



KHOA HỆ THỐNG THÔNG TIN
FACULTY OF INFORMATION SYSTEMS

TÀI LIỆU HƯỚNG DẪN THỰC HÀNH

MÔN CƠ SỞ DỮ LIỆU – IT004

BUỔI THỰC HÀNH 05

Hướng dẫn thực hành

Lê Võ Đình Kha – khalvd@uit.edu.vn



TÀI LIỆU HƯỚNG DẪN THỰC HÀNH

MÔN CƠ SỞ DỮ LIỆU – IT004

BUỔI THỰC HÀNH 05

Hướng dẫn thực hành

Lê Võ Đình Kha – khalvd@uit.edu.vn



TÀI LIỆU ĐƯỢC BIÊN SOẠN DỰA TRÊN CÁC NGUỒN TÀI LIỆU THAM KHẢO

1. Slides môn Cơ sở dữ liệu, Khoa Hệ thống Thông tin, Trường Đại học Công nghệ Thông tin, ĐHQG Tp. HCM.
2. Tài liệu thực hành môn Cơ sở dữ liệu Khoa HTTT, Giảng viên lý thuyết môn CSDL và quý Thầy Cô phụ trách giảng dạy môn CSDL.
3. Giáo trình Cơ sở dữ liệu, Đồng Thị Bích Thủy, Phạm Thị Bạch Huệ, Nguyễn Trần Minh Thư, NXB Khoa học và Kỹ thuật.

GIỚI THIỆU HỌC PHẦN THỰC HÀNH BUỔI SỐ 5

NỘI DUNG



Phần 1: Ràng buộc toàn vẹn.

- Loại phức tạp: Sử dụng TRIGGER

Phần 2: Bài tập thực hành

- CSDL Quản Lý Bán Hàng
- CSDL Quản Lý Giáo Vụ

RÀNG BUỘC TOÀN VỆN - TRIGGER

- Một quan hệ

- + Miền giá trị
- + Liên bộ
- + Liên thuộc tính

Ràng buộc toàn vẹn **Constraint**:

NOT NULL, UNIQUE, PRIMARY KEY,
FOREIGN KEY, CHECK, DEFAULT

- Nhiều quan hệ

+ Tham chiếu

- + Liên bộ, liên quan hệ
- + Liên thuộc tính, liên quan hệ
- + Thuộc tính tổng hợp
- + Chu trình

Ràng buộc toàn vẹn sử dụng **Trigger**

RÀNG BUỘC TOÀN VỆN - TRIGGER

1. KHÁI NIỆM VỀ TRIGGER

- **Trigger** là một thủ tục đặc biệt, có thể xem như một công cụ kiểm tra lỗi. Công cụ này được sử dụng để khai báo ràng buộc dữ liệu cho một đối tượng table hoặc view và tự động thực thi khi một trong 3 phát biểu **Insert, Update, Delete** làm 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 chạy khi toàn bộ câu lệnh **thêm, xóa, cập nhật** được thực hiện trọn vẹn. Lợi ích chính của trigger so với các công cụ kiểm tra khác đó là chúng có thể chứa các xử lý phức tạp trên các table có liên quan đến table đang thay đổi.

RÀNG BUỘC TOÀN VỆ - TRIGGER

1. KHÁI NIỆM VỀ TRIGGER

- **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**.

VÍ DỤ VỀ RÀNG BUỘC TOÀN VỌNG - TRIGGER

Kiểm tra giới hạn giá trị (CHECK constraint):

- Có thể sử dụng trigger để kiểm tra và đảm bảo rằng giá trị mới được thêm vào một cột thỏa mãn một điều kiện nhất định.

Auto-increment (DEFAULT constraint):

- Trigger có thể được sử dụng để tự động tăng giá trị của một cột khi một bản ghi mới được thêm vào bảng.

Xử lý các hành động phụ thuộc vào các sự kiện (AFTER INSERT, AFTER UPDATE, AFTER DELETE):

- Trigger có thể được sử dụng để tự động xử lý các hành động sau khi có các sự kiện nhất định xảy ra, chẳng hạn như sau khi thêm, cập nhật hoặc xóa dữ liệu.

RÀNG BUỘC TOÀN VỆN - TRIGGER

2. CÚ PHÁP TẠO TRIGGER

```
CREATE TRIGGER <trigger_name>  
ON <table name>  
( WITH <Encryption> )  
FOR | AFTER | INSTEAD OF  
(INSERT), (UPDATE), (DELETE)  
AS  
BEGIN  
    (Tập hợp các câu lệnh)  
END
```

RÀNG BUỘC TOÀN VỆN - TRIGGER

- **<trigger_name>**: Tên của trigger muốn đặt.
- **<table_name>**: Tên của bảng mà trigger sẽ được áp dụng.
- **(WITH <Encryption>)**: Tùy chọn này chỉ định xem trigger có được lưu trữ mã hóa hay không. Nếu muốn mã hóa trigger, có thể sử dụng từ khóa WITH ENCRYPTION.
- **FOR | AFTER | INSTEAD OF**: Xác định khi nào trigger sẽ được kích hoạt. Có thể là FOR (khi), AFTER (sau khi), hoặc INSTEAD OF (thay vì).
- **(INSERT), (UPDATE), (DELETE)**: Xác định sự kiện mà trigger sẽ phản ứng, có thể là INSERT, UPDATE, hoặc DELETE.
- **AS**: Chỉ định phần bắt đầu của trigger.
- **BEGIN và END**: Chứa tập hợp các câu lệnh SQL muốn thực hiện khi trigger được kích hoạt.

RÀNG BUỘC TOÀN VẠY - TRIGGER

Việc sử dụng **AFTER, FOR, DELETE, INSERT, UPDATE** và bảng tạm **INSERTED** và **DELETED**

FOR và AFTER:

- **FOR và AFTER** là tương đương nhau khi sử dụng trong ngữ cảnh của trigger. Chúng đều chỉ định thời điểm trigger sẽ được kích hoạt: sau khi thực hiện sự kiện thêm, cập nhật, hoặc xóa dữ liệu.
- **Bảng tạm INSERTED và DELETED:** Khi trigger được kích hoạt, SQL Server tạo **hai bảng tạm: INSERTED và DELETED**. **INSERTED** chứa các dòng dữ liệu mới được thêm vào hoặc cập nhật. **DELETED** chứa các dòng dữ liệu bị xóa hoặc cập nhật.
- **INSERT Trigger:** Khi thêm mới dữ liệu vào bảng thông qua câu lệnh **INSERT**, các dòng mới sẽ được lưu trong bảng **INSERTED**.

RÀNG BUỘC TOÀN VỆN - TRIGGER

- **DELETE Trigger:** Khi xóa dữ liệu từ bảng thông qua câu lệnh **DELETE**, các dòng bị xóa sẽ được di chuyển vào bảng **DELETED**.
- **UPDATE Trigger:** Khi cập nhật dữ liệu trong bảng thông qua câu lệnh **UPDATE**, bảng gốc và bảng **INSERTED** đều chứa các dòng dữ liệu mới, trong khi bảng **DELETED** chứa các dòng có nội dung cũ.
- **Thay đổi dữ liệu trên bảng INSERTED và DELETED:** Dữ liệu trong các bảng tạm INSERTED và DELETED không thể được thay đổi trực tiếp. Tuy nhiên, có thể sử dụng nó để xử lý dữ liệu trên các bảng có liên quan.
- **Cập nhật dữ liệu trong Trigger:** Trong trigger Insert và Update, có thể thay đổi nội dung của các dòng dữ liệu mới thay đổi bằng cách sử dụng lệnh UPDATE trên bảng hiện tại (bảng mà trigger đang thao tác).

RÀNG BUỘC TOÀN VỆN - TRIGGER

- Từ khóa **AS** là nơi bắt đầu phần thực thi của trigger trong SQL Server. Các hành động bên trong trigger được đặt giữa cặp từ khóa **BEGIN** và **END**.

Một số lệnh không thể xuất hiện trong trigger:

- Các Lệnh **CREATE**, **ALTER** và **DROP**: Trigger chứa các lệnh để tạo (CREATE), sửa đổi (ALTER), hoặc xóa (DROP) các đối tượng khác như bảng, trigger khác, hoặc thủ tục.
- **TRUNCATE TABLE**: Lệnh **TRUNCATE TABLE** không thể xuất hiện trong trigger. **TRUNCATE TABLE** được sử dụng để xóa toàn bộ nội dung của bảng, và việc này không phù hợp trong một ngữ cảnh trigger.
- **SELECT INTO**: Lệnh **SELECT INTO** cũng không được phép trong trigger vì nó tạo mới một bảng mới, và trigger thường được thiết kế để thực hiện hành động trên dữ liệu đã tồn tại.

RÀNG BUỘC TOÀN VẠY - TRIGGER

- Khai báo biến: Ví dụ: **DECLARE** @sl int, @hoten varchar(20)
- **IF ... ELSE** : tương tự trong lập trình C++.
- **Print 'In dòng chữ ra màn hình'** : in chuỗi ra màn hình.
- **@@ERROR**: biến toàn cục chứa mã lỗi.
- **@@ROWCOUNT**: biến toàn cục chứa số dòng dữ liệu.
- **Lệnh về transaction**: (SQL Server thường mặc định auto commit)
 - + Commit / Commit tran/ Commit transaction
 - + Rollback/ Rollback tran/ Rollback transaction
- **Các bảng tạm**: Inserted, Deleted

RÀNG BUỘC TOÀN VẠY - TRIGGER

- **Khai Báo Biến:** Có thể khai báo biến trong T-SQL bằng cách sử dụng từ khóa DECLARE. **Ví dụ:** DECLARE @sl INT, @hoten VARCHAR(20);
- **IF...ELSE:** T-SQL hỗ trợ cấu trúc điều kiện IF...ELSE, tương tự như trong lập trình C/C++.
- **PRINT:** Dùng để in thông báo hoặc giá trị ra màn hình trong quá trình thực thi câu lệnh. **Ví dụ:** PRINT 'In dòng chữ ra màn hình';
- **@@ERROR:** @@ERROR là biến toàn cục chứa mã lỗi của lệnh SQL thực thi gần nhất. Nếu giá trị của @@ERROR là 0, nghĩa là không có lỗi.
- **@@ROWCOUNT:** @@ROWCOUNT là biến toàn cục chứa số lượng dòng ảnh hưởng bởi lệnh SQL thực thi gần nhất.

RÀNG BUỘC TOÀN VỆ - TRIGGER

- **Lệnh Về Transaction:** **SQL Server** thường mặc định ở chế độ **auto-commit**, tức là mỗi câu lệnh SQL được thực thi mặc định sẽ tự động **commit**.
- **Các lệnh transaction** bao gồm **COMMIT** để xác nhận các thay đổi đã được áp dụng và **ROLLBACK** để hủy bỏ các thay đổi và quay trở lại trạng thái trước đó.
 - -- Bắt đầu một transaction **BEGIN TRANSACTION;**
 - -- Xác nhận thay đổi **COMMIT;**
 - -- Hoặc hủy bỏ thay đổi **ROLLBACK;**
- **Các Bảng Tạm: Inserted và Deleted** là các bảng tạm được sử dụng trong trigger để lưu trữ dữ liệu trước và sau khi một sự kiện thêm, cập nhật, hoặc xóa xảy ra.

RÀNG BUỘC TOÀN VỆN - TRIGGER

```
CREATE TRIGGER <trigger_name>
ON <table name>
FOR (INSERT), (UPDATE), (DELETE)
AS
BEGIN
    <Khai báo biến>
    If <Kiểm tra điều kiện>
        BEGIN
            Rollback Transaction
        END
    END
END
```

RÀNG BUỘC TOÀN VỆN - TRIGGER

3. KÍCH HOẠT/VÔ HIỆU HÓA MỘT TRIGGER

ALTER TABLE *table* ENABLE | DISABLE TRIGGER ALL | trigger_name(...n)

4. HIỆU CHỈNH TRIGGER

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.

STORE PROCEDURE

- Tạo Store Procedure

CREATE PROCEDURE < Tên Procedure > < Các tham số >

AS

< Khai báo biến >

< Nội dung chương trình >

GO

- Gọi Store Procedure

EXEC < Tên Procedure > < Truyền tham số tương ứng >

STORE PROCEDURE

- Tạo Store Procedure

```
CREATE PROCEDURE TIM_SP_NUOCSX @NUOC nvarchar(40)  
AS  
SELECT * FROM SANPHAM WHERE NUOCSX= @NUOC  
GO
```

- Gọi Store Procedure

```
EXEC TIM_SP_NUOCSX 'Trung Quoc'
```

ĐỌC THÊM - 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 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 khi 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:

- 1. 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.
- 2. 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.

6. CÁC LOẠI CURSOR

3. 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: Gửi 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 khi 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.

7. XỬ LÝ ĐỒNG THỜI

4 chọn lựa bao gồm:

- **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.

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))

- **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.

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))

- **RELATIVE {n | @nvar}**: Nếu n hoặc @nvar > 0, chuyển xuống n dòng tính từ dòng kế 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.

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))

- **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.

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 ) )
```

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).

VÍ DỤ 1 - TRIGGER

Yêu cầu: (Bài tập Quản lý Bán hàng – Phần 1 – Câu 11) 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).

- Tạo trigger trên bảng KHÁCH HÀNG để kiểm tra khi cập nhật ngày đăng ký.
- Tạo trigger trên bảng HÓA ĐƠN để kiểm tra khi cập nhật và thêm hóa đơn mới.

VÍ DỤ 1 - TRIGGER

Tạo trigger trên bảng KHÁCH HÀNG để kiểm tra khi cập nhật ngày đăng ký.

```
create TRIGGER TRG_CHECK_NGHD_NGDK      -- CẬP NHẬT NGÀY ĐĂNG KÝ CỦA KHÁCH HÀNG
ON KHACHHANG FOR UPDATE
AS
BEGIN
    DECLARE @NGHD SMALLDATETIME
    DECLARE @NGDK SMALLDATETIME
    SELECT @NGDK=NGDK FROM inserted
    IF (@NGDK > ANY (SELECT HOADON.NGHD FROM HOADON, inserted WHERE HOADON.MAKH=inserted.MAKH))
    BEGIN
        PRINT N'NGÀY ĐĂNG KÝ PHẢI NHỎ HƠN NGÀY MUA HÀNG'
        ROLLBACK TRANSACTION
    END
END
END
```

VÍ DỤ 1 - TRIGGER

Tạo trigger trên bảng HÓA ĐƠN để kiểm tra khi cập nhật và thêm hóa đơn mới.

```
CREATE TRIGGER TRG_CHECK_NGHD
ON HOADON FOR INSERT, UPDATE
AS
BEGIN
    DECLARE @NGDK SMALLDATETIME
    DECLARE @NGHD SMALLDATETIME
    SELECT @NGDK=NGDK, @NGHD= NGHD FROM inserted,KHACHHANG
    WHERE KHACHHANG.MAKH=inserted.MAKH
    IF (@NGDK>@NGHD)
    BEGIN
        PRINT N'Ngày lập hóa đơn phải lớn hơn ngày đăng ký của khách hàng'
        ROLLBACK TRANSACTION
    END
    ELSE
        PRINT N'CẬP NHẬT/THÊM THÀNH CÔNG'
END
```

VÍ DỤ 2- TRIGGER

Yêu cầu: Cài đặt trigger thực hiện việc cập nhật tự động doanh số tổng của các hóa đơn khi có một chi tiết hóa đơn mới được thêm vào.

```
CREATE TRIGGER TANGDOANHSO
```

```
ON HOADON
```

```
FOR INSERT
```

```
AS
```

```
BEGIN
```

```
    DECLARE @TRIGIA2 int, @MAKH2 char(4)
```

```
    SELECT @TRIGIA2 = TRIGIA, @MAKH2=MAKH FROM inserted
```

```
    UPDATE KHACHHANG SET DOANHSO= DOANHSO + @TRIGIA2 WHERE MAKH=@MAKH2
```

```
END
```

BÀI TẬP THỰC HÀNH

Yêu cầu: Sử dụng phần mềm **Microsoft SQL Server** và truy cập website môn học, tiến hành thực hiện các bài tập sau:

- Phần I bài tập **QuanLyBanHang** từ câu 11 đến câu 12.
- Phần III bài tập **QuanLyBanHang** từ câu 41 đến câu 44.

BÀI TẬP THÊM

- Phần I bài tập **QuanLyGiaoVu** câu 9, 10 và từ câu 15 đến câu 24.

HỎI - ĐÁP

