự kiện (Insert, Update, Delete) khai báo
- Instead of trigger: Các câu lệnh trong trigger thực hiện thay cho hành động Insert, Update, Delete

Trigger (tt)

- Giả sử có hai bảng HoaDon(SoHD, NgayHD) và ChiTietHD (SoHD, MaH, SLBan, DonGia). Tạo Trigger cho bảng ChiTietHD khi them vào bản ghi thì SoHD đó đã thêm vào bản HoaDon hay chưa

```
CREATE TRIGGER trgIns
ON ChiTietHD
FOR INSERT
AS
        IF NOT EXISTS (SELECT “True” FROM INSERTED WHERE
                Inserted.SoHD=HoaDon.SoHD)
BEGIN
        RAISERROR (60000, 16,1,’SoHD’, ‘ChiTietHD’, ‘SoHD’,
                ‘HoaDon’)
        ROLLBACK TRAN
END
```

Transaction

- Nhóm nhỏ các phát biểu: hoặc thực thi toàn bộ, hoặc không làm gì cả
- Begin: bắt đầu một tracsaction
  - **Begin tran**saction [<tên transaction|@biến transaction>]
- Commit: xác định kết thúc hay hoàn tất
  - **commit tran**saction [<tên transaction|@biến transaction>]

Transaction (tt)

- Rollback: các thao tác bị hủy bỏ từ begin hoặc điểm đánh dấu
  - **Rollback transaction** [<tên transaction| điểm đánh dấu| @biến transaction>]
- Save transaction
  - **Save transaction** [<điểm đánh dấu>]
- Biến @@trancount: cho biết số transaction hiện đang thực hiện (chưa được kết thúc với rollback hay commit) trong connection hiện hành

Transaction (tt)

```
SET XACT_ABORT ON
BEGIN TRAN
BEGIN TRY
-- lệnh 1
-- lệnh 2
-- ...
COMMIT
END TRY
BEGIN CATCH
      ROLLBACK
      DECLARE @ErrorMessage VARCHAR(2000)
      SELECT @ErrorMessage = 'Lỗi: '
            +ERROR_MESSAGE()
      RAISERROR(@ErrorMessage, 16, 1)
END CATCH
```

Transaction (tt)-ví dụ

```
BEGIN
DECLARE @Account_Id_A integer = 1;
DECLARE @Account_Id_B integer = 2;
DECLARE @Amount float = 10;
 BEGIN TRAN;
  BEGIN TRY
      UPDATE Account SET AVAIL_BALANCE = AVAIL_BALANCE -
       @Amount WHERE Account_Id = @Account_Id_A;
      UPDATE Account SET AVAIL_BALANCE = AVAIL_BALANCE +
       @Amount WHERE Account_Id = @Account_Id_B;
      COMMIT TRAN;
  END TRY
  BEGIN CATCH
       PRINT 'Error: ' + ERROR_MESSAGE();
       ROLLBACK TRAN;
  END CATCH;
END;
```

LOCK

- Shared Locks (khóa chia sẻ): cho phép đọc dữ liệu, không cho phép sự thay đổi nào của tài nguyên
- Exclusive Locks (Khóa độc quyền): ngăn hai người cùng thực hiện đọc, cập nhật, xóa, thêm bản ghi trong cùng một thời gian
- Update Lock: khi chưa cần cập nhật thì ở chế độ Shared Locks. Khi lênh Update thực sự thực thi thì ở chế dộ Exclusive Locks
- Intent Locks: Thông báo về việc sẽ khóa dữ liệu

Lock (tt) – Các ví dụ

- **Shared Locks**

```
BEGIN TRAN
USE AdventureWorks

SELECT * FROM Person.Address WITH (HOLDLOCK)
WHERE AddressId = 2

SELECT resource_type, request_mode,
resource_description FROM sys.dm_tran_locks
WHERE resource_type <> 'DATABASE'

ROLLBACK
```

Lock (tt) – Các ví dụ

- **Update Locks**

```
BEGIN TRAN
USE AdventureWorks

SELECT * FROM Person.Address WITH (UPDLOCK)
WHERE AddressId < 2

SELECT resource_type, request_mode,
resource_description FROM sys.dm_tran_locks
WHERE resource_type <> 'DATABASE'

ROLLBACK
```

Lock (tt) – Các ví dụ

- **Exclusive locks (X)**

```
BEGIN TRAN
USE AdventureWorks

UPDATE Person.Address SET AddressLine2 = 'Test
Address 2' WHERE AddressId = 5

SELECT resource_type, request_mode,
resource_description FROM sys.dm_tran_locks WHERE
resource_type <> 'DATABASE'

ROLLBACK
```

Lock (tt) – Các ví dụ

- **Intent locks (I)**

```
BEGIN TRAN
USE AdventureWorks

UPDATE TOP(5) Person.Address SET
AddressLine2 = 'Test Address 2' WHERE
PostalCode = '98011'

SELECT resource_type, request_mode,
resource_description FROM sys.dm_tran_locks
WHERE resource_type <> 'DATABASE'

ROLLBACK
```

Lock (tt)- sys.dm_tran_locks

```
SELECT dm_tran_locks.request_session_id,
    dm_tran_locks.resource_database_id,
    DB_NAME(dm_tran_locks.resource_database_id) AS dbname,
    CASE WHEN resource_type = 'OBJECT'
        THEN OBJECT_NAME(dm_tran_locks.resource_associated_entity_id)
        ELSE OBJECT_NAME(partitions.OBJECT_ID)
    END AS ObjectName,
    partitions.index_id, indexes.name AS index_name,
    dm_tran_locks.resource_type,
    dm_tran_locks.resource_description,
    dm_tran_locks.resource_associated_entity_id,
    dm_tran_locks.request_mode,
    dm_tran_locks.request_status
FROM sys.dm_tran_locks
LEFT JOIN sys.partitions ON partitions.hobt_id =
        dm_tran_locks.resource_associated_entity_id
LEFT JOIN sys.indexes ON indexes.OBJECT_ID = partitions.OBJECT_ID
        AND indexes.index_id = partitions.index_id
WHERE resource_associated_entity_id > 0
        AND resource_database_id = DB_ID()
ORDER BY request_session_id, resource_associated_entity_id
```

Thực thi lệnh SQL

- Các câu lệnh có thể được thực thi như là các câu lệnh đơn hoặc như một lô
- Xử lý lô
  - Một lô là một tập hợp của một hoặc nhiều câu lệnh SQL được gửi cùng một thời điểm từ một ứng dụng đến SQL Server để thực thi
  - Các câu lệnh này được biên dịch thành một đơn vị thực thi và được gọi là “execution plan”
  - Các câu lệnh trong “execution plan” được thực hiện cùng một lúc
- Scripts
  - Các câu lệnh SQL có thể được thực thi trong script bằng cách lưu trên tập tin. Phần mở rộng của file thường lưu dưới dạng *.sql. Tập tin sẽ được đọc khi được yêu cầu để thực thi.

DDL TRIGGER

DDL Trigger

- Kiểm tra hành động có xảy ra hay không trong một đối tượng trong SQL server hoặc trên Database hoặc trên server

Các sự kiện trên database

CREATE_TABLE	ALTER_TABLE	DROP_TABLE
CREATE_VIEW	ALTER_VIEW	DROP_VIEW
CREATE_FUNCTION	ALTER_FUNCTION	DROP_FUNCTION
CREATE_PROCEDURE	ALTER_PROCEDURE	DROP_PROCEDURE
CREATE_TRIGGER	ALTER_TRIGGER	DROP_TRIGGER
CREATE_EVENT_NOTIFICATION		DROP_EVENT_NOTIFICATION
CREATE_INDEX	ALTER_INDEX	DROP_INDEX
CREATE_STATISTICS	UPDATE_STATISTICS	DROP STATISTICS
CREATE_USER	ALTER_USER	DROP_USER
CREATE_ROLE	ALTER_ROLE	DROP_ROLE
CREATE_APPLICATION_ROLE	ALTER_APPLICATION_ROLE	DROP_APPLICATION_ROLE
CREATE_SCHEMA	ALTER_SCHEMA	DROP_SCHEMA
GRANT_DATABASE	DENY_DATABASE	REVOKE_DATABASE

Các sự kiện trên server

CREATE_LOGIN	ALTER_LOGIN	DROP_LOGIN
CREATE_HTTP_ENDPOINT		DROP_HTTP_ENDPOINT
CREATE_CERT	ALTER_CERT	DROP_CERT
GRANT_SERVER_ACCESS	DENY_SERVER_ACCESS	REVOKE_SERVER_ACCESS

- CREATE TRIGGER trigger_name
  ON {ALL SERVER|DATABASE}
  [WITH ENCRYPTION]
  {
  {{FOR |AFTER } {event_type,...}
  AS

  sql_statements}}

- DROP TRIGGER trigger_name ON {DATABASE|ALL SERVER}

- CREATE TRIGGER trgSprocs
  ON DATABASE
  FOR CREATE_PROCEDURE, ALTER_PROCEDURE, DROP_PROCEDURE
  AS
  IF DATEPART(hh,GETDATE()) > 8 AND DATEPART(hh,GETDATE()) < 18
  BEGIN
  DECLARE @Message nvarchar(max)
  SELECT @Message =
  'Completing work during core hours. Trying to release - '
  + EVENTDATA().value
  ('(/EVENT_INSTANCE/TSQLCommand/CommandText)[1]','nvarchar(max)')
  RAISERROR (@Message, 16, 1)
  ROLLBACK
  EXEC msdb.dbo.sp_send_dbmail
  @profile_name = 'ApressFinancialDBMA',
  @recipients = 'robin@fat-belly.com',
  @body = 'A stored procedure change',
  @subject = 'A stored procedure change has been initiated and rolled back during core hours'

  END

Tắt, bật trigger

- Tắt trigger:

```
DISABLE TRIGGER [schema_name.][trigger_name] ON
[object_name | DATABASE | ALL SERVER]
```

- Tắt toàn bộ trigger trên 1 bảng:

```
DISABLE TRIGGER ALL ON table_name;
```

- Tắt toàn bộ trigger trên 1 database:

```
DISABLE TRIGGER ALL ON DATABASE;
```

- Bật trigger trên 1 database:

```
ENABLE TRIGGER { [ schema_name . ] trigger_name [
,...n ] | ALL } ON { object_name | DATABASE | ALL
SERVER } [ ; ]
```

Cảm ơn các em đã chú ý lắng nghe

Những em chưa chú ý vẫn được cảm ơn bình thường