Test Objective

The Update Virtual Groups test script exercises the ability to create reasonably complex procedures to handle updates to multiple tables. One thing it does not do is update across sources. This test updates across sources in a single transaction. It also tests to ensure we roll back transactions which fail on one side.

Setting Up the Test

- Install the Teiid Designer
- Install, or have available to you, a JBoss application server with Teiid/EDS installed.
  - The test assumes user=user and admin=admin accounts are set up. Edit <profile>/conf/props/teiid-security-users.properties and soa-user.properties to enable the use of these accounts.
  - JDBC drivers, which support XA transactions, for the data sources (for this script, oracle14.jar and db2jcc4.jar)
- You need access to PartsSupplier on two RDBMSes – This script uses Oracle and DB2
- For testing with Groovy, a version of Groovy must be installed and the teiid-client.jar must be along Groovy's classpath

Two Ways to Test

- You may perform all the modeling tasks, manually deploy the data sources and VDB, all described below, or
- You may execute the provided Groovy script that creates its own data sources, and uses the supplied VDB
- Guess which one takes less time.

The Groovy Test

1. Unzip the TwoSourceUpdateTest_RequiredFiles.zip archive to a local directory, for example, /user/tester
2. Copy the teiid-client.jar file to the Groovy lib directory (or somewhere along its classpath)
3. The following example assumes the server is running locally, so the -s option is specified as “localhost”
4. cd to the groovy installation's /bin directory
5. Execute: `./groovy /user/tester/TwoSourceUpdate/TwoSourceTransTest.groovy -v CheckXa2.vdb -s localhost`
6. Execution will take a minute or so, depending on the server's capacity and load. After some amount of output, the test will conclude with:

```
PASSED: TestOne
PASSED: TestTwo
PASSED: TestThree

================================================
    Command line test
    Tests run: 3, Failures: 0, Skips: 0
================================================


================================================
Command line suite
Total tests run: 3, Failures: 0, Skips: 0
================================================
```

The key is there were no failures or skipped tests.

That's it.

The Manual Test

In the Admin Console

Using Jopr (RHQ?) or JON, set up two XA data sources. Here is the information you need to set up Oracle and SQL Server sources

Oracle

1. Expand the Datasources nnode in the tree
2. Select the XA Datasources node
3. Click **Add a new resource**
4. Select the Oracle XA template
5. Click Continue
6. Enter a JNDI name (XA_PartsOra, for example)
7. Enter the username: partssupplier
8. Enter the password: mm
9. Under XA Datasource Properties, click the **Add New** button
10. In the Name field, enter **URL**
11. In the Value field, enter
    jdbc:oracle:thin:@englxdbs11.mw.lab.eng.bos.redhat.com:1521:orcl
12. Click OK
13. Save the datasource definition.

DB2

1. Follow the same instructions as for the Oracle data source with the following changes:
2. Use the default template

3. Enter a JNDI name (XA_PartsSQL, for example)
4. Enter the username: partssupplier
5. Enter the password: mm
6. The driver class is: **com.ibm.db2.jcc.DB2XADataSource**
7. Under XA Datasource Properties, click the **Add New** button
8. In the Name field, enter **ServerName**
9. In the Value field, enter **slntds05.mw.lab.eng.bos.redhat.com**
10. Click **OK**
11. In the Name field, enter **PortNumber**
12. In the Value field, enter **50000**
13. Click **OK**
14. In the Name field, enter **DatabaseName**
15. In the Value field, enter **ds05**
16. Click **OK**
17. In the Name field, enter **DriverType**
18. In the Value field, enter **4**
19. Click **OK**
20. In the Name field, enter **User**
21. In the Value field, enter **partssupplier**
22. Click **OK**
23. In the Name field, enter **Password**
24. In the Value field, enter **mm**
25. Click **OK**
26. Save the datasource definition.

In the Designer

Modeling

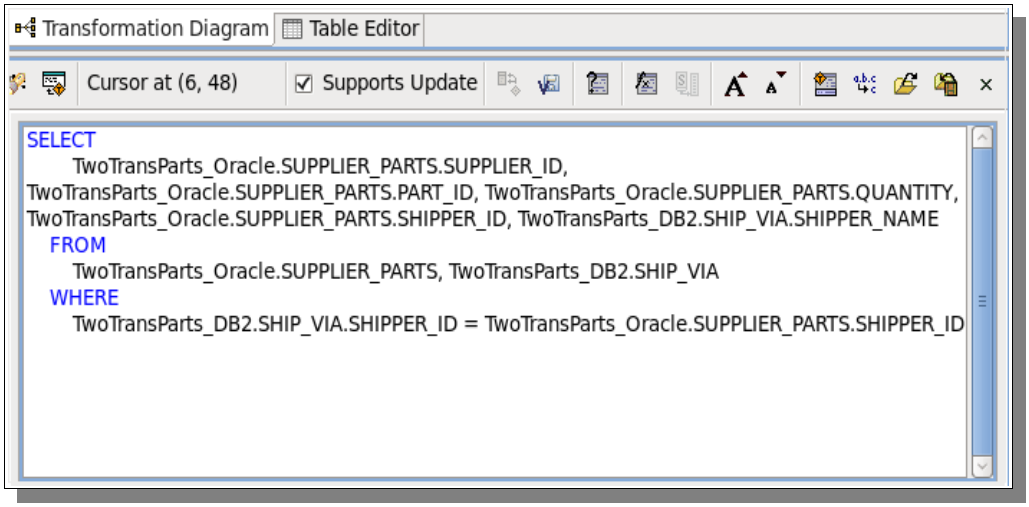
1 Create a Model Project
2 Import PartsSupplier from two RDBMSes
3 So you can easily use the SQL provided in this test script, name the imported PartsSupplier models:
3.1 PartsOne
3.2 PartsTwo
4 Create a Relational View model named Updates
5 Add to Updates a base table named UpdateParts
6 Use this SQL to define the transformation:

```
SELECT
            TwoTransParts_Oracle.SUPPLIER_PARTS.SUPPLIER_ID,
TwoTransParts_Oracle.SUPPLIER_PARTS.PART_ID,
TwoTransParts_Oracle.SUPPLIER_PARTS.QUANTITY,
TwoTransParts_Oracle.SUPPLIER_PARTS.SHIPPER_ID,
TwoTransParts_DB2.SHIP_VIA.SHIPPER_NAME
```

```
      FROM
            TwoTransParts_Oracle.SUPPLIER_PARTS, TwoTransParts_DB2.SHIP_VIA
      WHERE
            TwoTransParts_DB2.SHIP_VIA.SHIPPER_ID =
TwoTransParts_Oracle.SUPPLIER_PARTS.SHIPPER_ID
```

7 Check the Supports Update option
8 Click Yes to set all child columns to Updateable

(Note the transformation editor now has the SELECT, UPDATE,INSERT, and DELETE tabs.)



9 Go to the INSERT tab
10 Uncheck the Use default option
11 Enter the insert procedure SQL

```
CREATE PROCEDURE
BEGIN
      INSERT INTO TwoTransParts_Oracle.SUPPLIER_PARTS
(TwoTransParts_Oracle.SUPPLIER_PARTS.SUPPLIER_ID,
TwoTransParts_Oracle.SUPPLIER_PARTS.PART_ID,
TwoTransParts_Oracle.SUPPLIER_PARTS.QUANTITY,
TwoTransParts_Oracle.SUPPLIER_PARTS.SHIPPER_ID) VALUES ("INPUT".SUPPLIER_ID,
"INPUT".PART_ID, "INPUT".QUANTITY, "INPUT".SHIPPER_ID);
      VARIABLES.ROWS_UPDATED = INSERT INTO TwoTransParts_DB2.SHIP_VIA
(TwoTransParts_DB2.SHIP_VIA.SHIPPER_ID, TwoTransParts_DB2.SHIP_VIA.SHIPPER_NAME)
VALUES ("INPUT".SHIPPER_ID, "INPUT".SHIPPER_NAME);
ENDT".SHIPPER_ID, "INPUT".SHIPPER_NAME);
END
```

12 Switch to the DELETE tab
13 Uncheck the Use default option
14 Enter the delete procedure SQL

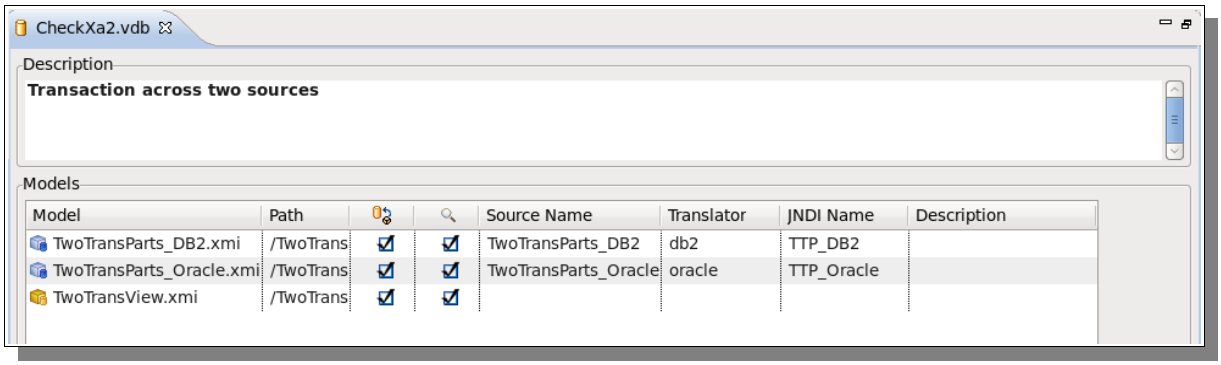
```
CREATE PROCEDURE
BEGIN
```

```
      DECLARE bigdecimal VARIABLES.ShipId;
      IF(HAS CRITERIA ON (TwoTransView.ttv.SHIPPER_ID,
TwoTransView.ttv.SUPPLIER_ID, TwoTransView.ttv.PART_ID))
      BEGIN
            VARIABLES.ShipId = SELECT SHIPPER_ID FROM
TwoTransParts_Oracle.SUPPLIER_PARTS WHERE TRANSLATE CRITERIA ON
(TwoTransView.ttv.SHIPPER_ID, TwoTransView.ttv.SUPPLIER_ID,
TwoTransView.ttv.PART_ID);
            VARIABLES.ROWS_UPDATED = DELETE FROM TwoTransParts_DB2.SHIP_VIA WHERE
SHIPPER_ID = VARIABLES.ShipId;
            DELETE FROM TwoTransParts_Oracle.SUPPLIER_PARTS WHERE TRANSLATE
CRITERIA ON (TwoTransView.ttv.SUPPLIER_ID, TwoTransView.ttv.PART_ID,
TwoTransView.ttv.SHIPPER_ID);
      END
END
```

15 On the UPDATE tab, uncheck the Update Enabled option.
16 Save the transformation and the model

The VDB

17 Create a new VDB
- 17.1 Add the Updates model to the VDB
- 17.2 To the PartsOne model, add:
  - 17.2.1 the translator: oracle
  - 17.2.2 JNDI Name is XA_PartsOra
- 17.3 To the PartsTwo model, add:
  - 17.3.1 the translator: db2
  - 17.3.2 JNDI Name is: XA_PartsDB2



Note: Your JNDI names will probably match the source names.

18 Save the VDB
19 Deploy the VDB to a server

Test Queries

- Using your favorite JDBC query tool, execute these queries

- Login to the VDB with this option: **autoCommitTxn=ON**

Initial Baseline

Make note of the counts.

1. SELECT COUNT(*) FROM TwoTransParts_Oracle.SUPPLIER_PARTS WHERE part_id = 'P302' AND supplier_id = 'S102'
2. SELECT COUNT(*) FROM TwoTransParts_db2.ship_via WHERE SHIPPER_ID = 94
3. SELECT COUNT(*) FROM TwoTransView.ttv WHERE SHIPPER_ID= 94

Insert a Row

We do this to demonstrate the VDB and sources are correctly wired.

4. INSERT INTO TwoTransView.ttv (SUPPLIER_ID, PART_ID, QUANTITY, SHIPPER_ID, SHIPPER_NAME) VALUES ('S102', 'P302', 135, 94, 'Shadowman Shipping')

Verify the counts have increased by one

5. SELECT COUNT(*) FROM TwoTransParts_Oracle.SUPPLIER_PARTS WHERE part_id = 'P302' AND supplier_id = 'S102'
6. SELECT COUNT(*) FROM TwoTransParts_db2.ship_via WHERE SHIPPER_ID = 94
7. SELECT COUNT(*) FROM TwoTransView.ttv WHERE SHIPPER_ID= 94

Delete the row we just inserted to demonstrate the delete function is also functioning correctly.

8. DELETE FROM TwoTransView.ttv WHERE (SUPPLIER_ID = 'S102') AND (PART_ID = 'P302') AND (SHIPPER_ID = 94)

The counts should revert to their original values.

9. SELECT COUNT(*) FROM TwoTransParts_Oracle.SUPPLIER_PARTS WHERE part_id = 'P302' AND supplier_id = 'S102'
10. SELECT COUNT(*) FROM TwoTransParts_db2.ship_via WHERE SHIPPER_ID = 94
11. SELECT COUNT(*) FROM TwoTransView.ttv WHERE SHIPPER_ID= 94

Rollback Test

First, insert a row into the Oracle to cause a rollback on our test insert .

12. INSERT INTO TwoTransParts_Oracle.SUPPLIER_PARTS (SUPPLIER_ID, PART_ID, QUANTITY, SHIPPER_ID) VALUES ('S102', 'P302', 135, 94)

Now verify our insert fails .

13. INSERT INTO TwoTransView.ttv (SUPPLIER_ID, PART_ID, QUANTITY, SHIPPER_ID, SHIPPER_NAME) VALUES ('S102', 'P302', 135, 94, 'Shadowman Shipping')

14. SELECT COUNT(*) FROM TwoTransParts_db2.ship_via WHERE SHIPPER_ID = 94
15. SELECT COUNT(*) FROM TwoTransView.ttv WHERE SHIPPER_ID= 94

Delete the row we added in preparation for testing the other source.

16. DELETE FROM TwoTransParts_Oracle.SUPPLIER_PARTS WHERE (SUPPLIER_ID = 'S102') AND (PART_ID = 'P302')

Insert a row in the DB2 source to cause a rollback on our test insert .

17. INSERT INTO TwoTransParts_DB2.SHIP_VIA (SHIPPER_ID, SHIPPER_NAME) VALUES (94, 'Bogus Shipping')

Now verify our insert fails .

18. INSERT INTO TwoTransView.ttv (SUPPLIER_ID, PART_ID, QUANTITY, SHIPPER_ID, SHIPPER_NAME) VALUES ('S102', 'P302', 135, 94, 'Shadowman Shipping')

Verify the row counts remained unaffected.

19. SELECT COUNT(*) FROM TwoTransParts_Oracle.SUPPLIER_PARTS WHERE part_id = 'P302' AND supplier_id = 'S102'
20. SELECT COUNT(*) FROM TwoTransView.ttv WHERE SHIPPER_ID= 94
21. Final cleanup: Delete the row we added earlier
22. DELETE FROM TwoTransParts_DB2.SHIP_VIA WHERE SHIPPER_ID = 94