# Unit 10

## Programming in Transact-SQL – TC2

# CURSORS

# CURSORS - Introduction

Applications, especially interactive online applications, cannot always work effectively with the entire result set of a SELECT as a unit, they need a mechanism to work with one row at a time. Cursors provide that mechanism:

- Allowing positioning at specific rows of the result set.
- Retrieving one row from the current position in the result set.
- Supporting data modifications to the rows at the current position in the result set.
- Supporting different levels of visibility to changes made by other users to the database entries that are presented in the result set.
- Providing Transact-SQL statements in scripts, stored procedures, and triggers access to the data in a result set.

# CURSOR PROCESS

The following general process is used with all SQL Server cursors:

1. Associate a cursor with the result set of a SELECT, and define characteristics of the cursor, such as whether the rows in the cursor can be updated. DECLARE.
2. Execute the Transact-SQL statement to populate the cursor.
3. Retrieve the rows in the cursor you want to see. The operation to retrieve one row from a cursor is called a fetch. Performing a series of fetches to retrieve rows in either a forward or backward direction is called scrolling.
4. Optionally, perform modification operations (update or delete) on the row at the current position in the cursor.
5. Close the cursor.

# DECLARE CURSOR

```
DECLARE cursor_name
        [ INSENSITIVE ] [ SCROLL ] CURSOR
        FOR select_statement
        [ FOR { READ ONLY |
                  UPDATE [ OF column_name [ ,...n ] ]
              } ] [;]
```

INSENSITIVE Makes a temporary copy of the data to be used by the cursor. Modifications made by other users are not reflected. Does not allow modifications.

# DECLARE CURSOR

*select_statement* Is a standard SELECT statement that defines the result set of the cursor.

SCROLL Specifies that all fetch options (FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE) are available. If not specified, NEXT is the only fetch option supported.

READ ONLY Prevents updates made through it.

UPDATE [OF *column_name* [,...*n*]] Defines updatable columns within the cursor. If UPDATE is specified without a column list, all columns can be updated.

# DECLARE CURSOR

*Examples:*

```
DECLARE vend_cursor CURSOR
FOR SELECT * FROM Vendors;

DECLARE vend_cursor INSENSITIVE CURSOR
FOR SELECT * FROM Vendors
FOR READ ONLY;

DECLARE vend_cursor CURSOR
FOR SELECT * FROM Vendors
FOR UPDATE  OF name, birth_date;
```

# CURSOR DATATYPE

Variables with a **cursor** data type are supported.

A cursor can be associated with a **cursor** variable by either of two methods:

```
DECLARE @MyVariable CURSOR;
DECLARE MyCursor CURSOR
        FOR SELECT LastName FROM Contacts;
SET @MyVariable = MyCursor;
--
DECLARE @MyVariable  CURSOR;
SET @MyVariable = CURSOR
        FOR SELECT LastName FROM  Contacts;
```

# POPULATE THE CURSOR

OPEN { cursor_name | cursor_variable_name }

Opens a Transact-SQL server cursor and populates the cursor by executing the Transact-SQL statement specified on the DECLARE CURSOR or SET *cursor_variable* statement.

Example:

OPEN MyCursor;

Or

OPEN MyVariable;

# CLOSE - DEALLOCATE

CLOSE { cursor_name | cursor_variable_name }

Closes an open cursor by releasing the current result set and freeing any cursor locks held on the rows on which the cursor is positioned. CLOSE leaves the data structures available for reopening, but fetches and positioned updates are not allowed until the cursor is reopened.

DEALLOCATE {cursor_name | c_variable_name }

Removes a cursor reference.

With c_*cursor_variable_name* removes only the reference of the named variable to the cursor.

# FETCH

Retrieves a specific row from a cursor, and updates the current row.

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

N is an integer and can be negative.

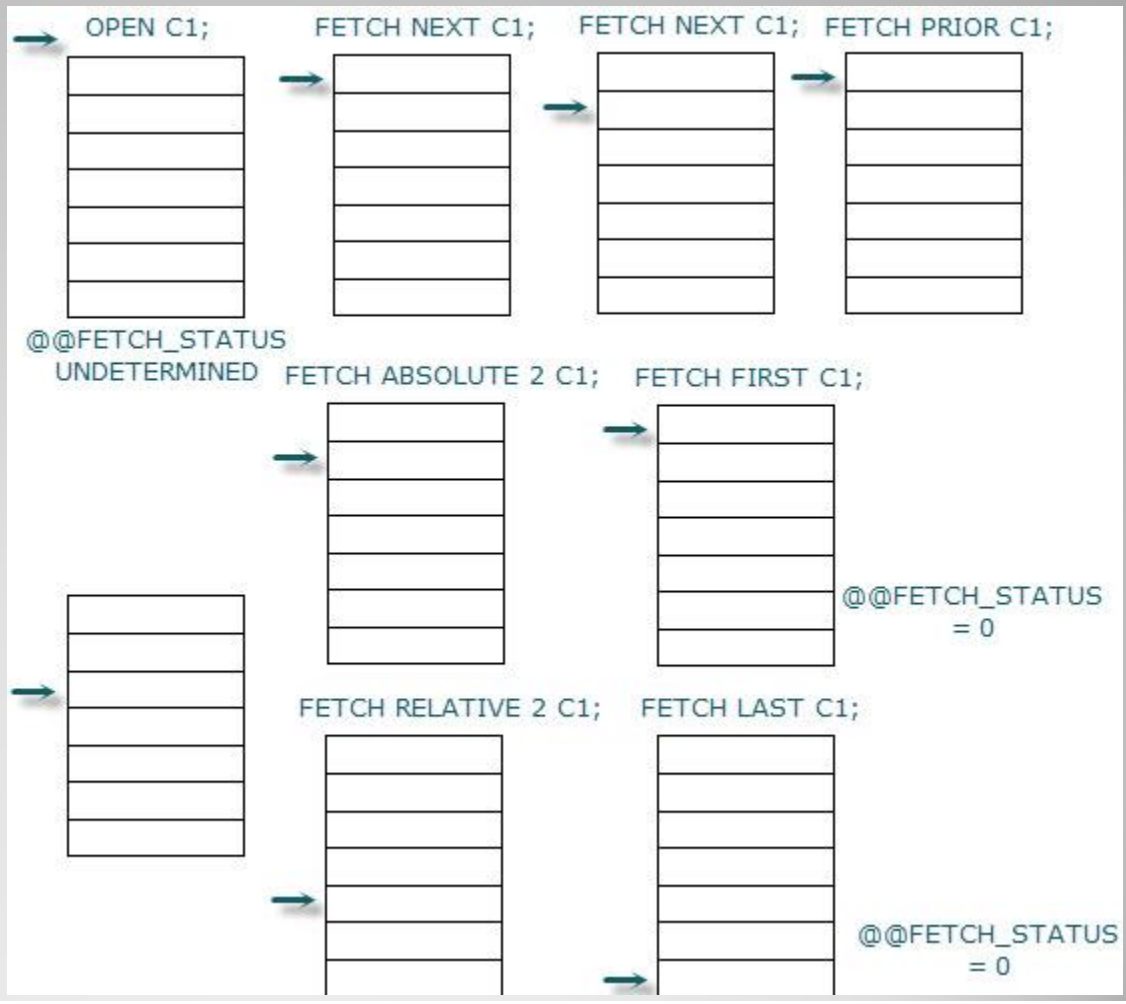
ABSOLUTE …the row n from the front of the cursor.

RELATIVE …the row n beyond the current row.

# FETCH

Example:

```
DECLARE  C1
SCROLL CURSOR
FOR SELECT *
FROM Vendors;
```



# CURSORS

**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.

```
DECLARE  C1 CURSOR FOR SELECT numclie, nombre, repclie FROM Clientes;
DECLARE @numclie INTEGER, @repclie INTEGER;
DECLARE @nombre VARCHAR(30);
OPEN C1;
-- Perform the first fetch.
FETCH NEXT FROM C1 INTO @numclie, @nombre, @repclie ;
-- Check @@FETCH_STATUS to see if there are any more rows to fetch.
WHILE @@FETCH_STATUS = 0
    BEGIN
      IF @repclie = 108 PRINT STR(@numclie,4) + ' - ' + @nombre + ' - '
                                              + STR(@repclie,3);
      FETCH NEXT FROM C1 INTO @numclie, @nombre, @repclie ;;
    END
CLOSE C1;
DEALLOCATE C1;
GO
```