Database mapping of XBRL instance documents based on the WIP taxonomy

Using Altova MapForce or RaptorXML+XBRL Server

Presented by: Alexander Falk
January 6, 2016
MapForce can be used to graphically design the XBRL → Database Mapping rules/logic.

Alternatively, if more control & programming is desired, XBRL instance documents can be analyzed and processed by RaptorXML+XBRL Server and data can be extracted into a database via Python scripts

If desired, FlowForce and MapForce Server can be used as a workflow & mapping engine to automate either of the above processes using date/time and/or event triggers (e.g. a WIP instance document arriving in a certain directory)
A sample target database

For the purpose of this demonstration, we created a very simple target DB that has just one table with columns that closely model the WIP spreadsheet:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>$29,831,262</td>
<td>$22,771,956</td>
<td>$7,059,306</td>
<td>$12,113,470</td>
<td>$9,248,924</td>
<td>$2,868,546</td>
<td>$11,987,630</td>
<td>$13,525,032</td>
<td>$3,159,000</td>
<td>$3,159,000</td>
<td>$3,159,000</td>
</tr>
<tr>
<td>201</td>
<td>$4,765,875</td>
<td>$3,915,859</td>
<td>$850,016</td>
<td>$4,761,592</td>
<td>$3,912,340</td>
<td>$849,252</td>
<td>$4,748,777</td>
<td>$3,519</td>
<td>$12,815</td>
<td>$319,663</td>
<td>$185,925</td>
</tr>
<tr>
<td>202</td>
<td>$3,165,949</td>
<td>$2,655,876</td>
<td>$500,273</td>
<td>$3,073,180</td>
<td>$2,553,445</td>
<td>$514,735</td>
<td>$3,092,382</td>
<td>$77,231</td>
<td>$19,152</td>
<td>$1,212,380</td>
<td>$1,019,868</td>
</tr>
<tr>
<td>203</td>
<td>$6,945,696</td>
<td>$5,348,200</td>
<td>$1,597,496</td>
<td>$5,955,890</td>
<td>$4,607,414</td>
<td>$1,298,476</td>
<td>$7,727,906</td>
<td>$710,786</td>
<td>$208,584</td>
<td>$2,085,189</td>
<td>$2,344,782</td>
</tr>
<tr>
<td>204</td>
<td>$3,202,917</td>
<td>$1,339,767</td>
<td>$1,962,150</td>
<td>$3,197,769</td>
<td>$1,236,328</td>
<td>$1,061,441</td>
<td>$3,199,404</td>
<td>$3,439</td>
<td>$100,000</td>
<td>$306,839</td>
<td>$241,974</td>
</tr>
<tr>
<td>206</td>
<td>$3,513,815</td>
<td>$2,260,925</td>
<td>$1,252,880</td>
<td>$2,839,759</td>
<td>$1,827,211</td>
<td>$1,012,584</td>
<td>$2,573,819</td>
<td>$433,714</td>
<td>$88,831</td>
<td>$205,940</td>
<td>$1,173,159</td>
</tr>
<tr>
<td>207</td>
<td>$3,915,079</td>
<td>$3,104,273</td>
<td>$800,506</td>
<td>$3,291,725</td>
<td>$2,849,640</td>
<td>$742,315</td>
<td>$3,509,274</td>
<td>$214,933</td>
<td>$86,381</td>
<td>$2,052,442</td>
<td>$2,039,028</td>
</tr>
</tbody>
</table>

Clearly, any real-world database will be more complex and have a relational database model involving multiple tables, but the same technology we will demonstrate here can be applied to mapping the data from XBRL to any number of relational tables.
Quick introduction to MapForce

- MapForce allows you to drop data sources, such as an XBRL-formatted WIP report, into a design surface.
- To develop a mapping from one data format to another, you then simply draw connecting lines – much like connecting circuits on a circuit board.
- The library pane on the left offers a palette of functions that allow you to transform the data or add conversions and calculations.
Adding an XBRL data source

When you use an XBRL formatted WIP report as the data source in MapForce, the WIP Taxonomy is automatically processed and you can show the structure of the XBRL WIP either as raw concepts or based on the XBRL financial table defined.
Locate and expand the surety:WorkInProcessTable

- The WIP taxonomy is based on and includes the full US-GAAP taxonomy, so there are many financial statement presentations included.
- Scroll down to the bottom of the XBRL control in MapForce and locate the 910100 – Disclosure – Work in Process presentation.
- Expand that node by clicking on the plus in front, then click on the surety:WorkInProcessTable XBRL table and hit the * key on the numeric keypad to expand all its children.
- This is the graphical representation of the source XBRL instance data that we will be mapping from. On the right side of each element/fact is a triangle where you can start connections from the source to the target.
Adding a database target

- Adding a database target involves first connecting to a database source using the connection wizard.
- Then you select which tables from that database you want to use in your mapping project – we’ll pick “contract”.

This creates a database object in our mapping project that looks like this.
Start making connections

- Now we can start making connections from the source to the target to create our mapping.
- We’re focusing on the ContractNumberAxis and will connect that to each row in the contract table in the DB.
- Start drawing lines from the triangle nodes on the left XBRL object to the triangle nodes on the right database table object...
Add datatype conversions where necessary

Sometimes you may find that the datatypes in the XBRL instance and in your target database are not compatible and you’ll get an error message in the mapping validation:

```
New Design1: Mapping validation failed - 1 error(s), 0 warning(s)
surety:ContractNumberAxis.domain ==> ContracNumberAxis_domain: Incompatible datatypes.
There are no valid values of the source type that are valid values of the target type.
Source: mf:node  Target: sqlite:integer
```

If that happens, add a manual conversion from the function library on the left – sometimes it is even more practical to convert to an intermediate datatype, like string.
Make more connections, then start looking at the output

- MapForce includes on-demand output preview as well as an interactive visual debugger that makes developing these mappings very easy.

- Once you’ve made a few more connections, it is time to start looking at the output.

- Since our target is a database, the output preview will be SQL commands that will be executed against the database server:

  ```sql
  INSERT INTO 'contract' ("ContractNumber", "ContractName", "EstRevenue", "EstCosts", "EstGrossProfit", "ForPeriodEstimatedRevenue", "ForPeriodCosts", "ForPeriodGrossProfit") VALUES (NULL, NULL, 29631262, 22771955, 7059306, NULL, NULL, NULL)
  INSERT INTO 'contract' ("ContractNumber", "ContractName", "EstRevenue", "EstCosts", "EstGrossProfit", "ForPeriodEstimatedRevenue", "ForPeriodCosts", "ForPeriodGrossProfit") VALUES (NULL, 4768879, 3615859, 950016, NULL, NULL, NULL, NULL)
  INSERT INTO 'contract' ("ContractNumber", "ContractName", "EstRevenue", "EstCosts", "EstGrossProfit", "ForPeriodEstimatedRevenue", "ForPeriodCosts", "ForPeriodGrossProfit") VALUES (NULL, 12107491, 13500000, -1312598, NULL, NULL, NULL, NULL)
  INSERT INTO 'contract' ("ContractNumber", "ContractName", "EstRevenue", "EstCosts", "EstGrossProfit", "ForPeriodEstimatedRevenue", "ForPeriodCosts", "ForPeriodGrossProfit") VALUES (208, 'C_208', NULL, NULL, 2193165, 3505674, -1312509)
  ```

- Clearly, that doesn't look quite right yet. We have three contracts in our WIP example XBRL instance, yet we’re trying to create six rows in the database with only some columns being filled with values and the rest being NULL...
Duration contexts vs. Instant contexts

- The reason for these six rows being created is that MapForce has found six contexts* in the XBRL instance document, and it has mapped each to a row in the database.

- So we need to understand the structure of our XBRL document a bit better: for each contract we have an instant context and a duration context in the XBRL file, because each contract is associated with data that is reported as an instant, such as % Complete; or reported as a duration such as Earned Contract Revenue, from Inception to Dec 31, 2014.

- Fortunately, we can easily resolve this in MapForce and map them both to the same row in the database using the ContractNumberAxis as the primary key.

- To do that, we right click on the contract table in our database object on the right and pick Add Duplicate Input After, which gives us two instances of the table that we can now map to separately from instant vs. duration contexts:

* All facts have an associated context which defines the reporting entity and time period, and it also defines dimensional information.
Next step: filter contexts and map them differently

Now that we have two representations of the table on the right, we can filter XBRL contexts depending on if they are instant or duration and map them differently – as long as we keep the primary key connected to both:
Database Table Actions

› Since we’re now mapping two separate contexts to the same table in the database, we also have to tell MapForce what database table actions to perform based on the primary key.

› We do that by clicking the small button to the right of each contract table and specify the database actions in the following dialog.

› We simply Insert another Action column before the “Insert All” column and specify that we want to Update the data if the primary key matches, otherwise we insert a new row into the DB.

› For all the input data items we then elect to insert the “mapped value” into the database.

› This will allow us to map some elements/facts from one context and then grab other facts from a different context as long as the primary key matches.
Calculations and if-else statements

- Last, but not least, you sometimes may need to do more complex calculations or map values differently depending on what inputs you have. We will look at one example of how to do that.
- In our XBRL taxonomy we have two separate facts that report billings over or under cost and earnings, but we want to map them to just one database field using either positive or negative values:
  - surety:BillingsInExcessOfCostAndEarnings
  - surety:CostsAndEarningsInExcessOfBillings
- We can do that by multiplying one of them by -1 and then use an if-else statement to map one or the other to the target column in the database, depending on if that fact exists in the source XBRL.
Putting it all together

Here is the complete mapping that we have now created with all components and logic included:
Looking at the final result

If we now switch to the Output tab we get a SQL Script that we can execute against the database and we see the following output, which produces precisely three rows in the target database as shown below.

```
UPDATE "contract" SET "ContractNum" = 200, "ContractName" = 'Really Big Contract', "ForPeriodEarnedRevenue" = 3740588, "ForPeriodCosts" = 2855269, "ForPeriodGrossProfit" = 885319 WHERE ("contract","ContraNumberAxis_domain")=1
---> OK. 0 row(s).
INSERT INTO "contract" ("ContraNumberAxis_domain", "ContractNum", "ContractName", "ForPeriodEarnedRevenue", "ForPeriodCosts", "ForPeriodGrossProfit") VALUES (1, 200, 'Really Big Contract', 3740588, 2855269, 885319)
---> OK. 1 row(s).
UPDATE "contract" SET "ContractNum" = 201, "ContractName" = 'Contract 201', "ForPeriodEarnedRevenue" = 319663, "ForPeriodCosts" = 185925, "ForPeriodGrossProfit" = 13738 WHERE ("contract","ContraNumberAxis_domain")=2
---> OK. 0 row(s).
INSERT INTO "contract" ("ContraNumberAxis_domain", "ContractNum", "ContractName", "ForPeriodEarnedRevenue", "ForPeriodCosts", "ForPeriodGrossProfit") VALUES (2, 201, 'Contract 201', 319663, 185925, 13738)
---> OK. 1 row(s).
UPDATE "contract" SET "ContractNum" = 208, "ContractName" = 'C_208', "ForPeriodEarnedRevenue" = 2193165, "ForPeriodCosts" = 3505674, "ForPeriodGrossProfit" = -1312509 WHERE ("contract","ContraNumberAxis_domain")=3
---> OK. 0 row(s).
INSERT INTO "contract" ("ContraNumberAxis_domain", "ContractNum", "ContractName", "ForPeriodEarnedRevenue", "ForPeriodCosts", "ForPeriodGrossProfit") VALUES (3, 208, 'C_208', 2193165, 3505674, -1312509)
---> OK. 1 row(s).
UPDATE "contract" SET "EstRevenue" = 29831262, "EstCosts" = 22771956, "EstGrossProfit" = 7059386, "FromInceptEarnedRevenue" = 12113470, "FromInceptIncurredCosts" = 9246924, "FromInceptContractBillings" = 11987630, "EstimatedCostToComplete" = 13525032, "PercentageComplete" = 40.60663037, "UnderOverBillings" = 125840 WHERE ("contract","ContraNumberAxis_domain")=1
---> OK. 1 row(s).
UPDATE "contract" SET "EstRevenue" = 4765875, "EstCosts" = 3915859, "EstGrossProfit" = 850016, "FromInceptEarnedRevenue" = 4761592, "FromInceptIncurredCosts" = 3912340, "FromInceptContractBillings" = 4748777, "EstimatedCostToComplete" = 3519, "PercentageComplete" = 0.99, "UnderOverBillings" = 12815 WHERE ("contract","ContraNumberAxis_domain")=2
---> OK. 1 row(s).
UPDATE "contract" SET "EstRevenue" = 12187491, "EstCosts" = 13500000, "EstGrossProfit" = -1312509, "FromInceptEarnedRevenue" = 2193165, "FromInceptIncurredCosts" = 3505674, "FromInceptContractBillings" = 2476537, "EstimatedCostToComplete" = 9994326, "PercentageComplete" = 0.18, "UnderOverBillings" = -283372 WHERE ("contract","ContraNumberAxis_domain")=3
---> OK. 1 row(s).
```

<table>
<thead>
<tr>
<th>ContractNumber</th>
<th>ContractName</th>
<th>EstRevenue</th>
<th>EstCosts</th>
<th>EstGrossProfit</th>
<th>ForPeriodEarnedRevenue</th>
<th>ForPeriodCosts</th>
<th>ForPeriodGrossProfit</th>
<th>EstimatedCostToComplete</th>
<th>PercentageComplete</th>
<th>UnderOverBillings</th>
<th>ForPeriodEarnedRevenue</th>
<th>ForPeriodCosts</th>
<th>ForPeriodGrossProfit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
<td>28531262</td>
<td>22771956</td>
<td>7059386</td>
<td>12113470</td>
<td>9246924</td>
<td>11987630</td>
<td>13525032</td>
<td>40.60663037</td>
<td>125840</td>
<td>3740588</td>
<td>2855269</td>
<td>885319</td>
</tr>
<tr>
<td>2</td>
<td>201</td>
<td>4765875</td>
<td>3915859</td>
<td>850016</td>
<td>4761592</td>
<td>3912340</td>
<td>4748777</td>
<td>3519</td>
<td>0.99</td>
<td>12815</td>
<td>850016</td>
<td>5013695</td>
<td>4761592</td>
</tr>
<tr>
<td>3</td>
<td>C_208</td>
<td>12187491</td>
<td>13500000</td>
<td>-1312509</td>
<td>2193165</td>
<td>3505674</td>
<td>2476537</td>
<td>9994326</td>
<td>0.18</td>
<td>-283372</td>
<td>12187491</td>
<td>13500000</td>
<td>-1312509</td>
</tr>
</tbody>
</table>

© 2015 Altova GmbH. All Rights Reserved.
Summary of MapForce data mapping approach

- As we have demonstrated here, it is easy to develop a data mapping from an XBRL-formatted WIP report to a database using MapForce.
- This data mapping can now be applied to any XBRL instance document that uses the same taxonomy!
- The mapping process for new instances can either be done interactively using MapForce itself.
- Or it can be automated by using MapForce Server and FlowForce Server to have the mapping be executed either based on a time-trigger or an event-trigger (e.g. when a WIP instance document is received in a certain directory).
- MapForce Server and FlowForce Server can be deployed either in your local IT infrastructure or in the cloud. They are available for Linux and Windows operating systems.
Alternative approach: RaptorXML+XBRL Server

- If more control, a more advance programming logic, or more complexity in the data model is required, or if the number of XBRL-formatted WIP reports to be ingested is huge so that performance optimizations for parallel processing are required, there is an alternative approach that we offer:

  - Altova’s RaptorXML+XBRL Server is an XBRL processing engine that is focused on high-speed and parallel processing on modern multi-core CPUs to achieve advanced throughput for XBRL validation

  - RaptorXML+XBRL Server comes with a built-in Python interpreter that allows a developer to add post-validation programming logic
How to get started with RaptorXML+XBRL Server

We have recently published the full sources for an example XBRL-to-database mapping project that is using RaptorXML+XBRL Server. This example is based on downloading the EDGAR company financial filings from the SEC, processing them, and writing them to a SQL database.

Using these sample sources can provide a great template for how to process XBRL instance documents for WIP, too:

1. Download, clone, or fork the sources from GitHub: https://github.com/altova/SECDB
3. You can request a free 30-day license key-code for all Altova products
Thank You!

For more information, please see our blog and website:

http://blog.altova.com

http://www.altova.com

Altova is a proud member of XBRL.US and the Center for Data Quality

Safe Harbor Statement

The presentation made during this meeting and other statements by Altova may contain forward-looking statements within the meaning of U.S. Private Securities Litigation Act of 1995 including without limitation plans with respect to future business or product strategy. Although Altova believes that these statements are based on reasonable assumptions within the bounds of its knowledge of its business and operations, forward-looking statements are subject to numerous assumptions, risks and uncertainties. By making these forward looking statements, the company undertakes no obligation to update these statements for revisions or changes after the date of this presentation. Additionally, Altova may revise its projections or plans as required during the course of its business. Actual results may differ materially from forward-looking statements or historical performance due to the factors discussed in this presentation and elsewhere. Potential factors that could impact results include such include increased competitive pressures, changes in general economic conditions, difficulties in the timely development of new products and services or other changes.

Altova, MobileTogether, MissionKit, XMLSpy, MapForce, StyleVision, UModel, DatabaseSpy, DiffDog, SchemaAgent, SemanticWorks, Authentic, and AltovaXML are trademarks and/or registered trademarks of Altova, Inc. in the United States and/or other countries. The names of and reference to other companies and products mentioned herein may be the trademarks of their respective owners.