# Cursors

Operations in a relational database act on a complete set of rows. This complete set of rows returned by the statement is known as the result set. Applications, especially interactive online applications, cannot always work effectively with the entire result set as a unit. These applications need a mechanism to work with one row or a small block of rows at a time. Cursors are an extension to result sets that provide that mechanism.

Cursors extend result processing by:

* Allowing positioning at specific rows of the result set.
* Retrieving one row or block of rows 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 data that is presented in the result set.
* Providing Transact-SQL statements in scripts, stored procedures, and triggers access to the data in a result set.

**Type of Cursors  
Forward-only**  
A forward-only cursor does not support scrolling; it supports only fetching the rows serially from the start to the end of the cursor. The rows are not retrieved from the database until they are fetched. The effects of all INSERT, UPDATE, and DELETE statements made by the current user or committed by other users that affect rows in the result set are visible as the rows are fetched from the cursor.

Because the cursor cannot be scrolled backward, most changes made to rows in the database after the row was fetched are not visible through the cursor.

**Static**  
The complete result set of a static cursor is built in **tempdb** when the cursor is opened. A static cursor always displays the result set as it was when the cursor was opened. Static cursors detect few or no changes, but consume relatively few resources while scrolling.

The cursor does not reflect any changes made in the database that affect either the membership of the result set or changes to the values in the columns of the rows that make up the result set. A static cursor does not display new rows inserted in the database after the cursor was opened, even if they match the search conditions of the cursor SELECT statement. If rows making up the result set are updated by other users, the new data values are not displayed in the static cursor. The static cursor displays rows deleted from the database after the cursor was opened. No UPDATE, INSERT, or DELETE operations are reflected in a static cursor (unless the cursor is closed and reopened), not even modifications made using the same connection that opened the cursor.

SQL Server static cursors are always read-only.

Because the result set of a static cursor is stored in a work table in **tempdb**, the size of the rows in the result set cannot exceed the maximum row size for a SQL Server table.

**Keyset**The membership and order of rows in a keyset-driven cursor are fixed when the cursor is opened. Keyset-driven cursors are controlled by a set of unique identifiers, keys, known as the keyset. The keys are built from a set of columns that uniquely identify the rows in the result set. The keyset is the set of the key values from all the rows that qualified for the SELECT statement at the time the cursor was opened. The keyset for a keyset-driven cursor is built in **tempdb** when the cursor is opened.

**Dynamic**  
Dynamic cursors are the opposite of static cursors. Dynamic cursors reflect all changes made to the rows in their result set when scrolling through the cursor. The data values, order, and membership of the rows in the result set can change on each fetch. All UPDATE, INSERT, and DELETE statements made by all users are visible through the cursor. Updates are visible immediately if they are made through the cursor using either an API function such as **SQLSetPos** or the Transact-SQL WHERE CURRENT OF clause. Updates made outside the cursor are not visible until they are committed, unless the cursor transaction isolation level is set to read uncommitted. Dynamic cursor plans never use spatial indexes.

**DECLARE CURSOR**

Transact-SQL Extended Syntax

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 ] ] ]

[;]

**OPEN CURSOR**

OPEN { { [ GLOBAL ] cursor\_name } | cursor\_variable\_name }

**FETCH CURSOR**

FETCH

[ [ NEXT | PRIOR | FIRST | LAST

| ABSOLUTE { n | @nvar }

| RELATIVE { n | @nvar }

]

FROM

]

{ { [ GLOBAL ] cursor\_name } | @cursor\_variable\_name }

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

**CLOSE CURSOR**

CLOSE { { [ GLOBAL ] cursor\_name } | cursor\_variable\_name }

**DEALLOCATE CURSOR**

DEALLOCATE { { [ GLOBAL ] cursor\_name } | @cu

rsor\_variable\_name }

DECLARE Employee\_Cursor CURSOR FOR

SELECT LastName, FirstName FROM AdventureWorks2012.HumanResources.vEmployee

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;