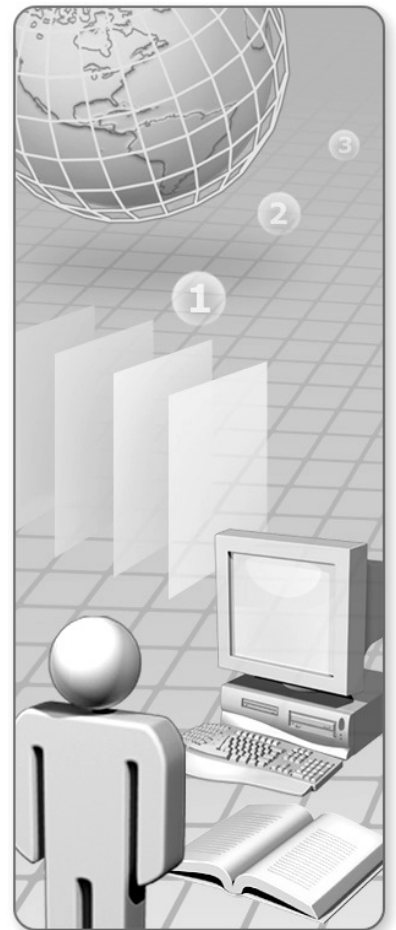


SQLHOL26: SQL Server 2008 Table Valued Parameters

Table of Contents

Before You Begin	1
Exercise: Working with Table Valued Parameters	3



Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links are provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Copyright © 2007 Microsoft Corporation. All rights reserved.

Microsoft, Excel, Office, and SQL Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Before You Begin

Estimated time to complete this lab

90 minutes

Objectives

After completing this lab, you will be able to:

- Work with Change Data Capture in SQL Server 2008

Prerequisites

Before working on this lab, you must have:

- Experience of Transact-SQL programming and SQL Server Management Studio.

Lab scenario

The objective of this Hands-on-Lab is to give you an overview of Change Data Capture.

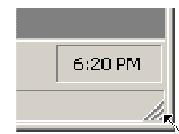
Change data capture is designed to capture insert, update, and delete activity applied to SQL Server tables, and to make the details of the changes available in an easily consumed relational format. The change tables used by change data capture contain columns that mirror the column structure of a tracked source table, along with the metadata needed to understand the changes that have occurred.

Virtual PC

This lab makes use of Microsoft Virtual PC 2007, which is an application that allows you to run multiple virtual computers on the same physical hardware. During the lab, you will use a virtual machine running Microsoft Windows Server® 2003.

Before you start the lab, familiarize yourself with the following basics of Virtual PC:

- To switch the focus for your mouse and keyboard to the virtual machine, click inside the virtual machine window.
- To remove the focus from a virtual machine, move the mouse pointer outside the virtual machine window.
- To mimic the CTRL+ALT+DELETE key combination inside a virtual machine, use <RIGHT>ALT+DEL. In Virtual PC, the <RIGHT>ALT key is called the host key.
- To enlarge the size of the virtual machine window, drag the lower-right corner of the window as seen in the screenshot.



- To switch to and from full-screen mode, press <RIGHT>ALT+ENTER.

Computers in this lab

This lab uses one computer as described in the following table. Before you begin the lab, you must start the virtual machines and then log on to the computer. In each exercise, you only have to start the virtual machine that is needed.

Virtual Machine	Computer Name	User Name	Password
SQL Server 2008 HOLs	MIAMI	Student	Pa\$\$w0rd

Start the Virtual Machine

1. Launch Microsoft Virtual PC from the Start menu or Desktop. If the Virtual PC Console does not appear, look for its icon in the System Tray, and double-click the Microsoft Virtual PC icon in the System Tray.
2. Select **SQL Server 2008 HOLs** and click **Start**.
3. When the virtual server is running, on the **Action** menu within the virtual server window, click **Ctrl+Alt+Del** (or press **Right Alt+Del** on your keyboard) to send a Ctrl+Alt+Del sequence to the login dialog box within the virtual server window.
4. Type the following information, and then click OK:
 - User name: **Student**
 - Password: **Pa\$\$w0rd**

Exercise: Working with Table Valued Parameters

Table-valued parameters are a new parameter type in SQL Server 2008. Table-valued parameters are declared by using user-defined table types. You can use table-valued parameters to send multiple rows of data to a Transact-SQL statement or a routine, such as a stored procedure or function, without creating a temporary table or many parameters.

Table-valued parameters are like parameter arrays in OLE DB and ODBC, but offer more flexibility and closer integration with Transact-SQL. Table-valued parameters also have the benefit of being able to participate in set-based operations

In this exercise your goal is to insert a entire set of data using multiple rows by using a single stored procedure.

Before SQL Server 2008, there was no integrated functionality that supported to call a stored procedure to achieve that goal, using one server round trip.

You will implement the workarounds that were existing pre-table valued parameters and then implement the solution using table valued parameters.

Start SQL Server Management Studio

1. Click **Start | All Programs | Microsoft SQL Server 2008 | SQL Management Studio** to start SQL Server Management Studio.
2. Click **Connect** in the **Connect to Server** dialog box after ensuring the following settings:
 - Server type: Database Engine
 - Server name: (local)
 - Authentication: Windows Authentication
3. Click on **'File | Open | File.**
4. Browse to the C:\SQHOLS folder and open the Labscript.sql file in the Table Valued Parameters folder

Inserting Data using Multiple Parameters in a stored procedure

1. Review and **Highlight** the following code and click **Execute**:

```
USE ADVENTUREWORKS
GO
CREATE TABLE dbo.Employee(
EmpID int NOT NULL,
```

```
EmpName nvarchar(100) NOT NULL,  
EmpEmail nvarchar(100) NOT NULL)
```

2. Review and **Highlight** the following code and click **Execute**:

```
USE AdventureWorks  
GO  
CREATE PROCEDURE NewEmployeeMS(@EmpID int,@EmpName nvarchar(100),@EmpEmail  
nvarchar(100))  
As  
BEGIN  
INSERT INTO dbo.Employee  
values(  
@EmpID, @EmpName, @EmpEmail)  
END
```

3. Review and **Highlight** the following code and click **Execute**:

```
USE AdventureWorks  
GO  
execute NewEmployeeMS 1,'John McLean','JohnMcLean@contoso.com'  
  
execute NewEmployeeMS 2,'Bob Smith','BobSmith@contoso.com'  
  
execute NewEmployeeMS 3,'Ted Connery','TedConnery@contoso.com'
```

4. Review and **Highlight** the following code and click **Execute**:

```
USE AdventureWorks  
select * from dbo.Employee;  
GO
```

Note: Drawbacks of the above solution:

1. Multiple round trips
2. Stored procedure needs to be executed multiple times
3. Inefficient code

Inserting data using local temporary tables

1. Review and **Highlight** the following code and click **Execute**:

```
USE ADVENTUREWORKS  
GO  
Truncate table dbo.Employee
```

2. Review and **Highlight** the following code and click **Execute**:

```
USE AdventureWorks  
GO  
CREATE PROCEDURE NewEmployeeTempTable  
As  
BEGIN  
INSERT INTO dbo.Employee
```

```
SELECT * FROM #EmployeeTempTable
ENDINSERT INTO dbo.Employee
        values(
            @EmpID, @EmpName, @EmpEmail)
END
```

3. Review and **Highlight** the following code and click **Execute**:

```
USE AdventureWorks
GO
CREATE TABLE dbo.#EmployeeTempTable(
EmpID int NOT NULL,
EmpName nvarchar(100) NOT NULL,
EmpEmail nvarchar(100) NOT NULL)
Go
```

Note: This temporary table is created on the client side at runtime, which will result in the stored procedure failing to execute on the server side as long as the client side is not defined.

4. Review and **Highlight** the following code and click **Execute**:

```
USE AdventureWorks
INSERT INTO #EmployeeTempTable
VALUES(1, 'John McLean', 'JohnMcLean@contoso.com')
INSERT INTO #EmployeeTempTable
VALUES(2, 'Bob Smith', 'BobSmith@contoso.com')
INSERT INTO #EmployeeTempTable
VALUES(3, 'Ted Connery', 'TedConnery@contoso.com')
```

Note: The insertion now occurs in the clients temporary table

5. Review and **Highlight** the following code and click **Execute**:

```
USE AdventureWorks
GO
exec dbo.NewEmployeeTempTable
```

6. Review and **Highlight** the following code and click **Execute**:

```
USE AdventureWorks
GO
SELECT * FROM dbo.Employee
GO
```

7. Modify the code as shown in the following code sample, and click **Execute**:

```
USE AdventureWorks
GO
Drop table dbo.#EmployeeTempTable
GO
```

Note:

1. A temporary table is created and populated on disk, increasing the disk I/O
2. They are created in tempdb and are prone to locking and blocking
3. You have to manually clean data when you are done with it by dropping the temporary table
4. Usage of temporary tables lead to frequent stored procedures re-compilations

Inserting data using table valued parameters

1. Review and **Highlight** the following code and click **Execute**:

```
USE ADVENTUREWORKS
GO
Truncate table dbo.Employee
```

2. Review and **Highlight** the following code and click **Execute**:

```
USE ADVENTUREWORKS
GO
CREATE TYPE EmployeeTableType AS TABLE
(EmpID INT, EmpName nvarchar(100), EmpEmail nvarchar(100))
```

Note: : It is required to create a table type to work with table-valued parameters

3. Review and **Highlight** the following code and click **Execute**:

```
USE ADVENTUREWORKS
GO
CREATE PROCEDURE NewEmployee(@EmployeeDetails EmployeeTableType READONLY)
As
BEGIN
    INSERT INTO dbo.Employee
    SELECT * FROM @EmployeeDetails
END
```

Note: Table-valued parameters must be passed as input READONLY parameters to Transact-SQL routines. You cannot perform DML operations such as UPDATE, DELETE, or INSERT on a table-valued parameter in the body of a routine

4. Review and **Highlight** the following code and click **Execute**:

```
USE AdventureWorks
GO
DECLARE @NewEmployees EmployeeTableType

INSERT INTO @NewEmployees
VALUES(1, 'John McLean', 'JohnMcLean@contoso.com')
INSERT INTO @NewEmployees
VALUES(2, 'Bob Smith', 'BobSmith@contoso.com')
INSERT INTO @NewEmployees
```

```
VALUES(3,'Ted Connery','TedConnery@contoso.com')  
EXECUTE NewEmployee @NewEmployees  
Go
```

Note: After the routine is out of scope, the table-valued parameter is no longer available.

5. Review and **Highlight** the following code and click **Execute**:

```
USE AdventureWorks  
GO  
select * from dbo.Employee
```

6. Modify the code as shown in the following code sample, and click **Execute**:

Note: Benefits of using Table-valued parameters

Table-valued parameters offer more flexibility and in some cases better performance than temporary tables or other ways to pass a list of parameters. Table-valued parameters offer the following benefits:

Have a well defined scope at the end of which they are automatically cleared.

Do not acquire locks for the initial population of data from a client.

Do not cause a statement to recompile.

Provide a simple programming model.

Enable you to include complex business logic in a single routine.

Reduce round trips to the server.

Can have a table structure of different cardinality.

Are strongly typed.

Enable the client to specify sort order and unique keys.

7. Close all applications and do not save changes.
8. Close Virtual PC and discard changes.