Transact-SQL: Cursor-Konzept

Stephan Karrer

Cursor-Konzept

- Ein Cursor verkörpert die private SQL-Area im Speicher des Benutzers, die für SELECT- und DML-Anweisungen angelegt wird

- expliziter Cursor:
  wird deklariert und programmtechnisch genutzt, meist um die Ergebnismenge einer SELECT-Anweisung zu bearbeiten:
  - Positionierung an bestimmten Zeilen der Ergebnismenge
  - Abrufen einer Zeile oder eines ZeilenBlocks von der aktuellen Position in der Ergebnismenge
  - Erlaubt Datenänderungen in den Zeilen an der aktuellen Position
  - Unterstützt unterschiedliche Sichtbarkeitsebenen bei Änderungen, die von anderen Benutzern an den Datenbankdaten, die im Resultset dargestellt werden, ausgeführt wurden

Programmiermodell

1) Verbinden eines Cursor mit dem Resultset einer SQL-Anweisung, Definition der Eigenschaften des Cursors (z. B., ob die Zeilen im Cursor aktualisiert werden können).

2) Ausführen der SQL-Anweisung zum Füllen des Cursor

3) Abrufen der gewünschten Zeilen

4) Ausführen optionaler Änderungen (UPDATE)

5) Schließen des Cursor

Beispiel für einen einfachen Cursor

```
DECLARE emp_cursor CURSOR FOR
    SELECT last_name FROM employees
        ORDER BY last_name
OPEN emp_cursor
-- Perform the first fetch.
FETCH NEXT FROM emp_cursor
-- Check @@FETCH_STATUS to see
-- if there are any more rows to fetch.
WHILE @@FETCH_STATUS = 0
BEGIN
   -- As long as the previous fetch succeeds.
   FETCH NEXT FROM emp_cursor
END
CLOSE emp_cursor
DEALLOCATE emp_cursor
```

- Durch die FETCH-Anweisung kommt die nächste Zeile in die Ausgabe

Zugriff auf Cursor-Attribute

- @@FETCH_STATUS : Status der letzten Fetch-Anweisung
- @@CURSOR_ROWS : Anzahl der Zeilen im Cursor
- CURSOR_STATUS : Ist der Cursor geöffnet und Information zum Resultset

Cursor-Nutzung mit Variablen

```
DECLARE @lname varchar(50), @fname varchar(50)

DECLARE emp_cursor CURSOR FOR
    SELECT last_name, first_name FROM employees
        ORDER BY last_name, first_name

OPEN emp_cursor

FETCH NEXT FROM emp_cursor INTO @lname, @fname

WHILE @@FETCH_STATUS = 0
BEGIN
   PRINT 'Employee: ' + @lname + ' ' + @fname
   FETCH NEXT FROM emp_cursor INTO @lname, @fname
END

CLOSE emp_cursor
DEALLOCATE emp_cursor
```

- Statt Ausgabe der getroffenen Records können auch für die weitere Verarbeitung Variablen gefüllt werden

Nutzung eines scrollable Cursor

```
DECLARE emp_cursor SCROLL CURSOR FOR
    SELECT last_name, first_name FROM employees
        ORDER BY last_name, first_name
OPEN emp_cursor

-- Fetch the last row in the cursor
FETCH LAST FROM emp_cursor

-- Fetch the row immediately prior to the current row
FETCH PRIOR FROM emp_cursor

-- Fetch the second row in the cursor
FETCH ABSOLUTE 2 FROM emp_cursor

-- Fetch the row that is three rows after the current row
FETCH RELATIVE 3 FROM emp_cursor

-- Fetch the row that is two rows prior to the current row
FETCH RELATIVE -2 FROM emp_cursor

CLOSE emp_cursor     DEALLOCATE emp_cursor
```

- Vorsicht:
  Scrollbare Cursor sind ein Performance-Thema!

Cursor-Syntax

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

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 ] ] ]
[;]
```

- SQL Server erlaubt eine erweiterte Parametrisierung des Cursors gegenüber dem ANSI-Standard

Verwendung von Cursor-Variablen

```
DECLARE @MyVariable CURSOR

DECLARE MyCursor CURSOR FOR
 SELECT last_name FROM employees

SET @MyVariable = MyCursor;

OPEN @MyVariable;

-- Use Cursor

DEALLOCATE MyCursor;
```

```
DECLARE @MyVariable CURSOR

SET @MyVariable =
   CURSOR SCROLL FOR
 SELECT last_name FROM employees

OPEN @MyVariable;

-- Use Cursor

DEALLOCATE @MyVariable
```

- Cursor können sowohl in benannter Form als auch anonym Variablen vom Typ CURSOR zugewiesen und über diese benutzt werden
- Auf diese Weise können auch Prozeduren und Funktionen Cursor als Parameter verwenden

Aktualisierungen (UPDATE/DELETE) via Cursor

- Es kann auch die gerade getroffene Zeile gelöscht oder aktualisiert werden
- Cursor muss schreibbar sein, das ist allerdings default
- Allerdings muss die Tabelle einen Unique Index besitzen, sonst ist der Cursor READONLY

```
BEGIN TRANSACTION

DECLARE region_cursor CURSOR FOR
    SELECT region_name FROM regions

DECLARE @lname varchar(50)

OPEN region_cursor

WHILE(1=1)
BEGIN
    FETCH NEXT FROM region_cursor INTO @lname
    IF @@FETCH_STATUS != 0  BREAK
    UPDATE regions SET region_name = 'Myregion'
          WHERE CURRENT OF region_cursor
END

DEALLOCATE region_cursor


ROLLBACK
```

Abfrage von Cursor - Verwaltungsinformationen

- sp_cursor_list : Liefert Liste der Cursor für die aktuelle Verbindung

- sp_describe_cursor,
  sp_describe_cursor_columns,
  sp_describe_cursor_tables :
  Informationen zu den Eigenschaften eines Cursors