Nguyen Tat Thanh Institute of International Education (NIIE)

DATABASE MANAGEMENT SYSTEMS

(Creadits 3)

MSc. Luong Tran Ngoc Khiet

May - 2021

Course content

Chap 3. Programming with Cursors

MSc. Luong Tran Ngoc Khiet

NTT Institute of International Education (NIIE)

Programming with cursors

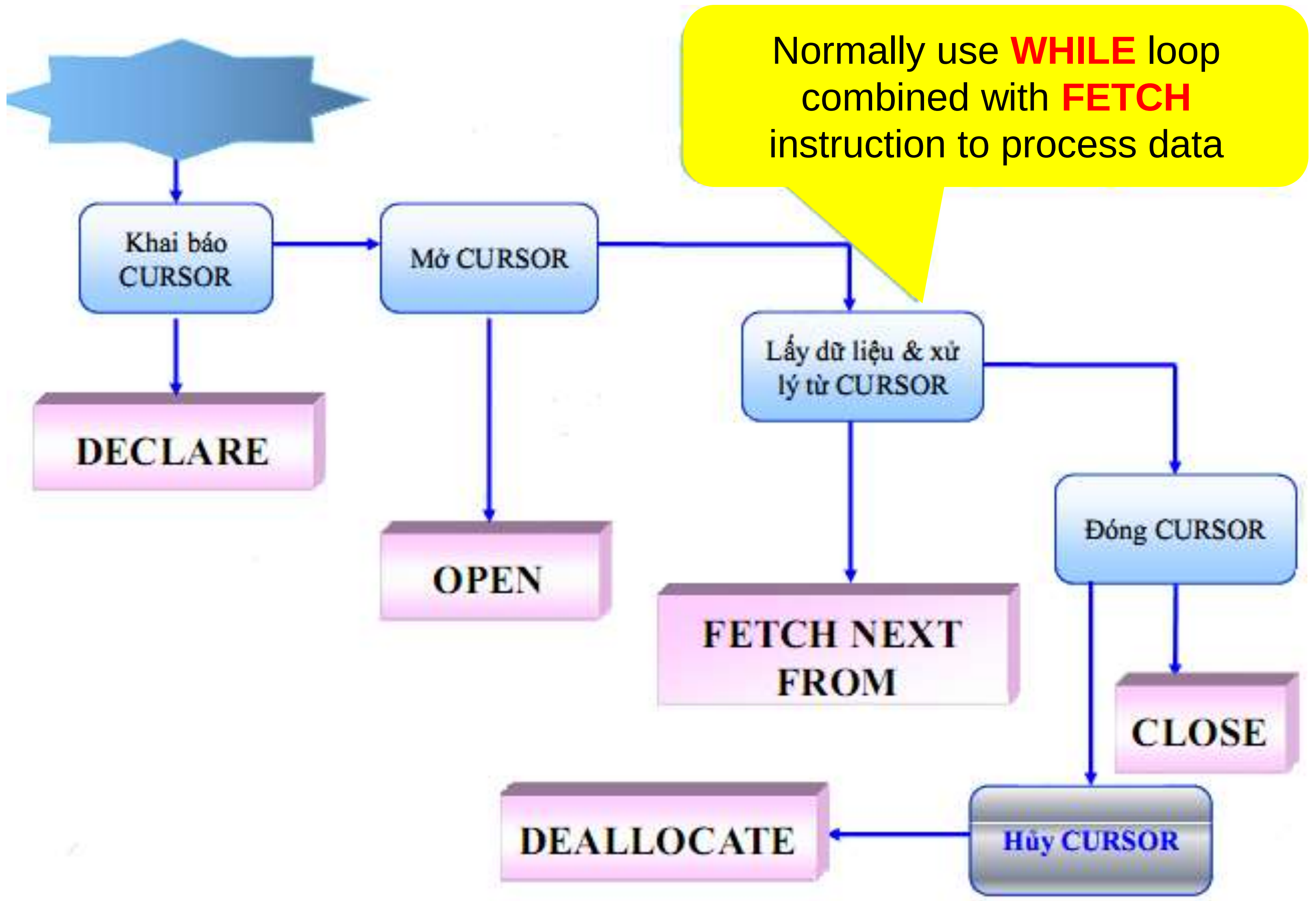
- A cursor is a database object used by an application to manipulate rows of data instead of collections of data.
- Pointers are used Procedure and Trigger

Programming with cursors

❑ With pointers we can:

- Allows positioning of specified rows of a result set.
- Gets a single row or set of rows from the current position of the result set.
- Supports modifying the row's data at the current position in the result set.
- Supports multiple levels of visibility into changes made by other users on the result set data.

Cursors handling process



Normally use WHILE loop combined with FETCH instruction to process data
Khai báo CURSOR
DECLARE
Mở CURSOR
OPEN
Lấy dữ liệu & xử lý từ CURSOR
FETCH NEXT FROM
Đóng CURSOR
CLOSE
Hủy CURSOR
DEALLOCATE

Declare cursor

- Command DECLARE used to create a cursor.
- It contains SELECT command to include records from the table.
- Syntax:

```
DECLARE  <Cursor_Name> CURSOR
FOR             <Select Statements>
[FOR UPDATE [OF Column_name[,….N]]]
```

Declare cursor

- Full syntax :

```
DECLARE  <Cursor_Name> CURSOR
[LOCAL | GLOBAL]
[FORWARD ONLY | SCROLL]
[STATIC | KEYSET | DYNAMIC |FAST_FORWARD]
[READ_ONLY | SCROLL_LOCKS | OPTIMISTIC]
[TYPE_WARNING]
FOR <Select Statements>
[FOR UPDATE [OF Column_name[,….N]]]
```

Declare cursor parameter

- Scope: [LOCAL | GLOBAL]
  - Local :chỉ sử dụng trong phạm vi khai báo(mặc định)
  - Global :sử dụng chung cho cả kết nối

- Move: [FORWARD ONLY | SCROLL]
  - ForWard_Only :chỉ di chuyển một hướng từ trước ra sau(mặc định)
  - Scroll : di chuyển tùy ý

Declare cursor parameter

- Status: [STATIC | KEYSET | DYNAMIC]
  - Static : dữ liệu trên Cursor không thay đổi mặt dù dữ liệu trong bảng nguồn thay đổi (mặc định)
  - Dynamic :dữ liệu trên Cursor sẽ thay đổi khi dữ liệu trong bảng nguồn thay đổi
  - KeySet :giống Dynamic nhưng chỉ thay đổi những dòng bị cập nhật

Declare cursor parameter

- Process: [READ_ONLY | SCROLL_LOCKS]
  - Read_Only :chỉ đọc(mặc định)
  - Scroll_Lock : đọc/ghi
- The select statement: does not contain Into, Compute, Compute by clauses
- Updated column list: is a list of columns that will be changed

Steps to use cursors

- Mở con trỏ:

  OPEN <Cursor_name>

- Duyệt và xử lý dữ liệu trong cursor :

  FETCH <Cursor_name>

- Đóng con trỏ:

  CLOSE <Cursor_name>

- Xoá các tham chiếu tới con trỏ:

  DEALLOCATE <Cursor_name>

Retrieve and traverse the cursor

- For example:

FETCH direction From cursor_name Into list varible

```
declare Cur_MatHang CurSor
for     select MaMH,tenmh from MatHang
open Cur_MatHang
declare @maMH char(4),@tenMH varchar(100)
while 0=0
    begin
        fetch next from Cur_MatHang into @maMH,@tenMH
        if @@fetch_status<>0 break
        print 'Mã mặt hàng :' + @maMH +' Tên mặt hàng :' +
        @tenMH
    end
close Cur_MatHang
deallocate Cur_MatHang
```

Retrieve and traverse the cursor

- FETCH FIRST: Retrieve the first row.
- FETCH NEXT: Retrieve the next row of the previous row.
- FETCH PRIOR: Retrieve the previous row of the previous retrieved row.
- FETCH LAST: Retrieve the last row.

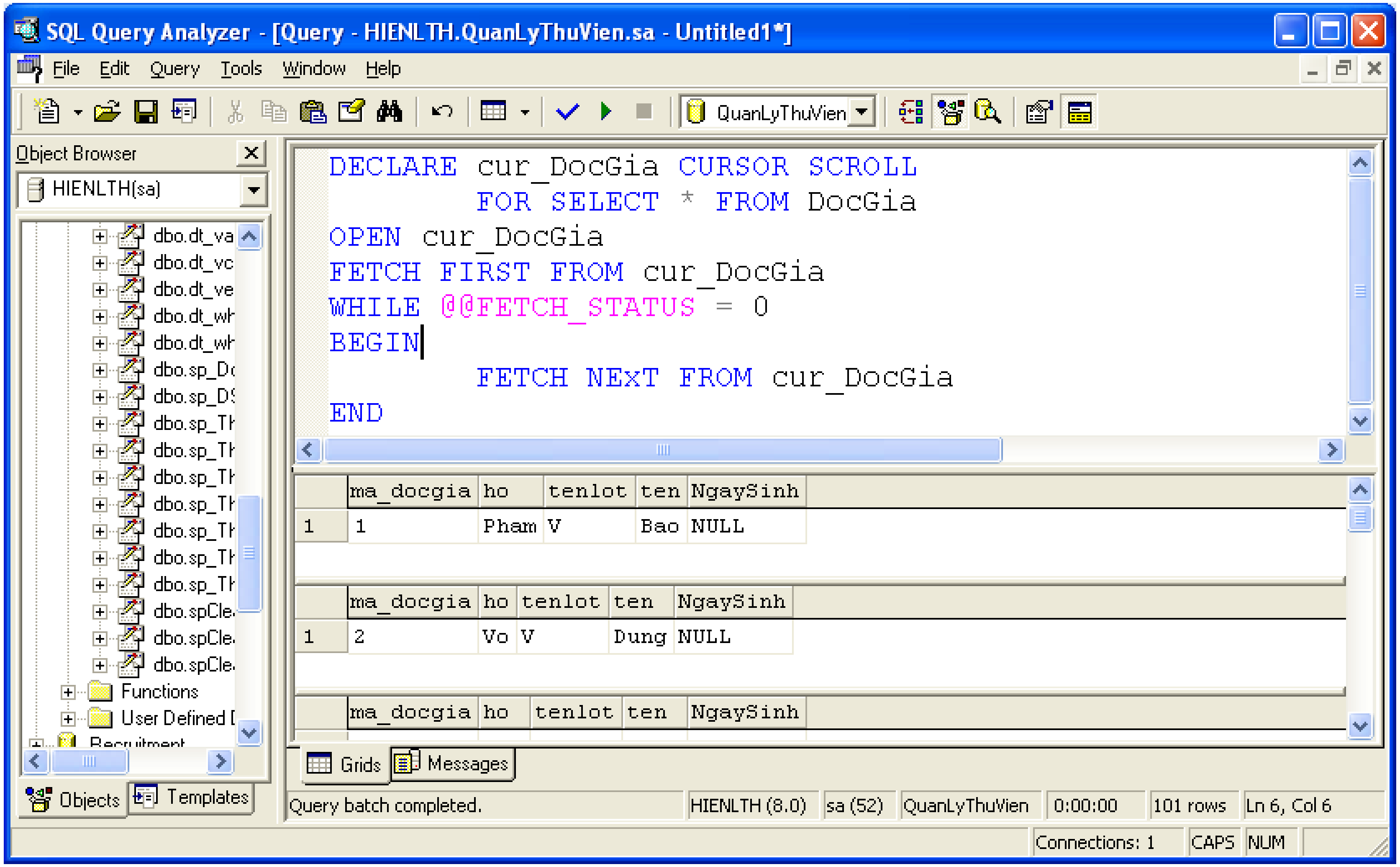
Retrieve and traverse the cursor

- FETCH ABSOLUTE *n*: Move to the n-th record from the first record
  - If n is a positive integer, it will retrieve n rows in the cursor.
  - If n is a negative integer, the n rows before the last row in the cursor are retrieved.
  - If n is 0, no rows are retrieved.
  - Example: FETCH Absolute 2 will display the second record of a table.
- FETCH RELATIVE *n*: Move to the nth record from the current record
  - If n is negative, n rows before the previously retrieved row are retrieved.
  - If n is 0, the current row is received.

Global variables of the FETCH command

- @@FETCH _STATUS: This variable returns an integer representing the result of the last access of the cursor..
  - **@@FETCH_STATUS return <>0 if failed**
  - **@@FETCH_STATUS return = 0 if sucessfull**
- @@CURSOR_ROWS: This variable returns the total number of rows currently in the open cursor.

For example using cursor



```
DECLARE cur_DocGia CURSOR SCROLL
        FOR SELECT * FROM DocGia
OPEN cur_DocGia
FETCH FIRST FROM cur_DocGia
WHILE @@FETCH_STATUS = 0
BEGIN
        FETCH NExT FROM cur_DocGia
END
```

	ma_docgia	ho	tenlot	ten	NgaySinh
1	1	Pham	V	Bao	NULL

	ma_docgia	ho	tenlot	ten	NgaySinh
1	2	Vo	V	Dung	NULL

	ma_docgia	ho	tenlot	ten	NgaySinh

Query batch completed. | HIENLTH (8.0) | sa (52) | QuanLyThuVien | 0:00:00 | 101 rows | Ln 6, Col 6

Connections: 1 | CAPS | NUM

Cursor

- A cursor is a database object used by an application to manipulate rows of data instead of collections of data. Using cursors, multiple operations can be performed row-by-row on the result set, which may or may not require the presence of the original table.

Cursor

- The cursor is created using the **DECLARE** command. First the pointer is declared and created in memory. Only then will it be opened..
- The **OPEN** command opens the cursor. Retrieving records from a cursor is called fetching. A user can only receive one record at a time.
- The **FETCH** instruction is used to read records from the cursor.

Con trỏ (tt)

- By default, a cursor is **forward only**. It can retrieve records sequentially from the first record to the last record. It cannot directly retrieve the 1st or last row in a table.
- When a pointer is temporarily not needed, it can be closed with the **CLOSE** command.
- Whenever the pointer is not used, references to it should be removed with the **DEALLOCATE** command

STORED PROCEDURE

Stored Procedure

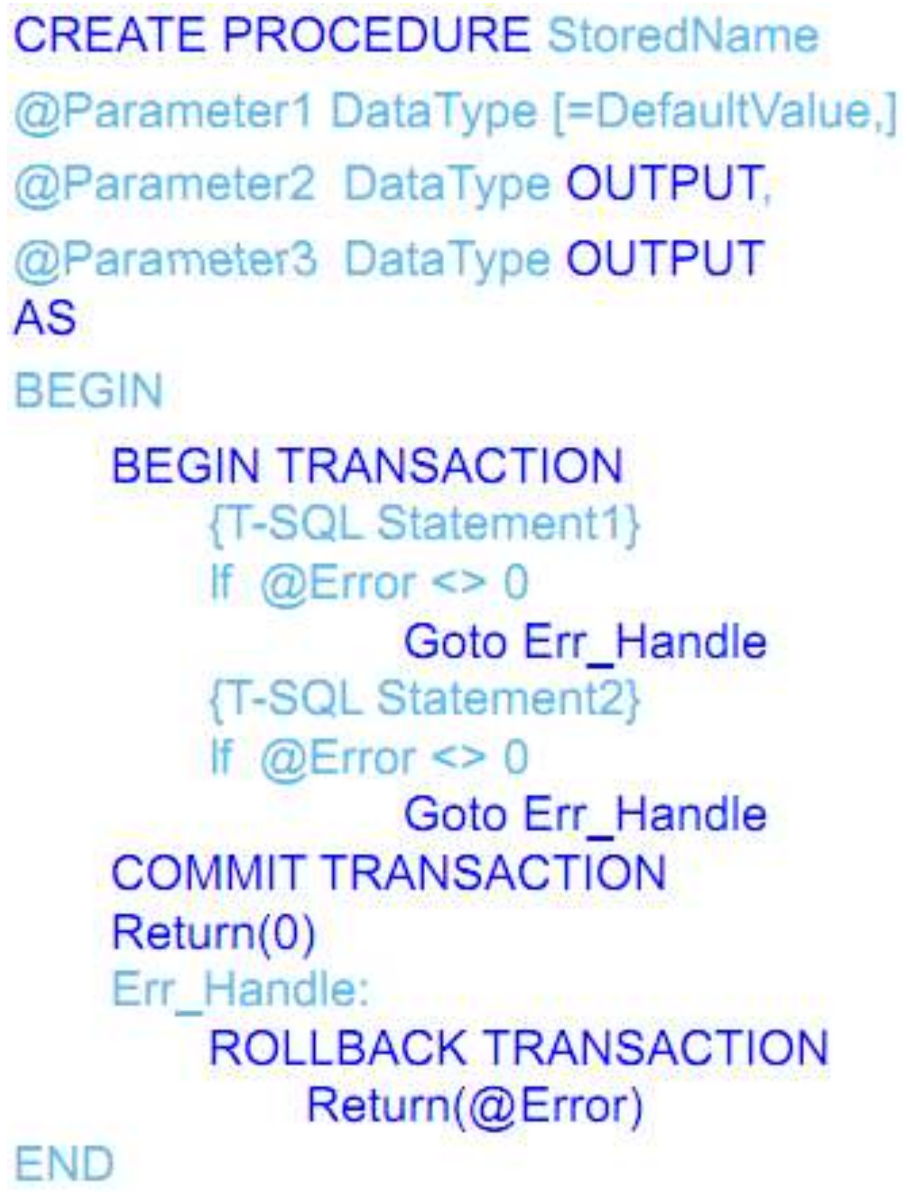
- Allows module-oriented programming
- Execute faster, reduce network connection usage
- Security
- Processing functions and sharing with other applications

Stored Procedure

- ***Syntax:***

```
CREATE PROCEDURE proc_name
AS
BEGIN
                sql_statement1
                sql_statement2
END
```

Stored Procedure Syntax



```
CREATE PROCEDURE StoredName
@Parameter1 DataType [=DefaultValue,]
@Parameter2  DataType OUTPUT,
@Parameter3  DataType OUTPUT
AS
BEGIN
    BEGIN TRANSACTION
        {T-SQL Statement1}
        If  @Error <> 0
                Goto Err_Handle
        {T-SQL Statement2}
        If  @Error <> 0
                Goto Err_Handle
    COMMIT TRANSACTION
    Return(0)
    Err_Handle:
        ROLLBACK TRANSACTION
            Return(@Error)
END
```

Example 1 – Store procedure without parameters

```
CREATE PROCEDURE  sp_XemDSSV
AS
BEGIN
    PRINT N'DANH SÁCH SINH VIÊN'
    SELECT MSSV, HoLot, Ten, NgaySinh,
        NoiSinh, DiaChi
    FROM SinhVien
END
```

Example 1 – Store procedure with parameters

```
CREATE PROCEDURE  sp_XemSV
        @MaSV nvarchar(11)
AS
BEGIN
    PRINT N'SINH VIÊN'
    SELECT HoLot, Ten, NgaySinh,
        NoiSinh, DiaChi
    FROM SinhVien
    WHERE MSSV = @MaSV
END
```

See Store Procedure content

- *Syntax:*

  sp_helptext proc_name

For example:

- *Open Query Analyzer, typing:*

  sp_helptext ***sp_XemDSSV***

  sp_helptext ***sp_XemSV***

- *Check spelling and procedure content.*

Excecute Stored Procedure

- ***Syntax:***

  EXECUTE proc_name parameter_list

  or

  EXEC proc_name parameter_list

  or

  proc_name parameter_list

Tips: Each parameter is separated by a comma

For example

- ***Open Query Analyzer, typing:***

  EXECUTE sp_XemDSSV

  EXECUTE sp_XemSV 'K29.103.010'

  or

  EXEC sp_XemDSSV

  EXEC sp_XemSV 'K29.103.010'

  or

  sp_XemDSSV

  sp_XemSV 'K29.103.010'

- ***Press F5 to execute***

Discussion