TRƯỜNG ĐH CÔNG NGHỆ THÔNG TIN TP.HCM
Khoa Hệ Thống Thông Tin
Chương trình đào tạo: Kỹ sư Công nghệ thông tin

Bài Thực hành số 5

Tên môn học: ***Cơ sở dữ liệu***

I. Tóm tắt bài thực hành

1. Yêu cầu lý thuyết

Sinh viên đã được trang bị kiến thức:

- Khai báo các RBTV có bối cảnh 1 quan hệ (Null, Not Null, Rule, Check)

2. Nội dung

❖ **Ôn lại các kiến thức về RBTV có bối cảnh một quan hệ**

- Ôn lại cách khai báo đã được thực hành ở các bài thực hành trước.

❖ **Tìm hiểu các kiến thức về RBTV có bối cảnh trên nhiều quan hệ**

- Khai báo RBTV có bối cảnh trên nhiều quan hệ.
- Tìm hiểu về kiểu dữ liệu Cursor.

❖ **Thực hiện được các bài tập sau**

- Sử dụng các câu lệnh khai báo RBTV có bối cảnh nhiều quan hệ trên CSDL Quản lý bán hàng và Quản lý giáo vụ.

II. Trigger

1. Khái niệm

Trigger là một thủ tục đặc biệt, dùng khai báo ràng buộc dữ liệu cho một đối tượng table hoặc view và tự động thực hiện khi một trong 3 phát biểu Insert, Update, Delete thay đổi dữ liệu trên đối tượng đó. Trigger không thể được gọi trực tiếp như các hàm, thủ tục bình thường, nó cũng không có tham số và giá trị trả về.

Trigger chỉ được thực hiện khi phát biểu cập nhật đã thoả mãn các ràng buộc đã khai báo trên table. Lợi ích chính của trigger là chúng có thể chứa các xử lý phức tạp trên các table có dữ liệu liên quan với table đang cập nhật.

Trigger có thể chứa phát biểu `ROLLBACK TRAN` ngay cả khi không có phát biểu `BEGIN TRAN`. Trong trường hợp phát biểu `ROLLBACK TRANSACTION` bên trong một trigger được thực hiện:

- Nếu trigger này được kích hoạt bởi 1 phát biểu cập nhật từ bên trong một transaction khác, thì toàn bộ transaction đó bị hủy bỏ.
- Nếu trigger được kích hoạt bởi 1 phát biểu cập nhật từ bên trong một gói lệnh, thì sẽ hủy bỏ toàn bộ gói lệnh đó.

Dựa vào ứng dụng, có 3 loại trigger như sau: Insert trigger, Update trigger, Delete trigger.

2. Tạo trigger cho table

Cú pháp Tạo trigger:

```
CREATE TRIGGER <trigger_name> ON <table name>
   [WITH ENCRYPTION]
   AFTER | FOR DELETE, INSERT, UPDATE
AS <Các phát biểu T-sql>
```

- ✓ `trigger_name`: Tên trigger phải phân biệt.
- ✓ `ON <tablename>`: tên table mà trigger sẽ thực hiện. Không sử dụng cú pháp trigger này cho View.
- ✓ `WITH ENCRYPTION`: Mã hóa trigger, không cho xem và sửa đổi câu lệnh tạo trigger..
- ✓ `FOR DELETE, INSERT, UPDATE`: Dùng chỉ định những phát biểu cập nhật nào nào trên table sẽ kích hoạt trigger. Khi thực hiện trigger, SQL sẽ tạo các bảng tạm: INSERTED và DELETED.
  - Khi chèn mẫu tin mới vào table thì mẫu tin mới đó cũng lưu trong table INSERTED
  - Khi xóa mẫu tin trong table: Thì các mẫu tin bị xoá đó được di chuyển sang table DELETED.
  - Khi cập nhật mẫu tin trong table: thì table đó và table INSERTED đều chứa mẫu tin có nội dung mới, còn table DELETED chứa mẫu tin có nội dung cũ.

  Bạn không thể thay đổi dữ liệu trên các table DELETED VÀ INSERTED. Nhưng bạn có thể dùng 2 table này để xử lý các mẫu tin trên các table liên quan. Ngoài ra, trong trigger Insert và Update, bạn có thể thay đổi nội dung của các mẫu tin mới bằng lệnh Update trên table có trigger.
- ✓ `AS`: Từ khóa bắt đầu các hành động bên trong trigger. Trigger có thể chứa hầu hết các lệnh của T-SQL ngoại trừ một số lệnh sau:
  - Các lệnh `CREATE, ALTER,` và `DROP`.
  - `TRUNCATE TABLE`
  - `SELECT INTO` (vì câu lệnh này cũng tạo ra bảng)

3. Kích hoạt/Vô hiệu một trigger

```
ALTER TABLE table ENABLE | DISABLE TRIGGER ALL | trigger_name[,…n]
```

4. Hiệu chỉnh trigger

Bạn có thể thay đổi các lệnh cần thực hiện cũng như hành động cập nhật mà Trigger sẽ được gọi thực hiện.

```
ALTER TRIGGER trigger_name …
```

5. Xóa trigger

```
DROP TRIGGER {trigger} [,…n]
```

Nếu xóa một table thì tất cả trigger của nó cũng bị xóa.

III. Kiểu dữ liệu cursor

1. Khái niệm

Cursor là kiểu dữ liệu cho phép truy xuất đến trên từng mẫu tin trong tập kết quả trả về bởi câu lệnh Select. Ngoài ra, bạn có thể sử dụng các phát biểu Update hoặc Delete để cập nhật hay xóa mẫu tin hiện hành trên các bảng cơ sở của Select bằng mệnh đề `WHERE CURRENT OF <Tên Cursor>`.

2. Các thao tác chung trên Cursor

- ✓ Khai báo cursor: `DECLARE <cursor_name> CURSOR FOR <lệnh Select>`
- ✓ Mở cursor : `OPEN <cursor_name>`
  Sau lệnh mở cursor, con trỏ mẫu tin hiện hành nằm ở vùng BOF.
- ✓ Xử lý mẫu tin trên cursor:
  Di chuyển mẫu tin hiện hành: `FETCH NEXT FROM cursor_name`
  Sử dụng phát biểu Update hoặc Delete để cập nhật hay xóa mẫu tin hiện hành
- ✓ Đóng cursor: `CLOSE <tên cursor>`
- ✓ Hủy bỏ cursor: `DEALLOCATE <TÊN CURSOR>`

3. Khai báo Cursor

```
DECLARE <CursorName> CURSOR
    [ LOCAL | GLOBAL ] -- Phạm vi hoạt động
    [ FORWARD_ONLY | SCROLL ] -- Phương thức di chuyển
    [ STATIC | KEYSET | DYNAMIC ] -- Loại Cursor
    [ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ] -Xử lý đồng thời
    [ TYPE_WARNING]
    FOR <lệnh Select>
    [ FOR UPDATE [ OF ColumnName [,…n]] ]
```

4. Phạm vi hoạt động của Cursor

Mặc định, cursor có phạm vi Global trên kết nối mà nó đã được tạo. Nghĩa là, bạn có thể sử dụng cursor trên các gói lệnh thực hiện trên kết nối đó, trừ phi bạn đóng và giải phóng Cursor. Nếu bạn mở Cursor chưa đóng thì sẽ bị lỗi và có khi bị treo cho đến khi đóng kết nối. Với lý do đó, khi không sử dụng Cursor Global, bạn nên đóng và giải phóng Cursor.

Cursor Local có phạm vi hoạt động bên trong gói lệnh đã tạo nó. Và tự giải phóng khi kết thúc gói lệnh.

5. Phương Thức Di Chuyển Trên Cursor

Có 2 phương thức di chuyển:

- ✓ `FORWARD_ONLY`: là phương thức mặc định, chỉ cho phép di chuyển sang mẫu tin kế tiếp.

- ✓ SCROLL: Cho phép di chuyển lên xuống trong tập mẫu tin.

6. Các Loại Cursor

Có 3 loại Cursor:

- ✓ STATIC: có thuộc tính READ ONLY, do đó không thể cập nhật các bảng gốc thông qua Cursor này. Khi tạo Cursor Static, dữ liệu từ các bảng gốc sẽ được Copy sang một bảng tạm trong CSDL tempdb. Do đó, Nếu các table nguồn của Cursor bị thay đổi dữ liệu thì các dữ liệu không xuất hiện trên Cursor.
- ✓ DYNAMIC: Cho phép cập nhật dữ liệu trên các table nguồn (dùng mệnh đề WHERE CURRENT OF <tênCursor> trong các phát biểu UPDATE or DELETE), và tự động hiển thị tất cả những thay đổi từ table nguồn. Tuy nhiên, dữ liệu và thứ tự của các mẫu tin trong tập mẫu tin có thể bị thay đổi.
- ✓ KEYSET: Giống như cursor Dynamic. Nhưng nó chỉ được tạo khi bảng nguồn có khai báo khóa, nếu không thì SQL tự động chuyển sang loại STATIC. Khi tạo Cursor KEYSET, Tập các khóa của bảng nguồn được lưu trên một table của CSDL tempdb. Do đó, việc xóa mẫu tin hoặc thay đổi giá trị khóa trên các bảng nguồn không thông qua Cursor sẽ không phản hồi trên tập mẫu tin.

  Cursor kiểu STATIC, KEYSET, và DYNAMIC mặc định dùng phương thức SCROLL.

TYPE_WARNING: Gởi thông báo chú ý về client nếu Cursor thực hiện chuyển đổi ngầm định từ kiểu yêu cầu sang một kiểu khác.

7. Xử lý đồng thời

Trong môi trường nhiều người dùng cùng làm việc trên cùng tập dữ liệu, Làm thế nào để người dùng chắc chắn rằng những thay đổi của họ không bị thay đổi bởi người dùng khác? Phụ thuộc vào kiểu Cursor mà bạn đã sử dụng, bạn không thể nhận thấy được những thay đổi cho đến khi bạn đóng Cursor và mở lại nó.

Trừ phi sử dụng cursor trong một transaction, nếu không các table nguồn của cursor không tự động khóa dữ liệu. SQL Server cung cấp 4 chọn lựa cho phép ngăn cản việc sửa đổi mẫu tin cho tới khi thực hiện xong hoặc bằng cách khóa các table nguồn của cursor để bảo vệ các thay đổi của bạn.

- ✓ READ_ONLY: Dùng khi chỉ truy xuất dữ liệu mà không sửa đổi dữ liệu.
- ✓ SCROLL_LOCKS: Khoá các dòng đã được đọc vào Cursor đối với các User khác.

- ✓ `OPTIMISTIC WITH VALUES`: Chỉ khóa các giá trị mà bạn vừa thay đổi. Nếu người dùng khác thay đổi các giá trị đó sẽ nhận được thông báo lỗi.
- ✓ `OPTIMISTIC WITH ROW VERSIONING`: Khi muốn cả dòng được cập nhật, không chỉ một vài Fields trong nó.

8. Khai báo cột trong Cursor được phép cập nhật

```
UPDATE [OF column_name [,...n]]
```

- ✓ Nếu chỉ định `OF column_name [,...n]` chỉ những cột liệt kê mới được sửa đổi.
- ✓ Nếu chỉ định UPDATE mà không chỉ định danh sách cột, thì tất cả các cột đều có khả năng cập nhật trừ phi chỉ định READ_ONLY.

9. Truy xuất dữ liệu trên Cursor

```
FETCH [ NEXT | PRIOR | FIRST | LAST
      | ABSOLUTE { n | @nvar } | RELATIVE { n | @nvar } ]
  FROM [ GLOBAL ] cursor_name
  [ INTO @variable_name [ ,...n ] ]
```

- ✓ `NEXT`: Chuyển sang mẫu tin kế tiếp.
- ✓ `PRIOR`: Chuyển về mẫu tin trước đó.
- ✓ `FIRST`: Chuyển về mẫu tin đầu tiên.
- ✓ `LAST`: Chuyển đến mẫu tin cuối cùng.
- ✓ `ABSOLUTE {n | @nvar}`: Nếu n hoặc @nvar > 0, tìm đến dòng thứ n tính từ dòng đầu tiên đếm xuống trong tập mẫu tin. Nếu n hoặc @nvar < 0, tìm đến dòng thứ n tính từ dòng cuối cùng đếm lên. Nếu n hoặc @nvar = 0, chuyển đến vùng BOF và không có giá trị trả về. Hằng số n phải là số nguyên và biến @nvar phải thuộc kiểu smallint, tinyint, hoặc int. Không sử dụng phương thức ABSOLUTE cho kiểu DYNAMIC.
- ✓ `RELATIVE {n | @nvar}`: Nếu n hoặc @nvar > 0, chuyển xuống n dòng tính từ dòng kề dưới dòng hiện hành. Nếu n or @nvar < 0, Chuyển lên n dòng trước dòng hiện hành. Nếu n or @nvar = 0, trả về dòng hiện hành. o cursor_name: Tên cursor đang mở. Nếu tồn tại cursor cục bộ và cursor toàn cục có cùng tên thì tên cursor được sử dụng sẽ là cursor cục bộ nếu không có từ khóa GLOBAL.
- ✓ `INTO @varname[,...n]`: Danh sách biến cục bộ nhận dữ liệu tương ứng từ các cột trên mẫu tin hiện hành, theo thứ tự từ trái sang phải. Số biến phải bằng số cột đã liệt kê trong câu lệnh Select khi tạo Cursor. Kiểu dữ liệu của mỗi biến phải tương thích với kiểu dữ liệu của cột hoặc được hỗ trợ chuyển kiểu ngầm định theo kiểu của cột.

Kiểm tra kết quả của lệnh FETCH: Sử dụng hàm @@FETCH_STATUS sau lệnh FETCH. Hàm trả về một trong 3 giá trị:

0	Nếu lệnh FETCH chuyển đến 1 mẫu tin trong danh sách.
-1	Nếu lệnh FETCH chuyển đến vùng BOF hoặc EOF
-2	Nếu chuyển đến 1 dòng đã bị xóa trên Server (Keyset).

IV. Ví dụ

1. Ví dụ 1

Yêu cầu: (Bài tập Quản lý Bán hàng – Phần 1 – Câu 11 – Trang 3) Cài đặt trigger kiểm tra Ngày mua hàng (NGHD) của một khách hàng thành viên sẽ lớn hơn hoặc bằng ngày khách hàng đó đăng ký thành viên (NGDK).

Thực hiện:

```
CREATE TRIGGER trg_ins_hd ON HOADON
FOR INSERT
AS
BEGIN
    DECLARE @NgayHD smalldatetime, @MaKH char(4), @NgayDK smalldatetime
    -- Lay thong tin cua HOADON vua moi them vao
    SELECT @NgayHD = NGHD, @MaKH = MAKH
    FROM INSERTED
    -- Lay thong tin KHACHHANG tuong ung
    SELECT @NgayDK = NGDK
    FROM  KHACHHANG
    WHERE MAKH = @MaKH
    -- So sanh
    IF (@NgayHD < @NgayDK) -- Neu Ngay hoa don nho hon ngay dang ky thanh vien
    BEGIN
        PRINT 'LOI: NGAY HOA DON KHONG HOP LE!'
        ROLLBACK TRANSACTION
    END
    ELSE
    BEGIN
        PRINT 'THEM MOI MOT HOA DON THANH CONG!'
    END
END
```

2. Ví dụ 2

Yêu cầu: Cài đặt trigger thực hiện việc cập nhật lại trị giá hóa đơn khi có một chi tiết hóa đơn mới được thêm vào.

Thực hiện:

```
CREATE TRIGGER trg_ins_cthd ON CTHD
FOR INSERT
AS
BEGIN
    DECLARE @SoHD int, @MaSP char(4), @SoLg int, @TriGia money
    -- Lay thong tin cua CTHD vua moi them vao
    SELECT @SoHD = SOHD, @MaSP = MASP, @SoLg = SL
    FROM INSERTED
    -- Tinh tri gia cua san pham moi them vao HOADON
    SET @TriGia = @SoLg * SELECT GIA FROM SANPHAM WHERE MASP = @MaSP
    -- Khai bao mot CURSOR duyet qua tat ca cac CTHD da co san trong HOADON
    DECLARE cur_cthd CURSOR
    FOR
        SELECT MASP, SL
        FROM CTHD
        WHERE SOHD = @SoHD

    OPEN cur_cthd
    FETCH NEXT FROM cur_cthd
    INTO @MaSP, @SoLg

    WHILE (@@FETCH_STATUS = 0)
    BEGIN
        -- Cong don tri gia cua tung san pham vao bien TriGia
        SET @TriGia = @TriGia + @SoLg * SELECT GIA FROM SANPHAM WHERE MASP = @MaSP
        FETCH NEXT FROM cur_cthd
        INTO @MaSP, @SoLg
    END

    CLOSE cur_cthd
    DEALLOCATE cur_cthd
    --Tien hanh cap nhat lai tri gia HOADON
    UPDATE HOADON SET TRIGIA = @TriGia WHERE SOHD = @SoHD
END
```

V. Bài tập yêu cầu

1. Bài tập 1

Sinh viên hoàn thành Phần I bài tập QuanLyBanHang từ câu 11 đến 14.

2. Bài tập 2

Sinh viên hoàn thành Phần I bài tập QuanLyGiaoVu câu 9, 10 và từ câu 15 đến câu 24.

~ HẾT ~