Applied SQL Presentation Samples

Event: DataFlex Entwickler Tag 2014, Frankfurt, Germany

Date: November 19h, 2014

Author/Trainer: Eddy Kleinjan, Data Access Europe

Contents

Intro

Requirements

- Visual DataFlex 18.0
- SQL Server 2008 SR2 or higher

Preparation Steps

- Make a copy of the "Order Entry" workspace and name it "Order Entry SQL"
- Start Studio 18 and open the copied workspace
- Configure the database driver.
  Database → Configure Database Drivers
- Create a new SQL database (OrderEntry180)
- Convert all existing tables to SQL using Database Builder
- Check the application and make sure it works with the new SQL Database
- INT files have been created

Copy a table using INTO

```
-- Create backup
SELECT *
INTO CustomerBackup
FROM Customer

-- Empty table
TRUNCATE TABLE Customer
-- Alternative: DELETE FROM Customer

-- Copy backup data back into table
INSERT INTO Customer
SELECT * FROM CustomerBackup

-- Drop the backup table
DROP TABLE CustomerBackup
```

OUTER APPLY

Select customers with their last order

```
SELECT
        [Customer_Number]
        ,[Name]
        ,[LastOrder].[Order_Total] [LastOrderTotal]

FROM
        [Customer]
        OUTER APPLY (
                SELECT TOP 1
                        [Order_Total]
                FROM
                        [OrderHea]
                WHERE
                        [OrderHea].[Customer_Number] = [Customer].[Customer_Number]
                ORDER BY
                        [OrderHea].[Order_Date] DESC
        ) [LastOrder]
```

Use CROSS APPLY (like INNER JOIN) to only show customer that have ordered something.

Alternative method using LEFT OUTER JOIN based on select expression

```
SELECT
        [Customer].[Customer_Number]
        ,[Customer].[Name]
        ,[OrderHea].[Order_Total] [LastOrderTotal]
FROM
        [Customer]
        LEFT OUTER JOIN [OrderHea] ON [OrderHea].[Order_Number] = (
                SELECT MAX([Order_Number])
                FROM [OrderHea]
                WHERE [OrderHea].[Customer_Number] = [Customer].[Customer_Number]
                )
```

Alternative method using column expression with query

```
SELECT
        [Customer].[Customer_Number]
        ,[Customer].[Name]
        ,(      SELECT TOP 1 [Order_Total]
                FROM [OrderHea]
                WHERE [OrderHea].[Customer_Number] = [Customer].[Customer_Number]
                ORDER BY [OrderHea].[Order_Number] DESC
                ) [LastOrderTotal]
FROM
        [Customer]
```

Create SQL View based Selection list

Create SQL View

```
CREATE VIEW [vwOrderHea] AS
SELECT
        [OrderHea].[Order_Number]
        ,[OrderHea].[Customer_Number]
        ,[Customer].[Name]
        ,[OrderHea].[Order_Date]
        ,[OrderHea].[Order_Total]
FROM
        [OrderHea]
        INNER JOIN [Customer] ON
                [Customer].[Customer_Number] = [OrderHea].[Customer_Number]
```

Connect to view

- **Puts SQL view into Filelist**
- **Creates Data Dictionary**

Change vwOrder.int

- **Create a 'virtual' index for each column**
  This will make each column sortable. The index number equals the fieldnumber.
  Use DataFlex 18.0; indexes will be 'virtual' on views

```
DRIVER_NAME  MSSQLDRV
SERVER_NAME  SERVER=.\SQL2008;Trusted_Connection=yes;DATABASE=OrderEntry180
DATABASE_NAME  vwOrderHea
SCHEMA_NAME  dbo

RECNUM_TABLE  NO
TABLE_CHARACTER_FORMAT  Ansi
USE_DUMMY_ZERO_DATE  YES

FIELD_NUMBER  4
FIELD_TYPE DATETIME

PRIMARY_INDEX 1

;Index for he
INDEX_NUMBER 1
INDEX_NUMBER_SEGMENTS  1
INDEX_SEGMENT_FIELD 1

INDEX_NUMBER 2
INDEX_NUMBER_SEGMENTS  2
INDEX_SEGMENT_FIELD 2
INDEX_SEGMENT_FIELD 1

INDEX_NUMBER 3
INDEX_NUMBER_SEGMENTS  2
INDEX_SEGMENT_FIELD 3
INDEX_SEGMENT_FIELD 1

INDEX_NUMBER 4
INDEX_NUMBER_SEGMENTS  2
INDEX_SEGMENT_FIELD 4
INDEX_SEGMENT_FIELD 1

INDEX_NUMBER 5
INDEX_NUMBER_SEGMENTS  2
INDEX_SEGMENT_FIELD 5
INDEX_SEGMENT_FIELD 1
```

Create vwOrderhea.sl

- Save OrderHea.sl as vwOrderHea.sl
- Replace the DD class from Orderhea_DD with cvwOrderHeaDataDictionary
- Replace OrderHea.* and Customer.* with vwOrderhea.*
- oSellist → Set peUpdateMode to umPromptCustom
- oSellist →

```
Function SeedData Handle hoUpdateColumn Handle hoInitialColumn Returns Boolean
    RowID riRec
    Handle hoServer hoInvServer
    Boolean bFound
    Get Server to hoServer
    Get Server of (phoInvokingObject(Self)) to hoInvServer
    Get CurrentRowId of (Server(phoInvokingObject(Self))) to riRec
    Move False to bFound
    If (not(IsNullRowID(riRec))) Begin
        Send FindByRowId of hoInvServer (Main_File(hoInvServer)) riRec // Refind current record
        Send Clear of hoServer
        Move OrderHea.Order_Number to vwOrderHea.Order_Number
        Send Find of hoServer EQ Index.1
        Move (Found) to bFound
    End
    Function_Return bFound
End_Function

Procedure OnMoveValueOutByCustom
    RowID riRec
    Integer[] SelRowsIndexes
    Handle hoServer hoInvServer hoDataSource
    Get Server to hoServer
    Get Server of (phoInvokingObject(Self)) to hoInvServer
    Get pSelectedRows to SelRowsIndexes
    If (SizeOfArray(SelRowsIndexes)>0) Begin
        Get phoDataSource to hoDataSource
        Get RowTag of hoDataSource SelRowsIndexes[0] to riRec
        Send FindByRowId of hoServer (Main_File(hoServer)) riRec // Refind selected record
        Send Clear of hoInvServer
        Move vwOrderHea.Order_Number to OrderHea.Order_Number
        Send Find of hoInvServer EQ Index.1
    End
End_Procedure
```

Change Order.vw

- Include Use vwOrderhea.sl
- Set Field_Prompt_Object Field OrderHea.Order_Number to OrderHea_SL

Create a free search filter

vwOrderheaFilter.sl

- Save vwOrderhea.sl as vwOrderheaFilter.sl
- Rename vwOrderHea_sl as vwOrderHeaFilter_sl

vwOrderheaSL: Create property for filter value

```
Property String psFilterValue
```

OrderheaDD: OnConstrain

Create a OnContrain procedure that will apply the types in filter value.

```
Set pbUseDDSQLFilters to True

Procedure OnConstrain
    String sValue sFilter
    Get psFilterValue to sValue
    Move (Trim(sValue)) to sValue
    If (sValue > "") Begin
        Move (sFilter + "CONVERT(varchar, [Order_Number]) LIKE '%" + sValue + "%'") to sFilter
        Move (sFilter + "OR CONVERT(varchar, [Customer_Number]) LIKE '%" + sValue + "%'") to sFilter
        Move (sFilter + "OR [Name] LIKE '%" + sValue + "%'") to sFilter
        Move (sFilter + "OR CONVERT(varchar, [Order_Date], 105) LIKE '%" + sValue + "%'") to sFilter
        Move (sFilter + "OR CONVERT(varchar, [Order_Total]) LIKE '%" + sValue + "%'") to sFilter
    End
    Set psSQLFilter to sFilter
End_Procedure
```

oSellist: AppyFilter

Create a procedure ApplyFilter that will apply the typed in value

```
Procedure ApplyFilter
    RowID riRow
    Handle hoServer
    Get Server to hoServer
    Send Rebuild_Constraints of hoServer
    Send Request_Read of hoServer GE (Main_File(hoServer)) Index.1
    If (not(Found)) Send Request_Read of hoServer LE (Main_File(hoServer)) Index.1
    If (not(Found)) Send Request_Clear of hoServer LE (Main_File(hoServer)) Index.1
    Move (GetRowID(Main_File(hoServer))) to riRow
    Send RefreshDataFromMatchingRow riRow 0
End_Procedure
```

oFilterform: New Form object

- Create a form object oFilterForm

```
Procedure OnChange
    Set psFilterValue to (Value(Self))
    Send ApplyFilter of oSelList
End_Procedure
```

Show extra information

Create a generic SQL handler for ease of use

```
Object oSqlExec is a cObject
    Property Handle phoSqlMgr // SQL Handle Manager
    Property Handle phoDbc // Database connection

    Procedure End_Construct_Object
        Handle hoSqlMgr hoDbc
        Forward Send End_Construct_Object
        Get Create (RefClass(cSQLHandleManager)) to hoSqlMgr
        Set phoSqlMgr to hoSqlMgr
        Get SQLFileConnect of hoSqlMgr Customer.File_Number to hoDbc
        Set phoDbc to hoDbc
    End_Procedure

    Function ExecuteSql String sSql Returns String[][]
        Handle hoSql hoDbc hoStmt
        String[][] asResult
        String sValue
        Integer iResult iCol iColCount iRow

        Get phoSqlMgr to hoSql
        Get phoDbc to hoDbc

        Get SQLOpen of hoDbc to hoStmt // Create and execute the statement
        Send SQLExecDirect of hoStmt sSql
        Repeat
            Get SQLStmtAttribute of hoStmt SQLSTMTATTRIB_COLUMNCOUNT to iColCount
            Get SQLFetch of hoStmt to iResult
            While (iResult <> 0)
                For iCol from 1 to iColCount
                    Get SQLColumnValue of hoStmt iCol to sValue
                    Move sValue to asResult[iRow][icol - 1]
                Loop
                Increment iRow
                Get SQLFetch of hoStmt to iResult
            Loop
            Get SQLNextResultSet of hoStmt to iResult
        Until (iResult = 0) // No more result sets
        Send SQLClose of hoStmt

        Function_Return asResult
    End_Function

End_Object
```

Execute a SQL command from the 'Refresh' procedure of a container

```
Procedure RefreshCustomerTotal
    String sSql
    String[][] asResult
    Integer iSep
    Move ("SELECT SUM([OrderHea].[Order_Total]) " ;
        + "FROM [OrderHea] " ;
        + "WHERE [Customer_Number] = " + String(OrderHea.Customer_Number)) ;
        to sSql
    Move (sSql + "; " + sSql) to sSql
    Get ExecuteSql of oSqlExec sSql to asResult
    Get_Attribute DF_DECIMAL_SEPARATOR to iSep
    Set Value of oCustomer_Order_Total to (replace(".", asResult[0][0], Character(iSep)))
End_Procedure
```

```
Procedure Refresh Integer eMode
    Boolean bRec
    Handle hoServer
    Get Server to hoServer
    Get HasRecord of hoServer to bRec
    Set Enabled_State of oPrintBtn to bRec
    Send RefreshCustomerTotal
    Send RefreshTheGrid of oOrderDtl_Grid (oOrderDtl_Grid(Self)) 0
End_Procedure

        Procedure RefreshTheGrid Handle hoGrid Integer iRetainSelectedRowMode
            Handle  hoDataSource
            Integer iRowCount
            Integer iSelectedRow
            RowID   riSelected
            Integer iTopRow
            RowID   riRecord
            Integer iFoundRow
            Get phoDataSource of hoGrid to hoDataSource

            // Check to see if there are any rows in the datasource.
            // No rows, don't do anything.
            Get RowCount of hoDataSource to iRowCount
            If (iRowCount = 0) Procedure_Return

            If (iRetainSelectedRowMode > 0) Begin
                Get SelectedRow of hoDataSource to iSelectedRow
                If (iRetainSelectedRowMode = 2) Begin
                    Get RowTag of hoDataSource iSelectedRow to riSelected
                End
            End
            Get ComTopRowIndex of hoGrid to iTopRow
            Get RowTag of hoDataSource iTopRow to riRecord
            If (not(IsNullRowID(riRecord))) Begin
                Send RefreshDataFromMatchingRow to hoGrid riRecord 0
                If (iRetainSelectedRowMode > 0) Begin
                    If (iRetainSelectedRowMode = 2) Begin
                        Get FindRowIdInCache of hoDataSource riSelected to iFoundRow
                        If (iFoundRow <> -1) Begin
                            Move iFoundRow to iSelectedRow
                        End
                    End
                    Send MoveToRow to hoGrid iSelectedRow
                End
            End
        End_Procedure  // RefreshTheGrid
```

Execute Stored Procedure

Create stored procedure to increase the price by 10%

```
-- [usp_IncreasePrice] @CustomerNumber = 1, @PriceFactor = 1.1

ALTER PROCEDURE [usp_IncreasePrice]
        @CustomerNumber int = NULL
        ,@PriceFactor float = 1
AS
BEGIN
        SET NOCOUNT ON;

        UPDATE [OrderDtl]
        SET
                [Price] = [Price] * @PriceFactor
                ,[Extended_Price] = [Qty_Ordered] * CONVERT(numeric(10,2), [Price] * @PriceFactor)
        FROM
                [OrderDtl]
                INNER JOIN [OrderHea] ON [OrderHea].[Order_Number] = [OrderDtl].[Order_Number]
        WHERE [OrderHea].[Customer_Number] = @CustomerNumber OR @CustomerNumber IS NULL

        UPDATE [OrderHea]
        SET [Order_Total] = (
                        SELECT SUM([Extended_Price])
                        FROM [OrderDtl]
                        WHERE [OrderDtl].[Order_Number] = [OrderHea].[Order_Number]
                        )
        WHERE [OrderHea].[Customer_Number] = @CustomerNumber OR @CustomerNumber IS NULL

        SELECT '@CustomerNumber' AS [Name], CONVERT(varchar(250), @CustomerNumber) AS [Value]
        UNION ALL
        SELECT '@PriceFactor' AS [Name], CONVERT(varchar(250), @PriceFactor) AS [Value]

END
```

Call the stored procedure

```
Procedure IncreasePrice
    String[][] asResult
    String sSql
    Handle hoServer
    Get Main_DD to hoServer
    Send Refind_Records of hoServer
    If (Customer.Customer_Number > 0) Begin
        Move ("EXEC [usp_IncreasePrice] " ;
            + "@CustomerNumber = " + String(Customer.Customer_Number) + " " ;
            + ",@PriceFactor = 1.1" ;
            ) to sSql
        Get ExecuteSql of oSqlExec sSql to asResult
        Send FindByRowId of hoServer (Main_File(hoServer)) (CurrentRowId(hoServer))
    End
End_Procedure
```

View to search through database dimensions

This view will select the data from each of the mentioned tables. It can be used to create a 'superfind' search of your database. Create a selection list on this source and add a filter. You could use the 'Source' make it jump to the right view to view the selected row

```
CREATE VIEW [vwDimensions] AS
SELECT
        'Customer' AS [Source]
        ,CONVERT(varchar(20), [CUSTOMER].[Customer_Number]) AS [Id]
        ,CONVERT(varchar(100), [CUSTOMER].[Name]) AS [Description]
FROM [CUSTOMER]
UNION ALL SELECT 'Vendor', CONVERT(varchar(20),[VENDOR].[ID]) ,[VENDOR].[Name] FROM [VENDOR]
UNION ALL SELECT 'Salesperson', [SALESP].[ID], [SALESP].[Name] FROM [SALESP]
UNION ALL SELECT 'Inventory', [INVT].[Item_ID], [INVT].[Description] FROM [INVT]
```

How to get a resultset at once

```
String[][] sJobTitles

// Execute SQL and get the result set
Send SQLExecDirect of hStatement ;
    "Select Distinct [HumanResources].[Employee].[JobTitle] From [HumanResources].[Employee]"
Get SQLFetchResultsetValues of hStatement to sJobTitles

// Process the result set
Move (SizeOfArray (sJobTitles) - 1) to iElements
For iElement From 0 To iElements
    Move iElement to aTheRows[iElement].sRowID
    Move sJobTitles[iElement][0] to aTheRows[iElement].aCells[0].sValue
Loop
```