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.

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*]]]

3. **Mở cursor** : OPEN { { [GLOBAL] *cursor_name* } | *cursor_variable_name* }

4. **Lấy dữ liệu** từ 1 mâu 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. **Đóng 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 ten 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

FETCH NEXT FROM Employee_Cursor
WHILE @@FETCH_STATUS = 0
BEGIN

   FETCH NEXT FROM Employee_Cursor
END

CLOSE Employee_Cursor
DEALLOCATE Employee_Cursor
```

2. Tạo biến cursor : ta có 2 cách:

a. 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
```

b. 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 cũng 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 | OPTIMISTIC ]
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. Đây 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 bị xóa, lệnh fetch sẽ trả @@FETCH_STATUS = -2.

+ Nếu Updates giá trị khóa ở ngoài cursor, sẽ tương đương với việc xóa dòng cũ, thêm dòng, lúc này việc fetch dòng với giá trị cũ 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 option ABSOLUTE fetch 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 cannot be specified if SCROLL or FOR_UPDATE is also specified. FAST_FORWARD and FORWARD_ONLY are mutually exclusive; if one is specified the other cannot be specified.

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.

OPTIMISTIC

Specifies that positioned updates or deletes made through the cursor do not succeed if the row has been updated since it was read into the cursor. SQL Server does not lock rows as they are read into the cursor. It instead uses comparisons of **timestamp** column values, or a checksum value if the table has no **timestamp** column, to determine whether the row was modified after it was read into the cursor. If the row was modified, the attempted positioned update or delete fails. OPTIMISTIC cannot be specified if FAST_FORWARD is also specified.

select_statement: là 1 phát biểu Select-SQL chuẩn, lệnh select này sẽ định nghĩa 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, nghĩa là chỉ có các cột được liệt kê trong OF mới được hiệu chỉnh

@@FETCH_STATUS

Returns the status of the last cursor FETCH statement issued against any cursor currently opened by the connection.

Return value	Description
0	FETCH statement was successful.
-1	FETCH statement failed or the row was beyond the result set.

-2	Row fetched is missing.

Syntax

@@FETCH_STATUS

Return Types

integer

Remarks

Because @@FETCH_STATUS is global to all cursors on a connection, use @@FETCH_STATUS carefully. After a FETCH statement is executed, the test for @@FETCH_STATUS must occur before any other FETCH statement is executed against another cursor. The value of @@FETCH_STATUS is undefined before any fetches have occurred on the connection.

For example, a user executes a FETCH statement from one cursor, and then calls a stored procedure that opens and processes the results from another cursor. When control is returned from the called stored procedure, @@FETCH_STATUS reflects the last FETCH executed in the stored procedure, not the FETCH statement executed before the stored procedure is called.

Examples

This example uses @@FETCH_STATUS to control cursor activities in a WHILE loop.

```
DECLARE Employee_Cursor CURSOR FOR
    SELECT LastName, FirstName FROM Northwind.dbo.Employees
OPEN Employee_Cursor
FETCH NEXT FROM Employee_Cursor
WHILE @@FETCH_STATUS = 0
BEGIN
  FETCH NEXT FROM Employee_Cursor
END
```

```
CLOSE Employee_Cursor
DEALLOCATE Employee_Cursor
```

FETCH

Retrieves a specific row from a Transact-SQL server cursor.

Syntax

```
FETCH
      [ [ NEXT | PRIOR | FIRST | LAST
            | ABSOLUTE { n | @nvar }
            | RELATIVE { n | @nvar }
         ]
         FROM
      ]
{ { [ GLOBAL ] cursor_name } | @cursor_variable_name }
[ INTO @variable_name [ ,...n ] ]
```

Arguments

NEXT

Returns the result row immediately following the current row, and increments the current row to the row returned. If FETCH NEXT is the first fetch against a cursor, it returns the first row in the result set. NEXT is the default cursor fetch option.

PRIOR

Returns the result row immediately preceding the current row, and decrements the current row to the row returned. If FETCH PRIOR is the first fetch against a cursor, no row is returned and the cursor is left positioned before the first row.

FIRST

Returns the first row in the cursor and makes it the current row.

LAST

Returns the last row in the cursor and makes it the current row.

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

If *n* or *@nvar* is positive, returns the row *n* rows from the front of the cursor and makes the returned row the new current row. If *n* or *@nvar* is negative, returns the row *n* rows before the end of the cursor and makes the returned row the new current row. If *n* or *@nvar* is 0, no rows are returned. *n* must be an integer constant and *@nvar* must be **smallint**, **tinyint**, or **int**.

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

If *n* or *@nvar* is positive, returns the row *n* rows beyond the current row and makes the returned row the new current row. If *n* or *@nvar* is negative, returns the row *n* rows prior to the current row and makes the returned row the new current row. If *n* or *@nvar* is 0, returns the current row. If FETCH RELATIVE is specified with *n* or *@nvar* set to negative numbers or 0 on the first fetch done against a cursor, no rows are returned. *n* must be an integer constant and *@nvar* must be **smallint**, **tinyint**, or **int**.

GLOBAL

Specifies that *cursor_name* refers to a global cursor.

cursor_name

Is the name of the open cursor from which the fetch should be made. If both a global and a local cursor exist with *cursor_name* as their name, *cursor_name* to the global cursor if GLOBAL is specified and to the local cursor if GLOBAL is not specified.

@cursor_variable_name

Is the name of a cursor variable referencing the open cursor from which the fetch should be made.

INTO *@variable_name*[**,***...n*]

Allows data from the columns of a fetch to be placed into local variables. Each variable in the list, from left to right, is associated with the corresponding column in the cursor result set. The data type of each variable must either match or be a supported implicit conversion of the data type of the corresponding result set column. The number of variables must match the number of columns in the cursor select list.

Remarks

If the SCROLL option is not specified in an SQL-92 style DECLARE CURSOR statement, NEXT is the only FETCH option supported. If SCROLL is specified in an SQL-92 style DECLARE CURSOR, all FETCH options are supported.
When the Transact_SQL DECLARE cursor extensions are used, these rules apply:

- If either FORWARD-ONLY or FAST_FORWARD is specified, NEXT is the only FETCH option supported.
- If DYNAMIC, FORWARD_ONLY or FAST_FORWARD are not specified, and one of KEYSET, STATIC, or SCROLL are specified, all FETCH options are supported.
- DYNAMIC SCROLL cursors support all the FETCH options except ABSOLUTE.

The @@FETCH_STATUS function reports the status of the last FETCH statement. The same information is recorded in the **fetch_status** column in the cursor returned by **sp_describe_cursor**. This status information should be used to determine the validity of the data returned by a FETCH statement prior to attempting any operation against that data. For more information, see @@FETCH_STATUS.

Permissions

FETCH permissions default to any valid user.

Examples

A. Use FETCH in a simple cursor

This example declares a simple cursor for the rows in the **authors** table with a last name beginning with B, and uses FETCH NEXT to step through the rows. The FETCH statements return the value for the column specified in the DECLARE CURSOR as a single-row result set.
USE pubs

```
GO
DECLARE authors_cursor CURSOR FOR
SELECT au_lname FROM authors
WHERE au_lname LIKE "B%"
ORDER BY au_lname

OPEN authors_cursor

-- Perform the first fetch.
FETCH NEXT FROM authors_cursor

-- Check @@FETCH_STATUS to see if there are any more rows to fetch.
WHILE @@FETCH_STATUS = 0
BEGIN
   -- This is executed as long as the previous fetch succeeds.
   FETCH NEXT FROM authors_cursor
END

CLOSE authors_cursor
DEALLOCATE authors_cursor
GO

au_lname
----------------------------------------
Bennet
au_lname
----------------------------------------
Blotchet-Halls
au_lname
----------------------------------------
```

B. Use FETCH to store values in variables

This example is similar to the last example, except the output of the FETCH statements is stored in local variables rather than being returned directly to the client. The PRINT statement combines the variables into a single string and returns them to the client.

```
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
```

au_lname	au_fname
Bennet	Abraham
Blotchet-Halls	Reginald
Carson	Cheryl
DeFrance	Michel
del Castillo	Innes
Dull	Ann
Green	Marjorie
Greene	Morningstar
Gringlesby	Burt
Hunter	Sheryl
Karsen	Livia
Locksley	Charlene

MacFeather	Stearns
McBadden	Heather
O'Leary	Michael
Panteley	Sylvia
Ringer	Albert
Ringer	Anne
Smith	Meander
Straight	Dean
Stringer	Dirk
White	Johnson
Yokomoto	Akiko

au_lname	au_fname
Yokomoto	Akiko

au_lname	au_fname
White	Johnson

au_lname	au_fname
Blotchet-Halls	Reginald

au_lname	au_fname
del Castillo	Innes

au_lname	au_fname
Carson	Cheryl