TRƯỜNG ĐẠI HỌC VINH

CHƯƠNG 2
LẬP TRÌNH CƠ BẢN VỚI T-SQL

Phan Anh Phong PhD.
Viện Kỹ thuật và Công nghệ

Nội dung

- Biến và các phép toán
- Xử lý lô (batch)
- Cấu trúc điều khiển
- Bài tập
- Hướng dẫn tự học

Biến

- Biến là 1 vùng nhớ dùng để lưu trữ dữ liệu, được xác định qua tên biến và kiểu dữ liệu
- Trong T-SQL có 2 loại biến:
  - Hệ thống:
  - Cục bộ
- Biến hệ thống do T-SQL định nghĩa sẵn và bắt đầu bằng ký hiệu @@.
  - Ví dụ:
  - Không thay đổi tên, kiểu của biến hệ thống, người dung có thể sử dụng biến hệ thống
- Biến cục bộ do người dùng tự định nghĩa và PHẢI KHAI BÁO bắt đầu bằng @
  - Ví dụ: DECLARE @ten_lop VARCHAR(50);

3

Ví dụ biến hệ thống

- **Một số biến hệ thống thường dùng**

–@@error - Mã lỗi trả về của lệnh T-SQL

–@@rowcount – số dòng bị ảnh hưởng bởi câu lệnh T-SQL

- @@SERVERNAME -trả về tên của máy chủ CSDL

–@@fetch_status – sử dụng với cursor, đọc dữ liệu trong bảng theo từng dòng cursor. Khi đọc mẫu tin thành công thì biến có giá trị =0

4

Khai báo biến

```
-- Khai báo một biến
 Declare @Result Int
  -- Khai báo một biến có giá trị 50
Declare @a Int = 50
  -- Khai báo một biến có giá trị 2020
Declare @b Int = 2020

 -- In ra màn hình Console
  -- Sử dụng Cast để ép kiểu Int về kiểu xâu ký tự.
  -- Sử dụng toán tử + để nối 2 xâu ký tự
  Print 'a= ' + Cast(@a as varchar(15))
   -- In ra màn hình Console
  Print 'b= ' + Cast(@b as varchar(15))
   -- Tính tổng
  Set @Result = @a + @b
   -- In ra màn hình Console
  Print 'Result= ' + Cast(@Result as varchar(15))
```

5

Biến cục bộ

- Biến cục bộ có giá trị trong một lô (*query batch)* hoặc trong một stored procedure/ function

```
DECLARE @cb_id int
SET @CB_ID=1
GO
PRINT @CB_id
```

BỊ LỖI, vì sao?

Giải thích ở phần sau

6

Gán giá trị cho biến

- Lệnh SET @<ten_bien> = <Biểu_thức>
- Lệnh SELECT @<ten_bien> = <Biểu_thức>

Ví dụ phép gán

- **Giả sử có bảng:**

CREATE TABLE canbo
(macb int primary key, ten varchar(15), namsinh int)
Với nội dung:
INSERT INTO canbo VALUES
(1,'May', 1990),
(2,'Hoc', 1991)

- **Gán bằng lệnh SET:**

```
USE HQTCSDL59
DECLARE @soluongcb int
SET @soluongcb=COUNT(*) FROM canbo
PRINT @soluongcb
```

LỖI!!

Ví dụ phép gán

```
USE HQTCSDL59
DECLARE @soluongcb int
SET @soluongcb=COUNT(*) FROM canbo
PRINT @soluongcb
```

LỖI!!

SỬA LẠI:

```
USE HQTCSDL59
DECLARE @soluongcb int
SET @soluongcb=(SELECT COUNT(*) FROM canbo)
PRINT @soluongcb
```

GIẢI THÍCH?

Ví dụ phép gán

- **Gán bằng lệnh SELECT:**

```
USE HQTCSDL59
DECLARE @soluongcb int
SELECT @soluongcb=COUNT(*) FROM canbo
PRINT @soluongcb
```

Ví dụ phép gán

- **Gán bằng lệnh SELECT:**

```
DECLARE @bien1 int, @bien2 varchar(15), @bien3 int
SELECT @bien1 = macb, @bien2 = ten, @bien3 =namsinh
FROM canbo
PRINT @bien1
PRINT @bien2
PRINT @bien3
```

- Lưu ý: nếu câu truy vấn trả về nhiều dòng, các biến chỉ nhận giá trị tương ứng của dòng cuối trong bảng
- Lấy giá trị dòng đầu: SELECT TOP 1 … FROM …

Khối lệnh

```
BEGIN
 -- khai báo biến
 -- các lệnh T-SQL
END
```

Nội dung

- Biến và các phép toán
- Xử lý lô (batch)
- Cấu trúc điều khiển
- Bài tập
- Hướng dẫn tự học

Xử lý lô (Batch)

Một chương trình viết bằng T-SQL có thể chia thành nhiều lô (batch) để xử lý.

Mỗi lô là một(một nhóm) các lệnh T-SQL được phân cách nhau bằng lệnh GO.

Mình lệnh GO viết trên một dòng, ngược lại bị LỖI

Ví dụ

```
SELECT * FROM canbo; GO
– LỖI
Msg 102, Level 15, State 1, Line 1
Incorrect syntax near 'GO'.
SỬA LẠI:
SELECT * FROM canbo
GO
```

Lưu ý: Lệnh GO là lệnh của SSMS chứ không phải lệnh của T-SQL, do vậy nếu dung GO ở môi trường khác SSMS sẽ bị lỗi

Xử lý lô (Batch)

DECLARE @cb_id int
SET @CB_ID=1
GO
PRINT @CB_id
BỊ LỖI, vì sao?
Chương trình trên có mấy lô?
Biến @cb_id là biến cục bộ hay toàn cục?
➔ CÁCH SỬA
DECLARE @cb_id int
SET @CB_ID=1
PRINT @CB_id
GO

Nội dung

- Biến và các phép toán
- Xử lý lô (batch)
- Cấu trúc điều khiển
- Bài tập
- Hướng dẫn tự học

IF … else…

- Dựa vào điều kiện để quyết định những lệnh T-SQL nào sẽ được thực hiện
- Cú pháp:

```
If <biểu_thức_điều kiện>
    <Lệnh| Khối_lệnh >
[Else Lệnh| Khối_lệnh]
```

- Khối lệnh là một hoặc nhiều lệnh nằm trong cặp từ khóa begin…end

17

IF … else…

```
USE HQTCSDL59
create table donvi
(madv char(10) primary key,
tendv nvarchar(30),
dienthoai char(15))

IF NOT EXISTS (SELECT madv FROM donvi WHERE madv
='2021')
        INSERT INTO Donvi(madv,tendv, dienthoai) VALUES
        ('2021', 'KỸ THUẬT - CÔNG NGHỆ', '02383686868')
ELSE
        PRINT N'Không thể nhập vì trùng mã đơn vị '
```

18

IF … else…

```
create table donvi
(madv char(10) primary key,
tendv nvarchar(30),
dienthoai char(15))

IF EXISTS (SELECT madv FROM donvi WHERE madv
='2021')
        PRINT N'Không thể nhập vì trùng mã đơn vị '
ELSE
        INSERT INTO Donvi(madv,tendv, dienthoai) VALUES
        ('2021', 'KỸ THUẬT - CÔNG NGHỆ', '02383686868')
```

19

IF … else…

- **Ví dụ cho CSDL có 2 bảng**

  HocPhan(MaHP char(5), TenHP varchar (30), SiSo int)

  DangKyHoc(MaSV char(10), MaHP char(5))

  Viết chương trình để thêm một đăng ký mới cho sinh viên có mã số 2020 vào lớp học phần HP01 (giả sử học phần này đã tồn tại trong bảng HocPhan). Qui định rằng mỗi học phần chỉ được đăng ký tối đa 30 sinh viên.

20

IF … else…

- **Ví dụ**

```
Declare @SiSo int
select @SiSo = SiSo from HocPhan where MaHP= "HP01"
if @SiSo < 30
    Begin
        insert into DangKyHoc(MaSV, MaHP) values("2020", "HP01")
        print N"Đăng ký thành công"
    End
Else
    print N "Học phần đã đủ số lượng"
```

21

CASE … WHEN…THEN …END

- Lệnh CASE là lệnh rẽ một hoặc nhiều nhánh
- Cú pháp:

```
CASE <biểu_thức>
    WHEN <biểu_thức_when> THEN <biểu_thức_kết_quả>
    …
    [ELSE <biểu_thức_kết_quả>]
END
```

22

CASE … WHEN…THEN …END

- Ví dụ về lệnh Case

```
DECLARE @n INT, @ketqua NVARCHAR(30)
SET @n=CONVERT(INT, RAND()*9)
SET @ketqua = CASE @n%2
        WHEN 0 THEN 'so chan'
        WHEN 1 THEN 'so le'
        END
PRINT CONVERT(NCHAR(2),@n)+ ' la ' +@ketqua
--hàm RAND() sinh số ngẫu nhiên trong khoảng (0, 1)
--hàm convert https://quantrimang.com/ham-convert-trong-sql-server-161965
```

23

CASE … WHEN…THEN …END

- Tự học lệnh CASE theo link sau:

https://quantrimang.com/ham-case-trong-sql-server-phan-1-36681#mcetoc_1cq2of1r53

24

Vòng lặp WHILE

- Vòng lặp WHILE dùng để lặp đi lặp lại một đoạn mã khi điều kiện cho trước trả về giá trị là TRUE
- Cú pháp

```
WHILE <dieu_kien >
        BEGIN
                <một lệnh>/<nhiều lệnh>
        END;
```

Trong đó, dieu_kien là 1 biểu thức logic trả về giá trị đúng/sai

Khi dieu_kien đúng thực hiện <một lệnh>/<nhiều lệnh>

Lưu ý có thể sử dụng BREAK để thoát lặp và CONTINUE để lặp

25

Vòng lặp WHILE – Ví dụ

- dùng vòng lặp while để tính tổng của 5 số nguyên ngẫu nhiên từ 1 đến 9

```
DECLARE @i int =1, @tong int =0, @k int
WHILE  @i<=5
        BEGIN
                SET @k=CONVERT(INT, RAND()*9)
                PRINT @k
                SET @tong = @tong+@k
                SET @i=@i+1
        END
PRINT 'Tong 5 so nguyen ngau nhien tu 1 den 9 la:' +
CONVERT(CHAR(2),@tong);
```

26

Vòng lặp WHILE – Ví dụ

```
USE tempdb
GO
CREATE TABLE SampleTable
(Id INT, CountryName NVARCHAR(100), ReadStatus TINYINT)
GO
INSERT INTO SampleTable ( Id, CountryName, ReadStatus)
Values (1, 'Germany', 0),
        (2, 'France', 0),
        (3, 'Italy', 0),
        (4, 'Netherlands', 0) ,
        (5, 'Poland', 0)
 SELECT * FROM SampleTable
```


Vòng lặp WHILE – Ví dụ

```
 USE tempdb
GO

DECLARE @Counter INT , @MaxId INT,  @CountryName NVARCHAR(100)
SELECT @Counter = min(Id) , @MaxId = max(Id)
FROM SampleTable

WHILE (@Counter <= @MaxId)
BEGIN
  SELECT @CountryName = CountryName
  FROM SampleTable WHERE Id = @Counter
  PRINT CONVERT(VARCHAR,@Counter) + '. country name is ' + @CountryName
  SET @Counter  = @Counter  + 1
END
```

Bài tập

- 1. Cho biết đoạn chương trình sau làm gì?

```
DECLARE @Songuyen INT
SET @Songuyen=10
WHILE (@Songuyen<13)
        BEGIN
                PRINT  @Songuyen
                SET @Songuyen = @Songuyen + 1
        END
```

29

Bài tập

- 2. Cho biết đoạn chương trình sau làm gì?

```
DECLARE @Songuyen INT
SET @Songuyen=10
WHILE (@Songuyen<14)
  BEGIN
      PRINT  @Songuyen
      IF @Songuyen=11 BREAK
      SET @Songuyen = @Songuyen + 1
  END
```

30

Bài tập

- 1. Cho biết đoạn chương trình sau làm gì?

```
DECLARE @Songuyen INT
SET @Songuyen=10
WHILE (@Songuyen<14)
  BEGIN
      PRINT  @Songuyen
      IF @Songuyen=11 CONTINUE
      SET @Songuyen = @Songuyen + 1
  END
```

31

Bài tập

- 4. Cho biết đoạn chương trình sau làm gì?

```
DECLARE @Songuyen INT
SET @Songuyen=10
WHILE (@Songuyen<14)
      BEGIN
            SET @Songuyen = @Songuyen + 1
            IF @Songuyen=12  CONTINUE
            PRINT  @Songuyen
      END
```

32

Lệnh WAITFOR

- Lệnh WAITFOR dùng để dừng thực hiện chương trình trong 1 khoảng thời gian hay đến một mốc thời gian nào đó
- Cú pháp

  WAITFOR DELAY <'time'>| TIME <'time'>
- Ví dụ

```
-- Trễ 30 giây
WAITFOR DELAY '00:00:30'
-- Trễ 1 giờ, 15 phút và 30 giây
WAITFOR DELAY '01:15:30'
-- Dừng xử lý cho đến 8:00am
WAITFOR TIME '08:00'
```

33

CURSOR trong T-SQL

```
CREATE TABLE tblKhach
(MaKhach char(4) PRIMARY KEY,
TenKhach nvarchar(100) NOT NULL,
DiaChi nvarchar(100) NOT NULL,
DienThoai nvarchar(20) )

INSERT INTO tblKhach VALUES
('K01',N'Xuân',N'Nghệ An','098xxxx'),
('K02',N'Hạ',N'Hà Tĩnh','091xxxx'),
('K03',N'Thu',N'Thanh Hóa','096xxxx'),
('K04',N'Đông',N'Hà Tĩnh',Null),
('K05',N'Tùng',N'Nghệ An','091xxxx')

--Cho biết kết quả đoạn chương trình sau:
DECLARE @name nvarchar(100)
SELECT @name= tenkhach FROM TBLKHACH
PRINT @name
```

34

CURSOR trong T-SQL

- CURSOR là gì?
- Các bước lập trình với CURSOR
- Lập trình CURSOR qua 1 số ví dụ
- Một số cách viết CURSOR trong T-SQL
- Bài tập
- Tự học

35

CURSOR là gì?

- Lệnh SELECT dùng để truy vấn dữ liệu và kết quả trả về dưới dạng bảng (tập hợp các hàng)
- Nhiều khi trong các ứng dụng cần làm việc với một hàng tại một thời điểm thay vì toàn bộ kết quả của bảng, một cách để thực hiện việc này là sử dụng CURSOR
- Cursor là kiểu dữ liệu đặc biệt dùng để dùng lưu trữ kết quả của câu lệnh SELECT và được xem như một tập hợp các dòng của một bảng

36

CURSOR trong T-SQL

- Các bước thực hiện Cursor :
  - Khai báo Cursor (Định nghĩa một Cursor)
  - Mở Cursor
  - Lấy dữ liệu từ Cursor
  - Đóng Cursor
  - Hủy bỏ Cursor

37

Các bước thực hiện CURSOR

- **Khai báo CURSOR**

DECLARE *cursor_name* CURSOR
[LOCAL | GLOBAL] [FORWARD_ONLY | SCROLL]
[STATIC | KEYSET | DYNAMIC]
FOR *select_statement*

Trong đó:

LOCAL | GLOBAL : Xác định phạm vi hoạt động của Cursor
LOCAL (mặc định),

GLOBAL, Cursor đó sẽ có phạm vi ảnh hưởng đến toàn bộ hoạt động của một kết nối.
FORWARD_ONLY | SCROLL: Dùng để điều khiển việc dịch chuyển trỏ bản ghi. Forward_only chỉ cho phép di chuyển bản ghi theo chiều tiến. SCROLL cho phép di chuyển trỏ bản ghi theo cả hai chiều là tiến hoặc lùi.

(*Forward_only là mặc định nếu không chỉ rõ STATIC | KEYSET | DYNAMIC khi khai báo kiểu Cursor.*

Ngược lại khi khai báo có chỉ rõ STATIC | KEYSET | DYNAMIC thì mặc định là SCROLL)

38

Các bước thực hiện CURSOR

- **Khai báo CURSOR**

DECLARE *cursor_name* CURSOR
[LOCAL | GLOBAL] [FORWARD_ONLY | SCROLL]
[STATIC | KEYSET | DYNAMIC]
FOR *select_statement*

Trong đó: STATIC | DYNAMIC| KEYSET Dùng để phân loại Cursor.
- STATIC: vùng làm việc của con trỏ không bị ảnh hưởng khi dữ liệu nguồn thay đổi

- DYNAMIC: mọi sự thay đổi dữ liệu trên các bảng nguồn của câu lệnh select sẽ tác động đến kết quả mà con trỏ đang duyệt

- KEYSET: gần giống DYNAMIC, các thay đổi dữ liệu trên các cột không là khóa chính trong bảng cơ sở sẽ được cập nhật trong dữ liệu cursor. Tuy nhiên đối với các dòng vừa thêm mới hoặc bị xóa bởi những người dùng khác sẽ không được hiển thị trong dữ liệu cursor có kiểu là KEYSET

Các bước thực hiện CURSOR

- **Mở cursor: OPEN <Tên_CURSOR>**
- **Lấy dữ liệu từ cursor**

Sau khi mở Cursor lệnh để lấy giá trị của các cột trong Cursor:

FETCH [Next | Prior | First | Last] FROM Tên_Cursor
INTO <các_biến_cục_bộ>

Trong đó:
FETCH: Sử dụng để lấy dữ liệu từ Cursor
NEXT: Nhảy đến dòng kế tiếp
PRIOR: Nhảy về dòng trước
FIRST: Nhảy đến dòng đầu tiên
LAST: Nhảy đến dòng cuối cùng
…
Chú ý: Nếu sử dụng FETCH NEXT cho lần đầu tiên trong Cursor sẽ tương đương với FETCH FIRST.

Các bước thực hiện CURSOR

Để đọc dữ liệu ta cần kiểm tra trạng thái Cursor:

- Sử dụng biến hệ thống @@Fetch_Status.

Giá trị của @@Fetch_Status là:

0: Nếu lấy mẩu tin thành công

-1: Nếu lấy mẩu tin thất bại (Do cursor đang ở BOF hoặc EOF)

-2: Nếu lấy mẩu tin thất bại, với lý do không tồn tại.

- Kết hợp biến hệ thống @@Fetch_Status với lệnh WHILE để duyệt từng dòng trong CURSOR

41

Các bước thực hiện CURSOR

Đóng và giải phóng CURSOR

- Đóng cursor:

  CLOSE <Cursor_name>

- Giải phóng cursor:

  DEALLOCATE <Cursor_name>

42

Ví dụ CURSOR trong T-SQL

- Cơ sở dữ liệu

```
CREATE TABLE tblSanPham
(id int PRIMARY KEY,
Ten NVARCHAR(100) NOT NULL,
Soluong int)
```

- Nhập dữ liệu

1	Bút bi	10
2	Phấn	5
3	Vở A4	20
…		

43

Ví dụ CURSOR trong T-SQL

```
DECLARE @id int, @title NVARCHAR(100), @quanlity int

DECLARE cursorSP CURSOR FOR
SELECT id, ten, soluong FROM tblSanPham

OPEN cursorSP

FETCH NEXT FROM cursorSP INTO @id, @title, @quanlity

WHILE @@FETCH_STATUS = 0
BEGIN
   PRINT 'ID:' + CAST(@id as nvarchar)
   PRINT 'TITLE:' + @title
         PRINT 'QUANLITY:' + CAST(@quanlity as nvarchar(10))
   FETCH NEXT FROM cursorSP INTO @id, @title, @quanlity
END
CLOSE cursorSP
DEALLOCATE cursorSP
```

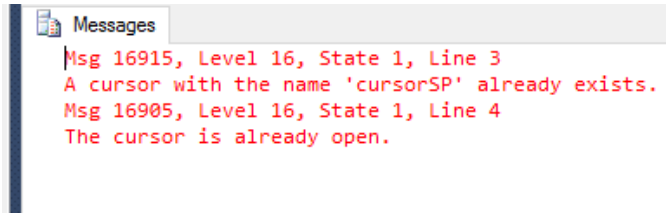
44

Ví dụ CURSOR trong T-SQL

Trong lập trình CURSOR mà thiếu 2 lệnh:

- CLOSE cursorSP
- DEALLOCATE cursorSP

Thì lỗi sẽ xảy ra khi chạy lại đoạn chương trình chứa cursor đó



CURSOR như 1 biến trong T-SQL

```
DECLARE @id int, @title NVARCHAR(100), @quanlity int

DECLARE @cursorSP CURSOR
SET @cursorSP = CURSOR
FOR SELECT id, ten, soluong FROM tblSanPham

OPEN @cursorSP
FETCH NEXT FROM @cursorSP INTO @id, @title, @quanlity
WHILE @@FETCH_STATUS = 0
BEGIN
   PRINT 'ID:' + CAST(@id as nvarchar)
   PRINT 'TITLE:' + @title
         PRINT 'QUANLITY:' + CAST(@quanlity as nvarchar(10))
   FETCH NEXT FROM @cursorSP INTO @id, @title, @quanlity
END
CLOSE @cursorSP
DEALLOCATE @cursorSP
-- Ưu điểm
```

Hướng dẫn tự học

- Thực hiện tạo lập và nhập dữ liệu vào cơ sở dữ liệu sử dụng giao diện trong SSMS (không viết code)

  Tham khảo: https://blogloi.com/tao-cau-truc-bang-trong-sql-server-2012/

  https://giasutinhoc.vn/labs/lab-sql-server/huong-dan-them-du-lieu-vao-bang-trong-sql-server/
- Ôn tập phép join, truy vấn lồng

47

Tóm tắt chương 2

- Biến
- Phép gán
- Xử lý lô (batch)
- Cấu trúc điều khiển
- Tạo lập và nhập dữ liệu sử dụng giao diện trong SSMS (không viết code)

48