Database – Inserting and Updating Rows

Overview

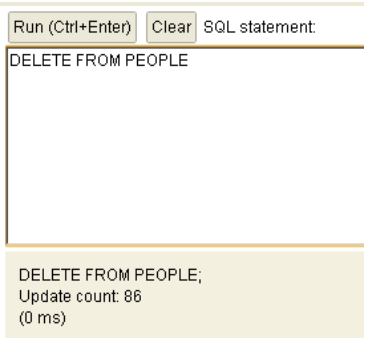
In this tutorial we'll cover using SQLXML to insert and update rows in a database table. This tutorial expands on concepts covered in “Database – Selecting Rows,” so users should be familiar with that material.

Steps

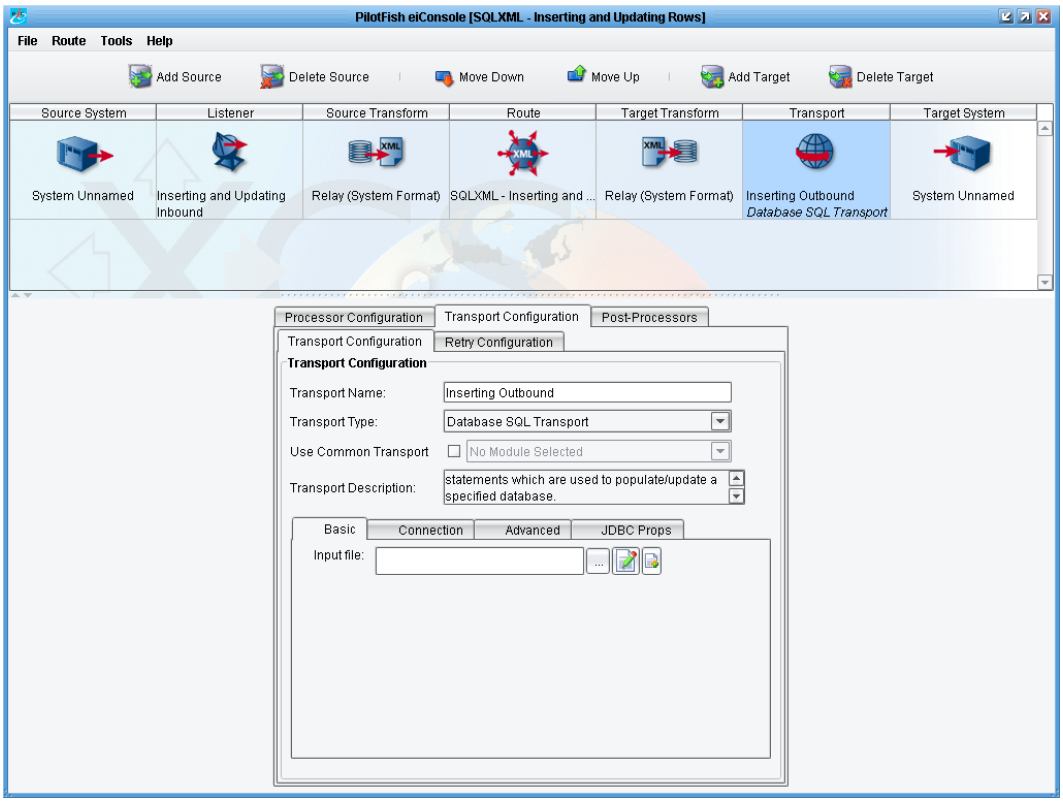
We'll start by clearing the rows from our existing “People” database table. Execute this query against it:

```
DELETE FROM PEOPLE
```

You should get a message indicating that rows have been affected:



We'll continue by creating a new Route called “Database – Inserting and Updating Rows,” adding and configuring a Directory Listener to check a folder for “**people.xml**” files, and adding a “Database SQL Transport”:



We've chosen to name this particular Transport “Inserting Outbound,” as we'll later be creating a second to handle updates.

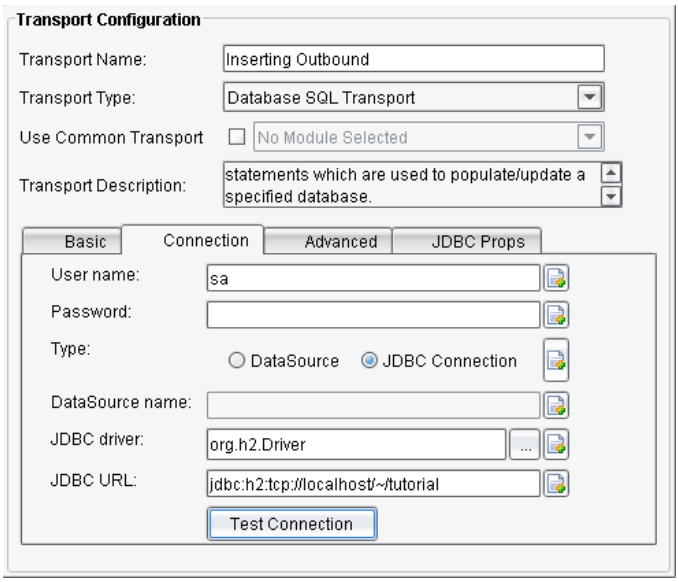
Configure the “Connection” tab using the credentials and information copied in the previous tutorial, then click the “Test Connection” button to verify it all works correctly:

Username: sa

Password:

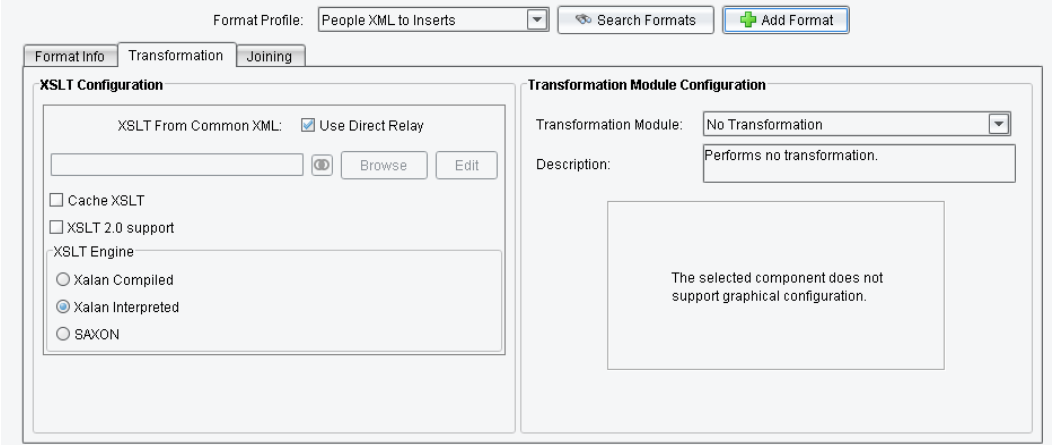
Driver: org.h2.Driver

URL: jdbc:h2:tcp://localhost/~/tutorial

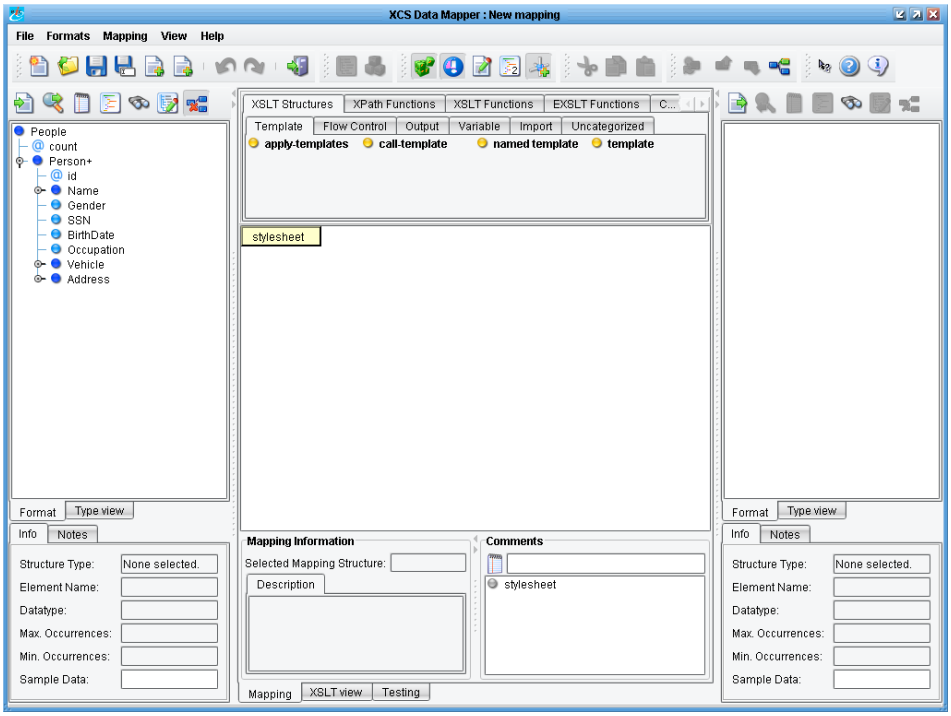


In the previous tutorial we configured a Listen using the “Input File” field to provide the SQLXML to be used. For this Transport, we'll do things a bit differently. Since we're expecting the “people.xml” sample to be picked up by our Listener, we'll want to make use of that data. We can therefore configure an XSLT document to transform the “**people.xml**” format to SQLXML, which in turn will be executed by the Transport.

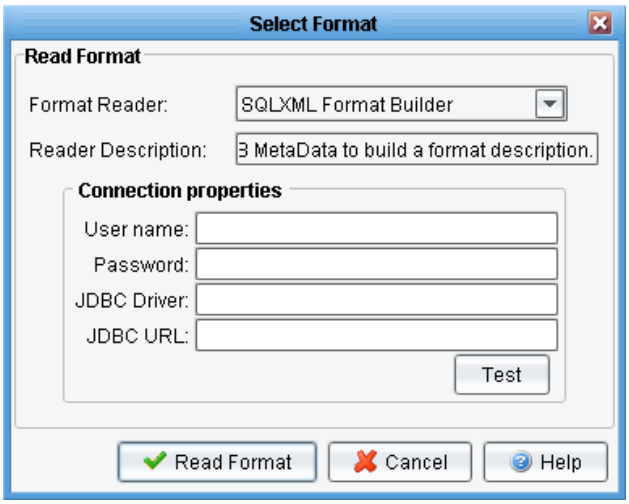
On the Target Transform stage, add a new Format called “People XML to Inserts”:



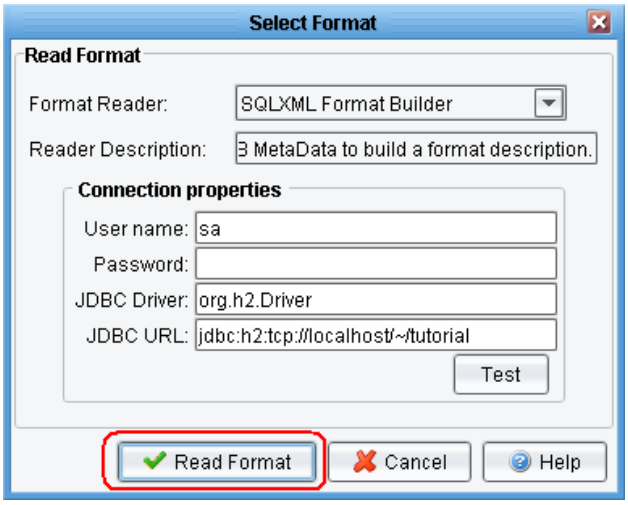
Uncheck the “Use Direct Relay” checkbox and click “Edit” to open the Data Mapper. Read “**people.xml**” in for the Source Format:



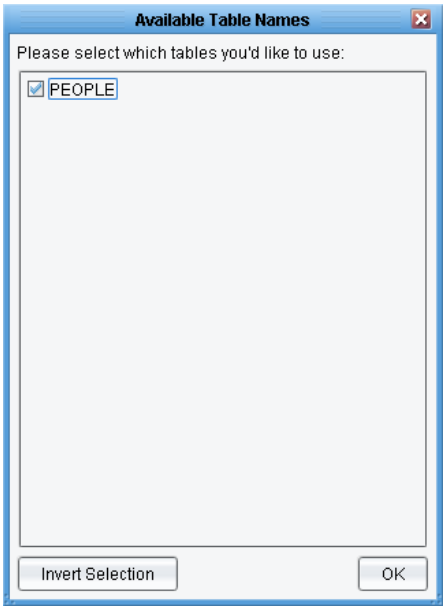
We'll now want to read in the Target format, for which we'll use a new type of Format Reader called the “SQLXML Format Reader”:



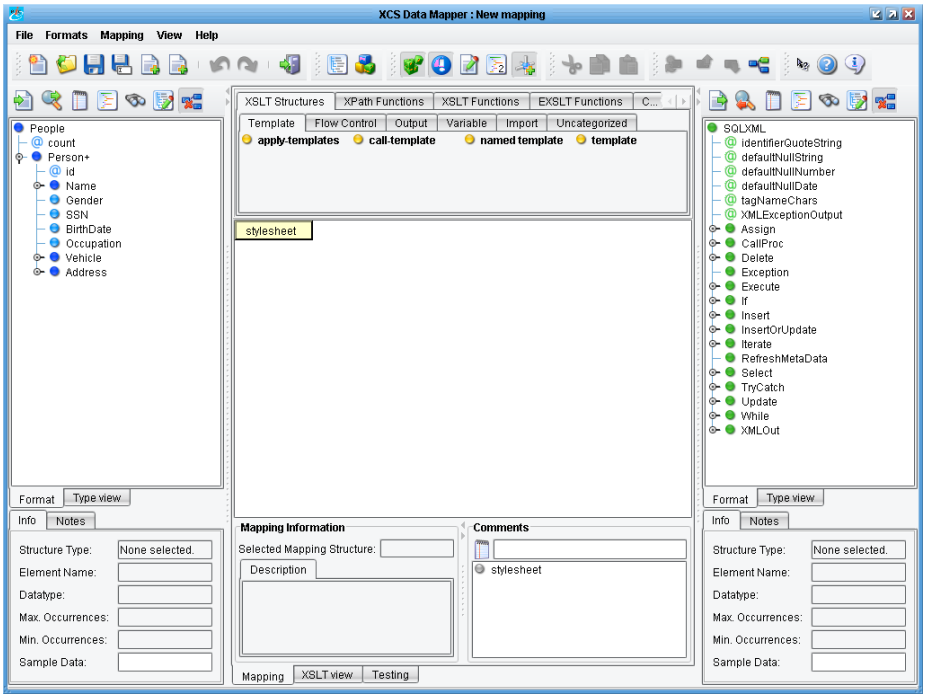
This dialog expects the same configuration values used by the Listeners and Transports we've covered. Provide the credentials and connectivity information, click “Test” to ensure they're correct and then click “Read Format”:



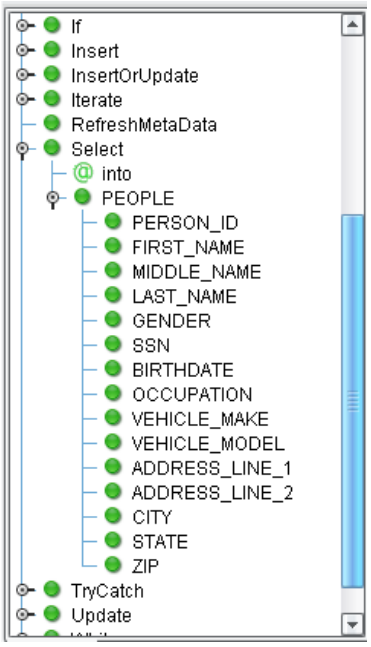
This will raise a dialog that allows you to select from a list of available database tables. Select “People” and hit “OK”:



This will populate the Target format with a fairly complex structure:

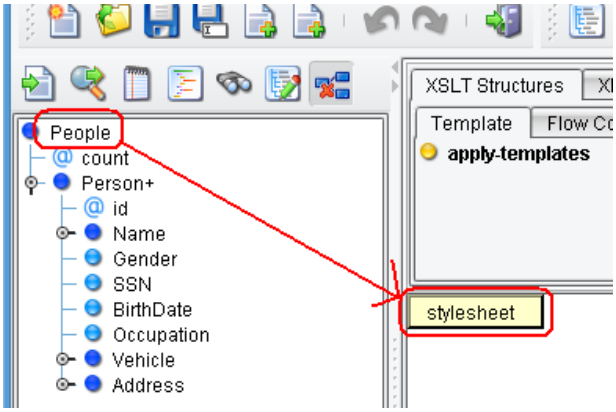


The Target format is now a representation of the entire set of SQLXML structures. Underneath these, you'll find your database-specific information, where applicable. For example, click on the “Select” we used for the last tutorial:

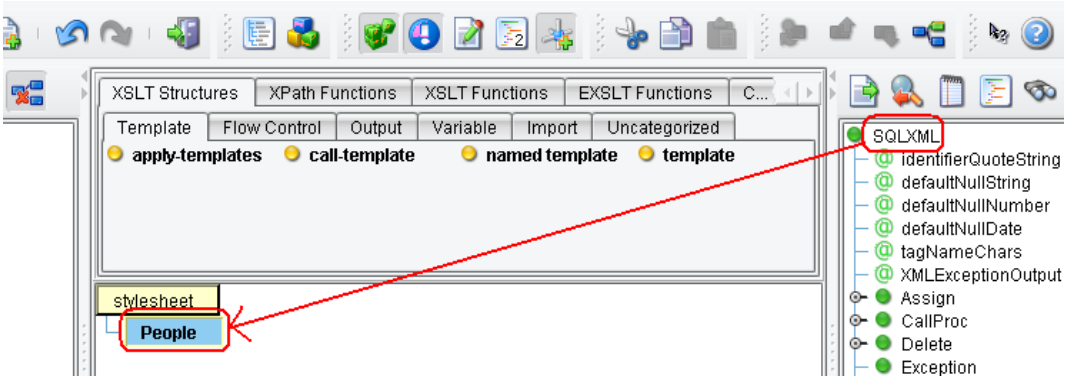


Working with this format is therefore a matter of dragging and dropping its items into the mapping panel while maintaining its provided structure.

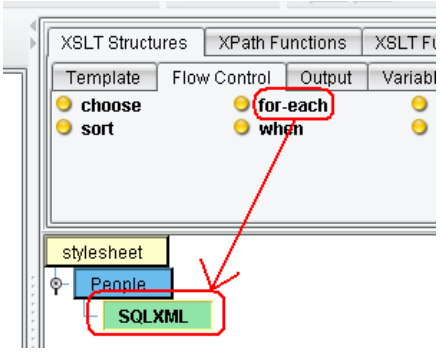
We'll want to perform an insert for each Person in our Source, so begin by mapping “People” onto the “stylesheet” element:



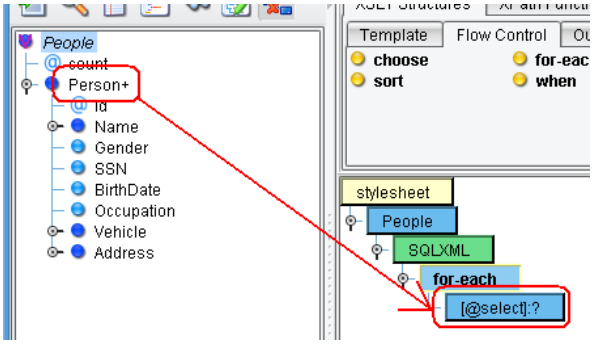
Drag the “SQLXML” root from the Target onto the added People element:



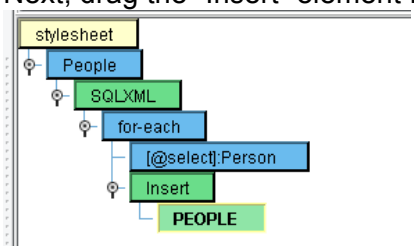
Drag XSLT Structures → Flow Control → for-each onto the added SQLXML element:



Drag the “Person” element from the Source onto the added “for-each”:

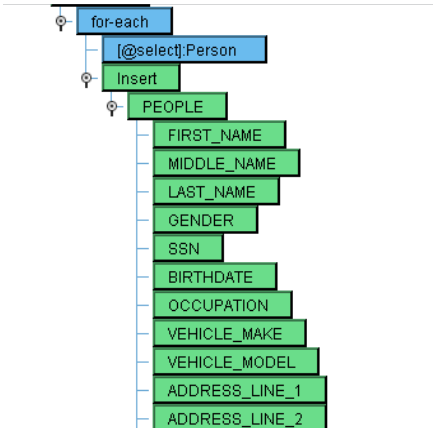


Next, drag the “Insert” element from the Target onto the “for-each” element, then the table

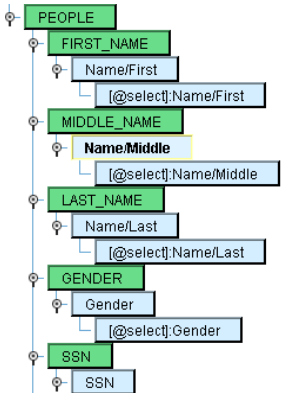


name, “People,” onto that:

Map each column name you're interested in populating (all except “PERSON_ID”) onto the table name:

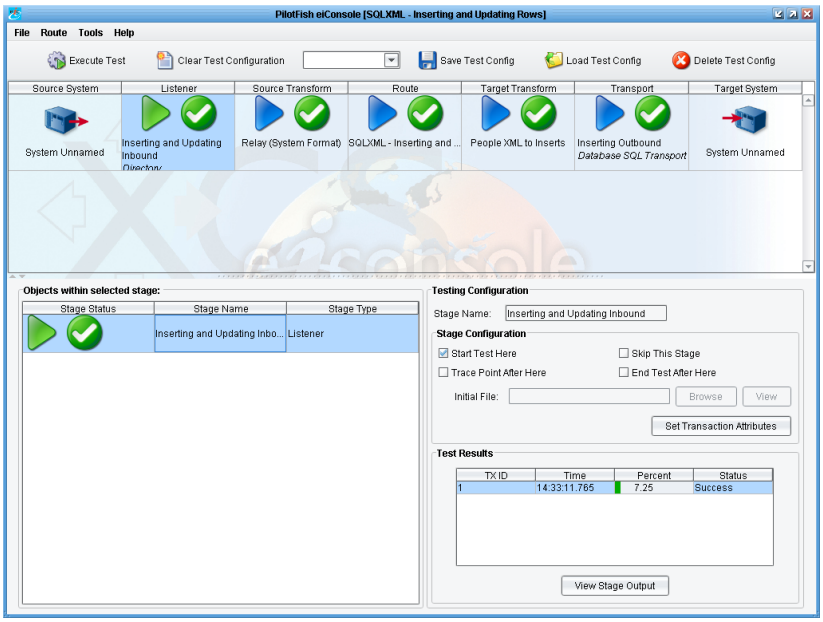


Now we just need to map each equivalent value from the Source onto these values:

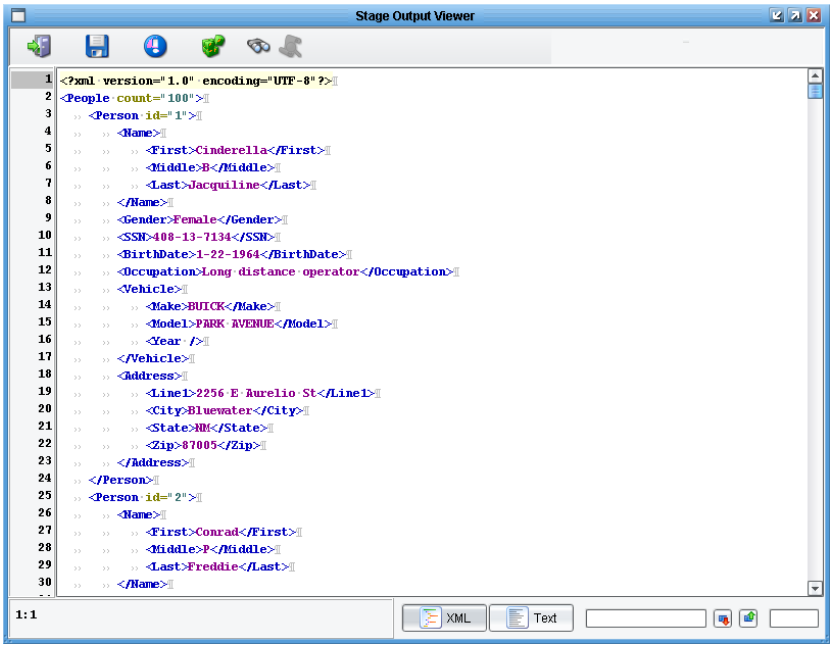


We've now mapped our "**people.xml**" to SQLXML, creating an "Insert" instruction for each "Person" in the source document.

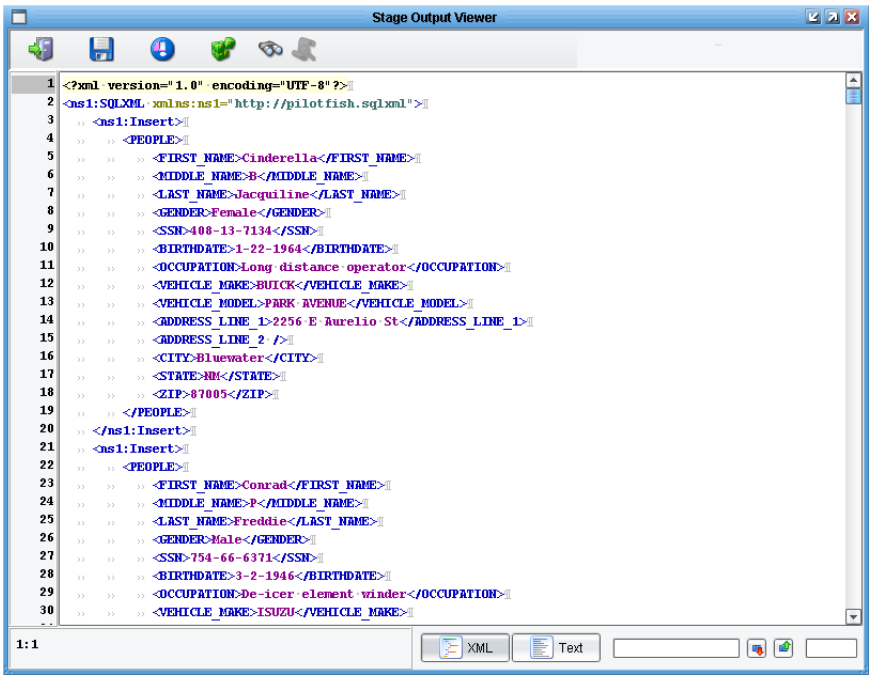
Return to the Console, switch to testing mode, and drop the "**people.xml**" file into source directory and watch it go all the way through:



View the stage output at the Listener stage:



Here we can see the original "**people.xml**" contents. In the Target Transform stage, view the output after the XSLT stage:



We can see the SQLXML generated. If you then look at the Transport stage, you'll see that it has executed successfully. There's no output to view, but we can go query our database with:

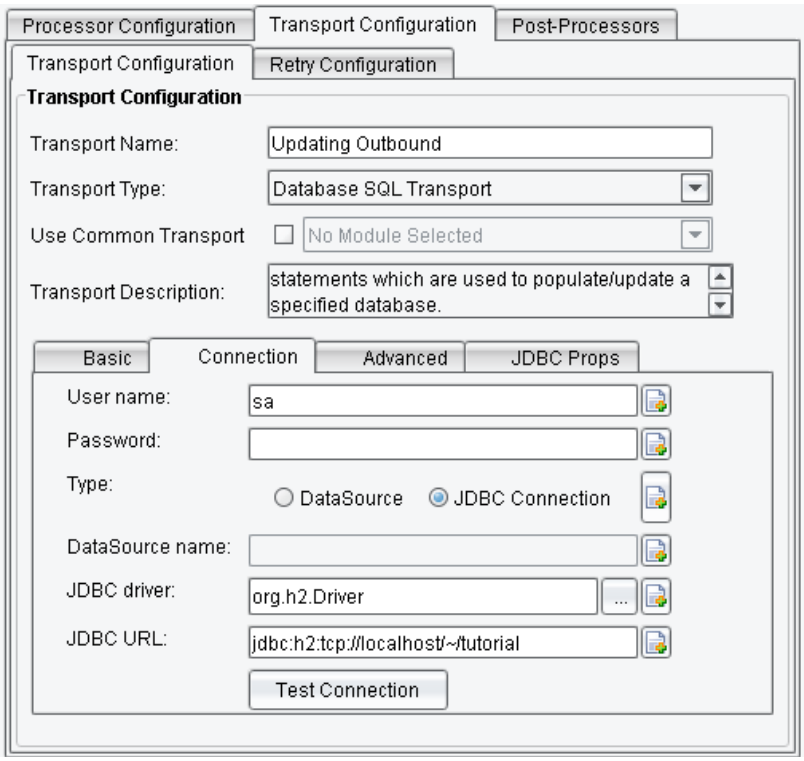
SELECT * FROM PEOPLE

SELECT * FROM PEOPLE;

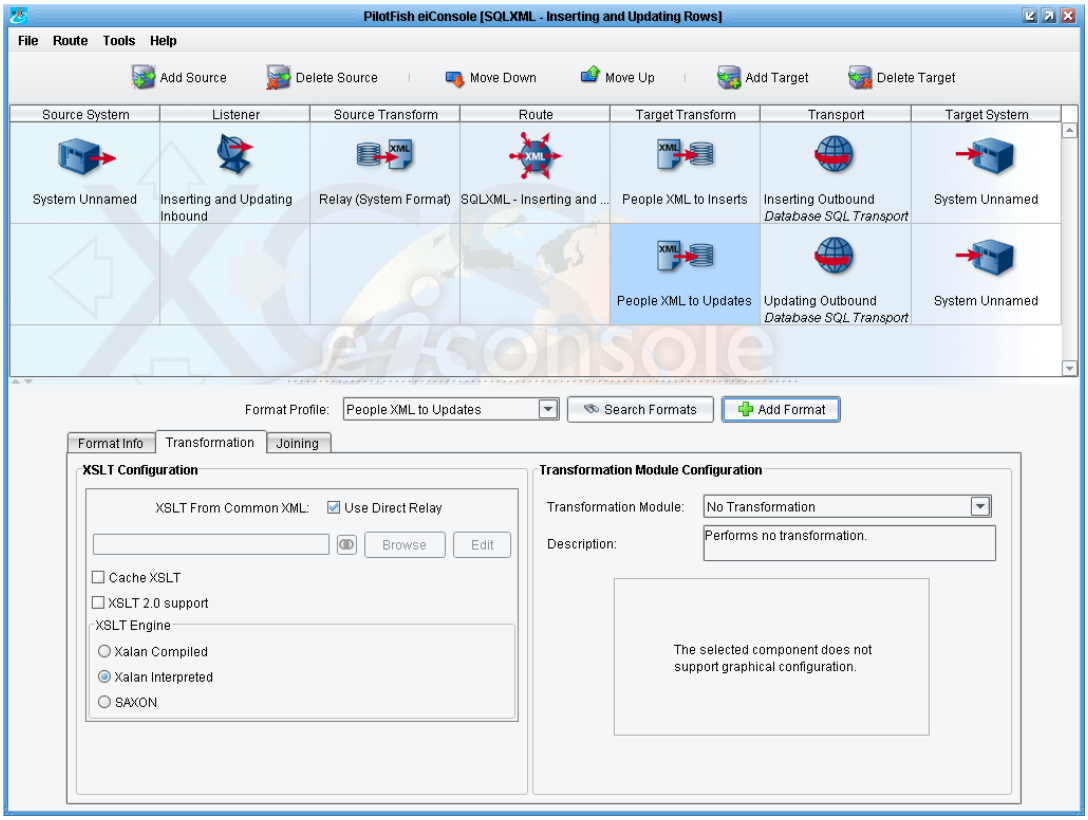
PERSON_ID	FIRST_NAME	MIDDLE_NAME	LAST_NAME	GENDER	SSN	BIRTHDATE	OCCUPATION	VEHICLE_MAKE	VEHICLE_MODEL	ADDRE
87	Cinderella	B	Jacquiline	Female	408-13-7134	1-22-1964	Long distance operator	BUICK	PARK AVENUE	2256 E
88	Conrad	P	Freddie	Male	754-66-6371	3-2-1946	De-icer element winder	ISUZU	TROOPER II	6788 E
89	Anita	R	Hedy	Female	536-22-0745	9-17-1934	Sales, automobile leasing	JEEP	COMMANDER	4922 E
90	Marcelino	W	Johnathon	Male	523-25-6908	1-25-1979	Supervisor, fabric coating	FORD	FIESTA	9606 W
91	Robby	E	Justin	Male	539-65-1181	11-26-1979	Design analyst	DODGE	LANCER	5485 S
92	Kristopher	U	Whitney	Male	573-32-4502	3-17-1977	Cooker cleaner	MINI	COOPER	1593 E
93	Patricia	U	Thi	Female	203-75-5842	5-5-1935	Sales agent	SATURN	RELAY	4673 W
94	Danyel	A	Katie	Female	230-78-6837	6-8-1982	Stenographer	MAZDA	MPV	3874 N
95	Eleanora	D	Luella	Female	395-47-6213	5-5-1966	Finishing trimmer	RENAULT	5	794 S L
96	Connie	C	Mose	Male	772-30-9001	3-15-1977	C.I.O. (chief information officer)	DAEWOO	LANOS	5912 E
97	Carl	Y	Ross	Male	261-57-4647	11-23-1968	Piece goods clerk	PLYMOUTH	COLT VISTA	4734 S
98	Delmer	K	Jarrod	Male	105-13-5500	4-12-1955	Sheriff	NISSAN	510	4992 W

We can see that our Insert statements worked successfully.

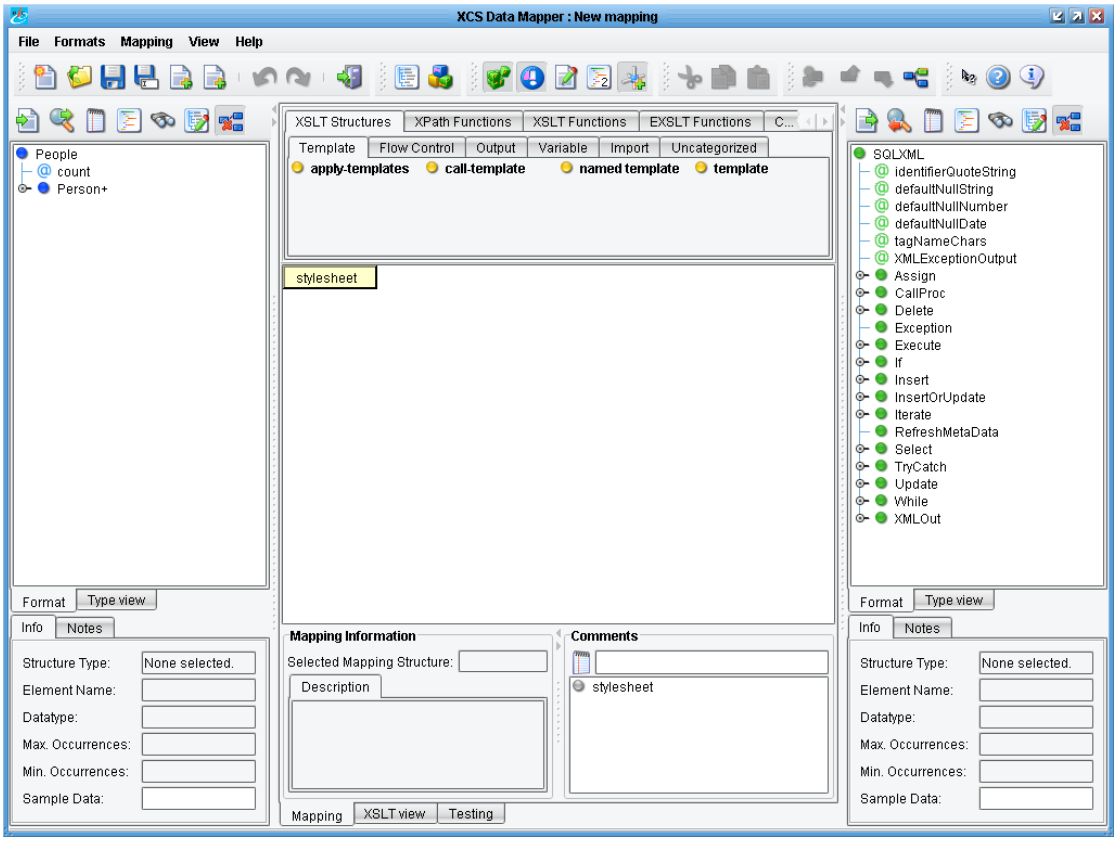
Let's now explore the “Update” instruction. Add and configure a new Target similar to the first, naming this one to indicate it's intended for updates:



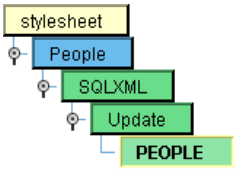
Add a new Format to this Target's Transform stage, this time called “People XML to Updates”:



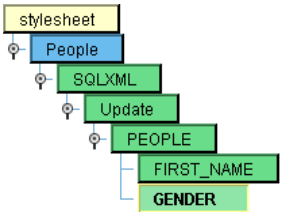
Open the Data Mapper once again and read the same Source and Target formats in:



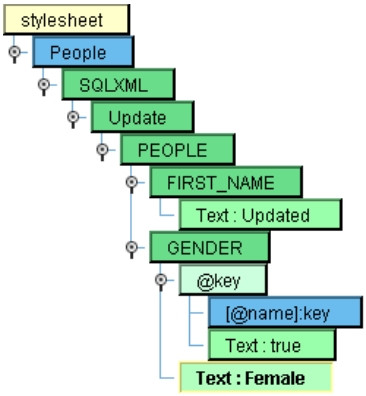
For this mapping, we'll just provide one Update (not under a “for-each):



We'll update each row with a “GENDER” value of “Female” to have a “FIRST_NAME” value of “Updated.” Add both columns to the mapping:

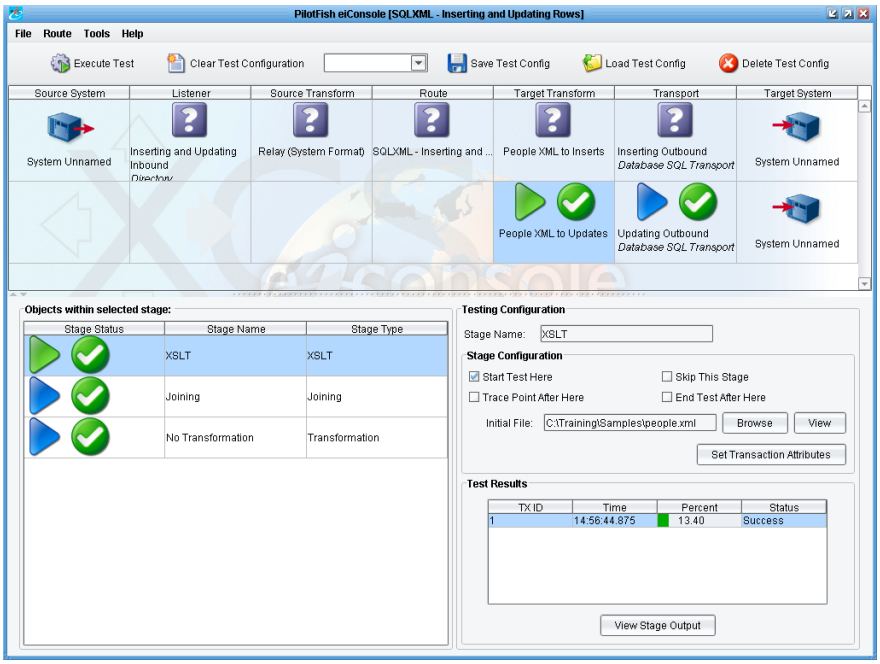


Add a “text” value to “FIRST_NAME” with the value “Updated,” add an “attribute” to “GENDER” with the name “key” and a “text” value of “true,” and then add a “text” value to “GENDER” of “female”:



This will now update all PEOPLE records where “GENDER” is “Female” with a “FIRST_NAME” of “Updated.”

Save the mapping, switch to the Console testing mode, and run the “**people.xml**” just through the new Target Transform onward:



Once again, we'll confirm our updates by querying the database directly. Try this query:

```
SELECT * FROM PEOPLE WHERE GENDER = 'Female'
```

You'll see that each such entry now has a “FIRST_NAME” of “Updated”:

SELECT * FROM PEOPLE WHERE GENDER = 'Female';

PERSON_ID	FIRST_NAME	MIDDLE_NAME	LAST_NAME	GENDER	SSN	BIRTHDATE	OCCUPATION
87	Updated	B	Jacquiline	Female	408-13-7134	1-22-1964	Long distance operator
89	Updated	R	Hedy	Female	536-22-0745	9-17-1934	Sales, automobile leasing
93	Updated	U	Thi	Female	203-75-5842	5-5-1935	Sales agent
94	Updated	A	Katie	Female	230-78-6837	6-8-1982	Stenographer
95	Updated	D	Luella	Female	395-47-6213	5-5-1966	Finishing trimmer
99	Updated	K	Glenna	Female	309-96-9438	2-5-1932	Home visitor
104	Updated	F	Chassidy	Female	423-98-6057	12-12-1984	Wood grinder, exc. head