**Cursors**

A cursor is a database object that points to a result set

Cursors can exist in two places:

API cursors are managed by the API on the client machine

Transact-SQL cursors exist on the server

**Uses**

1. Iterate through result sets in server-side scripts and procedures

Use to perform different operations on different rows of a given table

2. Provide access to applications that can’t use standard database APIs.

3. Do not use cursors if set-based access will suffice (cursors are slower)

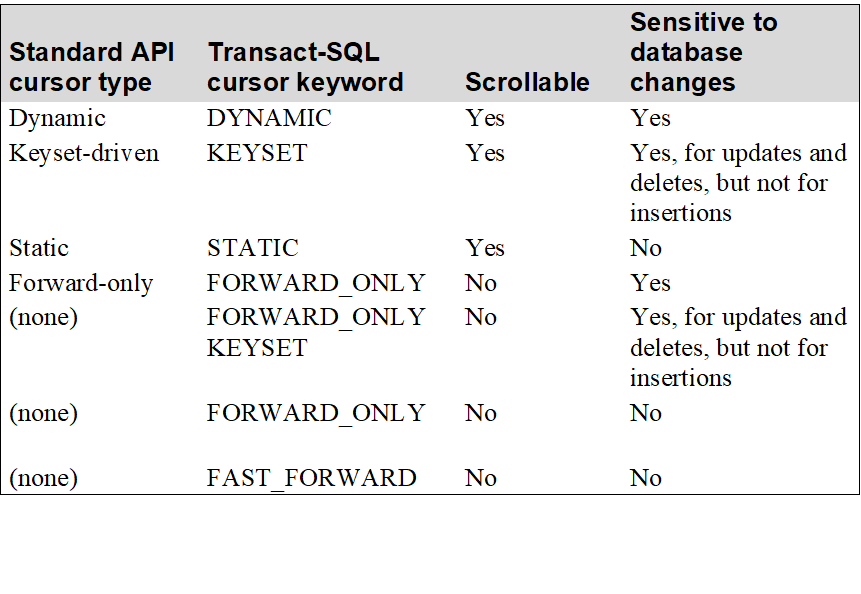
**Characteristics**

1. Scrollability

Forward-only cursor only iterates forward Scrollable cursors can iterate forward and backward (slower)

2. Sensitivity

Static cursors are insensitive to changes to the result set (fastest) Keyset-driven cursors are sensitive to Updates and Deletes (medium speed) Dynamic cursors are sensitive to all changes (slowest)



**Cursor Processing Statement**

DECLARE CURSOR

OPEN FETCH

CLOSE DEALLOCATE

**Example 1:**

DECLARE Vendors\_Cursor CURSOR

STATIC

FOR

SELECT VendorID, VendorName

FROM Vendors

ORDER BY VendorName;

OPEN Vendors\_Cursor;

FETCH NEXT FROM Vendors\_Cursor;

WHILE @@FETCH\_STATUS = 0

FETCH NEXT FROM Vendors\_Cursor;

CLOSE Vendors\_Cursor;

DEALLOCATE Vendors\_Cursor;

**Default Cursor:**

DECLARE Invoices\_Cursor CURSOR

FOR

SELECT \* FROM Invoices;

**Dynamic Cursor:**

DECLARE VendorInvoice\_Cursor CURSOR

LOCAL DYNAMIC

FOR

SELECT VendorName, InvoiceDate, InvoiceTotal

FROM Vendors JOIN Invoices

ON Vendors.VendorID = Invoices.VendorID

**Updating Through Cursor:**

DECLARE VendorUpdate\_Cursor CURSOR

GLOBAL SCROLL DYNAMIC SCROLL\_LOCKS

FOR

SELECT \* FROM Vendors

FOR UPDATE OF VendorName, VendorAddress1,

VendorAddress2, VendorCity,

VendorState, VendorZipCode,

VendorPhone;

**Fetch Statement:**

FETCH [NEXT|PRIOR|FIRST|LAST|ABSOLUTE n|RELATIVE n]

FROM [GLOBAL] cursor\_name [INTO @variable\_name [, ...]]

**Given:**

DECLARE Vendor\_Cursor CURSOR

STATIC

FOR SELECT VendorID, VendorName FROM Vendors;

**Possible Fetch statements**

-- Retrieves the next row

FETCH FROM Vendor\_Cursor;

-- Retrieves the next row

FETCH NEXT FROM Vendor\_Cursor;

-- Retrieves the previous row

FETCH PRIOR FROM Vendor\_Cursor;

-- Retrieves the first row

FETCH FIRST FROM Vendor\_Cursor;

-- Retrieves the last row

FETCH LAST FROM Vendor\_Cursor;

-- Retrieves the third row FETCH ABSOLUTE 3 FROM Vendor\_Cursor;

FETCH ABSOLUTE 3 FROM Vendor\_Cursor;

-- Retrieves the tenth row after the current row

FETCH RELATIVE 10 FROM Vendor\_Cursor;

-- Retrieves the row two rows before the current row

FETCH RELATIVE –2 FROM Vendor\_Cursor;

-- Retrieves the current row again

FETCH RELATIVE 0 FROM Vendor\_Cursor;

-- Retrieves the next row and assigns the values

-- to two local variables

FETCH FROM Vendor\_Cursor

INTO @VendorIDVar,@VendorNameVar;

**Fetch\_Status**

0 The FETCH was successful.

-1 The FETCH was unsuccessful because it reached the end of the result set.

-2 The FETCH was unsuccessful because the row was deleted.

@@FETCH\_STATUS is global to all of the cursors open on the current connection. So if you’re using multiple cursors, don’t code other statements between Fetch and @@FETCH\_STATUS.

**Fetch Loops:**

FETCH FIRST FROM Vendor\_Cursor;

WHILE @@FETCH\_STATUS = 0

BEGIN

...

FETCH NEXT FROM Vendor\_Cursor;

END;

FETCH LAST FROM Vendor\_Cursor;

WHILE @@FETCH\_STATUS = 0

BEGIN

...

FETCH PRIOR FROM Vendor\_Cursor;

END;

**Fetch Loops const:**

FETCH FIRST FROM Vendor\_Keyset\_Cursor;

WHILE @@FETCH\_STATUS <> -1

BEGIN

IF @@FETCH\_STATUS = -2

PRINT 'Missing row. Value was deleted.';

...

FETCH NEXT FROM Vendor\_Keyset\_Cursor;

END;

**Cursor\_Rows:**

n The number of eligible rows for the most recently opened cursor.

0 No cursor is open or no rows qualified for the cursor.

-1 The number of rows can change because this is a dynamic cursor.

**Example:**

USE AP;

DECLARE Vendor\_Cursor CURSOR

STATIC

FOR

SELECT VendorID, VendorName FROM Vendors

ORDER BY VendorName;

DECLARE @VendorIDVar int, @VendorNameVar varchar(50);

OPEN Vendor\_Cursor;

IF @@CURSOR\_ROWS > 0

BEGIN

FETCH NEXT FROM Vendor\_Cursor

INTO @VendorIDVar, @VendorNameVar;

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT CONVERT(varchar,@VendorIDVar) + ', ' + @VendorNameVar;

FETCH RELATIVE 20 FROM Vendor\_Cursor

INTO @VendorIDVar, @VendorNameVar;

END;

END;

CLOSE Vendor\_Cursor;

DEALLOCATE Vendor\_Cursor;

**Cursor Concurrency Options:**

OPTIMISTIC

No lock is placed on the row, so it can be modified by another process in the time between when you fetch and update the data.

SCROLL\_LOCKS

Each row is locked when it’s fetched, so no other process can modify the row until you release the lock by fetching a different row or closing the cursor.

READ\_ONLY

You can’t update data through the cursor, so rows are never locked

**Dynamic Cursor Example:**

DECLARE Dynamic\_Vendor\_Cursor CURSOR

DYNAMIC SCROLL\_LOCKS

FOR

SELECT \* FROM Vendors

ORDER BY VendorName;

OPEN Dynamic\_Vendor\_Cursor;

FETCH Dynamic\_Vendor\_Cursor;

Where Current Of cursor\_name

UPDATE Vendors

SET VendorName = 'Peerless Networking'

WHERE CURRENT OF Dynamic\_Vendor\_Cursor;

DELETE Vendors

WHERE CURRENT OF Dynamic\_Vendor\_Cursor;