I. Tóm tắt bài thực hành

1. Yêu cầu lý thuyết

Sinh viên đã được trang bị kiến thức:

- Các kiến thức cơ bản, nền tảng về CSDL, đặc biệt CSDL quan hệ: mô hình dữ liệu quan hệ (Relational Data Model).
- Các ngôn ngữ truy vấn, các kỹ năng khai báo, truy vấn một CSDL quan hệ với một hệ quản trị CSDL

2. Nội dung

❖ **Tìm hiểu cơ sở dữ liệu Quản lý thư viện**

- Tìm hiểu cấu trúc CSDL Quản lý Thư viện.
- Tìm hiểu các quy trình nghiệp vụ đăng ký, mượn sách, trả sách.
- Tìm hiểu các quy định và thống kê trong hệ thống Quản lý Thư viện.

❖ **Tìm hiểu các cấu trúc điều khiển.**

❖ **Tìm hiểu Cursor.**

❖ **Tìm hiểu Stored Procedure.**

II. Các cấu trúc điều khiển

1. Cấu trúc rẽ nhánh IF…ELSE

Cú pháp:

```
IF biểu_thức_điều_kiện BEGIN
      các_lệnh_xử_lý_khi_thỏa_điều_kiện
END
ELSE BEGIN
      các_lệnh_xử_lý_khi_không_thỏa_điều_kiện
END
```

Ví dụ:

```
DECLARE @soLuong int
SELECT @soLuong = COUNT(*) FROM SINHVIEN

IF @soLuong > 0 BEGIN
      PRINT @soLuong
END
ELSE BEGIN
      PRINT N'Không có sinh viên'
END
```

❖ **Kết hợp IF với truy vấn**

Ta có thể kết hợp lệnh IF với các câu truy vấn để kiểm tra điều kiện trả về trên câu truy vấn.

✓ Kết hợp IF với EXISTS

```
IF EXISTS(SELECT * FROM SINHVIEN) BEGIN
	PRINT N'Có dữ liệu sinh viên'
END
ELSE BEGIN
	PRINT N'Không có dữ liệu sinh viên'
END
```

✓ Kết hợp IF với các dạng truy vấn khác

```
IF (SELECT COUNT(*) FROM SINHVIEN) > 0 BEGIN
	PRINT N'Có dữ liệu sinh viên'
END
ELSE BEGIN
	PRINT N'Không có dữ liệu sinh viên'
END
```

2. Cấu trúc lặp WHILE

Cú pháp:

```
WHILE biểu_thức_điều_kiện BEGIN
	các_lệnh_xử_lý_khi_thỏa_điều_kiện
END
```

Ví dụ:

```
DECLARE @i INT
SET @i = 1

WHILE @i < 100 BEGIN
	PRINT @i
	SET @i = @i + 1
END
```

Ví dụ trên sẽ in ra dãy các con số từ 1 đến 99.

3. Cấu trúc CASE…WHEN

Câu lệnh CASE cho phép nhận một giá trị từ nhiều lựa chọn. Trường hợp có nhiều câu lệnh IF…ELSE lồng nhau, gây phức tạp câu lệnh, nên thay thế nó bằng câu lệnh CASE.

Cú pháp:

```
CASE <giá trị biểu thức Case>
	WHEN <điều kiện 1> THEN <kết quả tương ứng 1>
	WHEN <điều kiện 2> THEN <kết quả tương ứng 2>
	WHEN <điều kiện 3> THEN <kết quả tương ứng 3>
	…
```

```
        ELSE <kết quả tương ứng>
    END
```

Ngoài các cấu trúc điều khiển trên còn có các cấu trúc khác như: GOTO, WAITFOR… sinh viên sẽ tự tìm hiểu thêm ở nhà.

III. 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. Stored Procedure

1. Giới thiệu

Khi chúng ta tạo một ứng dụng sử dụng HQT CSDL Microsoft SQL Server, ngôn ngữ lập trình Transact-SQL sẽ là ngôn ngữ chính giao tiếp giữa ứng dụng và database. Khi chúng ta tạo các chương trình bằng Transact-SQL, hai phương pháp chính có thể dùng để lưu trữ và thực thi cho các chương trình là:

- ✓ Chúng ta có thể lưu trữ các chương trình cục bộ và tạo các ứng dụng để gởi các lệnh đến SQL Server và xử lý các kết quả,

- ✓ Chúng ta có thể lưu trữ những chương trình như các stored procedure trong SQL Server và tạo ứng dụng để gọi thực thi các stored procedure và xử lý các kết quả.

Đặc tính của Stored-procedure trong SQL Server :

- ✓ Chấp nhận những tham số vào và trả về những giá trị được chứa trong các tham số ra để gọi những thủ tục hoặc xử lý theo lô.
- ✓ Chứa các lệnh của chương trình để thực hiện các xử lý trong database, bao gồm cả lệnh gọi các thủ tục khác thực thi.
- ✓ Trả về các trạng thái giá trị để gọi những thủ tục hoặc thực hiện các xử lý theo lô để cho biết việc thực hiện thành công hay thất bại, nếu thất bại thì lý do vì sao thất bại.

Ta có thể dùng Transact–SQL EXCUTE để thực thi các stored procedure. Stored procedure khác với các hàm xử lý là giá trị trả về của chúng không chứa trong tên và chúng không được sử dụng trực tiếp trong biểu thức.

Stored procedure có những thuận lợi so với các chương trình Transact-SQL lưu trữ cục bộ là:

- ✓ **Stored procedure cho phép điều chỉnh chương trình cho phù hợp**: Chúng ta có chỉ tạo stored procedure một lần và lưu trữ trong database một lần, trong chương trình chúng ta có thể gọi nó với số lần bất kỳ. Stored procedure có thể được chỉ rõ do một người nào đó tạo ra và sự thay đổi của chúng hoàn toàn độc lập với source code của chương trình.
- ✓ **Stored procedure cho phép thực thi nhanh hơn**: nếu sự xử lý yêu cầu một đoạn source code Transact – SQL khá lớn hoặc việc thực thi mang tính lặp đi lặp lại thì stored procedure thực hiện nhanh hơn việc thực hiện hàng loạt các lệnh Transact-SQL. Chúng được phân tích cú pháp và tối ưu hóa trong lần thực thi đầu tiên và một phiên bản dịch của chúng trong đó sẽ được lưu trong bộ nhớ để sử dụng cho lần sau, nghĩa là trong những lần thực hiện sau chúng không cần phải phân tích cú pháp và tối ưu lại, mà chúng sẽ sử dụng kết quả đã được biên dịch trong lần đầu tiên.
- ✓ **Stored procedure có thể làm giảm bớt vấn đề kẹt đường truyền mạng**: giả sử một xử lý mà có sử dụng hàng trăm lệnh của Transact-SQL và việc thực hiện thông qua từng dòng lệnh đơn, như vậy việc thực thông qua stored procedure sẽ tốt hơn, vì nếu không khi thực hiện chúng ta phải gởi hàng trăm lệnh đó lên mạng và điều này sẽ dẫn đến tình trạng kẹt mạng.
- ✓ **Stored procedure có thể sử dụng trong vấn đề bảo mật của máy**: vì người sử dụng có thể được phân cấp những quyền để sử dụng các stored procedure này, thậm chí họ không được phép thực thi trực tiếp những stored procedure này.

2. Định nghĩa

Một Stored procedure được định nghĩa gồm những thành phần chính sau:

- ✓ Tên của Stored procedure

- ✓ Các tham số.
- ✓ Thân của stored procedure: bao gồm các lệnh của Transact-SQL dùng để thực thi procedure.

Một stored procedure được tạo bằng lệnh Create Procedure, và có thể thay đổi bằng cách dùng lệnh Alter

Procedure, và có thể xóa bằng cách dùng lệnh Drop Procedure trong lập lệnh của Transact – SQL

3. Cú pháp

3.1. Lệnh tạo Procedure

```
CREATE PROCEDURE procedure_name
{@parameter data_type input/output}/*các biến tham số vào ra*/
AS
BEGIN
      [khai báo các biến cho xử lý]
      {Các câu lệnh transact-sql}
END
```

Ghi chú:

- ✓ Trong SQL Server, có thể ghi tắt một số từ khóa mà tên có chiều dài hơn 4 ký tự. Ví dụ: có thể thay thế Create Procedure bằng Create Proc.
- ✓ Tên hàm, tên biến trong SQL Server không phân biệt hoa thường.

3.2. Khai báo biến và gán giá trị cho biến, ghi chú

```
/*Khai báo biến*/
DECLARE @parameter_name data_type
/*Gán giá trị cho biến*/
SET @parameter_name = value
SELECT @parameter_name = value
/*In thông báo ra màn hình*/
print N'Chuỗi thông báo unicode'
--Ghi chú 1, một dòng
/*
      Ghi chú 2
      Nhiều dòng
*/
```

3.3. Biên dịch và gọi thực thi một stored-procedure

Biên dịch : Chọn toàn bộ mã lệnh Tạo stored-procedure → Nhấn F5

Gọi thực thi một store-Procedure đã được biên dịch bằng lệnh exec:

```
EXECUTE procedure_name --Stored-proc không tham số
EXEC procedure_name Para1_value, Para2_value, … /*Stored-proc có tham số*/
```

3.4. Lệnh cập nhật Procedure

```
ALTER PROCEDURE procedure_name
[ {@parameter data_type } ]
AS
BEGIN
      [khai báo các biến cho xử lý]
     {Các câu lệnh transact-sql}
END
```

3.5. Lệnh xóa Procedure

```
DROP PROCEDURE procedure_name
```

4. Ví dụ

Tạo stored-procedure tính tổng của 2 số nguyên

```
--Tạo stored-procedure sp_Tong
CREATE PROCEDURE sp_Tong
      @So1 int, @So2 int, @Tong int out
AS
BEGIN
      SET @Tong = @So1 + @So2;
END
--Biên dịch stored-procedure → F5
--Kiểm tra
Declare @Sum int Exec sp_Tong 1, -2, out @Sum
Select @Sum
```

V. Bài tập yêu cầu

1. Bài tập 1

Cài đặt CSDL Quản lý Thư viện bằng hệ quản trị CSDL Microsoft® SQL Server. Tiến hành nhập một vài dữ liệu mẫu.

2. Bài tập 2

Vào Query Analysis, viết các Stored-procedure sau trong CSDL hiện hành

2.1. Stored procedure đơn giản

a. Viết stored-procedure In ra dòng 'Hello'
b. Viết stored-procedure In ra dòng 'Xin chào'.
c. Viết stored-procedure In ra dòng 'Xin chào' + @ten với @ten là tham số đầu vào là tên của bạn.
d. Viết stored-procedure In ra dòng 'Xin chào' + @ten với @ten là tham số đầu vào là tên Tiếng Việt có dấu của bạn. Gợi ý :
   - Sử dụng UniKey để gõ Tiếng Việt
   - Chuỗi unicode phải bắt đầu bởi N (vd: N'Tiếng Việt')
   - Dùng hàm cast (<biểuThức> as <kiểu>) để đổi thành kiểu <kiểu> của <biểuThức>
e. Viết stored-procedure Nhập vào 2 số @s1, @s2. In ra tổng @s1+@s2.
f. Viết stored-procedure Nhập vào 2 số @s1, @s2. In ra câu 'Tổng là : @tg' với @tg=@s1+@s2.
g. Viết stored-procedure Nhập vào 2 số @s1, @s2. Xuất tổng @s1+@s2 ra tham số @tong. Cho thực thi và in giá trị của tham số này để kiểm tra.
h. Viết stored-procedure Nhập vào 2 số @s1, @s2. Xuất tổng @s1+@s2 ra tham số @tong. Cho thực thi và in giá trị của tham số này để kiểm tra dưới dạng 'Tổng là: @tg' với @tg =@s1+@s2.
i. Viết stored-procedure Nhập vào 2 số @s1, @s2. In ra max của chúng.
j. Viết stored-procedure Nhập vào 2 số @s1, @s2. In ra câu 'Số lớn nhất của @s1 và @s2 là @max' với @s1, @s2, max là các giá trị tương ứng.
k. Viết stored-procedure Nhập vào 2 số @s1, @s2. Xuất min và max của chúng ra tham số @max, @min. Cho thực thi và in giá trị của các tham số này để kiểm tra.

2.2. Stored procedure có vòng lặp

l. Viết stored-procedure Nhập vào số nguyên @n. In ra các số từ 1 đến @n.
m. Viết stored-procedure Nhập vào số nguyên @n. In ra tổng các số chẵn từ 1 đến @n
n. Viết stored-procedure Nhập vào số nguyên @n. In ra tổng, và số lượng các số chẵn từ 1 đến @n
o. Viết stored-procedure Nhập vào 2 số. In ra ước chung lớn nhất của chúng theo gợi ý dưới đây.
   ✓ Không mất tính tổng quát giả sử a <= A

- ✓ Nếu A chia hết cho a thì: (a,A) = a ngược lại : (a,A) = (A%a,a) hoặc (a,A) = (a,A-a)
- ✓ Lặp lại b1,b2 cho đến khi điều kiện trong b2 được thỏa

p. Viết stored-procedure Nhập vào 2 số nguyên @s1, @s2. Xuất ước chung lớn nhất của @s1 và @s2 ra tham số @gcd. Cho thực thi và in bằng lệnh select giá trị của tham số này để kiểm tra dưới dạng 'Kết quả: `ucln(@s1,@s2) = @gcd`' trong đó thay thế @s1, @s2, @gcd bởi các giá trị tương ứng.

2.3. Stored procedure đệ quy

q. Viết stored-procedure Cài đặt có dùng đệ quy, thuật toán Euler tìm ước chung lớn nhất (a, A).

r. Viết stored-procedure Nhập vào 2 số. In ra ước chung lớn nhất của chúng. Bắt buộc viết bằng đệ quy.

s. Viết stored-procedure Nhập vào số nguyên @n <= 5. In ra tất cả các số nhị phân có @n bit.

Ví dụ: @n=2 thì kết quả in được là
00
01
10
11

3. Bài tập 3

t. Viết một stored proc có nội dung: Dùng lệnh print để in ra danh sách mã các tựa sách.

u. Viết một stored proc có nội dung: Dùng lệnh print để in ra danh sách mã và họ tên các độc giả.

4. Bài tập 4

Hoàn thành tất cả các bài tập Stored procedure, trigger trong file đặc tả cơ sở dữ liệu Quản lý thư viện.

~ HẾT ~