Lab 05: Stored Procedures

※\(^o^)/※

September, 2019

Contents

Introduction

This lab aims to help students get used to stored procedures, user-defined functions and triggers in T-SQL.

Lab Activities

Stored Procedures

A stored procedure with no parameter

Create the procedure

```
5  USE AccountPayables;
6  IF OBJECT_ID('spInvoiceReport') IS NOT NULL
7    DROP PROC spInvoiceReport;
```

```
GO

CREATE PROC spInvoiceReport
AS

SELECT VendorName, InvoiceNumber, InvoiceDate, InvoiceTotal
FROM Invoices JOIN Vendors
     ON Invoices.VendorID = Vendors.VendorID
WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0
ORDER BY VendorName;
GO
```

Test the procedure

```
USE AccountPayables;
EXEC spInvoiceReport;
GO
```

A stored procedure with input and output parameters

Create the procedure

```
USE AccountPayables;
IF OBJECT_ID('spInvTotal3') IS NOT NULL
    DROP PROC spInvTotal3;
GO

CREATE PROC spInvTotal3
       @InvTotal money OUTPUT,
       @DateVar smalldatetime = NULL,
       @VendorVar varchar(40) = '%'
AS

IF @DateVar IS NULL
   SELECT @DateVar = MIN(InvoiceDate) FROM Invoices;

SELECT @InvTotal = SUM(InvoiceTotal)
FROM Invoices JOIN Vendors
    ON Invoices.VendorID = Vendors.VendorID
WHERE (InvoiceDate >= @DateVar) AND
      (VendorName LIKE @VendorVar);
```

Test the procedure with parameters passed by position

```
USE AccountPayables;
DECLARE @MyInvTotal money;
EXEC spInvTotal3 @MyInvTotal OUTPUT, '2016-02-01', 'P%';
```

```
53
54  PRINT '$' + CONVERT(varchar,@MyInvTotal,1);
```

Test the procedure with parameters passed by name

```
58  USE AccountPayables;
59  DECLARE @MyInvTotal money;
60  EXEC spInvTotal3 @DateVar = '2016-02-01', @VendorVar = 'P%',
61      @InvTotal = @MyInvTotal OUTPUT;
62
63  PRINT '$' + CONVERT(varchar,@MyInvTotal,1);
```

A stored procedure with return

Create the procedure

```
67  USE AccountPayables;
68  IF OBJECT_ID('spInvCount') IS NOT NULL
69      DROP PROC spInvCount;
70  GO
71
72  CREATE PROC spInvCount
73         @DateVar smalldatetime = NULL,
74         @VendorVar varchar(40) = '%'
75  AS
76
77  IF @DateVar IS NULL
78     SELECT @DateVar = MIN(InvoiceDate) FROM Invoices;
79
80  DECLARE @InvCount int;
81
82  SELECT @InvCount = COUNT(InvoiceID)
83  FROM Invoices JOIN Vendors
84      ON Invoices.VendorID = Vendors.VendorID
85  WHERE (InvoiceDate >= @DateVar) AND
86         (VendorName LIKE @VendorVar);
87
88  RETURN @InvCount;
```

Test the procedure

```
92  USE AccountPayables;
93  DECLARE @InvCount int;
94  EXEC @InvCount = spInvCount '2016-02-01', 'P%';
95  PRINT 'Invoice count: ' + CONVERT(varchar, @InvCount);
```

A stored procedure for inserting invoices with data validation

Create the procedure

```
100 USE AccountPayables;
101 IF OBJECT_ID('spInsertInvoice') IS NOT NULL
102     DROP PROC spInsertInvoice;
103 GO
104
105 CREATE PROC spInsertInvoice
106         @VendorID       int = NULL,
107         @InvoiceNumber  varchar(50) = NULL,
108         @InvoiceDate    smalldatetime = NULL,
109         @InvoiceTotal   money = NULL,
110         @PaymentTotal   money = NULL,
111         @CreditTotal    money = NULL,
112         @TermsID        int = NULL,
113         @InvoiceDueDate smalldatetime = NULL,
114         @PaymentDate    smalldatetime = NULL
115 AS
116
117 IF NOT EXISTS (SELECT * FROM Vendors WHERE VendorID = @VendorID)
118     THROW 50001, 'Invalid VendorID.', 1;
119 IF @InvoiceNumber IS NULL
120     THROW 50001, 'Invalid InvoiceNumber.', 1;
121 IF @InvoiceDate IS NULL OR @InvoiceDate > GETDATE()
122         OR DATEDIFF(dd, @InvoiceDate, GETDATE()) > 30
123     THROW 50001, 'Invalid InvoiceDate.', 1;
124 IF @InvoiceTotal IS NULL OR @InvoiceTotal <= 0
125     THROW 50001, 'Invalid InvoiceTotal.', 1;
126 IF @PaymentTotal IS NULL
127     SET @PaymentTotal = 0;
128 IF @CreditTotal IS NULL
129     SET @CreditTotal = 0;
130 IF @CreditTotal > @InvoiceTotal
131     THROW 50001, 'Invalid CreditTotal.', 1;
132 IF @PaymentTotal > @InvoiceTotal - @CreditTotal
133     THROW 50001, 'Invalid PaymentTotal.', 1;
134 IF NOT EXISTS (SELECT * FROM Terms WHERE TermsID = @TermsID)
135     IF @TermsID IS NULL
136         SELECT @TermsID = DefaultTermsID
137         FROM Vendors
138         WHERE VendorID = @VendorID;
139     ELSE  -- @TermsID IS NOT NULL
140         THROW 50001, 'Invalid TermsID.', 1;
141 IF @InvoiceDueDate IS NULL
```

```
    SET @InvoiceDueDate = @InvoiceDate +
        (SELECT TermsDueDays FROM Terms WHERE TermsID = @TermsID);
ELSE  -- @InvoiceDueDate IS NOT NULL
    IF @InvoiceDueDate < @InvoiceDate OR
            DATEDIFF(dd, @InvoiceDueDate, @InvoiceDate) > 180
        THROW 50001, 'Invalid InvoiceDueDate.', 1;
IF @PaymentDate < @InvoiceDate OR
        DATEDIFF(dd, @PaymentDate, GETDATE()) > 14
    THROW 50001, 'Invalid PaymentDate.', 1;

INSERT Invoices
VALUES (@VendorID, @InvoiceNumber, @InvoiceDate, @InvoiceTotal,
        @PaymentTotal, @CreditTotal, @TermsID, @InvoiceDueDate,
        @PaymentDate);
RETURN @@IDENTITY;
GO
```

Test the procedure

```
USE AccountPayables;
BEGIN TRY
    DECLARE @InvoiceID int;
    EXEC @InvoiceID = spInsertInvoice
         @VendorID = 799,
         @InvoiceNumber = 'RZ99381',
         @InvoiceDate = '2016-04-12',
         @InvoiceTotal = 1292.45;
    PRINT 'Row was inserted.';
    PRINT 'New InvoiceID: ' + CONVERT(varchar, @InvoiceID);
END TRY
BEGIN CATCH
    PRINT 'An error occurred. Row was not inserted.';
    PRINT 'Error number: ' + CONVERT(varchar, ERROR_NUMBER());
    PRINT 'Error message: ' + CONVERT(varchar, ERROR_MESSAGE());
END CATCH;
GO
```

Passing tables to stored procedures

Create the procedure

```
USE AccountPayables;
-- drop stored procedure if it exists already
IF OBJECT_ID('spInsertLineItems') IS NOT NULL
    DROP PROC spInsertLineItems;
GO
```

```
186
187 -- drop table type if it exists already
188 IF  EXISTS (SELECT * FROM sys.types WHERE name = 'LineItems')
189     DROP TYPE LineItems;
190 GO
191
192 -- create the user-defined table type named LineItems
193 CREATE TYPE LineItems AS
194 TABLE
195 (InvoiceID         INT           NOT NULL,
196 InvoiceSequence    SMALLINT      NOT NULL,
197 AccountNo          INT           NOT NULL,
198 ItemAmount         MONEY         NOT NULL,
199 ItemDescription    VARCHAR(100)  NOT NULL,
200 PRIMARY KEY (InvoiceID, InvoiceSequence));
201 GO
202
203 -- create a stored procedure that accepts the LineItems type
204 CREATE PROC spInsertLineItems
205     @LineItems LineItems READONLY
206 AS
207     INSERT INTO InvoiceLineItems
208     SELECT *
209     FROM @LineItems;
210 GO
211 -- start snippet sp_table_passing_test
212 USE AccountPayables;
213 -- delete old line item data
214 DELETE FROM InvoiceLineItems WHERE InvoiceID = 114;
215
216 -- declare a variable for the LineItems type
217 DECLARE @LineItems LineItems;
218
219 -- insert rows into the LineItems variable
220 INSERT INTO @LineItems VALUES (114, 1, 553, 127.75, 'Freight');
221 INSERT INTO @LineItems VALUES (114, 2, 553, 29.25, 'Freight');
222 INSERT INTO @LineItems VALUES (114, 3, 553, 48.50, 'Freight');
223
224 -- execute the stored procedure
225 EXEC spInsertLineItems @LineItems;
226 -- end snippet sp_table_passing_test
227
228 -- start snippet sp_modify
229 -- create a store procedure
230 USE AccountPayables;
231 IF OBJECT_ID('spVendorState') IS NOT NULL
```

```
    DROP PROC spVendorState;
GO

CREATE PROC spVendorState
       @State varchar(20)
AS
SELECT VendorName
FROM Vendors
WHERE VendorState = @State;

EXEC sp_HelpText spVendorState

-- modify it
USE AccountPayables;
GO

ALTER PROC spVendorState
      @State varchar(20) = NULL
AS
IF @State IS NULL
   SELECT VendorName
   FROM Vendors;
ELSE
   SELECT VendorName
   FROM Vendors
   WHERE VendorState = @State;

EXEC sp_HelpText spVendorState
-- end snippet sp_modify
```

Test the procedure

```
USE AccountPayables;
-- delete old line item data
DELETE FROM InvoiceLineItems WHERE InvoiceID = 114;

-- declare a variable for the LineItems type
DECLARE @LineItems LineItems;

-- insert rows into the LineItems variable
INSERT INTO @LineItems VALUES (114, 1, 553, 127.75, 'Freight');
INSERT INTO @LineItems VALUES (114, 2, 553, 29.25, 'Freight');
INSERT INTO @LineItems VALUES (114, 3, 553, 48.50, 'Freight');

-- execute the stored procedure
EXEC spInsertLineItems @LineItems;
```

Modify stored procedures

Create the procedure

```
-- create a store procedure
USE AccountPayables;
IF OBJECT_ID('spVendorState') IS NOT NULL
    DROP PROC spVendorState;
GO

CREATE PROC spVendorState
       @State varchar(20)
AS
SELECT VendorName
FROM Vendors
WHERE VendorState = @State;

EXEC sp_HelpText spVendorState

-- modify it
USE AccountPayables;
GO

ALTER PROC spVendorState
      @State varchar(20) = NULL
AS
IF @State IS NULL
   SELECT VendorName
   FROM Vendors;
ELSE
   SELECT VendorName
   FROM Vendors
   WHERE VendorState = @State;

EXEC sp_HelpText spVendorState
```

Exercise 01

Create a procedure named spBalancedRange:
- Output: - The procedure should return a result set consisting of VendorName, InvoiceNumber, and Balance for each invoice with a balance due (InvoiceTotal - PaymentTotal - CreditTotal > 0). - Results should be sorted with largest balance due first. - Inputs: three optional parameters - @VendorVar is a mask that's used with a LIKE operator to filter by VendorName, e.g. @VendorVar = 'K%' - @BalanceMin and @BalanceMax are parameters used to specify the

requested range of balances due. If called with no parameters or with @Balance-Max = 0, the procedure should return all invoices with a balance due.

Exercise 02

Call the procedure from exercise 01 for the following situations: - Passed by position with @VendorVar='M%' and no balance range - Passed by name with @VendorVar omitted a balance range from $200 to $500 - Passed by position with a balance due that's less than $200, filtering for vendors whose name begin with C or F