**Transact-SQL Cursors**

Transact-SQL cursors là kiểu con trỏ thường được dùng trong stored procedures, triggers. Nó quản lý 1 tập các records - là kết quả trả về của 1 phát biểu SQL-Select liên kết với cursor. Ta dùng cursor trong các trường hợp khi muốn xử lý từng mẫu tin trong cursor theo các cách thức khác nhau, hoặc nhận kết quả trả về từ 1 Stored Procedure.

Tiến trình khi sử dụng cursor trong Stored Procedure hoặc Trigger là:

1. **Khai báo** các biến để chứa dữ liệu trả về từ cursor. Các biến này phải cùng kiểu dữ liệu với các fields trong SQL-Select.
2. **Liên kết 1** Transact-SQL cursor với 1 SELECT-SQL qua phát biểu DECLARE CURSOR như sau:

DECLARE ***cursor\_name*** CURSOR [ LOCAL | GLOBAL ]

[ FORWARD\_ONLY | SCROLL ]

[ STATIC | KEYSET | DYNAMIC | FAST\_FORWARD ] [ READ\_ONLY | SCROLL\_LOCKS | OPTIMISTIC ]

[ TYPE\_WARNING ]

FOR *select\_statement*

[ FOR UPDATE [ OF *column\_name* [ **,**...*n* ] ] ]

1. **Mở cursor** : OPEN { { [ GLOBAL ] *cursor\_name* } | *cursor\_variable\_name* }
2. **Lấy dữ liệu** từ 1 mẩu tin trong cursor và đưa dữ liệu đó vào các biến (đã khai báo trong bước 1) qua lệnh FETCH INTO. Transact-SQL cursors không cho phép lấy nhiều mẫu tin cùng 1 lúc.

# 5. Ěóng cursor

**I. Khai báo biến dữ liệu :** ta dùng lệnh declare để khai báo các biến nhằm để lưu trữ 1 record lấy từ cursor. Các biến này có thể được sử dụng trong các lệnh Transact SQL khác trong Stored Procedure hoặc Trigger.

**II. Liên kết 1 Transact-SQL cursor với 1 SQL-SELECT** : Ta có thể làm việc trực tiếp trên cursor hoặc qua biến cursor.

1. Làm việc trực tiếp trên cursor : chẳng hạn như ta tạo 1 cursor tên Employee\_Cursor qua phát biểu Declare như sau:

DECLARE Employee\_Cursor CURSOR FOR

SELECT LastName, FirstName FROM Northwind.dbo.Employees WHERE LastName like 'B%'

OPEN Employee\_Cursor -- Mở cursor

FETCH NEXT FROM Employee\_Cursor WHILE @@FETCH\_STATUS = 0 BEGIN

FETCH NEXT FROM Employee\_Cursor END

CLOSE Employee\_Cursor -- Ěóng cursor

DEALLOCATE Employee\_Cursor

1. Tạo biến cursor : ta có 2 cách:
   1. Tạo trước cursor, sau đó gán nó cho biến cursor: DECLARE @MyVariable CURSOR

DECLARE MyCursor CURSOR FOR

SELECT LastName FROM Northwind.dbo.Employees SET @MyVariable = MyCursor

* 1. Liên kết trực tiếp Select-SQL với biến cursor: DECLARE @MyVariable CURSOR

SET @MyVariable = CURSOR SCROLL KEYSET FOR

SELECT LastName FROM Northwind.dbo.Employees

Sau khi 1 cursor đã được liên kết với 1 biến **cursor**, biến **cursor** có thể được dùng thay cho tên cursor. Trong Stored procedure, ta cǜng có thể khai báo output parameters có kiểu dữ liệu là **cursor** và được liên kết với 1 cursor.

*Cú pháp lệnh tạo cursor:*

DECLARE *cursor\_name* CURSOR [ LOCAL | GLOBAL ]

[ FORWARD\_ONLY | SCROLL ]

[ STATIC | KEYSET | DYNAMIC | FAST\_FORWARD ] [ READ\_ONLY | SCROLL\_LOCKS ]

FOR *select\_statement*

[ FOR UPDATE [ OF *column\_name* [ **,**...*n* ] ] ]

* cursor\_name : tên cursor do ta đặt.
* LOCAL : phạm vi hoạt động của cursor là cục bộ trong 1 gói, SP, hoặc trigger (nơi mà cursor được tạo) . Cursor được tự động giải phóng khi ra khỏi phạm vi mà nó được tạo. Ěây là giá trị mặc định.
  + GLOBAL: phạm vi hoạt động của cursor là toàn cục trên 1 kết nối. Tên cursor có thể được

dùng trong trong các SP. Cursor được tự động giải phóng khi ra khỏi kết nối đó.

* + FORWARD\_ONLY: ta chỉ được quyền di chuyển con trỏ mẫu tin theo 1 chiều từ mẫu tin đầu đến mẫu tin cuối. Nếu cursor là FORWARD\_ONLY mà không có STATIC, KEYSET, or DYNAMIC , cursor sẽ hoạt động như là 1 DYNAMIC cursor. FORWARD\_ONLY là giá trị mặc định, nhưng nếu cursor có kiểu STATIC, KEYSET, or DYNAMIC thì cursor sẽ là SCROLL.
  + KEYSET: các fields và thứ tự của các dòng trong cursor được cố định khi cursor được mở. Tập khóa của các dòng trong cursor được lưu trong 1 table của **tempdb** gọi là **keyset.**

+ Nếu 1 dòng trong table liên kết với cursor bị xóa, lệnh fetch sẽ trả @@FETCH\_STATUS

= -2.

+ Nếu Updates giá trị khóa ở ngoài cursor, sẽ tương đương với việc xóa dòng cǜ, thêm dòng, lúc này việc fetch dòng với giá trị cǜ sẽ trả về @@FETCH\_STATUS = -2. Giá trị mới chỉ thấy được trong cursor, nếu lệnh được thực hiện với WHERE CURRENT OF <tên cursor>.

* + DYNAMIC: Nếu 1 user thay đổi dữ liệu trong base table, thì cursor kiểu này phản ánh được các dữ liệu đã thay đổi trong base table. Lưu ý rằng Fetch ABSOLUTE không được sử dụng trong trường hợp này.
  + FAST\_FORWARD: Với FORWARD\_ONLY, READ\_ONLY cursor sẽ thực thi nhanh, tối ưu hơn. FAST\_FORWARD không được sử dụng nếu SCROLL hoặc FOR\_UPDATE được sử dụng. FAST\_FORWARD và FORWARD\_ONLY không được khai báo đồng thời.
  + READ\_ONLY: không cho hiệu chỉnh các dòng trong cursor. Với READ\_ONLY, không thể dùng WHERE CURRENT OF <cursor> trong lệnh Update hoặc Delete.
  + SCROLL\_LOCKS: các dòng đã cập nhật hoặc xóa trong cursor sẽ bị khóa cho đến khi đóng cursor . SCROLL\_LOCKS không thể dùng kèm với FAST\_FORWARD.

*select\_statement*: là 1 phát biểu Select-SQL chuẩn, lệnh select này sẽ định nghƿa 1 tập các mẫu tin cho cursor. Lưu ý: các từ khóa COMPUTE, COMPUTE BY, FOR BROWSE, và INTO không được sử dụng trong *select\_statement* của 1 cursor.

UPDATE [OF *column\_name* [**,**...*n*]]: cho phép ta hiệu chỉnh các fields trong cursor. Nếu OF *column*\_*name* [**,**...*n*] được sử dụng, nghƿa là chỉ có các cột được liệt kê trong OF mới được hiệu chỉnh.

# @@FETCH\_STATUS

Trả về 1 số nguyên cho biết trạng thái của phát biểu FETCH:

|  |  |
| --- | --- |
| **Return value** | **Mô tả** |
| 0 | FETCH thành công. |
| -1 | FETCH không được hoặc đã ra khỏi phạm vi tập mẫu tin. |

-2

Mẫu tin cần lấy không có (do đã bị xóa ở base table).

**Lưu ý:** @@FETCH\_STATUS là biến toàn cục cho tất cả cursor trên 1 kết nối.

# FETCH: lấy dữ liệu của dòng đưa vào các biến

FETCH

[ [ NEXT | PRIOR | FIRST | LAST

| ABSOLUTE { *n* | **@***nvar* }

| RELATIVE { *n* | **@***nvar* }

] FROM

]

{ { *cursor\_name* | @*cursor\_variable\_name* } [ INTO **@***variable\_name* [ **,**...*n* ] ]

NEXT

Trả về dòng kết quả ngay sau dòng hiện tại và cho dòng kết quả trở thành dòng hiện hành. Nếu

FETCH NEXT là lệnh đầu tiên đối với cursor, nó sẽ trả về dòng đầu tiên trong tập kết quả.

PRIOR

Trả về dòng kết quả ngay trước dòng hiện tại và cho dòng kết quả trở thành dòng hiện hành. Nếu

FETCH PRIOR là lệnh đầu tiên đối với con trỏ, không có dòng nào được trả về .

FIRST

Trả về dòng đầu tiên của cursor và cho nó làm dòng hiện hành.

LAST

Trả về dòng cuối cùng của cursor và cho nó làm dòng hiện hành.

ABSOLUTE {*n* | **@***nvar*}

Nếu n hoặc @nvar dương, trả về dòng thứ n tính từ đầu cursor và và cho nó làm dòng hiện

hành

RELATIVE {*n* | **@***nvar*}

Nếu n hoặc @nvar dương, trả về dòng thứ n kể từ vị trí dòng hiện hành và cho nó làm dòng

hiện hành mới. Nếu n or @nvar =0 thì trả về dòng hiện hành

GLOBAL

Specifies that *cursor\_name* refers to a global cursor.

*cursor\_name:* tên cursor cung cấp dữ liệu

INTO *@variable\_name*[**,***...n*]

Ěưa dữ liệu từ các cột trong cursor vào các biến. Mỗi biến trong danh sách phải tương ứng với mỗi cột trong cursor từ trái sang phải. Số lượng các biến phải khớp với số lượng cột trong cursor.

**Examples**

**A. Use FETCH to store values in variables**

USE pubs GO

-- Declare the variables to store the values returned by FETCH. DECLARE @au\_lname varchar(40), @au\_fname varchar(20)

DECLARE authors\_cursor CURSOR FOR SELECT au\_lname, au\_fname FROM authors WHERE au\_lname LIKE "B%"

ORDER BY au\_lname, au\_fname OPEN authors\_cursor

-- Perform the first fetch and store the values in variables.

-- Note: The variables are in the same order as the columns

-- in the SELECT statement.

**FETCH** NEXT FROM authors\_cursor **INTO** @au\_lname, @au\_fname

-- Check @@FETCH\_STATUS to see if there are any more rows to fetch. WHILE @@FETCH\_STATUS = 0

BEGIN

-- Concatenate and display the current values in the variables.

PRINT "Author: " + @au\_fname + " " + @au\_lname

-- This is executed as long as the previous fetch succeeds. FETCH NEXT FROM authors\_cursor

INTO @au\_lname, @au\_fname END

CLOSE authors\_cursor DEALLOCATE authors\_cursor GO

Author: Abraham Bennet Author: Reginald Blotchet-Halls

**C. Declare a SCROLL cursor and use the other FETCH options**

This example creates a SCROLL cursor to allow full scrolling capabilities through the LAST, PRIOR, RELATIVE, and ABSOLUTE options.

USE pubs GO

-- Execute the SELECT statement alone to show the

-- full result set that is used by the cursor. SELECT au\_lname, au\_fname FROM authors ORDER BY au\_lname, au\_fname

-- Declare the cursor.

DECLARE authors\_cursor SCROLL CURSOR FOR SELECT au\_lname, au\_fname FROM authors ORDER BY au\_lname, au\_fname

OPEN authors\_cursor

-- Fetch the last row in the cursor. FETCH LAST FROM authors\_cursor

-- Fetch the row immediately prior to the current row in the cursor. FETCH PRIOR FROM authors\_cursor

-- Fetch the second row in the cursor.

FETCH ABSOLUTE 2 FROM authors\_cursor

-- Fetch the row that is three rows after the current row. FETCH RELATIVE 3 FROM authors\_cursor

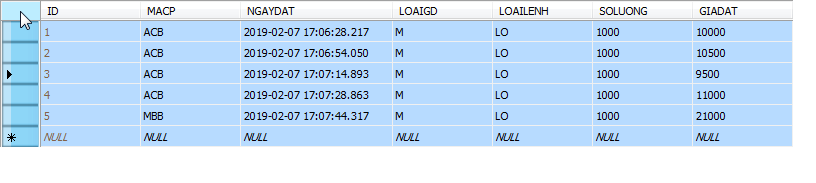
-- Fetch the row that is two rows prior to the current row. FETCH RELATIVE -2 FROM authors\_cursor

CLOSE authors\_cursor DEALLOCATE authors\_cursor GO

Ví dụ : Ta có cơ sở dữ liệu gồm có các table:

LENHDAT: chứa các lệnh đặt mua/bán cổ phiếu của các nhà đầu tư

|  |  |  |
| --- | --- | --- |
| **FieldName** | **Type** | **Description** |
| ID | int | Mã số lệnh đặt |
| MACP | char(7) | Mã cổ phiếu |
| NGAYDAT | datetime |  |
| LOAIGD | nchar(1) | Loại giao dịch : M : lệnh mua B : lệnh bán |
| LOAILENH | nchar(10) | Loại lệnh :  LO : khớp lệnh liên tục  ATO, ATC : khớp lệnh định kǶ |
| SOLUONG | int | Số lượng đặt |
| GIADAT | float | Giá đặt |
| TRANGTHAILENH | nvarchar(30) | Trạng thái lệnh : Chờ khớp  Khớp lệnh 1 phần Khớp hết  Ěã hủy Chưa khớp |



LENHKHOP: chứa các lệnh khớp khi thỏa qui tắc khớp lệnh

|  |  |  |
| --- | --- | --- |
| **FieldName** | **Type** | **Description** |
| IDKHOP | int | Mã số lệnh khớp |
| NGAYKHOP | datetime |  |
| SOLUONGKHOP | int |  |
| GIAKHOP | float |  |
| IDLENHDAT | int | Mã số lệnh đặt |

Viết Stored Procedure tính số lượng cổ phiếu khớp theo thuật tóan khớp lệnh liên tục khi có 1

lệnh mua hoặc bán được gởi đến bảng LENHDAT

**CREATE PROCEDURE CursorLoaiGD**

@OutCrsr CURSOR VARYING OUTPUT,

@macp NVARCHAR( 10), @Ngay NVARCHAR( 50), @LoaiGD CHAR

AS

SET DATEFORMAT DMY IF (@LoaiGD='M')

SET @OutCrsr=CURSOR KEYSET FOR

SELECT NGAYDAT, SOLUONG, GIADAT FROM LENHDAT

WHERE MACP=@macp

AND DAY(NGAYDAT)=DAY(@Ngay)AND MONTH(NGAYDAT)= MONTH(@Ngay) AND YEAR(NGAYDAT)=YEAR(@Ngay)

AND LOAIGD=@LoaiGD AND SOLUONG >0 ORDER BY GIADAT DESC, NGAYDAT

ELSE

SET @OutCrsr=CURSOR KEYSET FOR

SELECT NGAYDAT, SOLUONG, GIADAT FROM LENHDAT

WHERE MACP=@macp

AND DAY(NGAYDAT)=DAY(@Ngay)AND MONTH(NGAYDAT)= MONTH(@Ngay) AND YEAR(NGAYDAT)=YEAR(@Ngay)

AND LOAIGD=@LoaiGD AND SOLUONG >0 ORDER BY GIADAT, NGAYDAT

OPEN @OutCrsr GO

**CREATE PROC SP\_KHOPLENH\_LO**

@macp NVARCHAR( 10), @Ngay NVARCHAR( 50), @LoaiGD CHAR,

@soluongMB INT, @giadatMB FLOAT AS

BEGIN

SET DATEFORMAT DMY

DECLARE @CrsrVar CURSOR , @ngaydat NVARCHAR( 50), @soluong INT,

@giadat FLOAT, @soluongkhop INT, @giakhop FLOAT

IF (@LoaiGD='B')

**EXEC CursorLoaiGD @CrsrVar OUTPUT, @macp,@Ngay, 'M'**

ELSE

**EXEC CursorLoaiGD @CrsrVar OUTPUT, @macp,@Ngay, 'B'**

FETCH NEXT FROM @CrsrVar INTO @ngaydat , @soluong , @giadat SELECT @ngaydat , @soluong , @giadat

WHILE (@@FETCH\_STATUS <> -1 AND @soluongMB >0) BEGIN

IF (@LoaiGD='B' )

IF (@giadatMB <= @giadat) BEGIN

IF @soluongMB > @soluong -- khớp hết s.lg lệnh mua đặt trước

BEGIN

SET @soluongkhop = @soluong SET @giakhop = @giadat

SET @soluongMB = @soluongMB - @soluong UPDATE dbo.LENHDAT

SET SOLUONG = 0

WHERE CURRENT OF @CrsrVar

END ELSE BEGIN

SET @soluongkhop = @soluongMB SET @giakhop = @giadat

UPDATE dbo.LENHDAT

SET SOLUONG = SOLUONG - @soluongMB WHERE CURRENT OF @CrsrVar

SET @soluongMB = 0 END

--SELECT @soluongkhop, @giakhop

-- Cập nhật table LENHKHOP

END ELSE

GOTO THOAT

-- Còn Trường hợp lệnh gởi vào là lệnh mua

FETCH NEXT FROM @CrsrVar INTO @ngaydat , @soluong , @giadat END

THOAT:

CLOSE @CrsrVar DEALLOCATE @CrsrVar

END