Fetch: Mengenbearbeitung

Arbeitsumgebung (gibbix): bmWP2 (SQL-Server)

A. Fetch mit Cursor

Im folgenden Beispiel wird ein einfacher `CURSOR` für die Zeilen der Tabelle Person deklariert. Hier suchen wir alle Nachnamen, die mit B beginnen. Mit `FETCH NEXT` wird Zeile für Zeile eingelesen. Die `FETCH`-Anweisungen geben den Wert für die in `DECLARE CURSOR` angegebenen Spalten als einzeiliges Resultat zurück.

```
--Cursor declaration
DECLARE contact_cursor CURSOR FOR
   SELECT LastName FROM Person
     WHERE LastName LIKE 'B%'
       ORDER BY LastName;

--Open cursor
OPEN contact_cursor;

-- Perform the first fetch.
FETCH NEXT FROM contact_cursor;

-- Check @@FETCH_STATUS to see if there are any more rows to fetch.
WHILE @@FETCH_STATUS = 0
BEGIN
-- This is executed as long as the previous fetch succeeds.
   FETCH NEXT FROM contact_cursor;
END
-- Free the cursor memory
CLOSE contact_cursor;
DEALLOCATE contact_cursor;
GO
```

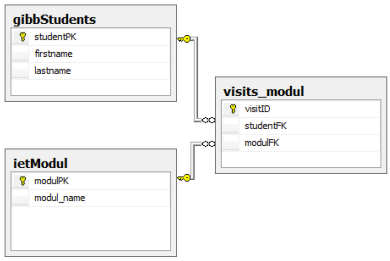
B. Verwenden von FETCH, um Werte in Variablen zu speichern

Das folgende Beispiel ähnelt Beispiel A. Allerdings wird hier die Ausgabe der `FETCH`-Anweisungen in lokalen Variablen gespeichert und nicht direkt an den Client zurückgegeben. Die `PRINT`-Anweisung fasst die Variablen zu einer Zeichenfolge zusammen und gibt sie an den Client zurück. IN SQL werden Variablen mit einem beginnenden `@` gekennzeichnet.

```
-- Declare the variables to store the values returned by FETCH.
DECLARE @LastName varchar(50), @FirstName varchar(50);

DECLARE contact_cursor CURSOR FOR
SELECT LastName, FirstName FROM Person
WHERE LastName LIKE 'B%'
ORDER BY LastName, FirstName;

OPEN contact_cursor;
-- Perform the first fetch and store the values in variables.
-- Note: The variables are in the same order as the columns
-- in the SELECT statement.
FETCH NEXT FROM contact_cursor
INTO @LastName, @FirstName;
-- Check @@FETCH_STATUS to see if there are any more rows to fetch.
WHILE @@FETCH_STATUS = 0
BEGIN
   -- Concatenate and display the current values in the variables.
   PRINT 'Contact Name: ' + @FirstName + ' ' +  @LastName

   -- This is executed as long as the previous fetch succeeds.
   FETCH NEXT FROM contact_cursor
   INTO @LastName, @FirstName;
END

CLOSE contact_cursor;
DEALLOCATE contact_cursor;
GO
```

Repetitorium AB104-08: C. Auftrag: Let's fetch

Im Ordner AB104-08_Script sind die folgenden Dateien vorhanden:
`000_CREATE_DB_AB104-08.sql`
`001_CREATE_TABLE_AB104-08.sql`
Mit diesem lässt sich folgendes DB-Modell erzeugen:



Nr. 1: Erstellen Sie ein Script `001-LETS-FETCH.sql`, mit dem Sie als erstes die Tabellen gibbStudents, ietModul und visits-modul leeren.
Achten Sie auf die richtige Reihenfolge, damit die referentielle Integrität nicht verletzt wird.

Nr. 2: Füllen Sie 20 Schüler in die Tabelle gibbStudents ein:

```
INSERT INTO gibbStudents
               (firstname
               ,lastname)
         VALUES
               ('John','Doe')
               ,('Bootsy','Collins')
               ,...
```

Nr. 3: Erfassen Sie folgende Module: 'Modul 104 Datenbank' und 'Modul 239 Internetserver in Betrieb nehmen'.

```
INSERT INTO ietModul (modulname)
               VALUES ('Modul 104 Datenbank')

INSERT INTO ietModul (modulname)
               VALUES ('Modul 239 Internetserver in Betrieb nehmen')
```

Nr. 4: Verbinden Sie mit Hilfe der „FETCH mit Cursor Anweisung" jeden erfassten Schüler mit den beiden Modulen. Sie benötigen hierzu mehrere Variablen:

```
DECLARE @student_pk bigint;
DECLARE @modul_104 int;
DECLARE @modul_239 int;
SET @modul_104 = (SELECT …)
SET @modul_239 = (SELECT …)
DECLARE student_cursor CURSOR FOR
       SELECT studentPK FROM gibbStudents
OPEN student_cursor;
-- Erster Schüler
FETCH NEXT FROM student_cursor
       INTO @student_pk;
WHILE @@FETCH_STATUS = 0
BEGIN
       -- Modul 104 Datenbank
       INSERT INTO …  VALUES…
       -- Modul 239 Datenbank
       INSERT INTO …
              VALUES …
      FETCH NEXT FROM student_cursor INTO …
END
CLOSE …
DEALLOCATE student_cursor;
```

Nr. 5: Überprüfen Sie das Resultat.

```
SELECT * FROM gibbStudents AS s
       INNER JOIN visits_modul AS vm ON vm.studentFK = s.studentPK
       INNER JOIN ietModul AS m ON m.modulPK = vm.modulFK
```

VIEL ERFOLG!