An overview of the SQL cursor @@FETCH\_STATUS function

SQL cursor is one of the most popular database objects. It is used to retrieve data from the result set of an SQL query one row at a time. Even if the cursor is not recommended from a performance perspective, they are still widely used especially when handling a small amount of data.

There are many related objects needed to use a cursor such as @@FETCH\_STATUS function. In this article, we will try to describe SQL cursor briefly and then we will explain @@FETCH\_STATUS function by providing some examples to clarify each of its returned values.

SQL cursor overview

As mentioned before, a cursor is a database object used to fetch the rows from a result set. There are different phases in a cursor life cycle:

Declaring cursor

First of all, we need to create a cursor object. Mainly there are 2 required parameters:

1. Cursor name
2. Source SQL query

You must use the following syntax to declare a cursor object:

|  |  |
| --- | --- |
| 1 | DECLARE <Cursor\_name> CURSOR FOR <Source\_SQL\_Query> |

Note that source SQL query should be a SELECT statement, as for example, the following cursor is to read the table schemas and names created in a database:

|  |  |
| --- | --- |
| 1 | DECLARE csrTables CURSOR FOR SELECT TABLE\_SCHEMA, TABLE\_NAME from INFORMATION\_SCHEMA.TABLES |

Opening cursor

Declaring SQL cursor doesn’t mean that you can start reading data from the result set, first, you need to retrieve the source SQL query result which is done using the OPEN command. The command syntax is very simple, just write OPEN keyword plus the name of the cursor as follows:

|  |  |
| --- | --- |
| 1 | OPEN <Cursor\_name> |

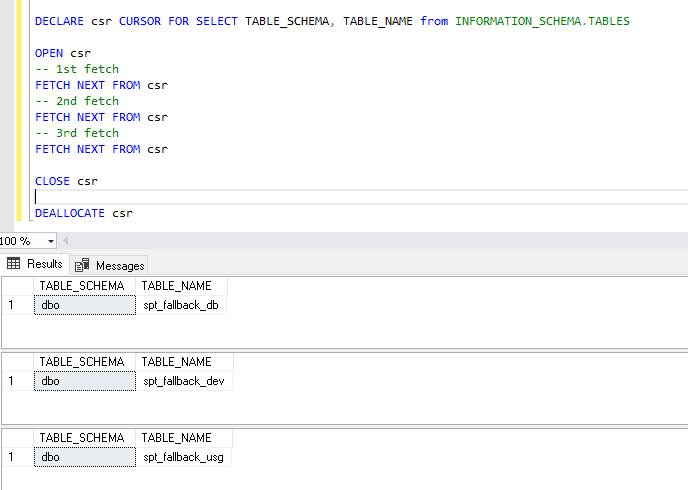
Fetching rows

After opening the cursor, the source SQL query is executed and the result set is stored in memory. The next step is to fetch rows and to visualize them or to store them within some declared variables. Note that each column in the result set must be mapped to a variable having a relevant data type.

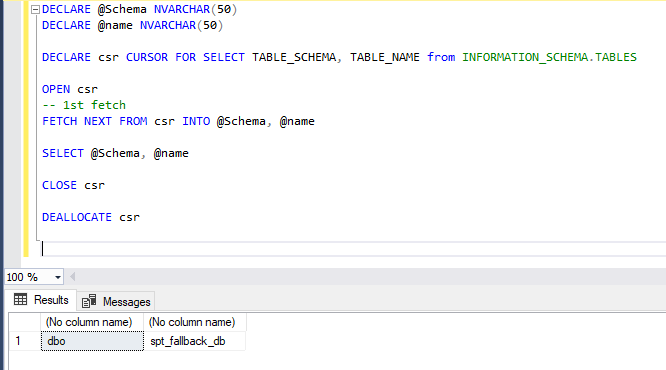
The FETCH command syntax is as follows:

|  |  |
| --- | --- |
| 1 | FETCH NEXT FROM <cursor\_name> [ INTO @variable\_name [ ,...n ] ] |

If you decide to only fetch rows without storing them within variables, the result will be displayed as a result of a select query:



Otherwise, if you choose to store the fetched rows within SQL variable you have to make sure that the order of the variables must be the same as the result set columns order:



Closing and deallocating cursor

After consuming rows, you should close it to release the current result set and to free any cursor locks held on the rows on which the cursor is positioned. The command syntax is very simple, just write CLOSE keyword plus the name of the cursor as follows:

|  |  |
| --- | --- |
| 1 | CLOSE <Cursor\_name> |

After closing the cursor, you should make sure that the cursor object reference is released from the memory by using the DEALLOCATE command as following:

|  |  |
| --- | --- |
| 1 | DEALLOCATE <Cursor\_name> |

@@FETCH\_STATUS function

After illustrating the main commands used to manipulate SQL cursors, there are some system functions that support cursors such as **@@FETCH\_STATUS** and **@@CURSOR\_ROWS**. In this section, we will explain the @@FETCH\_STATUS function by providing some examples.

What is the @@FETCH\_STAUTS function?

@@FETCH\_STATUS is a system function that returns the status of the last FETCH statement issued against any opened cursor. This function returns an integer value as mentioned in the table below (Reference: [@@FETCH\_STATUS (Transact-SQL)](https://docs.microsoft.com/en-us/sql/t-sql/functions/fetch-status-transact-sql)):

|  |  |
| --- | --- |
| **Value** | **Description** |
| 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 |
| -9 | The cursor is not performing a fetch operation |

One of the main use cases for this function is to implement it within a while loop to keep fetching rows while the fetch statement is successful instead of writing several FETCH statements for each cursor. As an example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | DECLARE @Schema NVARCHAR(50)  DECLARE @name NVARCHAR(50)    DECLARE csr CURSOR FOR SELECT TABLE\_SCHEMA, TABLE\_NAME from INFORMATION\_SCHEMA.TABLES    OPEN csr    FETCH NEXT FROM csr INTO @Schema, @name    WHILE @@FETCH\_STATUS = 0    BEGIN      FETCH NEXT FROM csr INTO @Schema, @name    END    CLOSE csr  DEALLOCATE csr |

In the example above, we have created a SQL cursor to fetch the result of a SQL command that selects the name and the schema of tables created within the database. After declaring and opening the cursor, we issued the first FETCH statement. If the SQL query returned at least one row the first FETCH statement should be successful, else it should fail. After that, we used a WHILE loop to check if the FETCH statement was successful and to keep fetching rows while there are more rows to be fetched. When the FETCH statement doesn’t return any rows @@FETCH\_STATUS function should return -1, and then the while loop is ended.

As mentioned in the previous section, it is very simple to give an example of how the @@FETCH\_STATUS returns 0 and -1 values. But what about -2 and -9? In the next section, we will briefly explain these values by providing some examples.

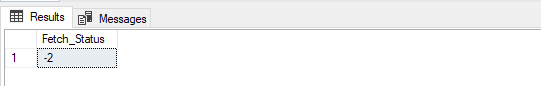
The row fetched is missing (@@FETCH\_STATUS = -2)

This FETCH status is returned if a row that should be returned in the FETCH statement is deleted in case that we used the KEYSET option while declaring the cursor (note that KEYSET option specifies that the membership and order of the rows in the cursor are fixed when the cursor is opened). This scenario happens when different users are working with the same data and one user deletes some rows while the other is trying to manipulate them using SQL cursor.

The following example is only to illustrate this scenario:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | DECLARE @Id int    CREATE TABLE #TBLTEMP (    Id int not null PRIMARY KEY    )    INSERT INTO #TBLTEMP (Id) VALUES (1),(2)    DECLARE csr CURSOR KEYSET FOR SELECT Id FROM #TBLTEMP ORDER BY Id  OPEN Csr  FETCH NEXT FROM csr INTO @Id    DELETE FROM #TBLTEMP WHERE Id = 2    FETCH NEXT FROM csr INTO @Id    SELECT @@FETCH\_STATUS as [Fetch\_Status]    CLOSE csr  DEALLOCATE csr  DROP TABLE #TBLTEMP |

As shown in the following screenshot you can see that the @@FETCH\_STATUS function returned a value of -2:



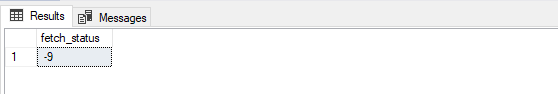
For more information about KEYSET and other options that can be used while declaring SQL cursors you can refer to the following documentation: [DECLARE CURSOR (Transact-SQL)](https://docs.microsoft.com/en-us/sql/t-sql/language-elements/declare-cursor-transact-sql)

The cursor is not performing a fetch operation (@@FETCH\_STATUS = -9)

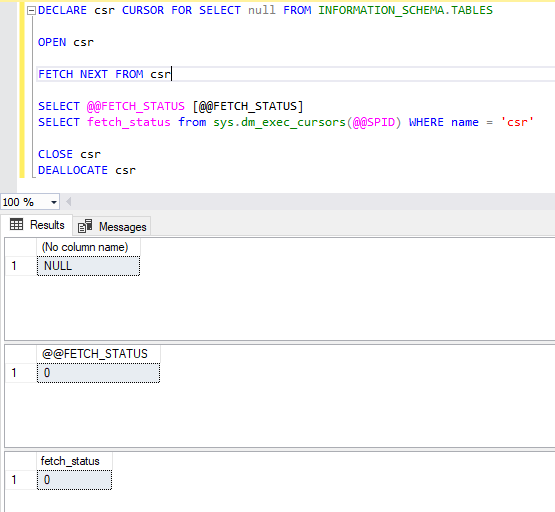
This value is not returned by the @@FETCH\_STATUS function. It is a value stored within SQL Server internals and it is shown in the System Dynamic Management Views when the SELECT statement is defined. Declaring the SQL cursor will never fetch data (no information about the result set columns) regardless the cursor is open or not. The simplest example is to declare a cursor with a SELECT query that has no columns and only select NULL:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | DECLARE csr CURSOR FOR SELECT null FROM INFORMATION\_SCHEMA.TABLES    OPEN csr    SELECT fetch\_status from sys.dm\_exec\_cursors(@@SPID) WHERE name = 'csr'    CLOSE csr  DEALLOCATE csr |

As shown in the following screenshot, the result will show -9:



Note that, if you try to issue a FETCH statement it will return 0 or -1 regardless of the number of rows in the result set:



For more information about system dynamic management views in SQL Server and sys.dm\_exec\_cursors, you can refer to the following links:

* [System Dynamic Management Views](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/system-dynamic-management-views?view=sql-server-ver15)
* [Sys.dm\_exec\_cursors (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-cursors-transact-sql?view=sql-server-ver15)

Also, you can refer to the following Stackoverflow.com posts for more information about @@FETCH\_STATUS:

* [What are the examples for @@fetch\_status value -2 and -9](https://stackoverflow.com/q/41307323/7031230)
* [What does the value -9 means for fetch\_status in SQL Server?](https://stackoverflow.com/q/36271942/7031230)

Conclusion

In this article, we described briefly SQL cursor database object by illustrating the different phases in this object life cycle. In addition, we explained the @@FETCH\_STATUS system function that is used to check the status of the FETCH statement of the cursor and we illustrated the different values that it returns by providing some scenarios that can yield for each value.