End-To-End Virtual Procedures Test Script

Objective

This test script exercises the Designer by adding virtual procedures to a model and then verifies the server executes the procedures correctly. **If any part of modeling or querying does not succeed, it should be marked as having failed.**

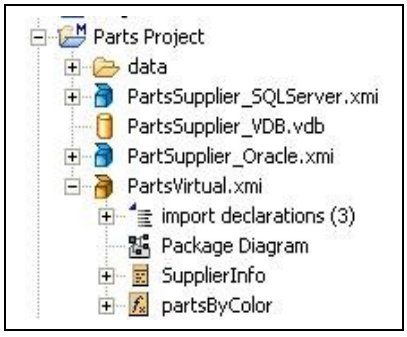
Specific patterns, identified by development and customer use cases, drove the design of these procedures.

Set Up

This script uses the Parts Project that is included with the Designer installation in the examples folder.

1. Start the Designer
2. Help > Project Examples...
3. Navigate to, and expand, the Teiid Designer node in the Projects tree
4. Select Parts Model Project Set Example
5. Click Finish
6. Expand **Parts Project**
7. Expand **PartsVirtual.xmi**

   The Model Explorer tree will contain the project and will resemble this:



Test

Using the PartsVirtual model as a starting point, a series of virtual procedures will be created. When creating the procedures, copy from this document and paste into the transformation editor.

Creating the Virtual Procedures

For **each of the following test cases**, follow these steps to create the virtual procedure.

1. Right-click PartsVirtual.xmi > New Child > Procedure
2. Name the procedure (use the name of the test case, for example MMSP01)
3. Expand the virtual procedure
4. Right-click the procedure (MMSP01, in this case) > Edit
5. Select the procedure's source from this document
6. Paste into the transformation editor
7. Click the Save/Validate SQL button

Virtual Procedure Test Cases

MMSP03

```
CREATE VIRTUAL PROCEDURE
BEGIN
 DECLARE short x;
 LOOP ON (SELECT quantity FROM PartsVirtual.SupplierInfo ORDER BY
quantity) AS quantityCursor
 BEGIN
  x = convert((quantityCursor.quantity - 1), short);
  if(x = 100)
  BEGIN
   BREAK;
  END
  ELSE
  BEGIN
   CONTINUE;
  END
 END
 SELECT quantity, x FROM PartsVirtual.SupplierInfo WHERE quantity>=x;
END
```

MMSP04

```
CREATE VIRTUAL PROCEDURE
BEGIN
 DECLARE short x;
 x= convert(0, short);
```

```
 WHILE (x < 200)
 BEGIN
  x = convert((x + 1), short);
  if(x = 100)
  BEGIN
   BREAK;
  END
  ELSE
  BEGIN
   CONTINUE;
  END
 END
 SELECT supplier_id FROM PartsVirtual.SupplierInfo WHERE quantity=x;
END
```

MMSP06

```
CREATE VIRTUAL PROCEDURE
BEGIN
 SELECT supplier_id, supplier_name, quantity INTO #tempSupplierInfo FROM
PartsVirtual.SupplierInfo WHERE quantity > 100;
 SELECT supplier_id, supplier_name, quantity INTO #tempSupplierInfo FROM
PartsVirtual.SupplierInfo WHERE quantity > 200;
 INSERT INTO #tempSupplierInfo (supplier_id, supplier_name, quantity)
VALUES('S116', 'T E S T E R', convert(100,short));
 SELECT supplier_name FROM #tempSupplierInfo WHERE quantity=100 OR
quantity=200 OR quantity=300;
END
```

MMSP07

```
CREATE VIRTUAL PROCEDURE
BEGIN
 DECLARE string x;
 SELECT supplier_id INTO #tempSupplierInfo FROM
PartsVirtual.SupplierInfo;
 LOOP ON (SELECT supplier_id FROM #tempSupplierInfo) AS
supplier_idCursor
 BEGIN
  x = supplier_idCursor.supplier_id;
 END
 SELECT supplier_id FROM PartsVirtual.SupplierInfo WHERE supplier_id=x;
END
```

MMSP08

- Right-click MMSP08 > New Child > Procedure Parameter

- Name the parameter **param1**
- Define the datatype of param1 as **short**
- Complete by adding the following as the transformation SQL:

```
CREATE VIRTUAL PROCEDURE
BEGIN
 DECLARE short x;
 LOOP ON (SELECT quantity FROM PartsVirtual.SupplierInfo WHERE
quantity < param1 ORDER BY quantity) AS quantityCursor
 BEGIN
   x = convert((quantityCursor.quantity - 1), short);
 END
 SELECT x, supplier_id FROM PartsVirtual.SupplierInfo WHERE quantity <=
x;
END
```

MMSP09

- Right-click MMSP09 > New Child > Procedure Parameter
- Name the parameter **param1**
- Define the datatype of param1 as **short**
- Complete by adding the following as the transformation SQL:

```
CREATE VIRTUAL PROCEDURE
BEGIN
 SELECT supplier_name, convert(quantity, integer) AS quantity INTO
#tempSupplierInfo FROM PartsVirtual.SupplierInfo WHERE quantity <=
param1;
 SELECT supplier_name FROM #tempSupplierInfo WHERE quantity=400;
END
```

MMSP10

- Right-click MMSP10 > New Child > Procedure Parameter
- Name the parameter **param1**
- Define the datatype of param1 as **short**
- Complete by adding the following as the transformation SQL:

```
CREATE VIRTUAL PROCEDURE
BEGIN
DECLARE string VARIABLES.X;
 SELECT DISTINCT
        PartsVirtual.SupplierInfo.SUPPLIER_ID,
PartsVirtual.SupplierInfo.SUPPLIER_NAME,
PartsVirtual.SupplierInfo.QUANTITY
 INTO #tempSupplierInfo
FROM
```

```
        PartsVirtual.SupplierInfo
WHERE
        PartsVirtual.SupplierInfo.QUANTITY <=
PARTSVIRTUAL.MMSP10.param1;
VARIABLES.X = SELECT supplier_name FROM #tempSupplierInfo WHERE
(quantity = 100) AND (LEFT(supplier_name, 1) = 'P');
SELECT
        VARIABLES.X, (PARTSVIRTUAL.MMSP10.param1 - 1) AS
pMinusOne, 23 AS twentyThree;
END
```

MMSP11

This procedure uses two input parameters. (The use of named procedures was introduced in 4.3 SP3. The first two tests of MMSP11 will work with other versions.)

- Right-click MMSP11 > New Child > Procedure Parameter
- Name the parameter **QtyIn**
- Define the datatype of QtyIn as **short**
- Right-click QtyIn > New Sibling > Procedure Parameter
- Name the parameter **SortMode**
- Define the datatype of SortMode as **string**
- Complete by adding the following as the transformation SQL:

```
CREATE VIRTUAL PROCEDURE
BEGIN
        DECLARE string VARIABLES.SORTDIRECTION;
        VARIABLES.SORTDIRECTION =
PARTSVIRTUAL.MMSP11.SORTMODE;
        IF(ucase(VARIABLES.SORTDIRECTION) = 'ASC')
        BEGIN
                SELECT * FROM PartsVirtual.SupplierInfo
WHERE QUANTITY > PARTSVIRTUAL.MMSP11.QTYIN ORDER BY
PartsVirtual.SupplierInfo.PART_ID ;
        END
        ELSE
        BEGIN
                SELECT * FROM PartsVirtual.SupplierInfo
WHERE QUANTITY > PARTSVIRTUAL.MMSP11.QTYIN ORDER BY
PartsVirtual.SupplierInfo.PART_ID DESC ;
        END
END
```

At this point you should have a total of 9 virtual procedures in the PartsVirtual model (partsByColor plus the 8 you created).

Next Steps:

- Save All
- Create connection profiles and data sources for PartsSourceA and PartsSourceB
- Save All
- Open PartsSupplier_VDB.vdb
- Ensure the correct translator and JNDI names are used for the source models
- Synchronize as necessary

Executing the Virtual Procedures

The virtual procedures created may be executed in either the Database Development perspective or deployed to a server and executed using a JDBC tool.

Executing the VDB from the Designer:

- Right-click the VDB > Modeling > Execute VDB
- Follow the connection dialogs to connect to the preview version of the VDB
- Right-click PartsVdb > Open SQL Scrapbook
- Opening the SQL Scrapbook, you can paste the SQL from the table below into the editor. Select the line you want to execute, right-click > Execute Current Text (or press ALT-S).

Deploying the VDB to the Server:

- Right-click the VDB > Modeling > Deploy VDB
- Use the JDBC tool of your choice to execute the queries in the table below.

<<EXPECTED RESULTS>> If your results do not exactly match the expected results for each test case, that test case has failed. DOCUMENT any results that are different and share them with a test team member upon completion of the test.

SQL	Expected Results
SELECT * FROM PartsSourceA.SUPPLIER WHERE SUPPLIER_ID IN (EXEC PartsVirtual.MMSP04())	15 rows with supplier_id S100 – S115 (except S114)
Exec PartsVirtual.MMSP03( )	42 rows of quantities of 500; x = 499
Exec PartsVirtual.MMSP04( )	71 rows
Exec PartsVirtual.MMSP06( )	115 rows; last is ‘T E S T E R’
Exec PartsVirtual.MMSP07( )	16 rows of ‘S115’
Exec PartsVirtual.MMSP08(300)	71 rows
Exec PartsVirtual.MMSP09(400)	42 rows
Exec PartsVirtual.MMSP10(200)	1 row of ‘Park’, 199, 23
EXEC PartsVirtual.MMSP11(401, 'desc')	42 rows starting with part P313 and ending with part P306
EXEC PartsVirtual.MMSP11(401, 'asc')	42 rows starting with part P306 and ending with part P313
EXEC PartsVirtual.MMSP11(SortMode =	42 rows starting with part P313 and ending

'desc', QtyIn = 401)	with part P306

Post-Test Cleanup

No manipulation of data was involved with this test, so the database will already be in the same state it was at the beginning of the test.

Please see a member of the test team for any questions regarding the accuracy of the script or the expected results.