

Base de Données, I3306

Chap 5 : Curseurs / Cursors

UL-Nabatieh

December 18, 2018



Outline

- 1 Cursor overview
- 2 Cursors operations
- 3 Types of cursors

Cursor overview

- A method for iterating over a result set in T-SQL
- One row at a time
- A mechanism to navigate **tuple-by-tuple** over a relation;
- Typically used inside triggers, stored procedures, or functions
- Main idea:
 - When we execute a query, a relation is returned;
 - It is stored in private work area for the query;
 - Cursor is a pointer to this area;
 - Move the cursor to navigate over the tuples.



Outline

- 1 Cursor overview
- 2 Cursors operations
- 3 Types of cursors



Cursors operations

- **Declare cursor**: declare variables to hold the output from the cursor

```
1 declare @c cursor
```

- **Assign cursor**: assign the query to the cursor

```
1 set @c = cursor for  
2 select artNum, artDesc  
3 from Article
```



Cursors operations

- **Open cursor:** execute the query and put the pointer at the first tuple

```
1 open @c
```

- **Fetch next tuple:** fetch the first row

```
1 declare @idArt int, @descp varchar(30)
2 fetch next from @c into @idArt, @descp
```

- **Fetch next tuple:** loop until there are no more results

```
1 while @@FETCH_STATUS = 0
2 begin
3     print cast(@idArt as varchar(50)) + ' ' + @descp
4     fetch next from @c into @idArt, @descp
5 end
```

Cursors operations

- **Close cursor**: close the cursor

```
1 close @c;
```

- **Close cursor**: deallocate the cursor to free up any memory or open result sets.

```
1 deallocate @c
```



Cursor: example

```
1 declare @idArt int
2 declare @descp nvarchar(50)
3
4 declare @c cursor
5
6 set @c = cursor for
7 select idArt, descp
8 from Article
9
10 open @c
11 begin
12     fetch next from @c into @idArt, @descp
13     while @@FETCH_STATUS = 0
14     begin
15         print cast(@idArt as varchar(50)) + ' ' + @descp
16         fetch next from @c into @idArt, @descp
17     end
18
19 close @c
20 deallocate @c
```

Cursor functions : @@FETCH_STATUS

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

- **0** : the FETCH statement was successful.
- **-1** : the FETCH statement failed or the row was beyond the result set.
- **-2** : the row fetched is missing.



Cursor functions : @@CURSOR_ROWS

Returns the **number of qualifying rows currently** in the last cursor opened on the connection.

- ❶ **0** : no cursors have been opened, no rows qualified for the last opened cursor, or the last-opened cursor is closed or deallocated.
- ❷ **< 0** : the cursor is populated asynchronously
- ❸ **> 0** : the returned value is the total number of rows in the cursor



Outline

- 1 Cursor overview
- 2 Cursors operations
- 3 Types of cursors



Types of cursors

- 1 Static cursors
- 2 Dynamic cursors
- 3 Forward only cursors



Cursors: syntax

```
1 DECLARE cursor_name CURSOR  
2   [ FORWARD_ONLY | SCROLL ]  
3   [ STATIC | DYNAMIC | FAST_FORWARD ]  
4   FOR select_statement
```



Static cursors

A **static** cursor:

- Defines a cursor that makes a temporary copy of the data to be used by the cursor.
- All requests to the cursor are answered from this temporary table in tempdb
- No UPDATE, INSERT, or DELETE operations are reflected in a static cursor (unless the cursor is closed and reopened)



Static cursors: example

```
1 declare @des varchar(30)
2 declare @c cursor
3 set @c= cursor static for
4 select descArt from Article
5 open @c
6 if @@cursor_rows > 0
7 begin
8     fetch next from @c into @des
9     while @@fetch_status = 0
10    begin
11        print 'Description: ' + convert(varchar(20),@des)
12        fetch next from @c into @des
13    end
14 end
15 close @c
16 deallocate @c
```

Dynamic cursors

A **dynamic** cursor:

- 1 Allows you to see the data updation, deletion and insertion in the data source while the cursor is open;
- 2 Is sensitive to any changes to the data source and supports update, delete operations;
- 3 By default dynamic cursors are **scrollable**.



Forward only Cursor

A **forward only** cursor:

- Is the fastest cursor among the all cursors;
- It doesn't support backward scrolling;
- It is sensitive to any changes to the original data source.



SCROLL Cursor



Specifies that all fetch options (FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE) are available.

- ❶ **FETCH FIRST**: fetches the first row in the cursor;
- ❷ **FETCH NEXT**: fetches the row after the last row fetched;
- ❸ **FETCH PRIOR**: fetches the row before the last row fetched;
- ❹ **FETCH LAST**: fetches the last row in the cursor;



SCROLL Cursor

Specifies that all fetch options (FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE) are available

- 1 **FETCH ABSOLUTE n** : fetches the n th row from the first row in the cursor if n is a positive integer. If n is a negative integer, the row n rows before the end of the cursor is fetched. If n is 0, no rows are fetched.
- 2 **FETCH RELATIVE n** : fetches the row n rows from the last row fetched. If n is positive, the row n rows after the last row fetched is fetched. If n is negative, the row n rows before the last row fetched is fetched. If n is 0, the same row is fetched again.



Remarks

- If FORWARD_ONLY is specified without the STATIC, or DYNAMIC keywords, the cursor operates as a DYNAMIC cursor.
- When neither FORWARD_ONLY nor SCROLL is specified, FORWARD_ONLY is the default, unless the keywords STATIC or DYNAMIC are specified.
- STATIC and DYNAMIC cursors default to SCROLL
- FORWARD_ONLY is supported with STATIC and DYNAMIC Transact-SQL cursors.

