UIT VNUHCM University of Information Technology

CHƯƠNG 3:
XỬ LÝ THÔNG TIN TRÊN MÁY TÍNH: LẬP TRÌNH CƠ SỞ DỮ LIỆU

Khoa Khoa học và kỹ thuật thông tin
Bộ môn Thiết bị di động và Công nghệ Web

NỘI DUNG

1. Stored procedure.
2. Trigger.
3. Function.
4. Cursor.

Lập trình Procedure

Giới thiệu

- Một Stored Procedure là bao gồm các câu lệnh Transact-SQL và được lưu lại trong cơ sở dữ liệu.
- Để thực thi chỉ cần gọi ra.
- Transact-SQL (T-SQL) là một ngôn ngữ lập trình được sử dụng làm trung gian giữa cơ sở dữ liệu và các ứng dụng. Nó tương đối dễ học vì thực chất nó được tạo bởi hầu hết là các lệnh SQL.

Lợi ích của Store procedure

- Module hóa: Chỉ cần viết Stored Procedure 1 lần, sau đó có thể gọi nó nhiều lần ở trong ứng dụng.
- Thực thi nhanh hơn: Stored Procedure sẽ được biên dịch và lưu vào bộ nhớ khi được tạo ra - thực thi nhanh hơn so với việc gửi từng đoạn lệnh SQL tới SQL Server.
- Giảm tải băng thông: gom các câu lệnh SQL vào 1 Stored Procedure và chỉ phải gọi đến 1 lần duy nhất qua network thay vì phải gọi nhiều lần.

Cú pháp

– Khai báo một store procedure:

```
CREATE PROCEDURE procedure_name
AS
     sql_statement
GO;
```

– Thực thi store procedure:

```
EXEC procedure_name;
```

Ví dụ

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Ví dụ

– Viết procedure liệt kê danh sách tất cả các khách hàng.

```
CREATE PROCEDURE Danh_sach_khach_hang
AS
     SELECT * FROM Customers
GO;
```

– Thực thi PROCEDURE:

```
EXEC Danh_sach_khach_hang;
```

PROCEDURE có tham số

- Ta có thể truyền vào các tham số đầu vào cho một Procedure. Một Procedure có thể có 1 hoặc nhiều tham số.
- Có 3 trường hợp tham số cho Procdure là:
  + Một tham số vào (input).
  + Nhiều tham số vào (multiple input).
  + Tham số ra (output).

Ví dụ

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Ví dụ trường hợp 1 tham số

– Khai báo Procedure:

```
CREATE PROCEDURE SelectAllCustomers @City nvarchar(30)
AS
     SELECT * FROM Customers WHERE City = @City
GO;
```

– Gọi thực thi Procedure:

```
EXEC SelectAllCustomers @City = 'London';
```

Ví dụ trường hợp nhiều tham số

– Khai báo procedure:

```
CREATE PROCEDURE SelectAllCustomers @City nvarchar(30),
@PostalCode nvarchar(10)
AS
      SELECT * FROM Customers WHERE City = @City AND
      PostalCode = @PostalCode
GO;
```

– Thực thi procedure:

```
EXEC SelectAllCustomers @City = 'London', @PostalCode =
'WA1 1DP';
```

Ví dụ trường hợp tham số ra

– Khai báo procedure:

```
CREATE PROCEDURE SelectCustomerName @CustomerID char(2),
@CustomerName nvarchar(100) OUTPUT
AS
      SELECT @CustomerName=CustomerName FROM Customers WHERE
      CustomerID = @CustomerID
GO;
```

– Thực thi procedure:

```
DECLARE @CName nvarchar(100)
EXEC SelectCustomerName @CustomerID = 2, @CName OUTPUT
print @CName
```

Lập trình Trigger

Giới thiệu

– **Trigger** là stored procedure đặc biệt sẽ tự động thực hiện khi có hành động bổ sung DL lên 1 table mà trigger bảo vệ . Trigger có thể bao gồm hầu hết các phát biểu T_SQL.

Các tính chất của Trigger

- Liên kết với Table.
  + Trigger được định nghĩa trên 1 table cụ thể gọi là Trigger liên kết Table.
- Thực hiện tự động.
  + Nếu có các hành động INSERT, UPDATE, DELETE mà trigger định nghĩa nó sẽ tự đông thực hiện.
  + Nó không được gọi trực tiếp và không chấp nhận tham số.

Các tính chất của Trigger (tt)

- Là 1 giao tác:
  - + Trigger và phát biểu tạo ra nó được thực hiện như là 1 giao tác , có thể roolback bất cứ đâu trong trigger. Tong Trigger có thể có Rollback Trans mà không có Begin Trans , SQL Server tự hiểu có Begin trans ảo ở đây.
  - + Nếu có rollback và nó được thực hiện thì toàn Transaction sẽ Rollback.
  - + Nếu trong 1 batch có rollback và nó được thực hiện thì toàn batch sẽ bị hủy các phát biệu sau Roolback sẽ không thực hiện.
- Nên tránh lạm dụng Rollback trong Trigger vì nó phải Làm undo các thao tác trước đó ➔ kiểm tra tính hợp lệ trước khi bắt đầu transaction

Lợi ích của Trigger

- Bổ sung dây chuyền (cascade).
  - + Các bổ sung dây chuyền bằng Trigger làm giảm ước các coding cần thiết cho sự thay đổi trên các bảng liên quan.
- Làm tăng cường toàn vẹn dữ liệu.
  - + Do có thể tham chiếu nhiều cột trên các bảng nên có tác dụng hơn check.Tuy vậy các rang buộc ưu tiên kiểm tra trước , nếu vi phạm trigger không thực thi.
- Thông báo lỗi do người dùng định nghĩa.
  - + Các thông báo Rule, Default, check
- Bảo quản dữ liệu không tiêu chuẩn hóa.

Bảo quản dữ liệu không tiêu chuẩn hóa.

Ví dụ:

HOADON (MSHD.NGAYHD,**TONGTIEN**)

CTHOADON (MSHD, MSMH, SOLUONG, DONGIA)

Khi sửa số lượng, đơn giá dẫn đến ảnh hưởng tổng tiền.

Lưu ý

- Trigger là phản ứng, ràng buộc được thực hiện trước
- Ràng buộc được kiểm tra đầu tiên.
- 1 Bảng có thể viết nhiều trigger.
- Trigger không thể tạo trên View và bảng tạm.
- Trigger không thể trả về các tập kết quả.

Các thao tác với trigger

- Tạo trigger.
- Sửa trigger.
- Xoá trigger.

Tạo trigger

```
Create Trigger  TenTrigger
On table [with (… )]
[For insert, update, delete]
        [With append]
        [Not for Relication]
    As
        phát biểu SQL
```

Các tính chất của trigger

- Khi tạo Trigger, thông tin trigger được insert vào các bảng hệ thống sysobject và syscomment.
- Có sẵn hai bảng đặc biệt trong các trigger là: **deleted** và **inserted**.
- Bảng **deleted** chứa các bản sao các dòng bị tác động bởi các phát biểu **Update và delete**.
- Bảng **inserted** chứa các bản sao các dòng bị tác động bởi các phát biểu **insert và Update**.
- Không thể thay đổi DL trên các bảng deleted và inserted trực tiếp, có thể dùng phát biểu select.

Chú ý

– SQL không cho phép các phát biểu sau dùng Trigger :

+ Tất cả các phát biểu create, alter, drop.
+ Grant, Revoke, Deny.
+ Load, restore.
+ Reconfigure
+ Truncate Table.
+ Update Statistic.
+ Select into.

Ví dụ 1: Phát biểu: Một sinh viên (MSSV) không được thi quá 2 lần.

```
Create Trigger KiemTraThi2Lan
      On KETQUA for Insert
      As
      Begin
            If (Select  Count(*) from KETQUA  a, INSERTED b
            Where a.MSSV=b.MSSV and  a.MSMH=b.MSMH)>1
            Begin
                  Print ' Khong the qua 2 lan'
                  Rollback Transaction
            End
      End
```

Ví dụ 2: Phát biểu: Khi xoá một hoá đơn thì xoá luôn các CTHD tương ứng.

```
Create Trigger XoaHD
      On HOADON for delete
      As
      Begin
            declare  @ms int
            Select @ms = (Select MSHD from
      Deleted)
            Delete From CTHD where  MSHD =          @ms
      End
```

– Lưu ý: Nếu có ràng buộc phải loại bỏ trước vì sẽ vi phạm khi xóa.

Sửa trigger

Alter Trigger TenTrigger
On Tentable [with encryption]
[for *inser, update, delete*]
[not fro replication]
AS
Phát biểu SQL

Sửa trigger KiemTraThi2Lan.

```
Alter  Trigger KiemTraThi2Lan
      On KETQUA for Insert
      As begin
      If (Select  Count(*) from KETQUA a,      INSERTED  b
      Where a.MSSV=b.MSSV and  a.MSMH=b.MSMH)>2
      Begin
            Print ' Khong the qua 2 lan'
            Rollback Transaction
      End
```

Xoá trigger

- Cú pháp: `Drop Trigger TenTrigger`
- Có thể làm mất hiệu lực tạm thời của Trigger lên 1 table
- Cú pháp :

```
Alter Table   tenTable
{Enable/ Disable } Trigger {all/TenTrigger}
```

- Vd :

```
Alter    Table  KETQUA
Disable Trigger  KiemTraThi2Lan
```

- Muốn có hiệu lực lại : dùng **Enable**

Các hoạt động của trigger.

- Hoạt động khi insert.
- Hoạt động khi update.
- Hoạt động khi delete

Hoạt động khi insert

- Phát biểu Insert thực hiện trên bảng Trigger định nghĩa.
- Phát biểu Insert được ghi .
- Trigger bị bắn phá và phát biểu trong Trigger thực thi khi Trigger Insert bắn phá, các dòng mới được thêm vào hai bảng Trigger và Inserted. Bảng Inserted nắm giữ một bản sao các dòng đã Insert. Bảng Inserted chứa các hoạt động từ phát biểu trên. Trigger có thể khảo sát trên bản Inserted để quyết định hành động của mình.

Hoạt động khi update

- Phát biểu Update thực hiên trên bản Trigger định nghĩa.
- Phát biểu Update được ghi lại.
- Trigger bị bắn phá và phát biểu trong trigger thực hiện có thể xem như 2 bước :
  + Xóa (delete)
  + Chèn (insert)
- Như thế các dòng gốc sẽ di chuyển đến các bảng deleted và các dòng cập nhật được chèn vào bảng inserted.

Hoạt động khi update

- Có thể định nghĩa 1 trigger để giám sát việc cập nhật DL trên 1 cột đặc biệt bằng cách dùng phát biểu **if update**, nó cho phép thực hiện khi có sự cập nhật trên các cột đã chỉ định.
- Cú pháp: If update (column) [{AND | OR} update (**cols**)
- VD:

```
Create Trigger CapnhatMSSV
      If update(MSSV)
      Begin
            Print 'Không được cập nhật mã số NV'
            Rollback transaction
        End
```

Hoạt động khi Delete

- Phát biểu Delete thực hiên trên bản Trigger định nghĩa.
- Phát biểu Delete được ghi lại.
- Trigger bắn phá và các phát biểu trong Trigger thực thi.
- Khi Trigger bắn phá các dòng bị xóa được đặt trong bản Deleted. Bảng Deleted lưu giữ một bản sao các dòng bị xóa.
- Chú ý:
  + Các dòng Delete không tồn tại trong cơ sở dữ liệu, vì vậy bản xóa và cơ sở dữ liệu không có các “dòng chung”
  + Trigger bắn phá bởi Delete không thực hiện phát biểu:TRUNCATE TABLE

Các dạng đặc biệt của Trigger

- Trigger lồng.
- Trigger đệ quy.

Trigger lồng

- Trigger có thể lồng nhau **32 mức**. Bất cứ trigger nào trong chuỗi lồng bị loop (mức lồng vượt quá mức 32) transaction sẽ ***rollback***.
- Chú ý:
  - + Mặc định, cấu hình lồng bằng ON.
  - + Trigger bị lồng không bị bắn phá 2 lần trong 1 transaction, mặt khác, trigger không thể tự bắn phá chính nó. Trong trường hợp này ta nói trigger đệ quy, sẽ bàn sau.
  - + Một trigger là 1 transaction, 1 lỗi xảy ra tại bất kỳ ở đâu, tất cả việc bổ sung DL sẽ rollback, có thể thêm phát biểu print để kiêm tra lỗi xảy ra tại đâu.

Trigger lồng

– Mức lồng sẽ tăng lên khi có 1 trigger lồng bị bắn phá. Để tránh mức lồng vượt quá mức 32, nên dùng hàm **@@nestlevel**.

Lợi ích của trigger lồng

- Dùng trigger là công cụ hữu hiệu đảm bảo DL toàn vẹn khi install SQL Server, lồng nhau là default. Có thể làm mất tạm thời cho phép trigger lồng bằng cú pháp
  + sp_configure ‘nested Triggers’, 0.
  + Muốn ngược lại: sp_configure ‘nested Triggers’, 1

Điểm yếu của trigger lồng

- Làm mất tính log của trigger do Tính phức tạp của nó.
- Có thể thay thế chức năng log của trigger bằng con đường *mỗi trigger được khởi tạo sẽ bổ sung tất cả dữ liệu cần thiết* nhờ trợ giúp của **stored-Procedure.**

Trigger đệ quy

- Trigger có thể có các phát biểu Update, insert, delete đến cùng một bảng hay bảng khác. Khi option đệ quy bật lên trigger có thể thay đổi table mà bắn phá chính nó. Execute option đệ quy mặc định là không, có thể thay đổi VLOOKUP.
- Cú pháp:

  Sp-dboption database, 'recursive triggers', (True False)

Các loại đệ quy

- Đệ quy trực tiếp:
  + Ví dụ: một ứng dụng cập nhật table A, gây nên trig1. Trig1 lại cập nhật A 1 lần nữa, và trig1 bị gọi.
- Đệ quy gián tiếp:
  + Ví dụ: một ứng dụng cập nhật table A, gây nên trig1 bắn phá, trig1 cập nhật table B, gây nên trig2, trig2 cập nhật table A, trig1 lại bắn phá 1 lần nữa

Lập trình Function

Giới thiệu

- Function (Hàm) là một đối tượng trong cơ sở dữ liệu bao gồm một tập nhiều câu lệnh được nhóm lại với nhau và được tạo ra với mục đích sử dụng lại.
- Trong SQL Server, hàm được lưu trữ và bạn có thể truyền các tham số vào cũng như trả về các giá trị.

Các thao tác với hàm

- Khai báo hàm trong SQL:
  - + Dạng 1: Dạng đầy đủ.
  - + Dạng 2: Inline Table-valued Functions.
  - + Dạng 3: Multi-statement Table-valued Functions.
- Xoá hàm.

Cú pháp: `DROP FUNCTION function_name;`

Cú pháp Dạng 1: Dạng đầy đủ.

```
CREATE  FUNCTION [ owner_name. ] function_name
    ( [ { @parameter_name [AS] scalar_parameter_data_type [
= default ] } [ ,...n ] ] )
RETURNS scalar_return_data_type
 [ AS ]
BEGIN
    function_body
    RETURN scalar_expression
END
```

Ví dụ 1

– Khai báo:

```
Create Function Cong(@x1 int, @x2 int) returns int
As
begin
     return @x1+@x2
end
```

– Sử dụng

```
select dbo.Cong(1, -5) as KQ
```

Cú pháp Dạng 2: Inline Table-valued Functions.

```
CREATE FUNCTION [ owner_name. ] function_name
    ( [ { @parameter_name [AS] scalar_parameter_data_type [
= default ] } [ ,...n ] ] )
RETURNS TABLE
 [ AS ]
RETURN [ ( ] select-stmt [ ) ]
```

Ví dụ 2

– Khai báo:

```
Create Function DiemNhoHon(@x1 int) returns table
As
     return (select * from SVMON2 where Diem <@x1 )
```

– Sử dụng

```
select * from dbo.DiemNhoHon(5)
```

Cú pháp Dạng 3 Multi-statement Table-valued Functions.

```
CREATE FUNCTION [ owner_name. ] function_name
    ( [ { @parameter_name [AS] scalar_parameter_data_type [
= default ] } [ ,...n ] ] )
RETURNS @return_variable TABLE < table_type_definition >
[ WITH < function_option > [ [,] ...n ] ]
[ AS ]
BEGIN
    function_body
    RETURN
END
```

Ví dụ 3

```
Create Function DiemNhoHon5(@x1 int) returns @x2
table (MSGV int, MSM int, Diem int)
As
begin
     insert @x2
     select * from SVMON where Diem <@x1
     return
end
```

Lập trình Cursor

Giới thiệu

- **Cursor** (con trỏ) là một kỹ thuật lập trình CSDL cao cấp trong SQL. Dùng tính toán và chọn một trường ô (duyệt – traversal) trong một bảng (Table) trong CSDL.
- Các cursor tạo điều kiện xử lý tiếp theo kết hợp với việc traversal, chẳng hạn như thu hồi, bổ sung và loại bỏ các bản ghi cơ sở dữ liệu
- Trong các thủ tục SQL, một cursor sẽ làm cho nó có thể định nghĩa một tập kết quả (một tập hợp các dòng dữ liệu) và thực hiện logic phức tạp trên cơ sở hàng bằng hàng.

Vấn đề

- Các câu lệnh trong SQL đều thao tác lên nhiều dòng dữ liệu thỏa điều kiện WHERE cùng lúc mà không thể thao tác trên từng dòng cụ thể.
- Cursor là kiểu dữ liệu có thể duyệt qua từng dòng kết quả trả về của câu lệnh SELECT giúp chúng ta có thể xử lý khác nhau cho từng kết quả mà ta mong muốn.
- Nhưng lại tồn tại khuyết điểm là xử lý rất chậm.

Các bước sử dụng một Cursor

- Để sử dụng con trỏ trong cơ sở dữ liệu, chúng ta cần:
  1. Khai báo một con trỏ xác định một tập kết quả.
  2. Thiết lập kết quả cho con trỏ.
  3. Gán dữ liệu cho các biến cục bộ cần thiết cho con trỏ và một hàng.
  4. Đóng cursor khi hoàn thành.

Các cú pháp khai báo 1 cursor

1- Cú pháp khai báo một Cursor

```
DECLARE ten_con_tro [SCROLL] CURSOR
        FOR
                SELECT
                FROM
                WHERE
        [FOR {READ ONLY | UPDATE [OF ten_cot [truong]]}]
```

Cú pháp khai báo một Cursor

- SCROLL: cho phép con trỏ di chuyển lên xuống, qua lại giữa các mẫu tin.
- READ ONLY: không cho phép thực thi các hành động như update,...
- UPDATE: xác định khả năng cập nhật của con trỏ, nếu OF được chỉ định thì chỉ có những cột, những trường trong danh sách được chỉnh sửa. Ngoài ra, chúng ta cũng có thể khởi tạo riêng con trỏ rồi mới gán lệnh SELECT cho con trỏ, như sau:

```
DECLARE @ten_con_tro CURSOR
SET @ten_con_tro = CURSOR FOR SELECT...
```

2- Cú pháp mở một Cursor

```
OPEN [GLOBAL] ten_con_tro | @ten_con_tro
```

Lưu ý: GLOBAL là biến toàn cục.

3 – Cú pháp truy cập một con trỏ

```
FETCH [NEXT | PRIOR | FIRST | LAST | ABSOLUTE {n | @nVar} |
RELATIVE {n | @nVar}]
FROM [GLOBAL] ten_con_tro | @ten_con_tro
[INTO @ten_bien[du_lieu]]
```

Cú pháp truy cập một con trỏ

- **NEXT, PRIOR, FIRST, LAST**: chỉ định cách đọc dữ liệu.
- **ABSOLUTE**: chỉ định n số dòng dữ liệu cần đọc. Nếu:
  - n = 0: không có giá trị trả về.
  - n < 0: xuất phát từ phần đáy dữ liệu.
  - n > 0: xuất phát từ phần đỉnh dữ liệu.
- **RELATIVE** cũng giống như ABSOLUTE nhưng bắt đầu từ vị trí hiện tại.
- Ngoài ra, chúng ta còn có lệnh @@FETCH_STATUS để check xem hệ thống đọc dữ liệu thành công hay thất bại.

4 – Cú pháp đóng Cursor

```
CLOSE [GLOBAL] ten_con_tro | @ten_con_tro
```

hoặc

```
DEALLOCATE [GLOBAL] ten_con_tro | @ten_con_tro
```

Cú pháp đóng Cursor

- **Lưu ý:** CLOSE và DEALLOCATE sẽ có sự khác biệt. Với CLOSE chúng ta sẽ đóng cursor lại nhưng có thể tái sử dụng lại ở lần sau. Còn DEALLOCATE sẽ giải phóng hoàn toàn cursor ra khỏi bộ nhớ, vì vậy nếu có lệnh nào tham chiếu tới cursor có thể gây ra lỗi.

Ví dụ

– Tính điểm trung bình của sinh viên cho CSDL sau:

KQ		
MASV	TENMH	DIEM
s1	m1	7
s2	m1	8
s3	m1	6
s1	m3	4
s2	m3	9
s5	m4	3
s2	m4	8
s1	m4	9

SV		
MASV	TENSV	DIEMTB
s1	B	---
s2	A	---
s3	C	---
s5	D	---

Dữ liệu mẫu

- s3 (1 môn: 6)
- s1 (3 môn: 9,4,7)
- s2 (3 môn: 8,9,8)
- s5 (1 môn: 2)

Ví dụ 1

- Khai báo biến:
  + DECLARE @a CHAR(10), @b FLOAT.
- Khai báo con trỏ:
  + DECLARE x cursor for select MASV from SV.
- Mở con trỏ:
  + OPEN x.

Ví dụ 1

- Truy cập con trỏ:
  + Fetch next from x into @a
- Kiểm tra xem con trỏ đọc dữ liệu được hay không:
  + while (@@fetchstatus = 0)
- Tính điểm TB dựa vào con trỏ:
  + Begin
    - Select @b = AVG(DIEM) from KQ WHERE MASV = @a
    - UPDATE SV SET DIEMTB = @b WHERE MASV = @a.
    - Fetch next from x into @a.
  + End.

Ví dụ 1

- Đóng con trỏ:
  + close x.
  + deallocate x.

Thực thi Cursor

- Để thực thi Cursor, ta dùng lệnh EXEC.

- VD:
  + EXEC TINHDIEM

Full CODE SQL mẫu

```
PROCEDURE TINHDIEM
AS
BEGIN
        DECLARE @a char(10), @b FLOAT.
        DECLARE x cursor for select MASV from SV.
        OPEN x
        Fetch next from x into @a.
        while (@@fetchstatus = 0)
       Begin
              SELECT @b=AVG(DIEM) FROM KQ WHERE MASV = @a
              UPDATE SINHVIEN SET @b = DTB WHERE MASV = @a.
              Fetch next from x into @a
        End
        close x
        deallocate x
END
```

Ví dụ 2

– Tính điểm trung bình của từng sinh viên

Ví dụ 2

```
DECLARE @a char(10), @b FLOAT
DECLARE x cursor for
      SELECT MASV, AVG(DIEM) AS DTB
      FROM KQ
      GROUP BY MASV
OPEN x
Fetch x into @a, @b
While (@@fetchstatus=0)
      Fetch next from x into @a, @b.
End.
```

Tổng kết

- T-SQL là một dạng ngôn ngữ lập trình CSDL, kết hợp các nhóm lệnh SQL lại với nhau.
- Store procedure là một dạng thủ tục, dùng để nhóm các câu lệnh SQL lại với nhau. Có 2 dạng là: không tham số và có tham số (tham số gồm 2 dạng là tham số vào và tham số ra).
- Trigger được thực thi tự động khi có hành vi thay đổi trên CSDL. Trigger không có tham số.
- Function là một dạng đối tượng nhận tham số đầu vào và trả về giá trị cụ thể. Dùng để tái sử dụng lại các lệnh SQL.
- Cursor là một kỹ thuật lập trình nâng cao, cho phép truy xuất dữ liêu phức tạp theo từng dòng và từng ô.

Bài tập cursor

Bài 1. Viết Cursor trả về **SỐ MÔN HỌC** của SV.

Bài 2. Viết Cursor trả về **SỐ MÔN HỌC LẠI** của SV. (Sinh viên học lại khi ***DIEM < 5***)

Bài 3. Viết Cursor cập nhật số môn học của sinh viên từ bảng KQ vào bảng SV.

TÀI LIỆU THAM KHẢO

1. Nguyễn Gia Tuấn Anh, Trương Châu Long, *Bài tập và bài giải SQL Server*, NXB Thanh niên (2005).
2. Đỗ Phúc, Nguyễn Đăng Tỵ, *Cơ sở dữ liệu*, NXB Đại học quốc gia TPHCM (2010).
3. *Nguyễn Gia Tuấn Anh, Mai Văn Cường, Bùi Danh Hường*, *Cơ sở dữ liệu nâng cao*, NXB Đại học quốc gia TPHCM (2019).
4. Itzik Ben-Gan, *Microsoft SQL Server 2012- TSQL Fundamentals*.


Q
A