

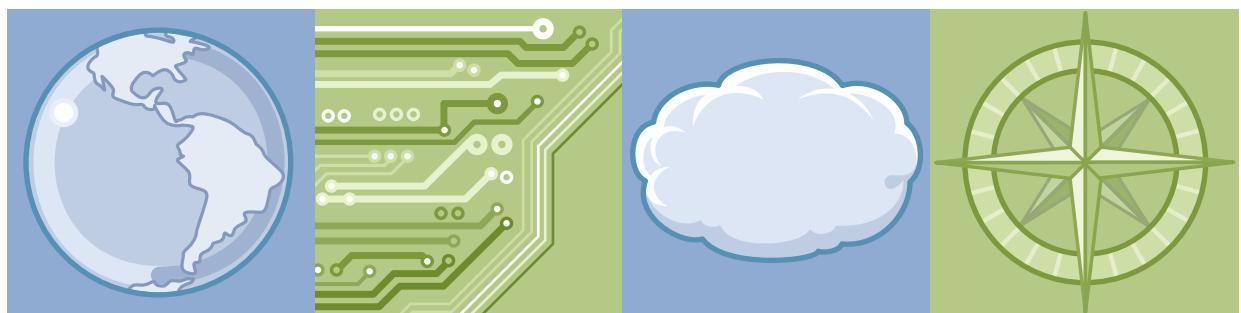


IBM Training

Student Exercises

InfoSphere MDM Algorithms

Course code ZZ780 ERC 1.0



Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

DataStage®

DB™

DB2®

InfoSphere®

Notes®

Rational®

Tivoli®

WebSphere®

Pentium is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Lenovo and ThinkPad are trademarks or registered trademarks of Lenovo in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

April 2014 edition

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Contents

Trademarks	vii
Exercise 2. Creating a new Algorithm	2-1
Part 1: Opening the PERSON algorithm	2-2
Part 2: Configure the National ID Number attribute	2-3
Part 3: Configure the sex attribute	2-13
Part 4: Configure the legal name attribute	2-14
Part 5: Configure the birth date attribute	2-18
Part 6: Configure the home address attribute	2-20
Part 7: Configure the phone attribute	2-22
Part 8: Adding default threshold values	2-24
Exercise 3. Loading Members and viewing Virtual data model	3-1
Part 1: Starting the InfoSphere MDM server	3-2
Part 2: Deploying the MDM Configuration	3-2
Part 3: Deploying Sample Data	3-4
Exercise 4. Bucket Analysis	4-1
Part 1: Running Attribute Completeness	4-2
Part 2: Running bucket analytics	4-4
Part 3: Configuration modification	4-10
Exercise 5. Weight Generation	5-1
Part 1: Generating weights	5-2
Part 2: Redeploy the Configuration	5-6
Part 3: Viewing the weights	5-8
Exercise 6a. Bulk Cross Load	6a-1
Part 1: Run a bulk cross match and load entity data	6a-2
Exercise 6b. Pair Manager and Threshold Calculations	6b-1
Part 1: Generate and review Threshold Analysis sample pairs	6b-2
Part 2: Review Threshold Analysis sample pairs	6b-7
Part 3: Setting new thresholds using Threshold Calculator	6b-9
Part 4: Re-deploy the configuration	6b-14
Exercise 6c. Testing Our Algorithms	6c-1
Part 1: Entity analysis overview	6c-2
Part 2: Member comparison	6c-4
Exercise 7a. Reset the MDM database	7a-1
Part 1: Stop the WAS server	7a-2
Part 2: Database reset	7a-2
Part 3: Setting the multi timezone settings	7a-5
Exercise 7b. Running Probabilistic Search with the Physical PME	7b-1
Part 1: Setup	7b-2

Part 2: Adding the Physical MDM record	7b-2
Part 3: Running the Probabilistic Searches	7b-6
Exercise 8a. Loading the PME Algorithm	8a-1
Part 1: Changing to the MDM Configuration perspective	8a-2
Part 2: Create the PME Configuration project	8a-2
Part 3: Import the Default PME Configuration	8a-4
Part 4: Examining the OOTB data model	8a-9
Part 5: Adding a new Attribute	8a-13
Part 6: Adding a new Source	8a-15
Exercise 8b. Viewing the PME Derived Data	8b-1
Part 1: Switch to the MDM Developer perspective	8b-3
Part 2: Accessing the MDM 11 tables	8b-3
Part 3: View the derived data	8b-5
Part 4: Viewing the Critical Data tables	8b-8
Exercise 9a. Customizing the Physical algorithms	9a-1
Part 1: Adding the Driving License to the Algorithm	9a-6
Part 2: Creating the Username Algorithm	9a-13
Exercise 9b. Bucket Analysis	9b-1
Part 1: Deploying the MDM Configuration	9b-2
Part 2: Deploying Sample Data	9b-4
Part 3: Running bucket analytics	9b-17
Exercise 9c. Generating Weights	9c-1
Part 1: Generating weights	9c-2
Part 2: Testing the previous weights	9c-5
Part 3: Loading the new weights	9c-13
Part 4: Deploying the Physical PME Algorithm	9c-15
Part 5: Testing our new Configuration	9c-22
Exercise 10. Customizing MDM Converters	10-1
Part 1: Creating our new Development Project	10-2
Part 2: Creating Custom Converters	10-6
Part 3: Resolving Errors and configuration	10-8
Part 4: Testing our new Configuration	10-23
Appendix A. MDM Virtual Data Model	A-1
Part 1: Creating a new Virtual MDM Configuration project	A-1
Part 2: Adding a new member type	A-7
Part 3: Adding attributes	A-10
Part 4: Adding the Entity Types	A-16
Part 5: Adding composite views	A-17
Part 6: Adding definitional sources	A-19
Part 7: Adding informational sources	A-21
Part 8: Incorporating strings	A-22
Appendix B. Algorithm Configuration	B-1
Part 1: Configuring the SEX Algorithm	B-1

Part 2: Configure the legal name attribute	B-5
Part 3: Configure the birth date attribute	B-19
Part 4: Configure the home address attribute	B-26
Part 5: Configure the phone attribute	B-28
Appendix C. Bulk Cross Load.....	C-1
Part 1: Run a bulk cross match and load entity data	C-2
Appendix D. Pair Manager and Threshold Calculations.....	D-1
Part 1: Generate and review Threshold Analysis sample pairs	D-2
Part 2: Review Threshold Analysis sample pairs	D-7
Part 3: Setting new thresholds using Threshold Calculator	D-10
Part 4: Re-deploy the configuration	D-16
Appendix E. Entity Analytics.....	E-1
Part 1: Entity analysis overview	E-2

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

DataStage®

DB™

DB2®

InfoSphere®

Notes®

Rational®

Tivoli®

WebSphere®

Pentium is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Lenovo and ThinkPad are trademarks or registered trademarks of Lenovo in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

Exercise 2. Creating a new Algorithm

What this exercise is about

This exercise covers using Workbench to create and configure a PME algorithm based on a preconfigured Member Model (.imm).

What you should be able to do

At the end of this exercise, you should be able to:

- Create an algorithm with the following settings:
- National ID (used for bucketing and comparison)
- Gender (used for comparison)
- Legal Name (used for multiple buckets and comparison)
- Birth Date (used for bucketing and comparison)
- Home Address (used for bucketing and comparison)
- Mobile and Home Phones (Used for bucketing and comparison)
- Configure default thresholds

Introduction

Typically, a generic algorithm is imported along with your project configuration, but in class we will begin with an empty algorithm. This algorithm will need to be configured to address the attributes you are using, the standardization that you want to use on those attributes, the comparisons that you would like to use, and the bucketing strategy that you would like to employ. You can use Workbench to make your edits. The tool will validate your design and present you with a list of errors if there are any inaccuracies in your algorithm design.

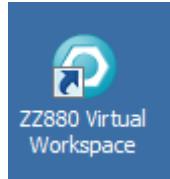
Requirements

A data model has been configured using the Basic Template and adding the attributes to our Person Member Type that includes: Birth Date (BIRTHDT), Gender (SEX), Home Address (HOMEADDR), Home Phone (HOMEPHON), Mobile Phone (MOBILEPHON), Name (LGLNAME), National ID Number (NATID). If you are starting from a blank MDM v11 installation, see Appendix A for details on how the configuration file has been setup.

Exercise instructions

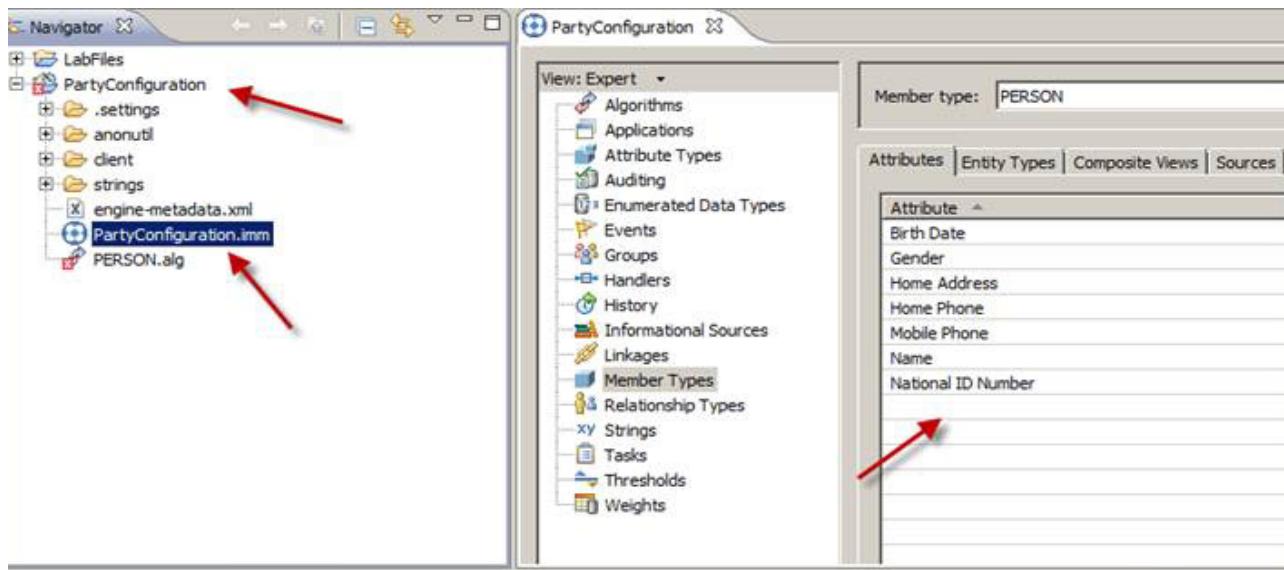
Part 1: Opening the PERSON algorithm

- __ 1. Login to the VMWare image using the following credentials:
 - __ a. Username: **Administrator**
 - __ b. Password: **passw0rd**
- __ 2. Once you are logged in, open the RAD environment by double clicking on the **ZZ880 Virtual Workspace** link on the desktop.



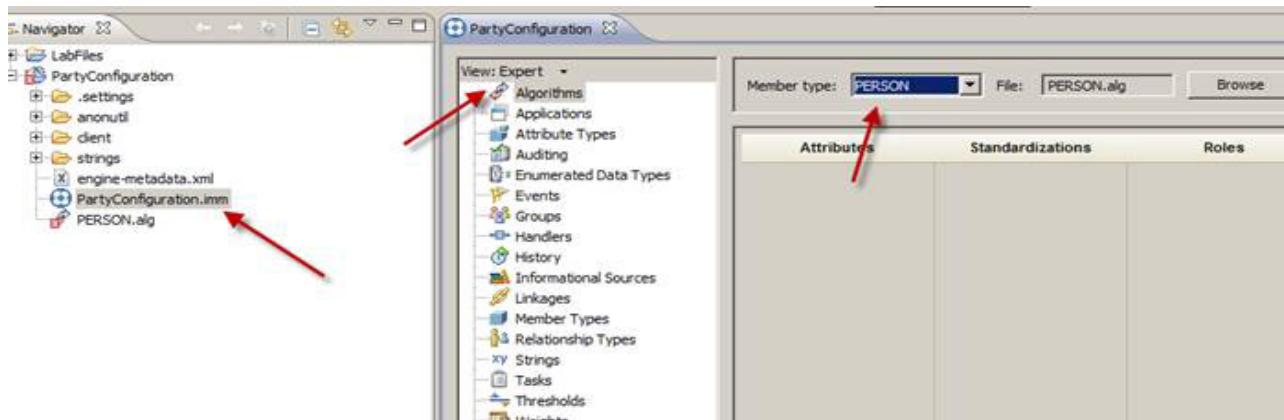
In the workspace, you'll find a PartyConfiguration project that was already created. (See Appendix A on how this was created). The PartyConfiguration was created using the basic template and the following Attributes:

- Birth Date
- Gender
- Home Address
- Home Phone
- Mobile Phone
- Name
- National ID Number



The PartyConfiguration data model is what we will work with for our algorithm implementation. You'll may see errors in our workspace, this is because we have not configured the algorithm yet for this configuration project.

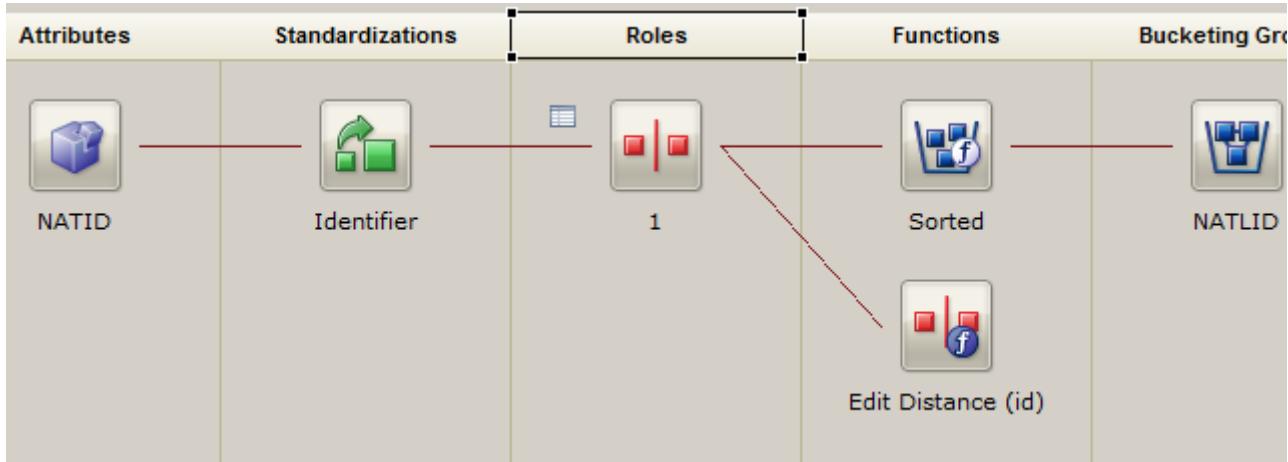
3. Open the configuration file by double clicking on the **PartyConfiguration.imm**. In the Configuration perspective, select the Algorithms tab. You'll notice that the algorithm is currently empty for our only Member Type (PERSON).



Part 2: Configure the National ID Number attribute

We will start off by configuring the National ID Number algorithm piece. For the National ID Number we will create a bucket to allow the comparison of records with similar National ID Number. The image below is the configuration of our National ID that we would like to create:

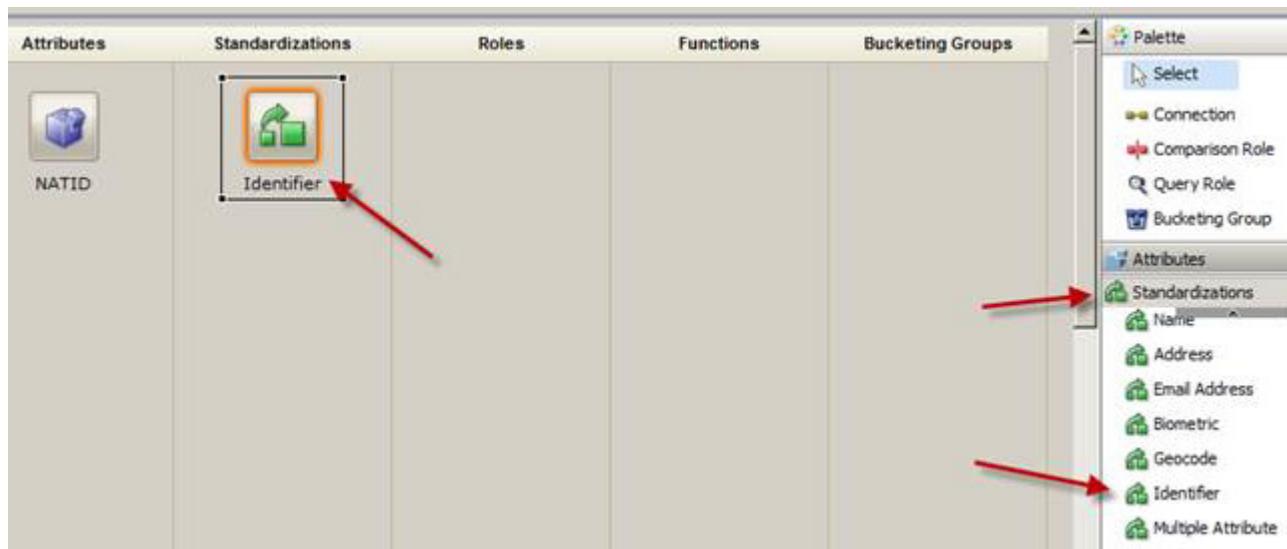
- Standardization: **Identifier** (special characters are filtered out, e.g. 675-asd-7008 is converted to 675ASD7008)
- Bucketing Function: **Sorted** (sorts the characters for buckets, e.g. ADS0056778)
- Comparison: **Edit Distance**



1. We start off by adding the National ID Attribute to our algorithm. On the right hand side, expand the **Attributes** view in the palette. Select the **NATID (IDENT)** and place the attribute in the first column.



2. Next, we will add the standardization function. On the right hand side of the algorithm, expand the **Standardizations** view in the palette and select the Identifier function. Place the Identifier standardization in the second column beside the NATID.



- 3. To connect the NATID to the Identifier, select the **Connection** icon in the palette. Once the Connection tool is selected, click the **NATID** icon in the Attributes column and then click the **Identifier** icon under the Standardization column.



- 4. Press the **ESC** key or select the **Select** icon in the palette to exit the Connection tool.
- 5. Next we need to customize the function's properties. Select the **Identifier** icon in the Standardizations column. In RAD, you will find a Properties window (bottom left hand corner). Select the **Properties** window tab.

The screenshot shows the IBM InfoSphere MDM Workbench interface. On the left is the Navigator pane with a tree view of project files including 'LabFiles', 'PartyConfiguration', '.settings', 'anonutil', 'client', 'strings', 'engine-metadata.xml', 'PartyConfiguration.imm', and 'PERSON.alg'. The main area is titled 'PartyConfiguration' and contains a list of configuration items like 'Attribute Types', 'Auditing', 'Events', etc. To the right are two tabs: 'Attributes' and 'Standardizations'. Under 'Attributes', there is a box labeled 'NATID'. Under 'Standardizations', there is a box labeled 'Identifier' with a green checkmark icon. A red arrow points from the 'Properties' tab in the bottom navigation bar to the 'Field arguments' row in the Properties table.

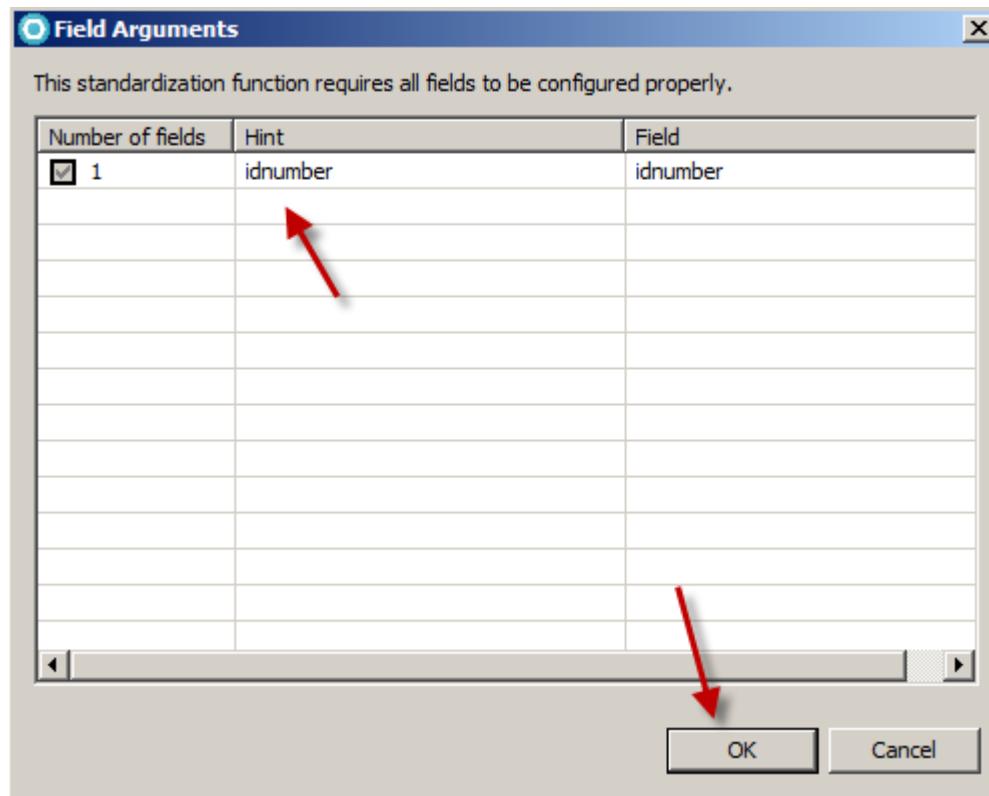
Property	Value
Anonymous string code	
Character mapping string code	
Derivation arguments	
Description	
Equivalent string code	
Field arguments	
Include Source	False
Primary standardization role	1
Primary standardization role label	
Record status filter	AI
Standardization function code	IDENT1
Type	Alphanumeric

6. Click in the **Value** column for property **Field arguments**. Click the ... button to display available field arguments for you to choose from.

The screenshot shows the same IBM InfoSphere MDM Workbench interface. The 'Properties' tab is active in the bottom navigation bar. The Properties table has the 'Field arguments' row selected. A red arrow points from the ellipsis button (...) in the 'Value' column of the 'Field arguments' row to the button itself.

Property	Value
Anonymous string code	
Character mapping string code	
Derivation arguments	
Description	
Equivalent string code	
Field arguments	
Include Source	False
Primary standardization role	1
Primary standardization role label	
Record status filter	AI
Standardization function code	IDENT1

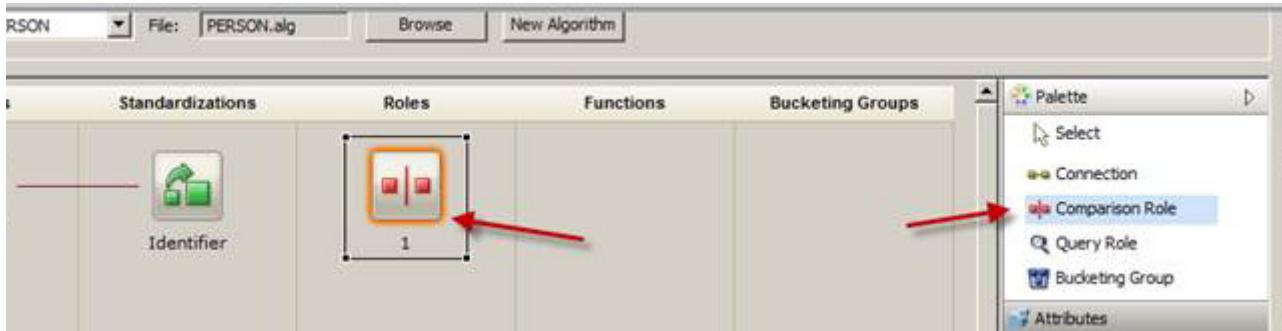
7. Since there is only one field from an IDENT attribute type and the Identifier standardization function accepts one field, the **idnumber** will already be selected. Click the **OK** button.



- 8. Under the **Properties** window, change the following values:
- a. Record Status Filter: **AI**.
 - b. Type: **Numeric**.

Property	Value
Anonymous string code	
Character mapping string code	
Derivation arguments	
Description	
Equivalent string code	
Field arguments	idnumber
Include Source	False
Primary standardization role	1
Primary standardization role label	AI
Record status filter	IDENT1N
Standardization function code	
Type	Numeric

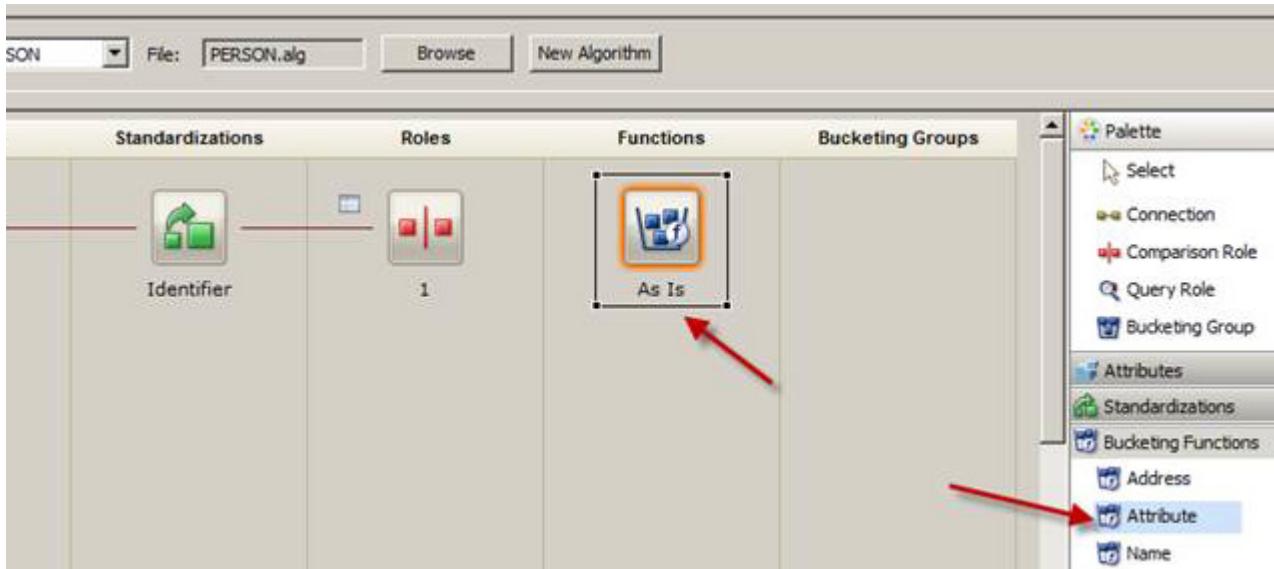
- 9. Once the value has been standardized, all the values get placed in a Comparison Role (this feeds the bucketing and comparison functions). Back in the Algorithms window, select the **Comparison Role** icon in the palette and place it in the 3rd column beside the Identifier standardization.



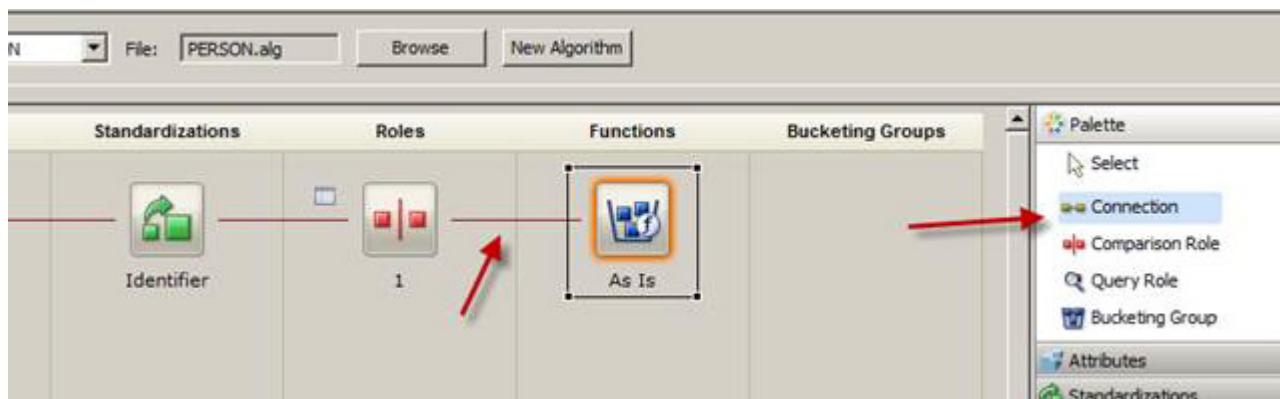
10. Connect the **Identifier** Icon to the **Comparison Role 1** using the Connection tool from the palette.



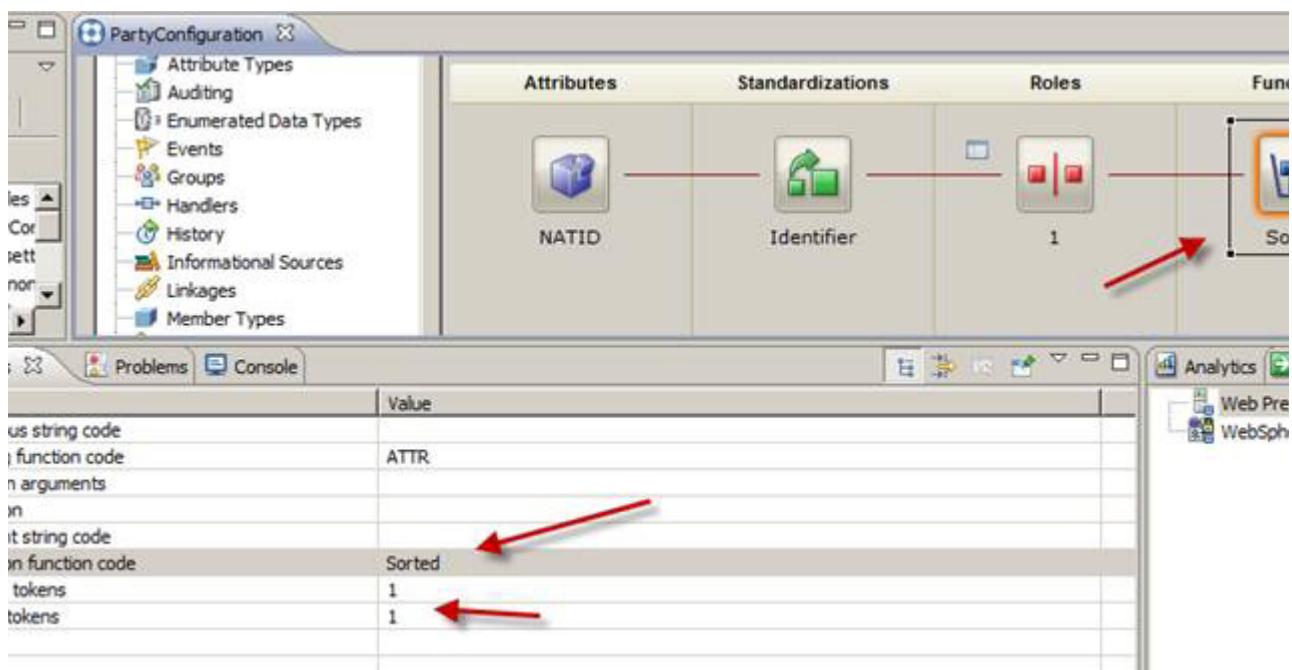
11. Next we will add our bucketing, this will be based on a sorted value of our Identifier, meaning it will be compared to other ID with the same character values (but possibly in a different order. Expand the **Bucketing Functions** view in the palette and select the **Attribute** icon. Place the Attribute bucketing function in the fourth row of the algorithm.



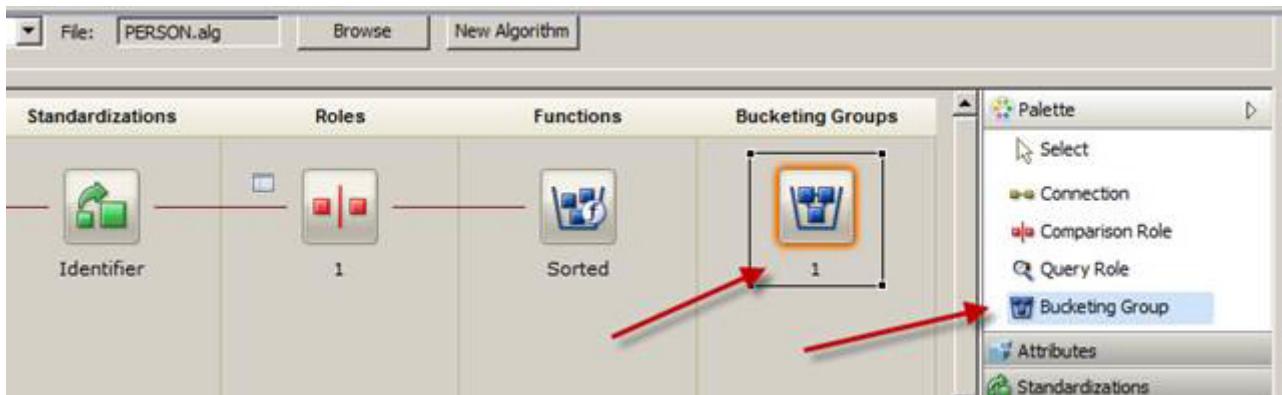
12. Connect the **Comparison Role 1** to the **As Is** bucketing function using the Connection tool from the palette.



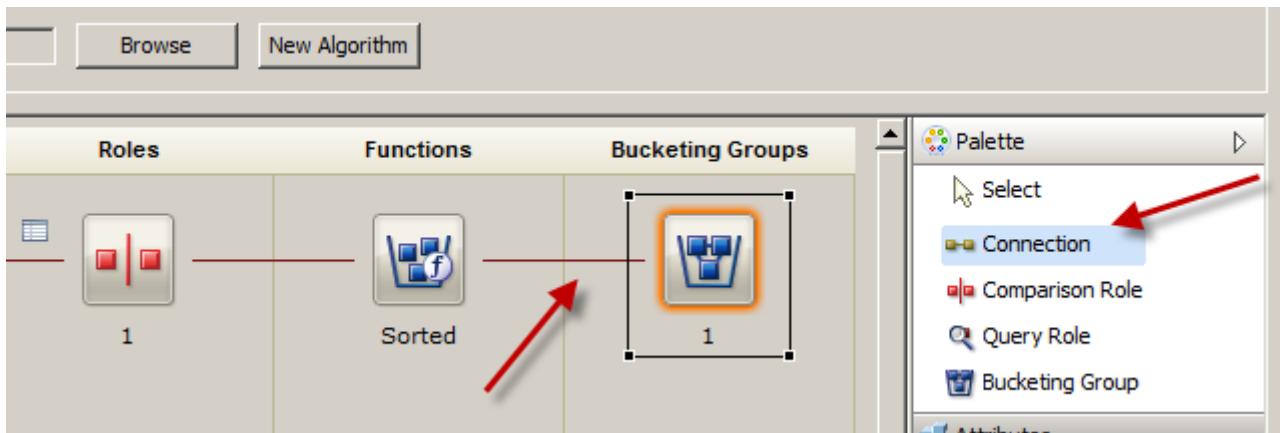
13. Select the **As Is** bucketing function icon in our algorithm and open the **Properties** window in RAD. Change the following values in the Properties window:
- Generation Function Code: **Sorted**.
 - Max Tokens: **1**.
 - Min Tokens: **1**.



14. Now that the bucketing function is set up, we need the actual bucket group in place. Select the **Bucketing Group** from the palette and place the Bucketing Group in the 5th column.

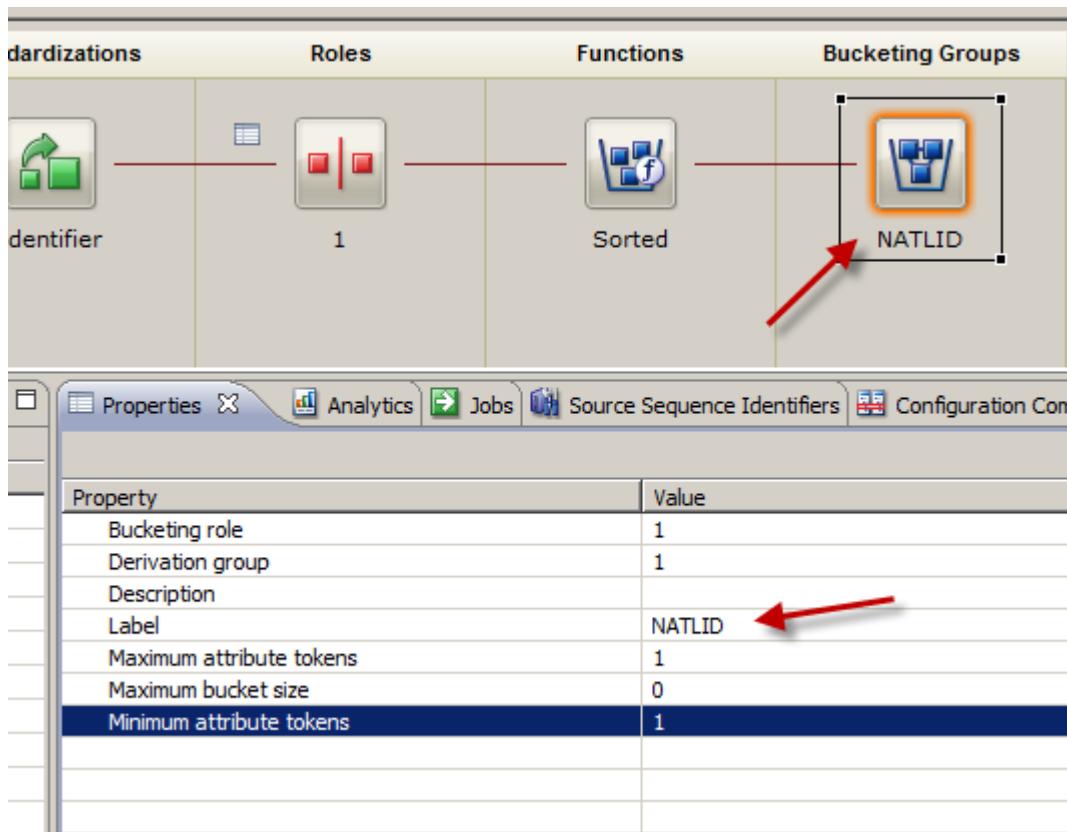


- ___ 15. Connect the **Sorted** function to the new Bucketing Group 1 using the **Connection** tool from the palette.

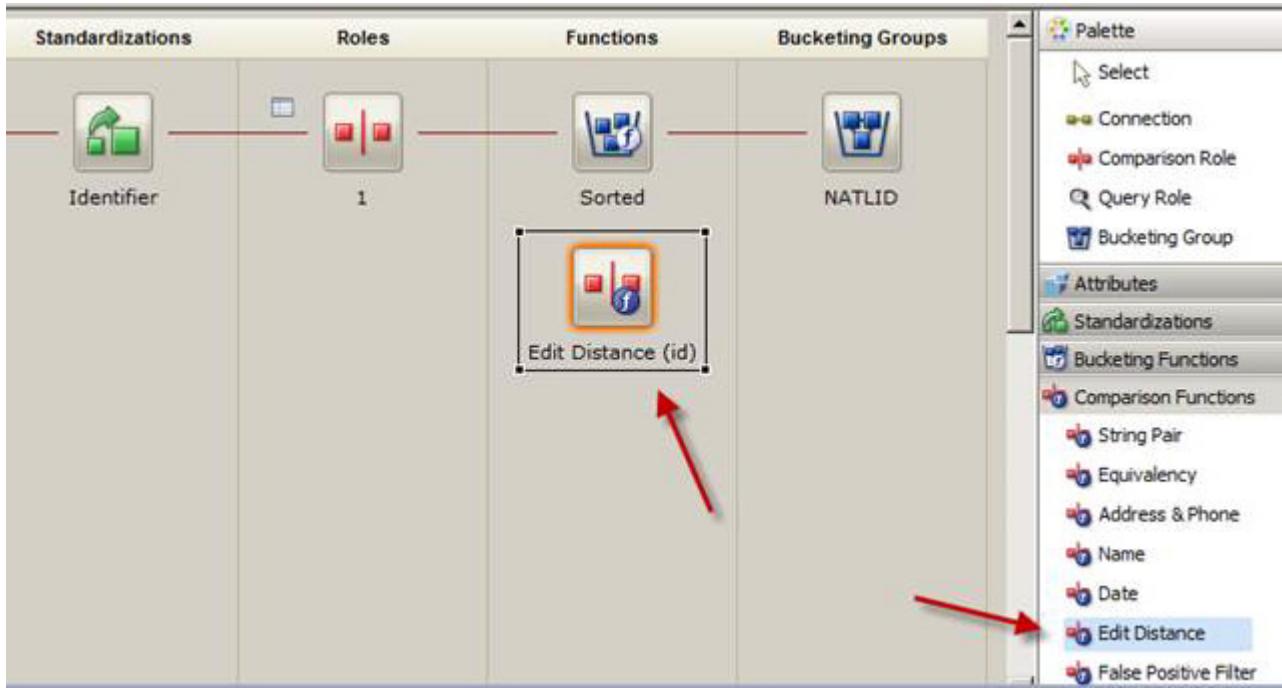


- ___ 16. Select the new **Bucketing Group 1** and open the RAD Properties window. Change the following values:

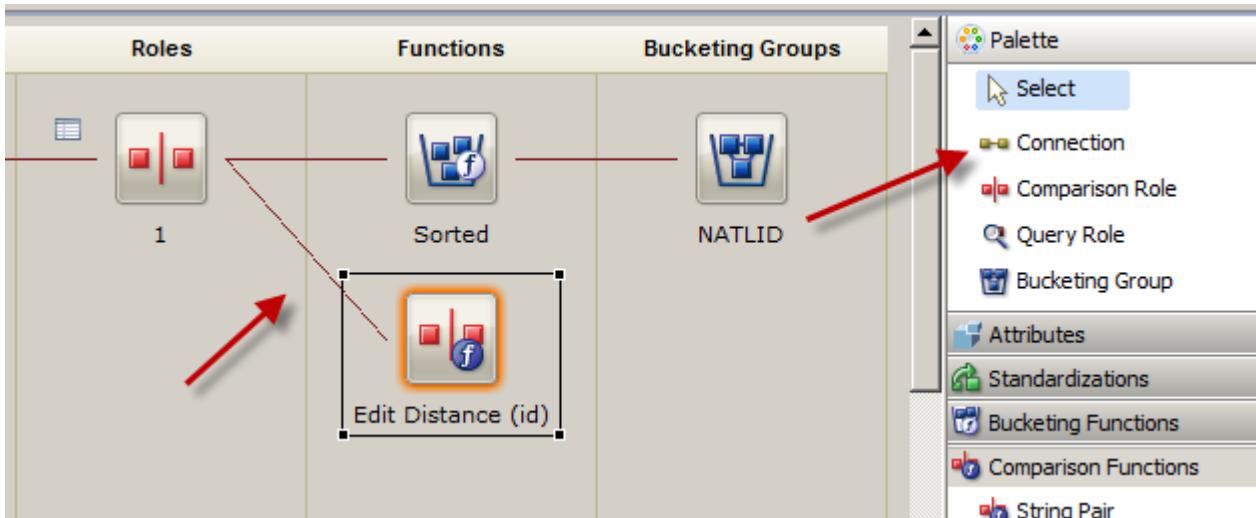
- ___ a. *Label: NATLID.*



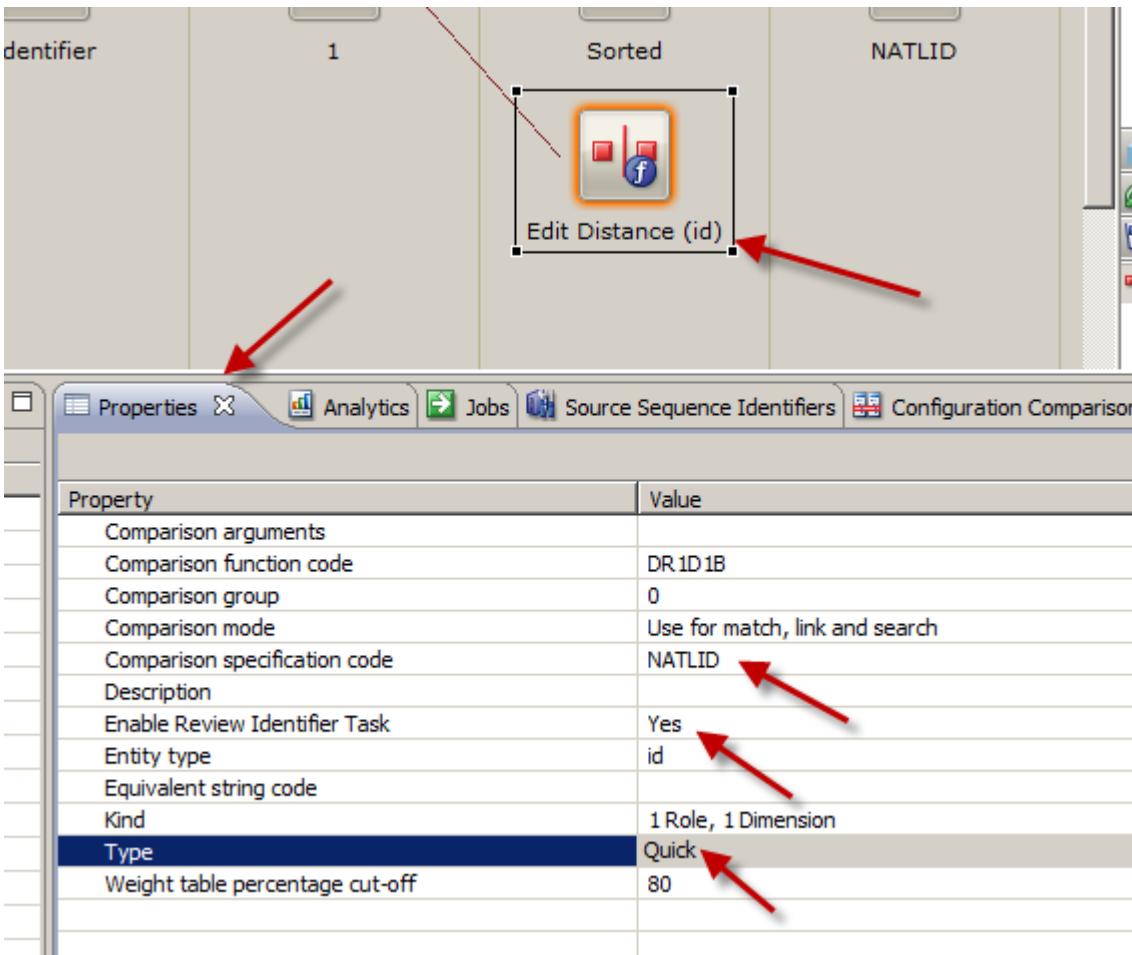
17. Not only will we compare members that have a similar national id, we will also use the edit distance between the national ids to contribute to the overall comparison score. To add a comparison function, expand the **Comparison Functions** view in the palette, select the **Edit Distance** and place the function in the 4th column under the **Sorted** bucketing function.



- ___ 18. Using the Connection tool from the palette, connect the **Comparison Role 1** to the **Edit Distance (id)** comparison function.



- ___ 19. Press the **ESC** key to exit from the Connection tool.
- ___ 20. Select the **Edit Distance (id)** icon in the Functions column and open the **Properties** tab in RAD. Change the following values for our comparison function:
- ___ a. Comparison Spec Code: **NATLID**.
 - ___ b. Enable Review ID Task: **Yes**.
 - ___ c. Kind: **1 Role, 1 Dimension**.
 - ___ d. Type: **Quick**.



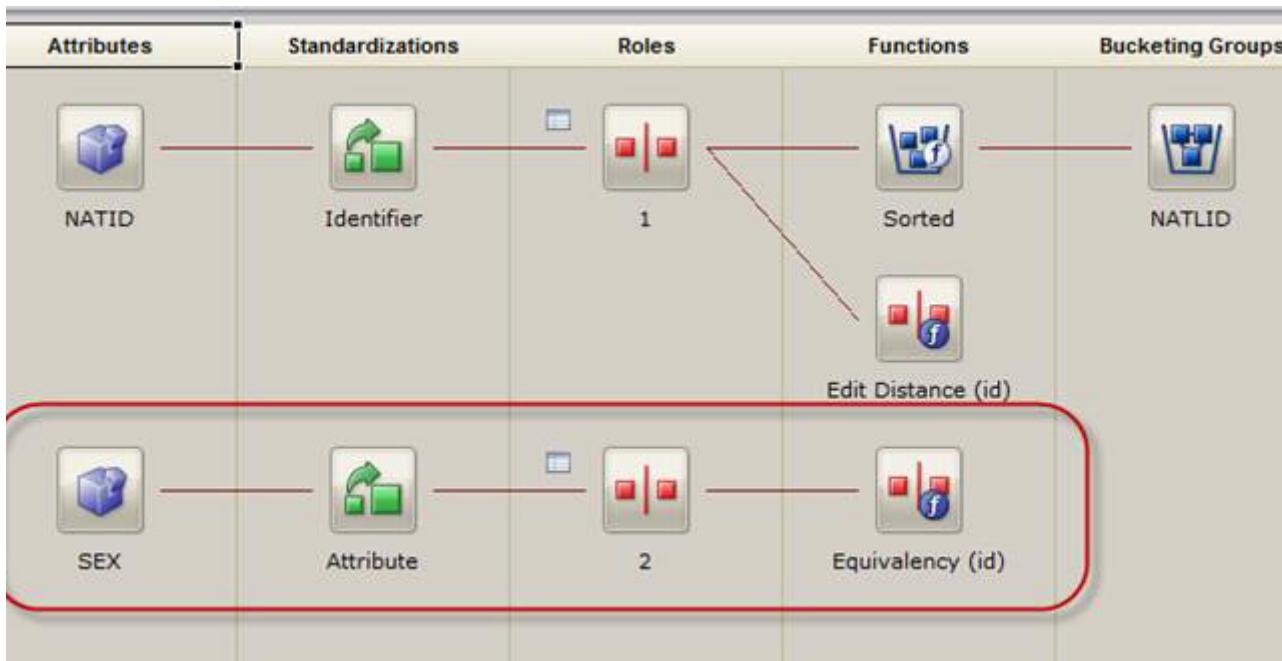
___ 21. Save the project (CTRL+S)

Part 3: Configure the sex attribute

The sex (or gender) attribute will not be used for bucketing (since they would create huge buckets) but we would like to give a small value to the comparison score if the gender matches. Since we've walked you through creating the algorithm in the first part, in this part you will attempt to create the Sex algorithm without the steps. (if you get stuck we provided the steps in Appendix B).

- ___ 1. Create the following algorithm pieces (and connections) for the SEX attribute with the following values:
 - ___ a. Attribute: **SEX (ATTR)**
 - ___ b. Standardization: **Attribute**
 - i. Field arguments: **1 - attrval**
 - ii. Record Status Filter: **All**
 - iii. Type: **Alphanumeric**.
 - ___ c. Comparison Role
 - ___ d. Comparison Function: **Equivalency**

- iv. Comparison Spec Code: SEX.
- v. Enable Review ID Task: No.
- vi. Type: Alphanumeric.



Part 4: Configure the legal name attribute

For the Person's Name, we will create 2 Bucketing functions, one that includes the Name + Birth Date and another that includes only the Last Name and Zip code. The reason we don't want to create a bucket for just the Name is that the buckets might become very large and impact performance, so we will compare people that have similar sounding names (phonetic) and have the same Birth Data or Zip Code.

For the comparison of the Name, we will use the Name comparison function. The Name comparison function will compare each token in the Names, providing equivalency names, bonuses for position and frequency based scores (i.e. common names score lower than unique names)

Legal name plus Birth Date

We will start off creating the Bucketing for the Legal Name and Birth Date. The detailed steps to this algorithm piece are also found in **Appendix A** if you require additional information.

- 1. Create the following algorithm pieces (and connections) for the **LGLNAME** attribute with the following values:
 - a. Attribute: **LGLNAME**
 - b. Standardization: **Name**
 - vii. Field arguments: **onmlast**, **onmfirst**, **onmmiddle**, **onmprefix**, **onsuffix**.
 - viii. Primary standardization role label: **Name Tokens**.

- ix. Record Status Filter: **AI**.
 - x. Type: **Person**.
 - xi. Anonymous string code: **ANONNAME**.
- ___ c. Comparison Role


Hint

For the Field argument of the Standardization Function, you will need to check each of the checkbox 1 to 5 and select the values in the order above.

- ___ d. Bucketing Function: **Name**

- i. Description: **Full Name**.
- ii. Generation function code: **Equivalence & Phonetic**.
- iii. Derivation arguments: **NORMPHONE**
- iv. Equivalent string code: **ALTNAMES**.
- v. Maximum tokens: **1**.
- vi. Minimum tokens: **1**.
- vii. Type: **Person**.


Note

By setting the Maximum token and minimum token to 1, this means the buckets that are generated will only have one token from the Legal Name in order to create the bucket. This is important because we want our buckets to use one token from the Name and the other from the Birth Date.

- ___ e. Bucketing Group

- i. Description: **Use 1 name token + dob to build**.
- ii. Label: **1 Name token + DOB**.
- iii. Maximum attribute tokens: **2**.
- iv. Maximum bucket size: **0**.
- v. Minimum attribute tokens: **2**.

**Note**

By setting the maximum attribute and minimum attribute tokens to 2, this means the buckets will be created using 2 tokens from the input. The name bucketing function specified that only one token will be sent to the bucketing group, which means the other token will always be taken from the Birth Date.

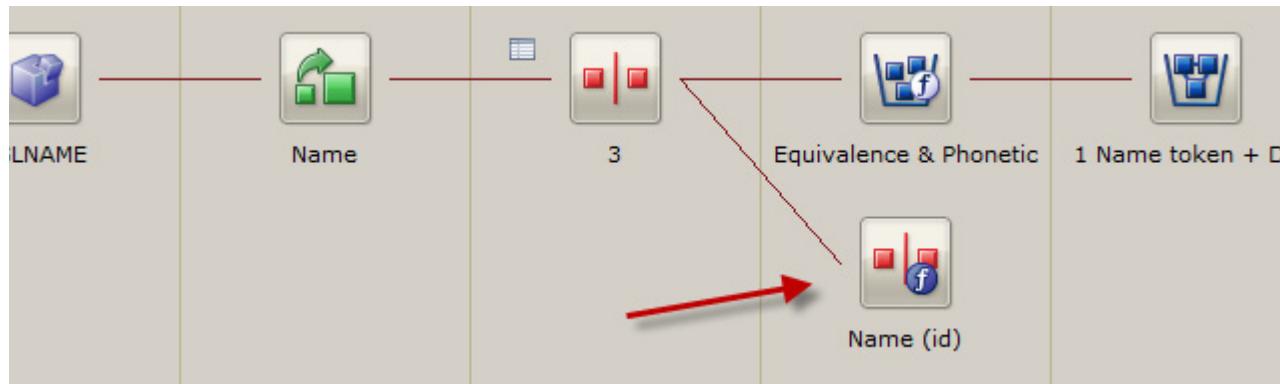


This bucketing group is still missing the Birth Data attribute, but we will leave it for now and connect the Birth Date once we add it to our algorithm.

Name Comparison Function

We will also use the name for the comparison of our Members. For the Comparison Function we can reuse the standardization of the name (since the values that are included in the Name standardization we just created include all the tokens we want to compare.)

- 1. Add the following comparison function to the algorithm and attach it to the **Comparison Role 3**.
 - a. Comparison Function: **Name**
 - i. Comparison specification code: **NAME**.
 - ii. Equivalent string code: **ALTNAMES**.
 - iii. Type: **Person (comprehensive)**.



Last Name plus Zip Code

For the Last Name and Zip Code bucket, we will create a new Standardization function since the first one we created included 5 attribute tokens and we only need one (omnlst). Use the LGLNAME attribute that we already added to algorithm and start by create a second standardization function for the attribute.

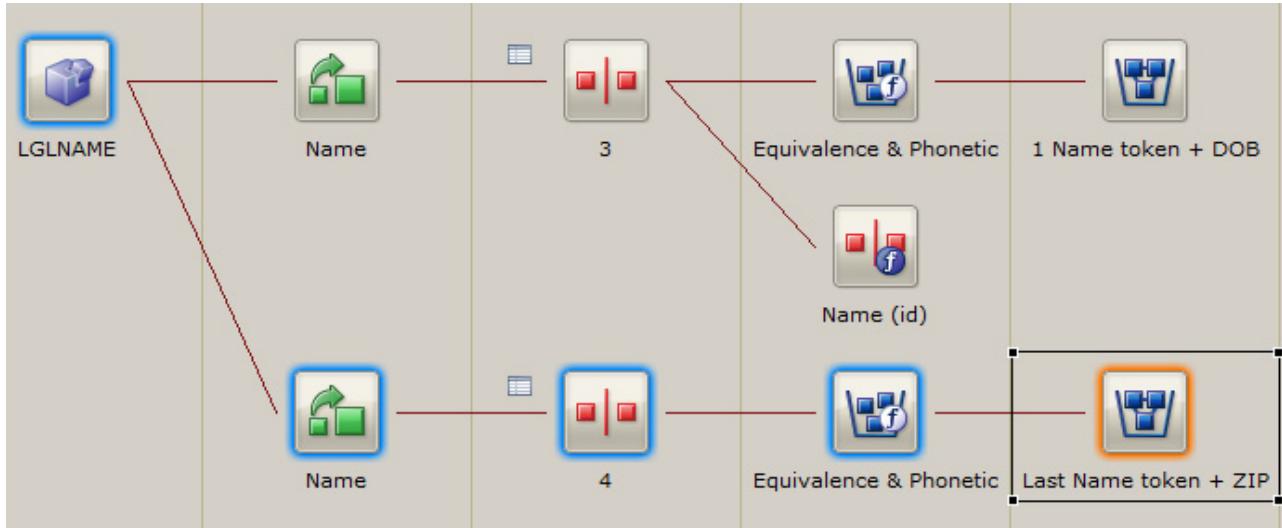
- __ 1. Create the following algorithm pieces (and connections) for the LGLNAME attribute with the following values:
 - __ a. Standardization: **Name**
 - i. Field arguments: **omnlst**
 - ii. Primary standardization role label: **Last Name Token**
 - iii. Record Status Filter: **All**.
 - iv. Type: **Person**.
 - v. Anonymous string code: **ANONNAME**.



Note

Connect the existing LGLNAME attribute to your new Standardization function.

- __ a. Comparison Role
- __ b. *Bucketing Function: Name*
 - i. Description: **Last Name token**.
 - ii. Generation function code: **Equivalence & Phonetic**.
 - iii. Derivation arguments: **NORMPHONE**
 - iv. Equivalent string code: **ALTNAMES**.
 - v. Maximum tokens: **1**.
 - vi. Minimum tokens: **1**.
 - vii. Type: **Person**
- __ c. Bucketing Group
 - i. Description: **Last name token + zip to build**.
 - ii. Label: **Last Name token + ZIP**.
 - iii. Maximum attribute tokens: **2**.
 - iv. Maximum bucket size: **0**.
 - v. Minimum attribute tokens: **2**.

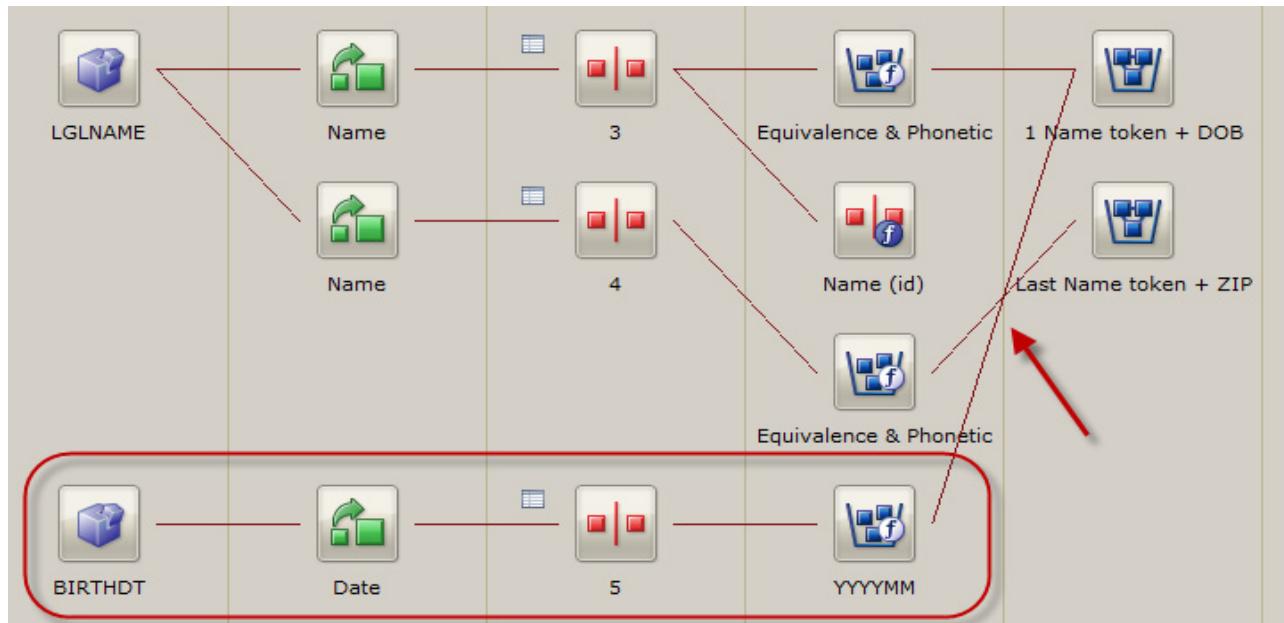


Part 5: Configure the birth date attribute

For the Birth Date, we won't create a bucket by itself (buckets would be too large), but will attach the attribute to our Name plus DOB bucket for the second token in that bucket. We will also create a date comparison function for the Birth Date that will provide a score when the day, month or year match.

- ___ 1. Create the following algorithm pieces (and connections) for the BIRTHDT attribute with the following values:
 - ___ a. Attribute: **BIRTHDT**
 - ___ b. Standardization: **Date**
 - i. Field Arguments: **dateval**
 - ii. Anonymous string code: **ADATE**.
 - iii. Primary Standardization role label: **Birth Date**.
 - iv. Record status filter: **AI**.
 - v. Type: **Standard**.
 - ___ c. Comparison Role
 - ___ d. *Bucketing Function: Date*
 - i. Description: **Birthdate**.
 - ii. Generation function code: **YYYYMM**.
 - iii. Maximum tokens: **1**.
 - iv. Minimum tokens: **1**.
 - v. Type: **Normal**

- __ e. Bucketing Group: Connect the Bucketing Function to the 1 Name token + DBO Bucketing Group.

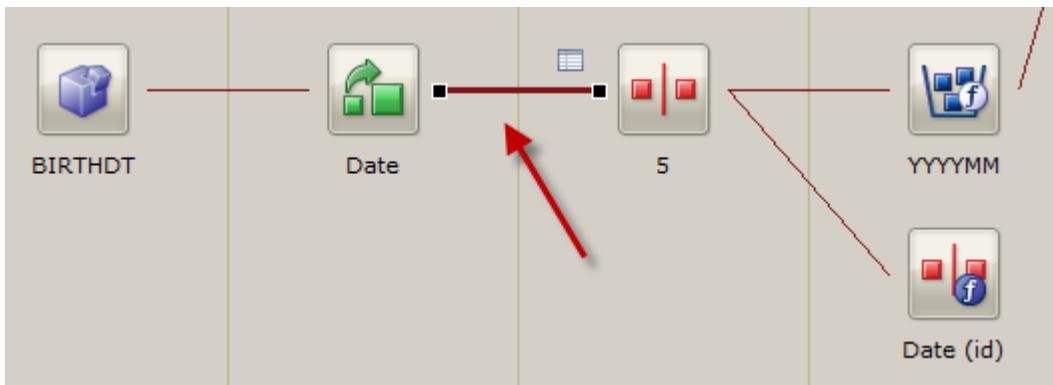


- __ 2. For the Comparison of our Birth Dates, add the following Comparison Function attaching it to the Comparison Role 5
- Comparison specification code: **DOB**.
 - Type: **Date**.

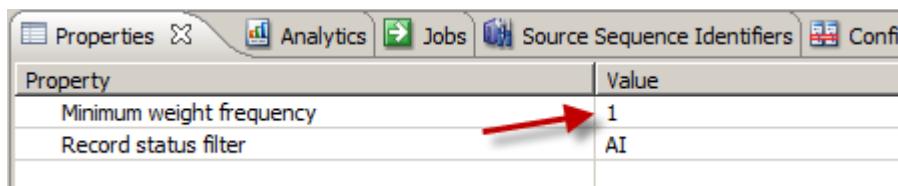


The Comparison function will use values specified for the various combinations of birth dates. To generate these values, we will populate the frequency of each date. To generate a frequency for each date regardless if there is only one value we need to modify the minimum frequency value.

- __ 3. Click the connection between the **Date** Standardization icon and the **Comparison Role 5**.



- ___ 4. Under the **Properties** window, change the Minimum weight frequency to **1**.



- ___ 5. Click the **Save** button to save your project.

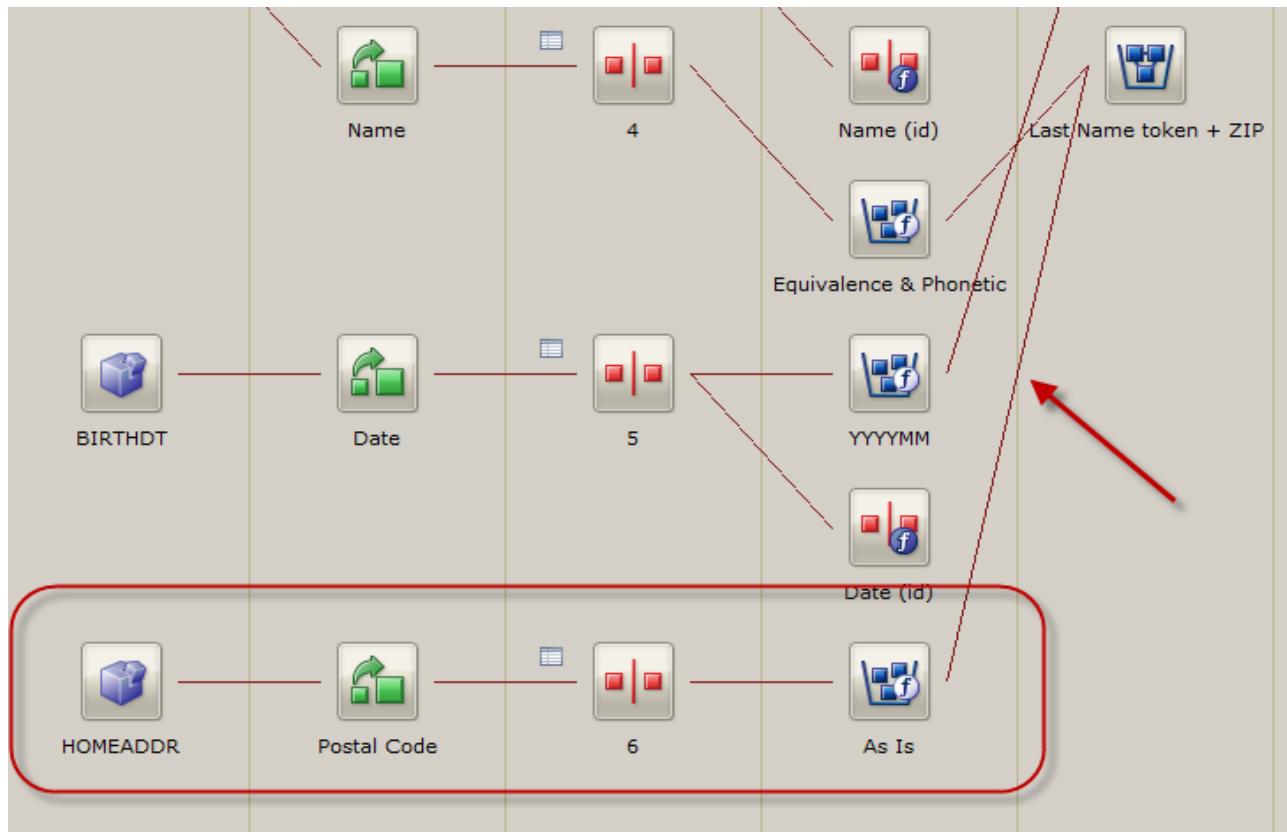
Part 6: Configure the home address attribute

To configure the home address, we will break it into 2 steps. The first step is to separate the Zip Code (or Postal Code) from the address so that we can use it in the Last Name token + Zip code bucket. The second step will be to create a comparison algorithm that will use the entire address (and phone) for the comparison. We will not create a bucket for the address alone, because even with multiple tokens in the buckets (e.g. New York, St, NY), you could have too many members. If we were storing business addresses however, it might work to create a bucket for a complete match of the address. By using a 2 dimensional comparison with Address and Phone, we can give someone bonus points for have both matches (Phone and Address), and adjust the weights as they get further away on both attributes.

Postal Code (Zip Code for Last Name bucket)

- ___ 1. Create the following algorithm pieces (and connections) for the **HOMEADDR(ADDR)** attribute with the following values:
- ___ a. Attribute: **HOMEADDR(ADDR)**
 - ___ b. Standardization: **Postal Code**
 - i. Field arguments: **zipcode**
 - ii. **Locale: United States.**
 - iii. **Record status filter: AI.**
 - ___ c. **Comparison Role**

- ___ d. Bucketing Function: **Attribute**
 - i. Description: Zip code.
- ___ e. Generation function code: **As Is**.
 - i. Maximum tokens: 1.
 - ii. Minimum tokens: 1.
- ___ f. Bucketing Group: **Connect the Bucketing Function to the Last Name token + Zip Bucketing Group**

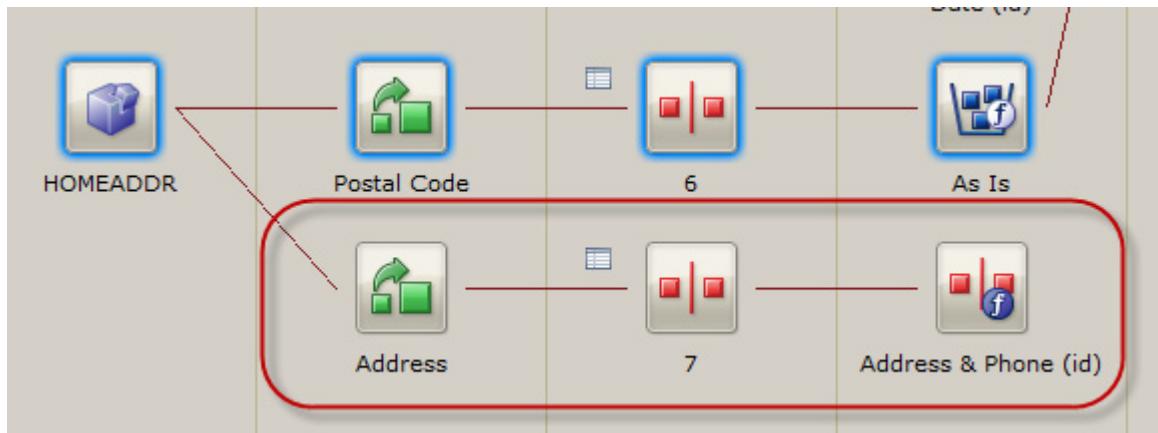


Address and Phone Comparison

For our address comparison, we will need to use all the tokens, not just the Zip Code (or Postal Code). To pull all tokens we can reuse the HOMEADDR attribute, but will need to create a new

Standardization function that takes all tokens. (NOTE: We will deal with the Phone part of the comparison later in this exercise)

- ___ 1. Create the following algorithm pieces (and connections) for the **HOMEADDR(ADDR)** attribute with the following values:
 - ___ a. Standardization: **Address** (Connect the existing HOMEADDR to this standardization function)
 - i. Field arguments: ***stline1,stline2,stline3,stline4,city,state,zipcode***
 - ii. *Record status filter*: AI.
 - iii. *Region*: **United States - expanded**.
 - ___ b. Comparison Role
 - ___ c. Comparison Function: **Address & Phone**
 - i. Comparison specification code: **AXP**



- ___ 2. Save your changes (**CTRL+S**)

Part 7: Configure the phone attribute

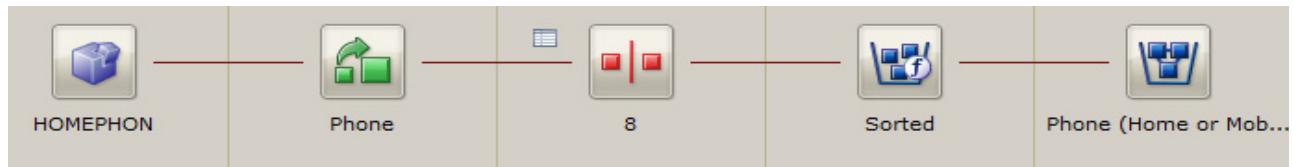
We have 2 phone types defined in our data model: Home Phone and Mobile Phone. We will combine both types into the same Comparison Roles as the same data type (e.g. someone might enter a mobile phone as their Home Phone).

We will also create a bucket for our phone based on the sorted values of the phone number (e.g 555-4212-9053 would be bucketed as 01223455559).

Creating the Phone Bucket

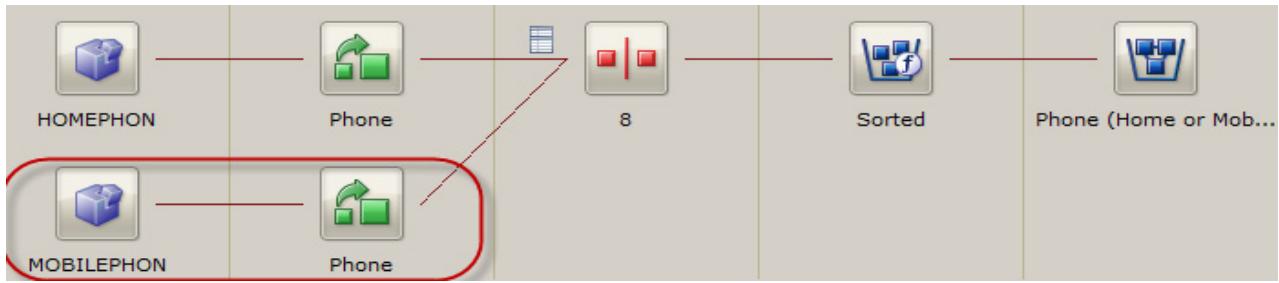
- ___ 1. Create the following algorithm pieces (and connections) for the **HOMEPHON (PHONE)** attribute with the following values:
 - ___ a. Attribute: **HOMEPHON (PHONE)**
 - ___ b. Standardization: **Phone**
 - i. Field arguments: ***phnumber***

- ii. Primary Standardization role label: **Home Phone**.
- iii. Record status filter: AI.
- iv. Type: **North America**.
- c. Comparison Role
- d. *Bucketing Function: Attribute*
 - i. Description: **Home Phone**.
 - ii. Generation function code: **Sorted**.
 - iii. Maximum tokens: **1**.
 - iv. Minimum tokens: **1**
- e. *Bucketing Group*
 - i. Description: **Phone (Home or Mobile) to build**.
 - ii. Label: **Phone (Home or Mobile)**.
 - iii. Maximum attribute tokens: **1**.
 - iv. Maximum bucket size: **0**.
 - v. Minimum attribute tokens: **1**



We are not complete yet, remember that we also want to include the Mobile Phone in this bucket (and comparison in the next part of the exercise), so we want to add it to the same Comparison Role (since it's the same data, just different type).

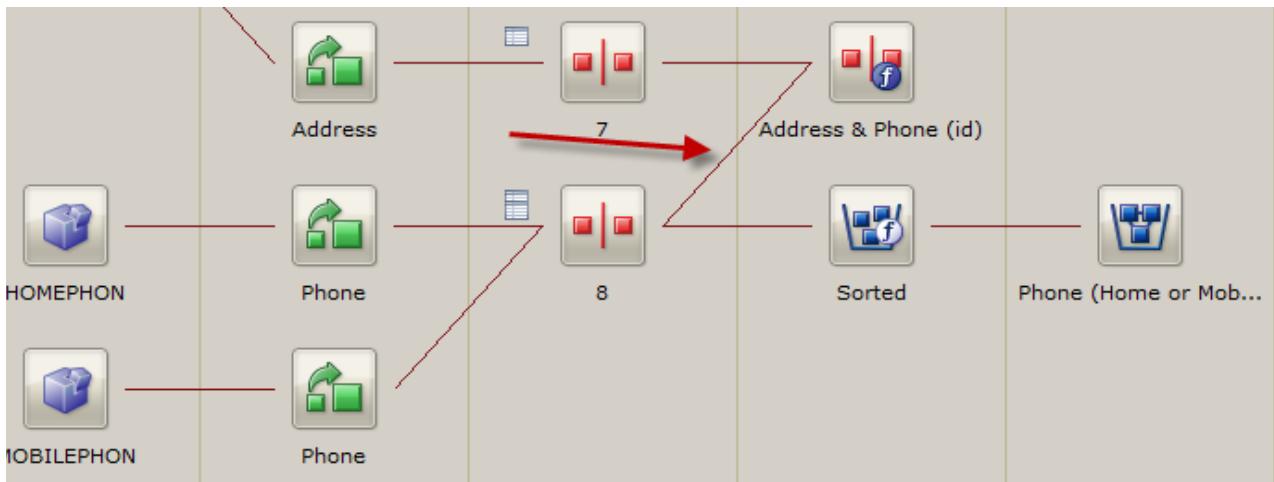
- 2. Create the following algorithm pieces (and connections) for the **MOBILEPHON (PHONE)** attribute with the following values:
 - a. Attribute: **MOBILEPHON (PHONE)**
 - b. Standardization: **Phone**
 - i. Field arguments: **phnumber**
 - ii. Primary Standardization role label: **Mobile Phone**.
 - iii. Record status filter: AI.
 - iv. Type: **North America**.
 - c. Comparison Role: **connect the Standardization function (Phone) to the same comparison role as the HOMEPHON attribute..**



Phone Number comparison

To compare our phone numbers, we have already created a 2 dimensional comparison with the Address, all that is left is to connect our Home Phone and Mobile Phone comparison role to our Address and Phone comparison function.

- 1. Using the palette's connection tool, connect the **Comparison Role 8** (the phone comparison role) to the **Address & Phone (id)** comparison function.

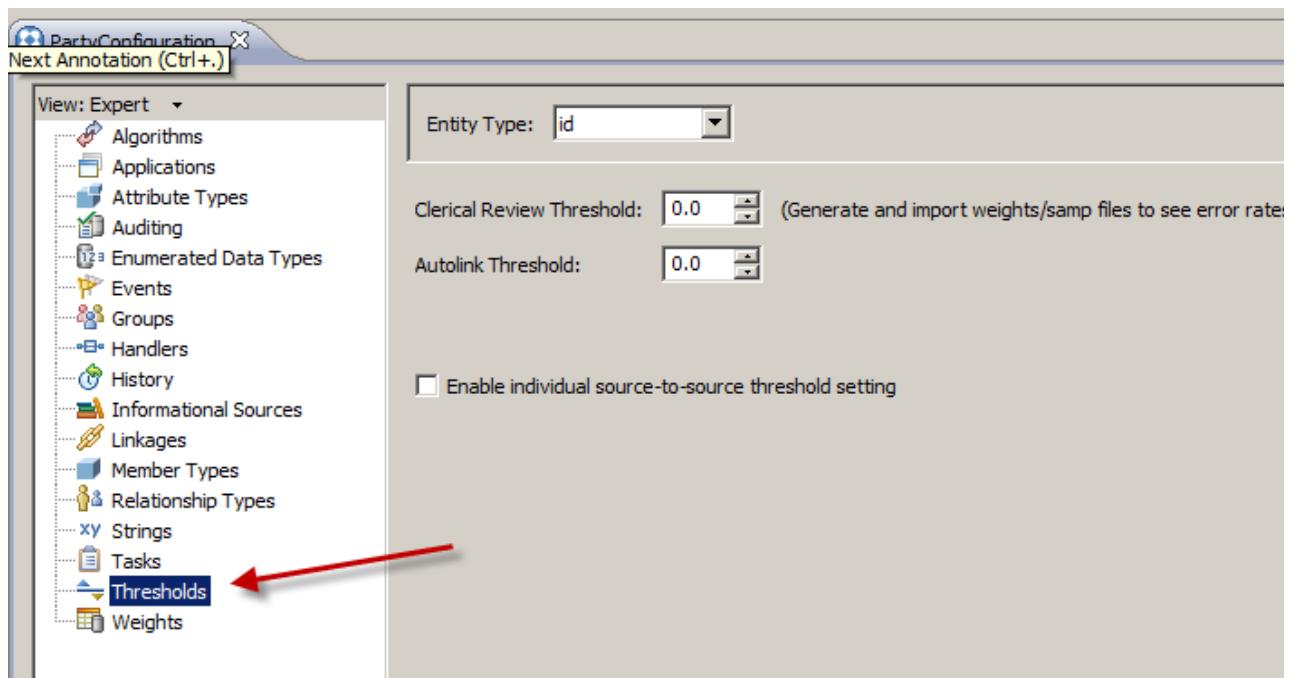


- 2. Save your changes (**CTRL+S**)

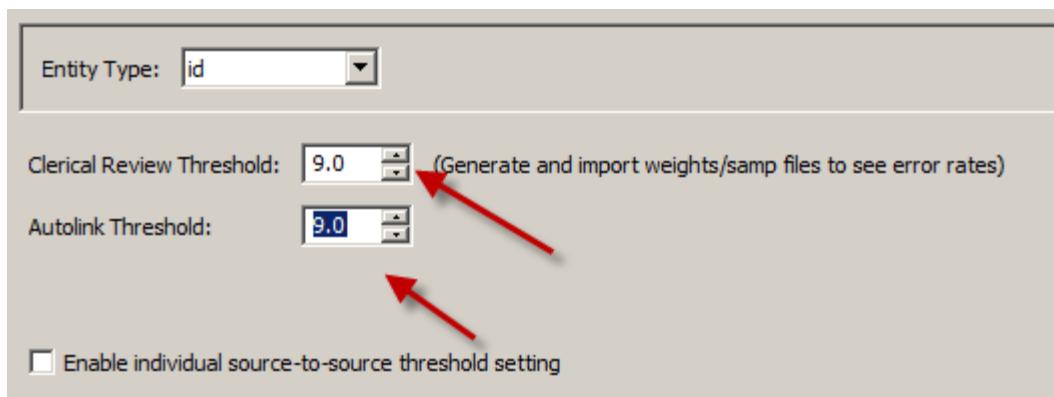
Part 8: Adding default threshold values

In this part of the exercise we will create some thresholds for linking our members. These thresholds are only a starting point, we will be running some reports and evaluating the thresholds later in this course.

- 1. In the Party Configuration, select the **Thresholds** tab.



- 2. Set both **Clerical Review** and **Autolink** thresholds to 9.0.



- 3. Save your changes (CTRL+S).

End of exercise

Exercise 3. Loading Members and viewing Virtual data model

What this exercise is about

This exercise covers loading the InfoSphere MDM with Sample Data and viewing the database tables that store the Member Data, Derived Data, Algorithm and Buckets.

What you should be able to do

At the end of this exercise, you should be able to:

- Load a delimited file of Member data into the Virtual MDM
- View the Member data in the database
- View the Algorithm data in the database
- View the Buckets in the database
- View the Derived data in the database

Introduction

Now that we have our algorithm in place we need to load some test data to see how our buckets are performing and also to generate our weights.

Luckily we've been provided with a file that contains 450,000 members that we can load into our database.

Requirements

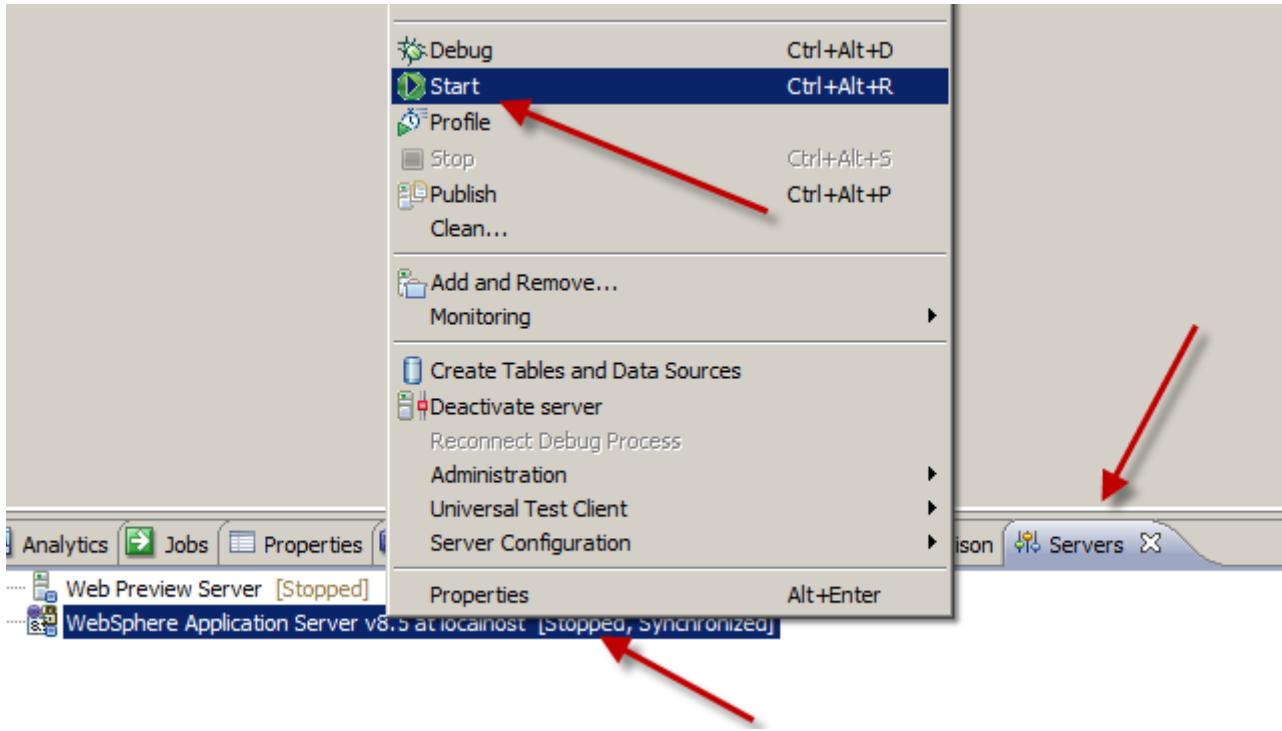
Each of these exercise depend on the previous exercise being complete. For this exercise you must have the data model and algorithms defined in the configuration file.

Exercise instructions

Part 1: Starting the InfoSphere MDM server

Before we can do anything, we must start the InfoSphere MDM.

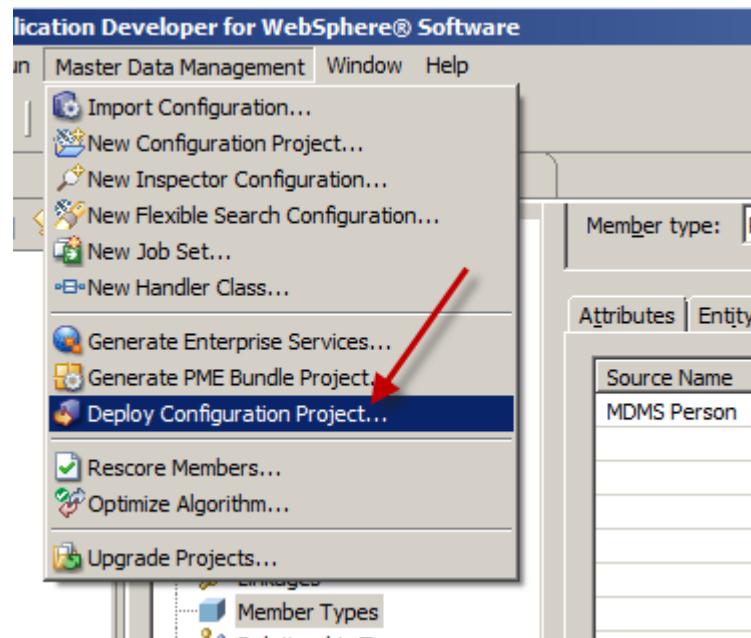
- 1. Inside RAD, open the **Servers** tab. Right click on the **WebSphere Application Server** and select **Start**.



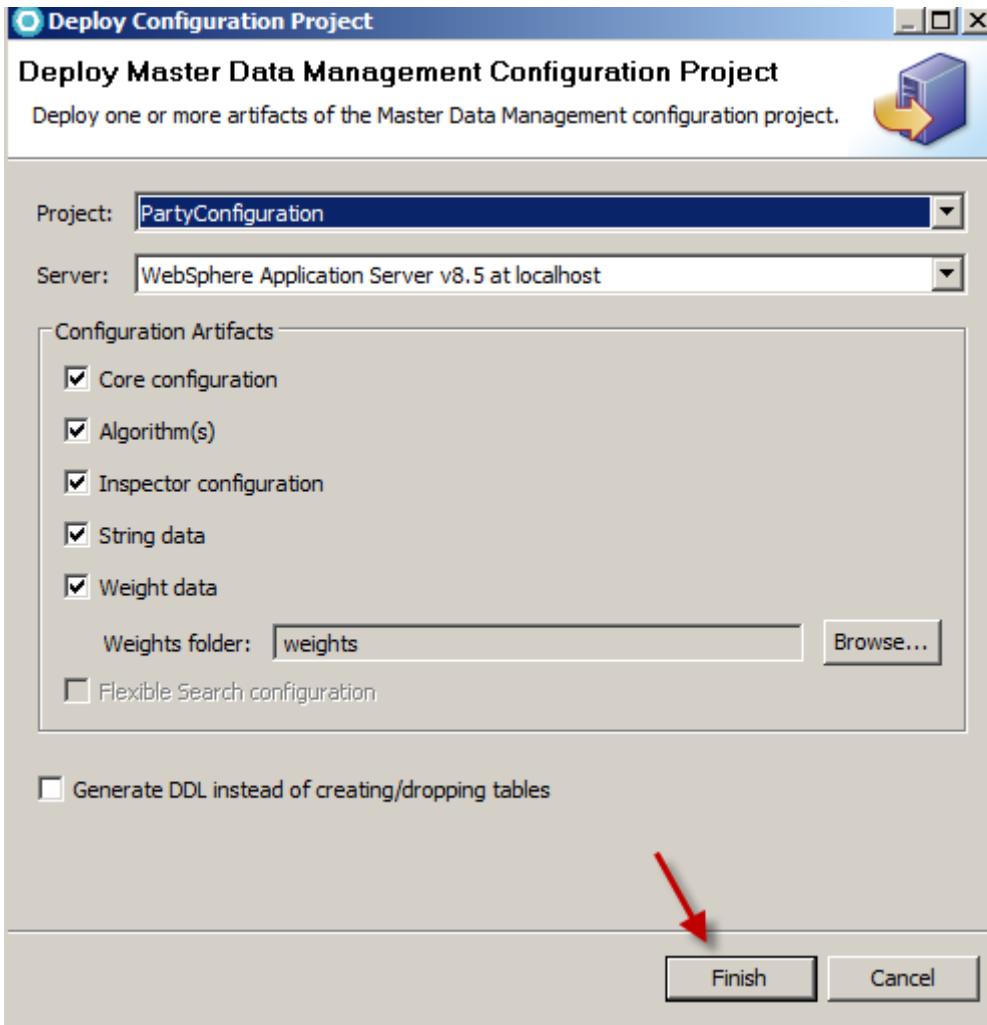
Part 2: Deploying the MDM Configuration

Although we configure the algorithms, it is not deployed (published to the MDM database tables). In this part we will deploy our configuration.

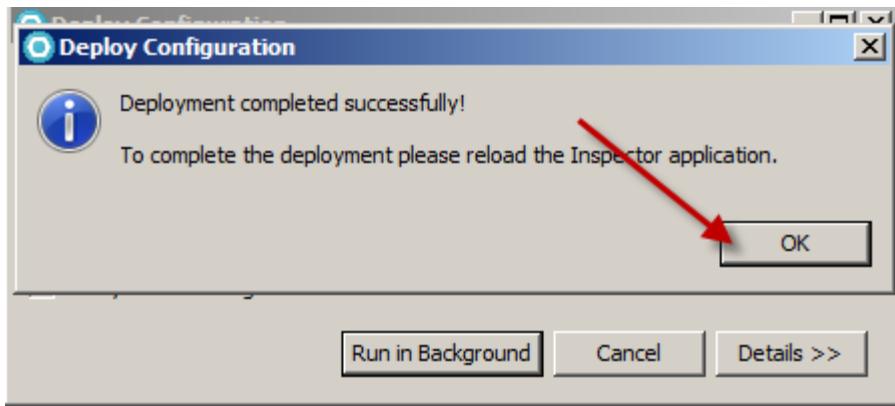
- 1. Once the server has started, we can deploy the algorithm to the InfoSphere MDM. From the RAD file menu, select **Master Data Management > Deploy Configuration Project ...** (NOTE: you need to be in the MDM Configuration perspective to see this menu option.)



2. Leave all Configuration Artifacts selected and click the **Finish** button.



3. The deployment will take several minutes, once it is complete, click the **OK** button.



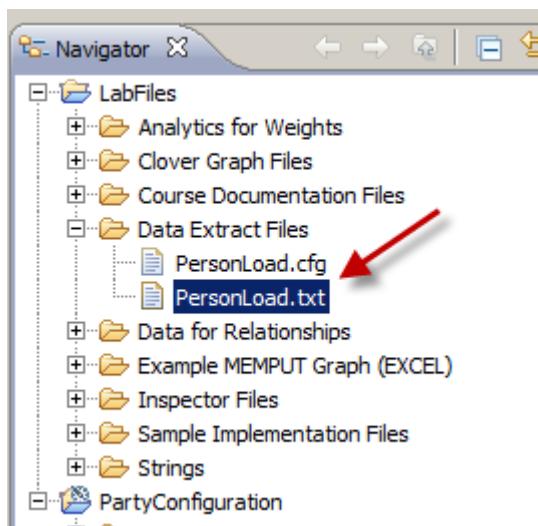
Part 3: Deploying Sample Data

Now that the configuration (including the algorithm) is deployed, we can load some sample data. Luckily we've already produced a data load file for you that will populate the InfoSphere MDM with sample data that will test our algorithms.

**Note**

When working with the algorithms, bucket analysis, weight generations, etc. You will want to have a sample data load of at least 250,000 records to provide proper analysis and weight generation. The higher the number the better and the sample set should be representative of the data that will be used in production.

- 1. To see the sample data file that we will use in this exercise, under the Navigator window, open **LabFiles > Data Extract Files > PersonLoad.txt** file. (NOTE: this file could be produced by any ETL tool, such as DataStage from a source database)



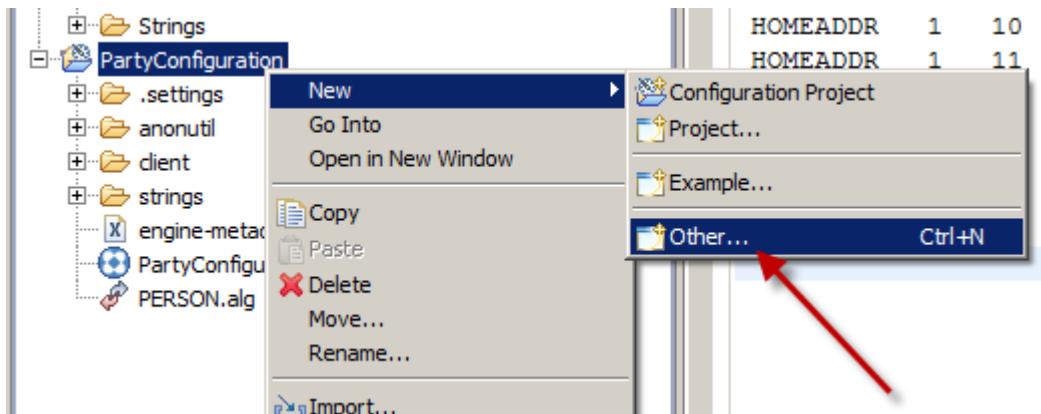
- 2. In the **PersonLoad.txt** you'll find 465,142 records delimited by a | character.

```
A|435202|M|BROSE|PERRY|O|III|1943-09-22|5520 BUCHANAN ST.|APACHE JUNCTION|AZ|85217|4801275588|480
A|435203|M|TEEPLE|DANIAL|I||1952-11-10|30829 SIXTEENTH ST.|TEMPE|AZ|85283|4803524457|4802837410|3
A|435204|F|HOWSE|ADELAIDE|W||1929-06-24|14553 TRIANON PLAZA|PHOENIX|AZ|00000|4803986314|480305002
A|435205|F|CORNETT|DIANN|R||1900-02-26|25298 BENTON ST.|APACHE JUNCTION|AZ|85217|4807259216|48072
A|435206|M|BLAYLOCK|BABYBOY|A||2006-09-02|16724 CARROLLTON ST.|GLENDALE|AZ|85318|||366929687
A|435207|M|SCHRAMEK|GREGORY|||1987-10-20|25694 CUMBERLAND STREET|LITCHFIELD PARK|AZ|85340|6231826
A|435208|M|WOLD|BABYBOY|L|III|2006-05-03|3872 OLIVER ST.|APACHE JUNCTION|AZ|85219|4801486228|4801
A|435209|F|BLASH|AILENE|A||1921-04-25|25621 FLOW COURT|PHOENIX|AZ|00000|9999999999||203992948
A|435213|F|GOREE|RAY|R||1945-03-09|16773 HENRY STREET|TEMPE|AZ|85287|4802036411|4802873309|233372
A|435214|M|LOAIZA|WARREN|A|II|2001-03-06|2527 ADAMSVILLE ROAD|MESA|AZ|85202|4801550526|4802025506
A|435217|M|JUHASZ|JULES|||1952-06-25|12616 PILIE ST.|PHOENIX|AZ|00000|||611841116
A|435218|M|GARLICK|SEBASTIAN|R||1926-06-09|10402 DUNCAN STREET|GLENDALE|AZ|85304|6232064926|62330
A|435219|M|WESTRICK|BABYBOY|S||1900-03-16|6569 RANELAGH STREET|PHOENIX|AZ|85061|4802655724|480206
A|435220|M|EUSTICE|BABYBOY|S||2007-03-17|16290 KENILWORTH ST.|GLENDALE|AZ|85318|6238657151|623831
A|435221|M|CROZIER|BABYBOY|O||1909-07-01|13569 AVENUE "A".|PHOENIX|AZ|85045|4802035924|4802457401
A|435222|F|SERNA|KRISTIAN|R||1959-10-19|4089 FRONT ST.|MESA|AZ|85212|9999999999||123456789
A|435223|M|HAGIN|BABYBOY|G||1900-07-01|30926 MONROE ST.|PHOENIX|AZ|85038|4801508330|4801038191|21
A|435224|F|HIGGENBOTHAM|BETHANY|G||1905-12-01|3692 HOME ST.|PHOENIX|AZ|85001|4804591107|480401730
A|435225|M|ROMAN|TIMMY|M||1947-10-01|26513 HIGH PARK STREET|PHOENIX|AZ|85030|4802886676||
```

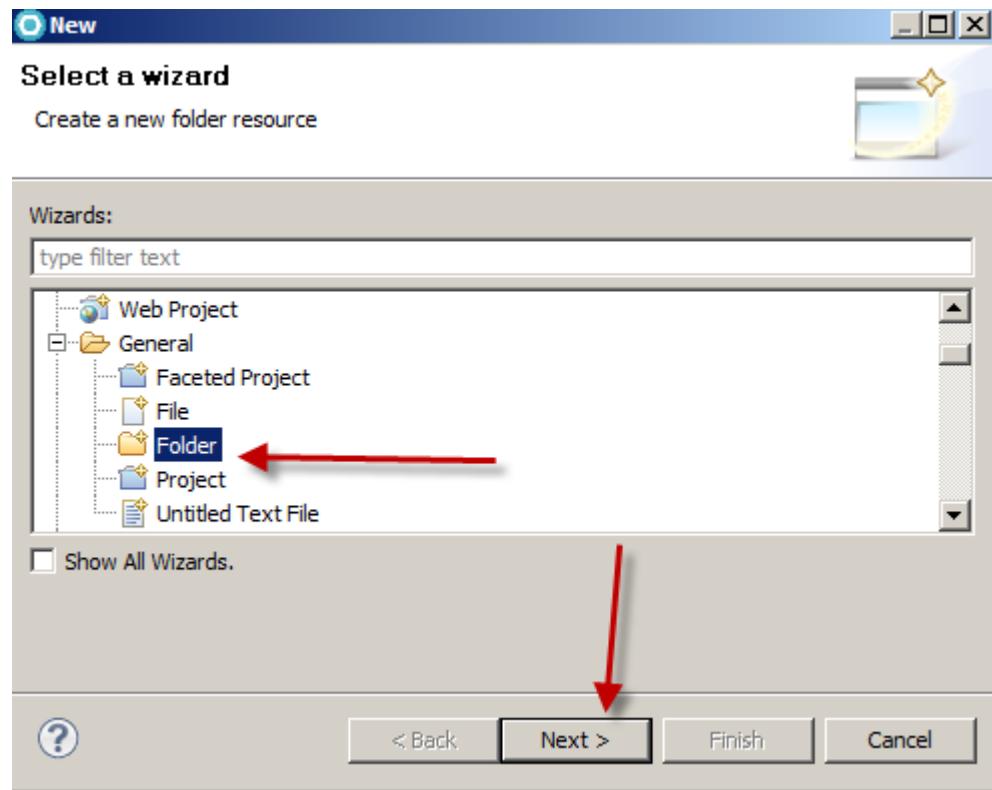
3. To find out which column represents which value in the delimited file, open the **PersonLoad.cfg** file found under the same directory. This file maps the field position to the data model attribute. We will use this file when loading our data.

# Attribute	#	Position	Transform	Assignment	Field	Constant
MEMHEAD	1	1	NA	SetString	srcCode	
MEMHEAD		1	2	BL	SetString	memidnum
SEX	1	3	BL	SetString	attrVal	
LGLNAME	1	4	BL	SetString	onmLast	
LGLNAME	1	5	BL	SetString	onmFirst	
LGLNAME	1	6	BL	SetString	onmMiddle	
LGLNAME	1	7	BL	SetString	onmsuffix	
BIRTHDT	1	8	BL	SetDate_Y4MD	dateVal	
HOMEADDR	1	9	BL	SetString	stLine1	
HOMEADDR	1	10	BL	SetString	city	
HOMEADDR	1	11	BL	SetString	state	
HOMEADDR	1	12	ZX	SetString	zipCode	
HOMEPHON	1	13	ZL	SetString	phnumber	
MOBILEPHON	1	14	ZL	SetString	phnumber	
SSN	1	0	NA	SetString	idIssuer SSA	
SSN	1	15	BL	SetString	idNumber	

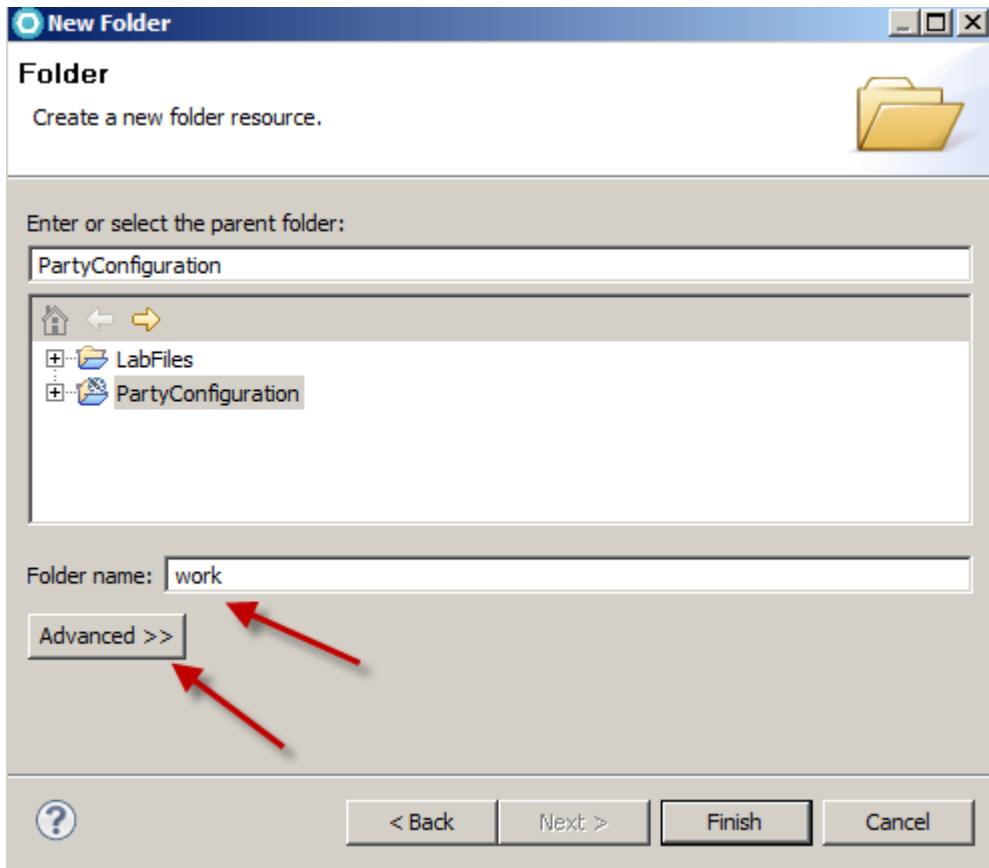
4. To load our data into the InfoSphere MDM, we will create a folder that is linked to the work directory of the Virtual MDM implementation. Right click on the **PartyConfiguration** project and select **New > Other ...**



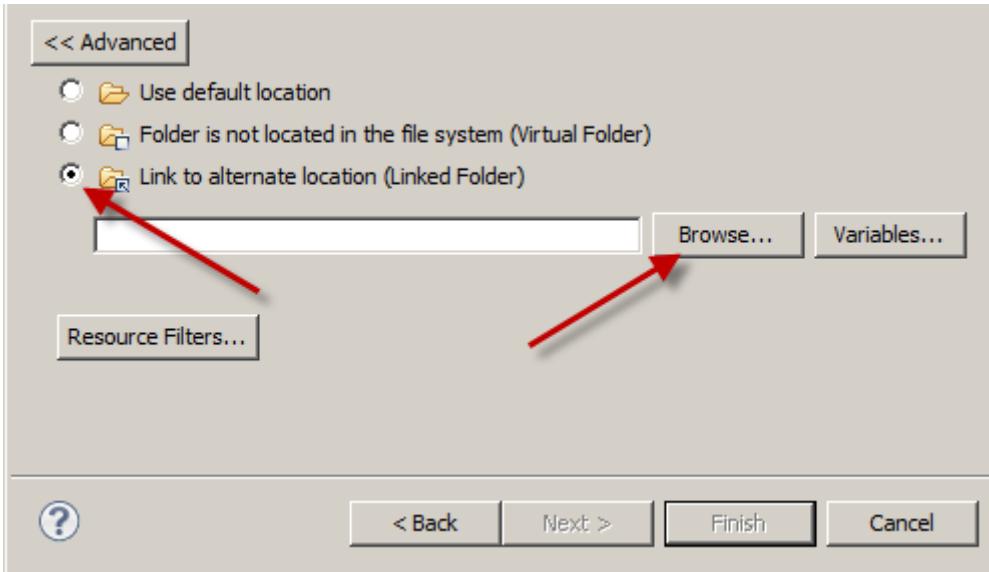
5. Select **General > Folder** and click the **Next >** button.



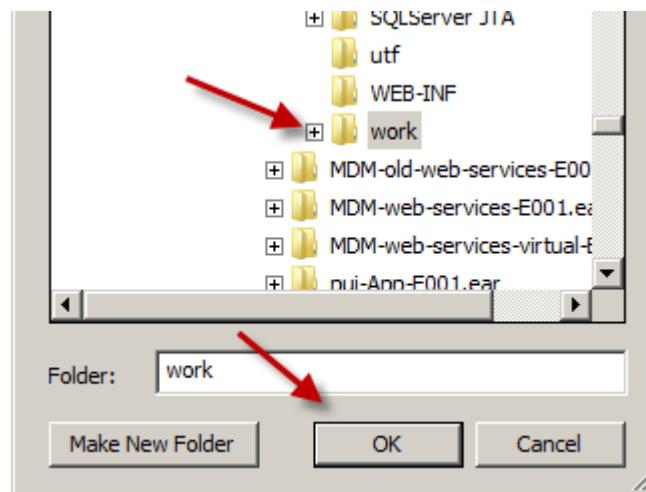
- 6. Enter **work** for the folder name and click the **Advanced >>** button.



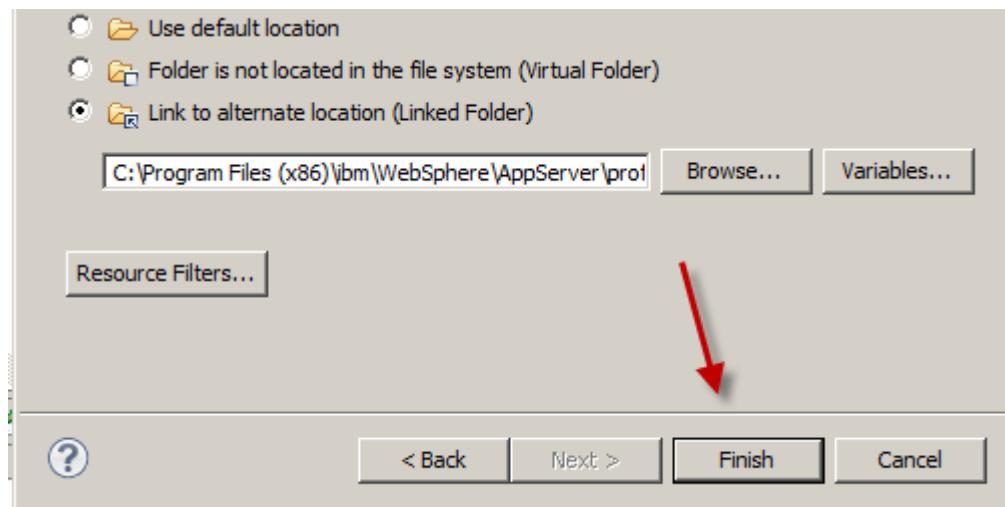
7. Select the **Link to alternate location (Linked Folder)** checkbox and click the **Browse** button.



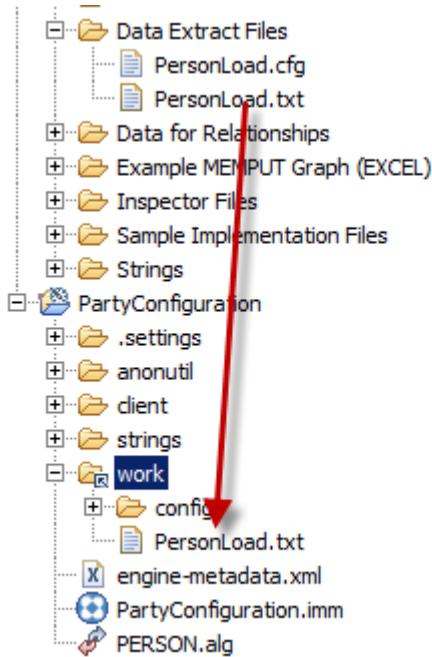
8. Navigate to: **C:\Program Files (x86)\ibm\WebSphere\AppServer\profiles\AppSrv01\installedApps\mdmCell\MDM-native-E001.ear\native.war\work\PartyConfiguration\work** and click the **OK** button. (NOTE: there are 2 work folders in the path)



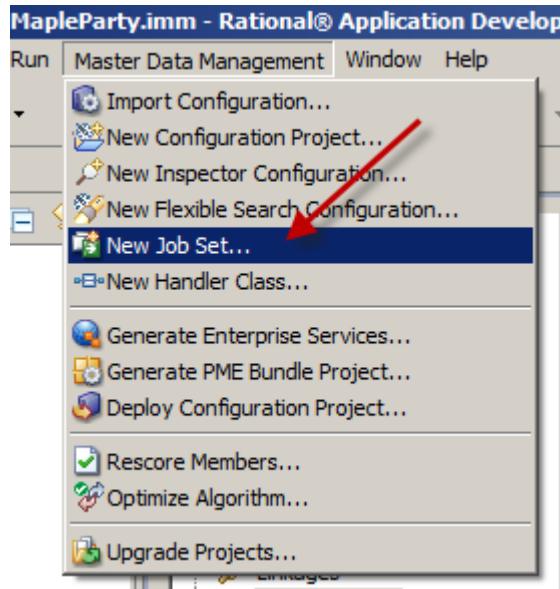
- 9. Click the **Finish** button to link the directory to your Workbench project.



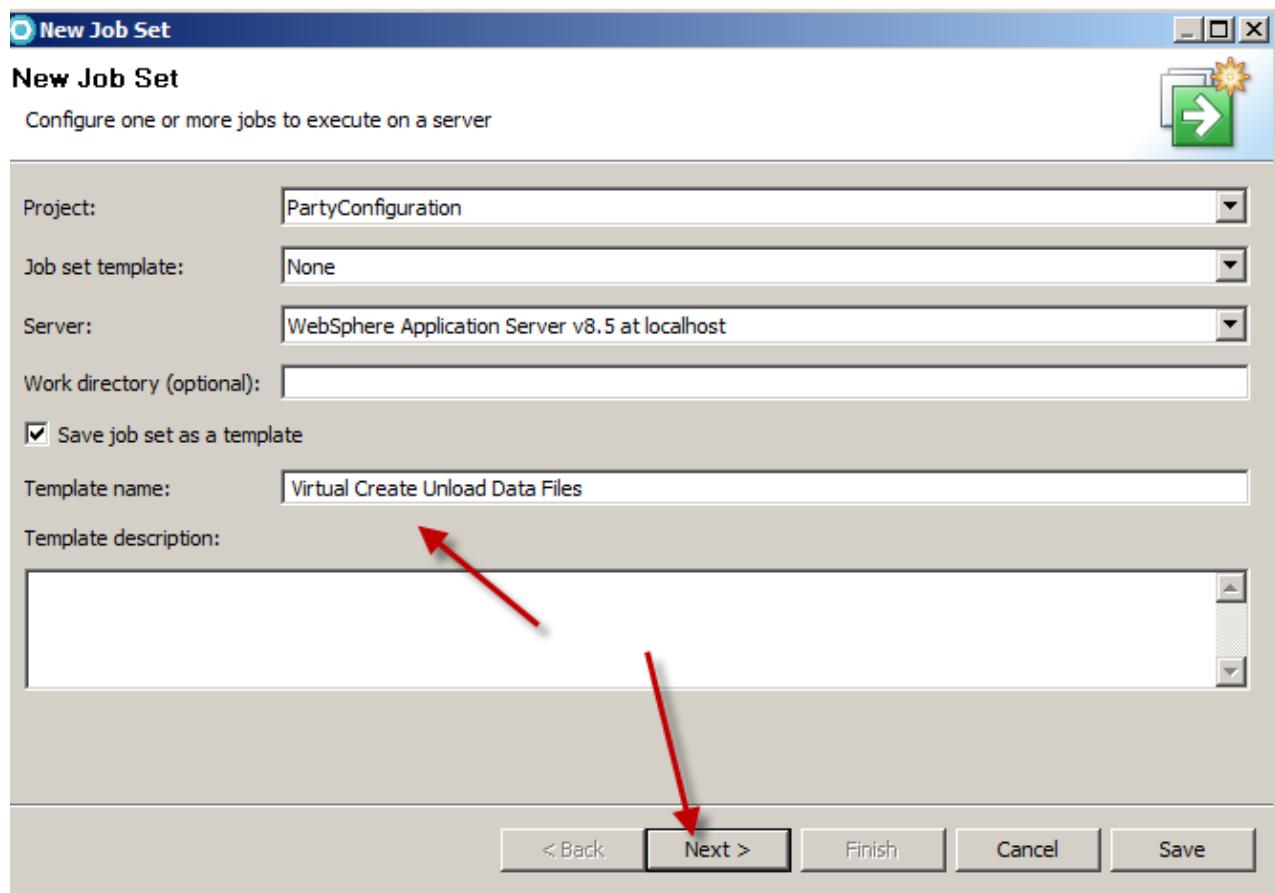
- 10. In order for MDM to access our data load, it must exist in the work path. In RAD, copy the **PersonLoad.txt** file into our work folder.



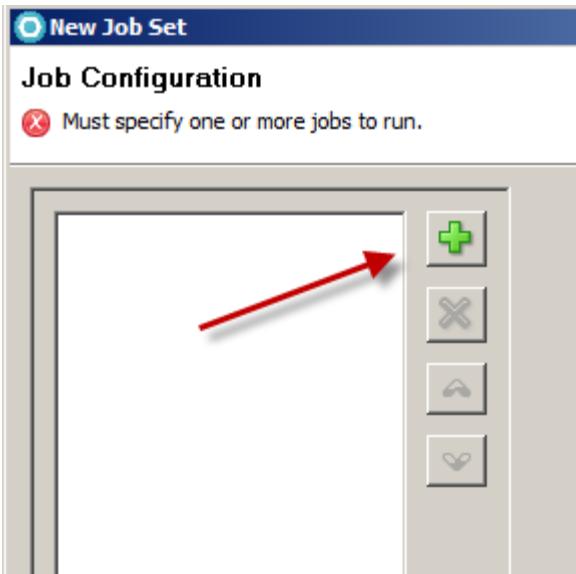
11. To load our data into the InfoSphere MDM, we will first create Unload files (these are files that are divided into a similar structure as the database tables. To create these files from our sample data file, we can run a create UNL job. From the RAD file menu, select **Master Data Management > New Job Set...**



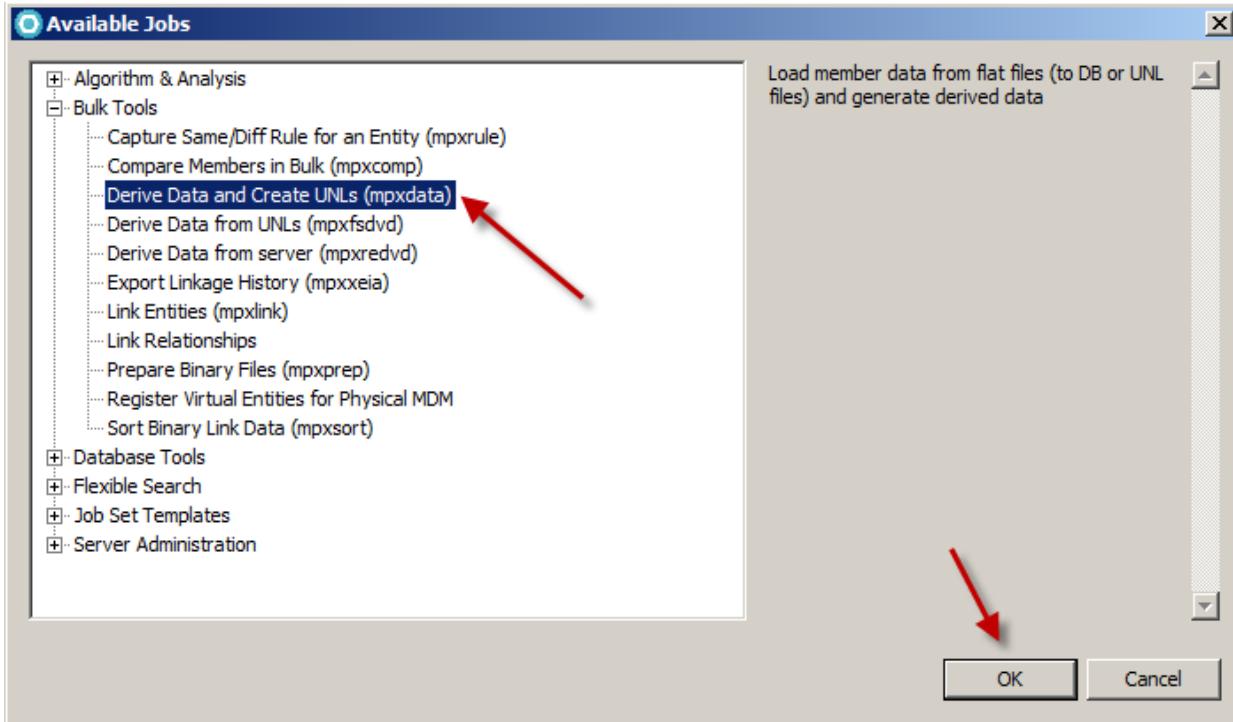
12. We will want to run this job a number of times, so we will save the job as a template. Select the **Save job as a template** checkbox and enter **Virtual Create Unload Data Files** as the Template name. Click the **Next >** button when complete.



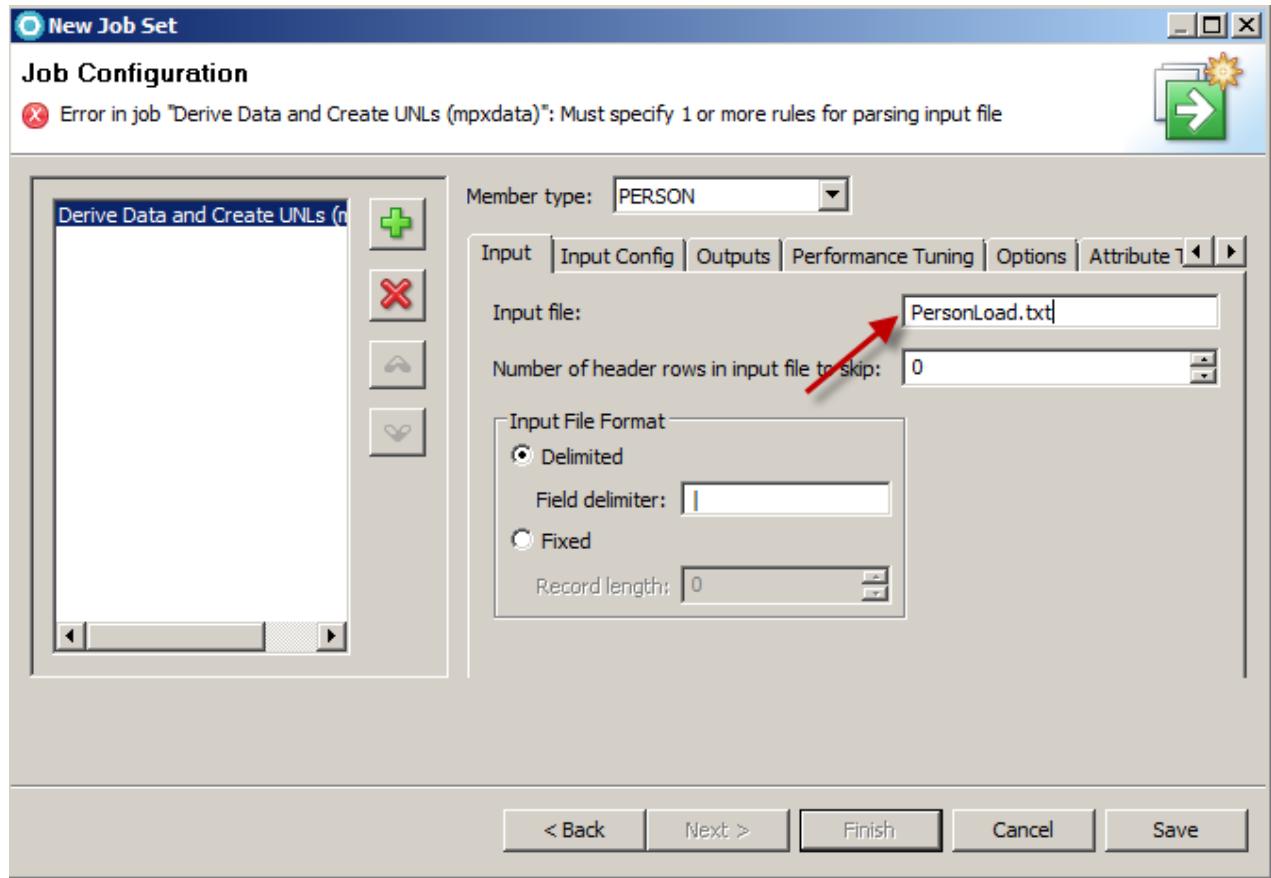
13. Inside the New Job Set window, click the green plus button to add a new job task.



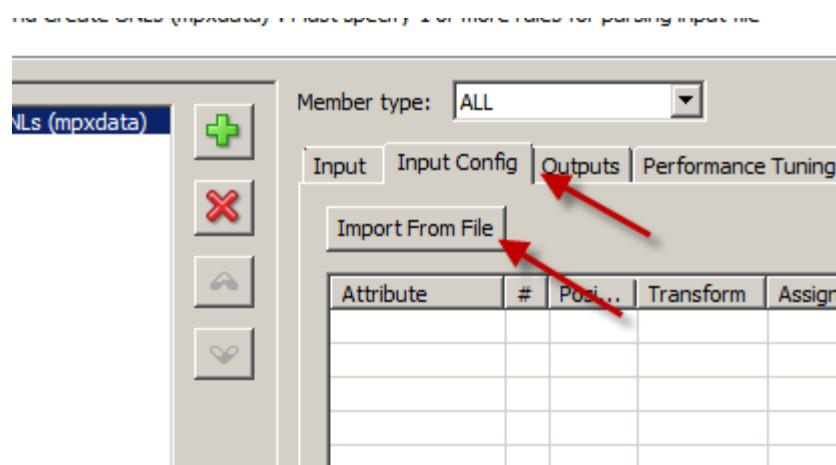
14. Select Bulk Tools > Derive Data and Create UNLs (mpxdata) and click the OK button.



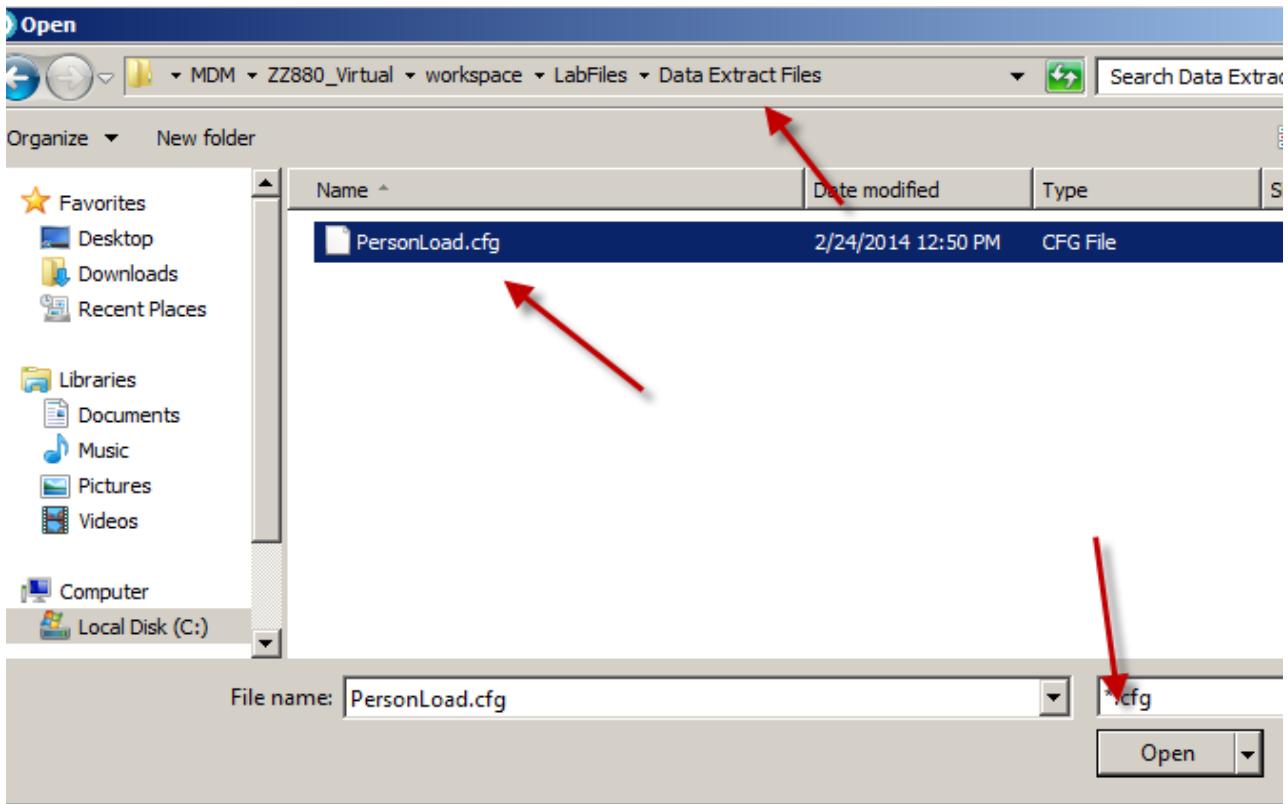
15. Under the Input tab, enter **PersonLoad.txt** for the Input file.



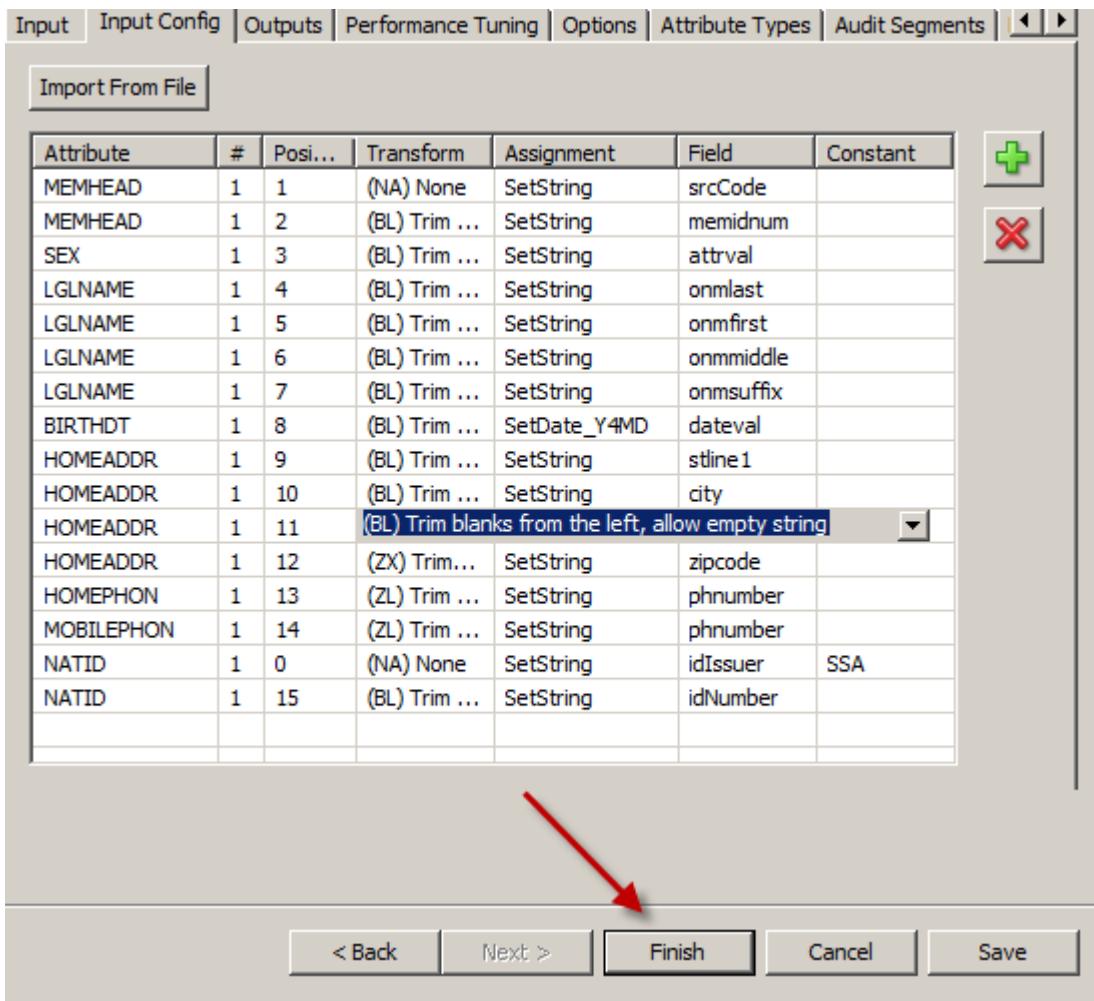
- 16. Select the Input Config tab and click the Import From File button.



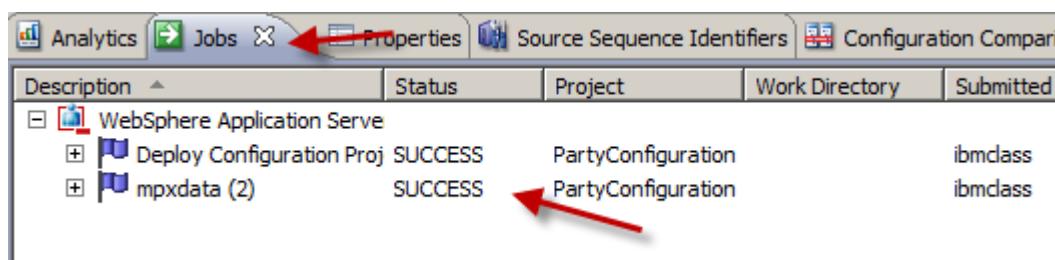
- 17. Navigate to C:\MDM\ZZ880_Virtual\workspace\LabFiles\Data Extract Files and select the PersonLoad.cfg file. Click the Open button.



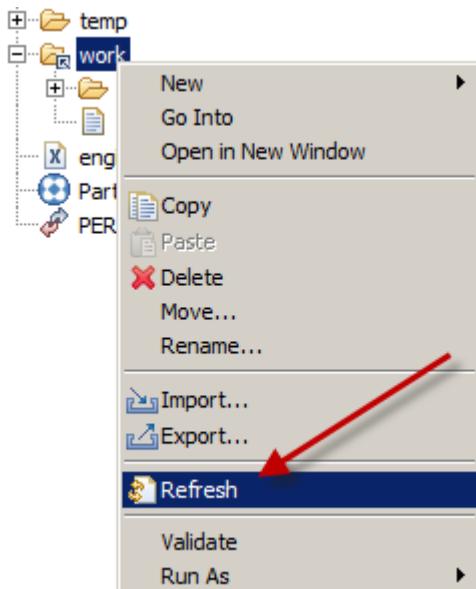
18. The configuration file will be imported into our Job. In this view you can see what each column in the configuration file represented. Click the **Finish** button to run the job.



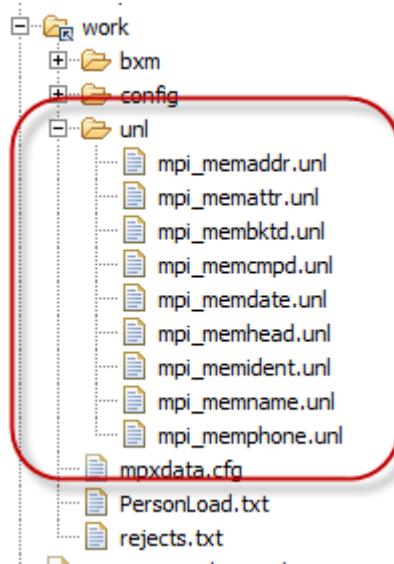
19. Open the **Jobs** window in RAD and view the status of the last job. Wait until the status changes to SUCCESS (NOTE: you may need to refresh the view from time to time using the refresh button.)



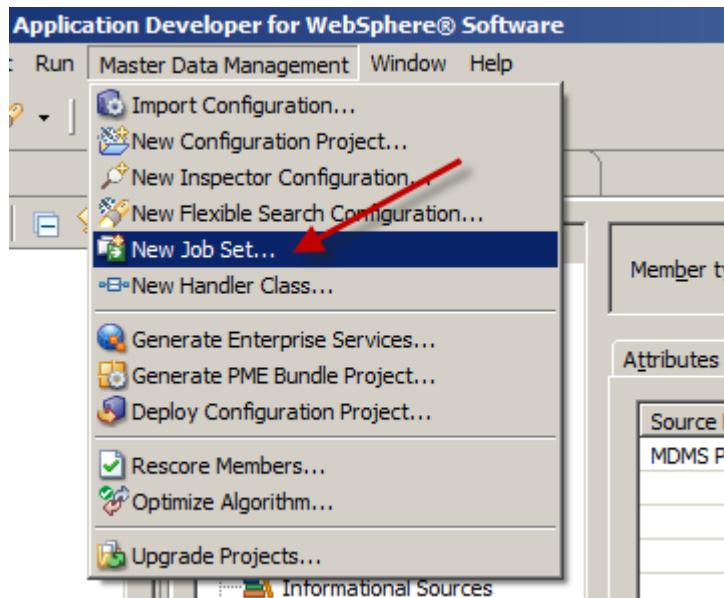
20. Once the job is complete you can check to see if it created the UNL files. Right click on the **work** directory and select **Refresh**



- ___ 21. You should see an unl folder that contains the unl files. If you open one you'll see the data that will be loaded into that table.



- ___ 22. Next we will load the UNL files into the InfoSphere MDM database. To do this we will run another job. From the RAD file menu, select **Master Data Management > New Job Set...**



- ___ 23. Select the **Save job set as a template** and name the template **Virtual Load UNLs to DB**. Click the **Next >** button.

New Job Set

New Job Set
Configure one or more jobs to execute on a server

Project: PartyConfiguration

Job set template: None

Server: WebSphere Application Server v8.5 at localhost

Work directory (optional):

Save job set as a template

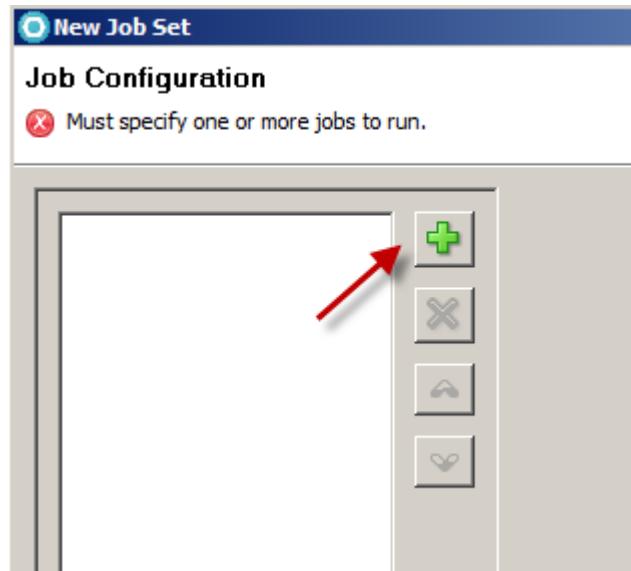
Template name: Virtual Load UNLs to DB

Template description:

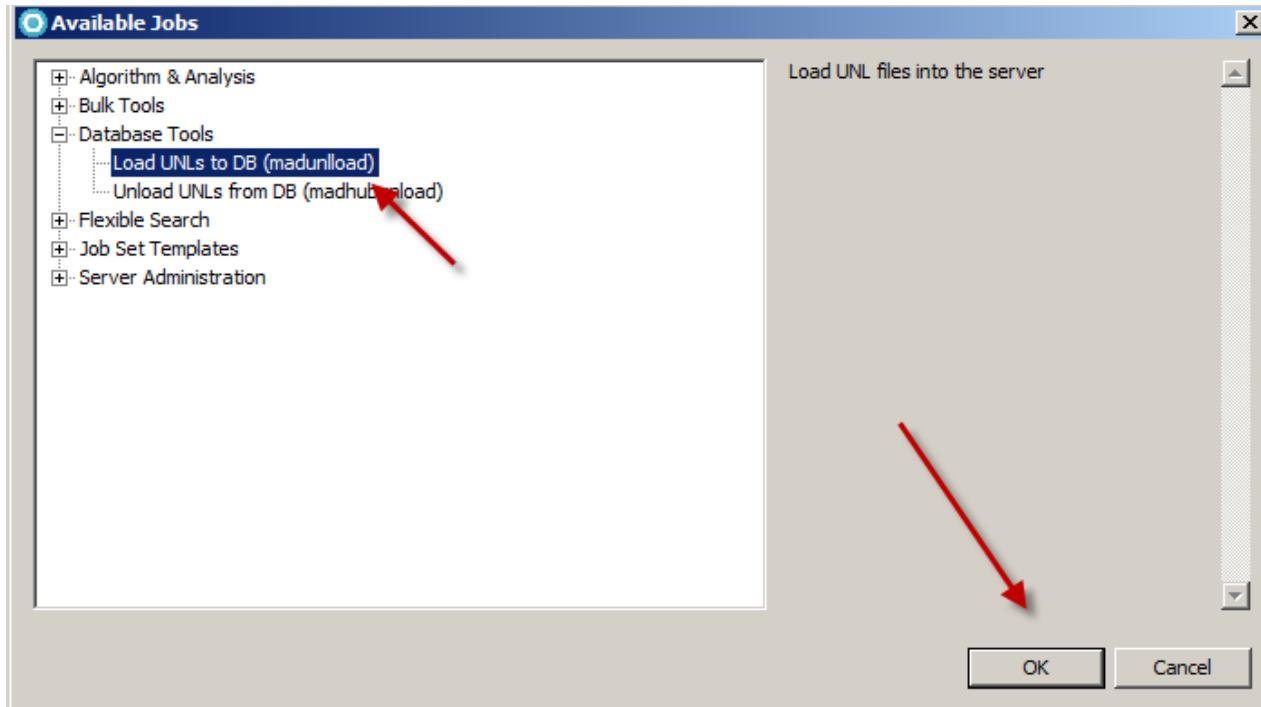
< Back Next > Finish Cancel Save

Red arrows point from the text descriptions to the 'Save job set as a template' checkbox, the 'Template name:' field containing 'Virtual Load UNLs to DB', and the 'Template description:' area.

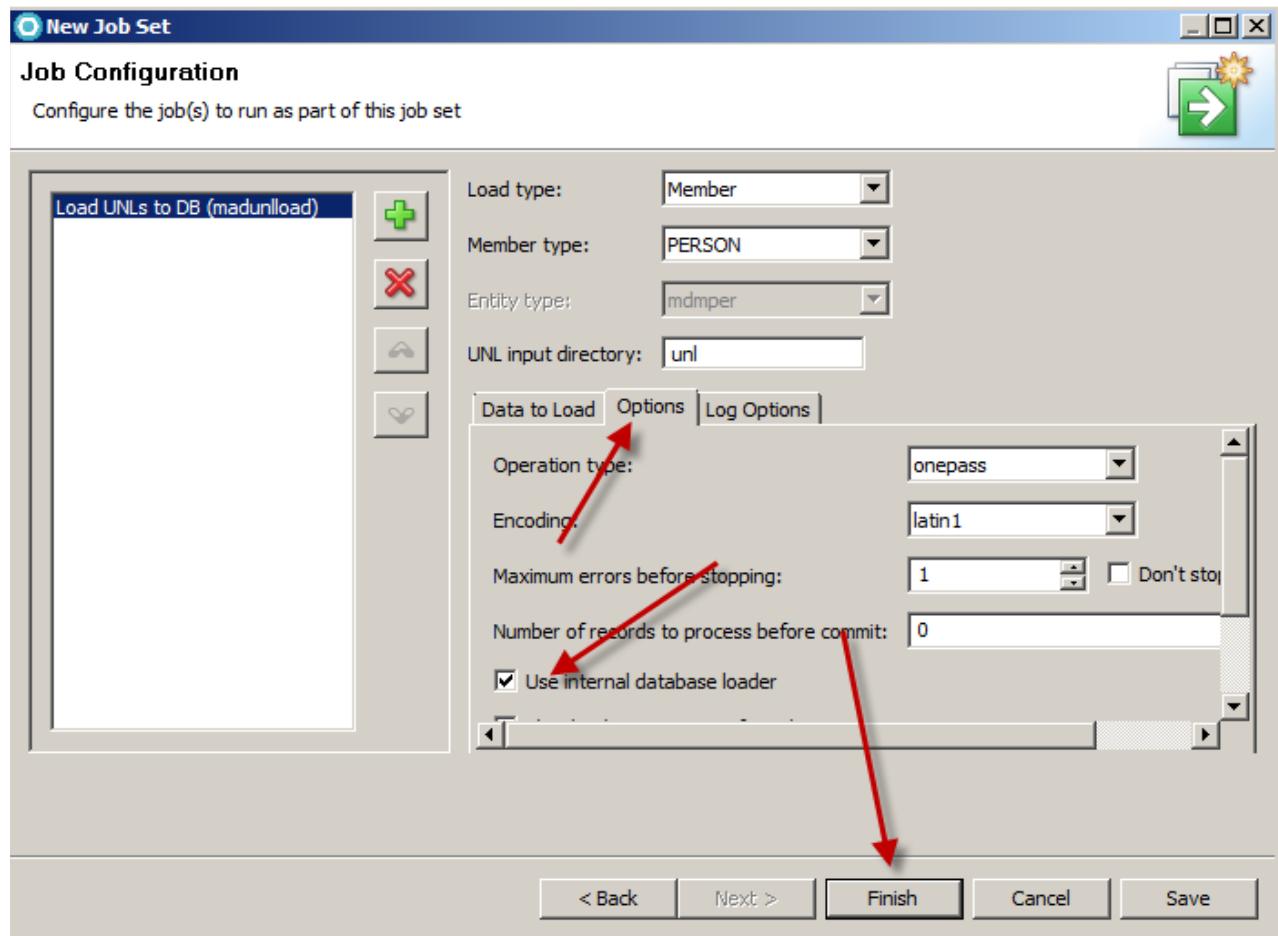
- ___ 24. Inside the New Job Set window, click the green plus button.



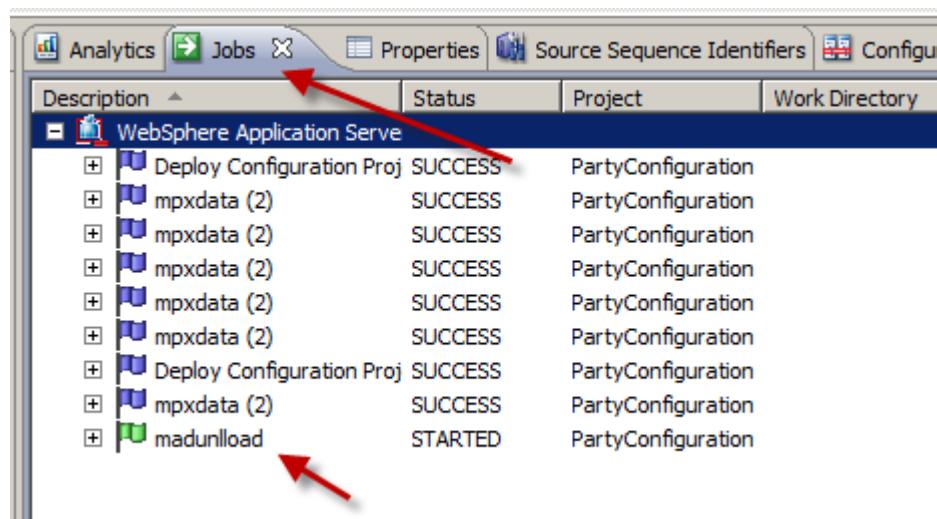
___ 25. Select Database Tools > Load UNLs to DB (madunload) and click the **OK** button.



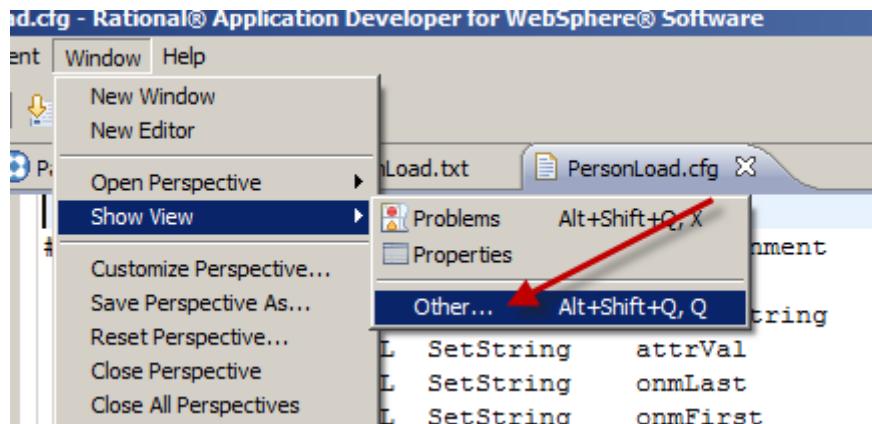
___ 26. Under the Options tab, select the **Use internal database loader checkbox** and click the **Finish** button



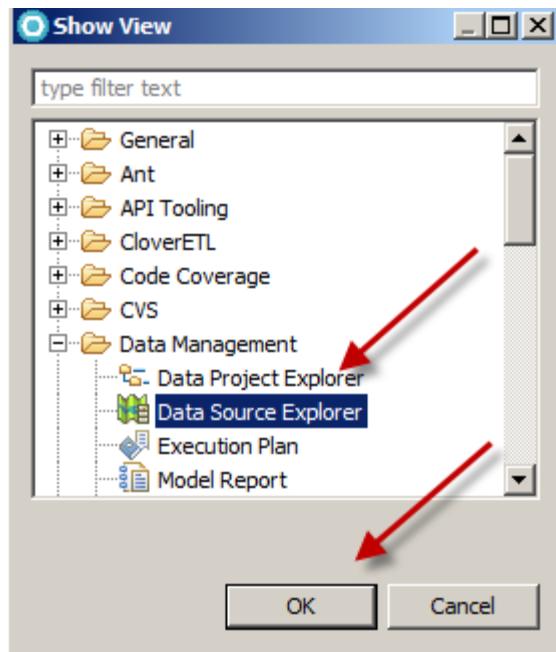
27. This job will take a little longer than the previous one as it is loading all the records (~500000 records) into the database, you can check the status under the **Jobs** window in RAD.



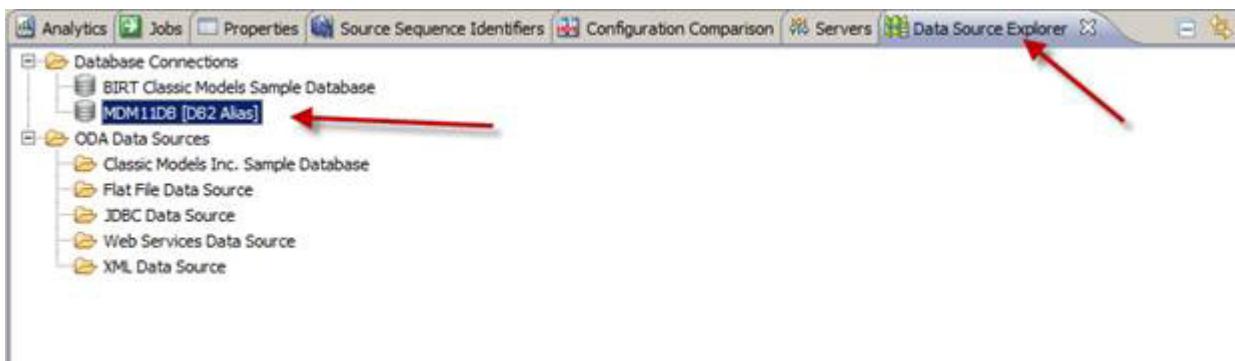
- 28. Once the job is complete, we can check the Virtual database table to see if the data was populated. To see the database, we will open the Data Source Explorer view. From the RAD file menu, select **Show View > Other ...**



- 29. Select **Data Management > Data Source Explorer** and click the **OK** button.

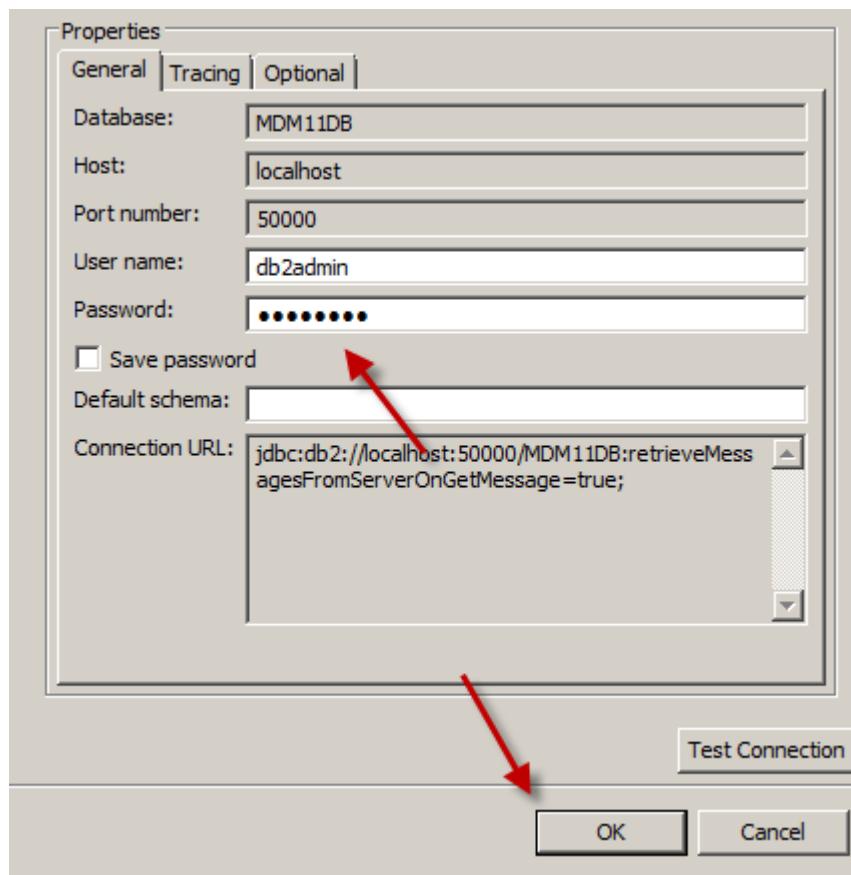


- 30. Under the Data Source Explorer window, double click the **MDM11DB**.

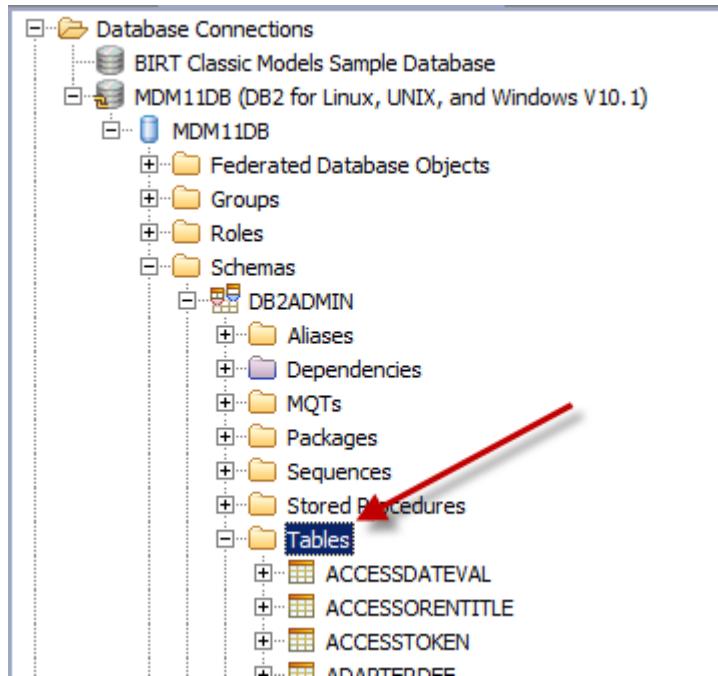


___ 31. Enter the following credentials and click the **OK** button:

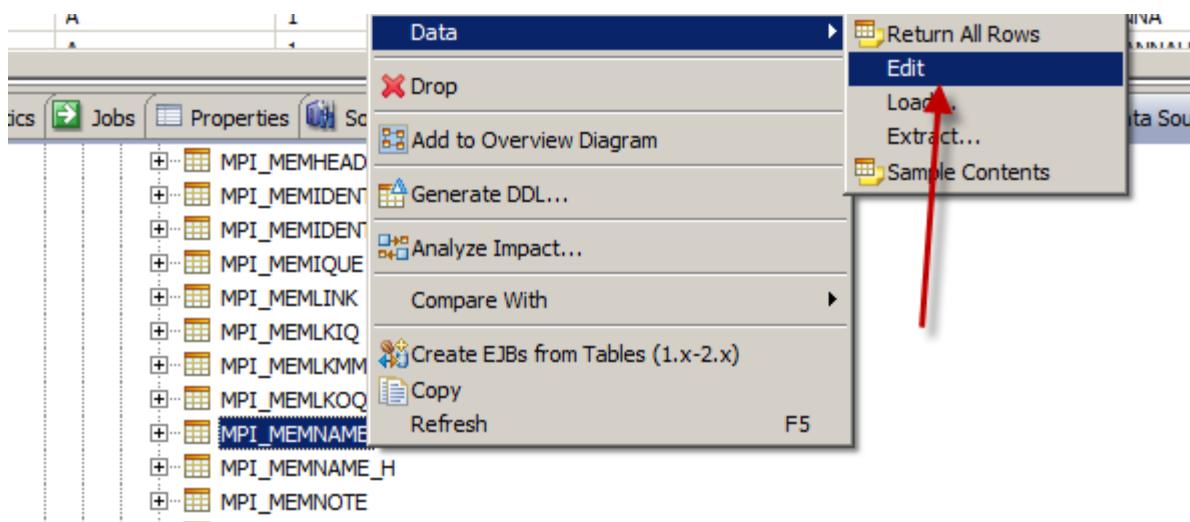
- ___ a. User name: **db2admin**
- ___ b. Password: **db3Admin**



___ 32. Under the **Data Source Explorer**, navigate to the Tables folder (under **MDM11DB > MDM11DB > Schemas > DB2ADMIN**).



33. Open the **MPI_MEMNAME** table (right click on table and select **Data > Edit**) to see if our records have been populated. If they are empty, there was an error loading the records (The Console tab in the RAD environment should provide more details if there was an error).



MEMRECCNO [BIGINT]	MEMSEQNO [SMALLINT]	CAUDRECCNO [BIGINT]	MAUDRECCNO [BIGINT]	RECSTAT [CHAR(1)]	ATTRRECCNO [SMALLINT]	ASAIDXNO [SMALLINT]	ONMLAST [VARCHAR(300)]	ONMFIRST [VARCHAR(300)]
374114	4	2	2	A	1	0	KEITHLEY	BRANDE
374115	5	2	2	A	1	0	LADAROLA	BENITO
374116	4	2	2	A	1	0	FOVIVES	RONALD
374117	3	2	2	A	1	0	WADA	NORMAND
374118	5	2	2	A	1	0	SPEISER	GORDON
374119	5	2	2	A	1	0	SORKIN	TEENA
374120	4	2	2	A	1	0	MONTETIME	NOEMI
374121	4	2	2	A	1	0	WEEDON	ALPHA
374122	4	2	2	A	1	0	DEANDRE	RAPOZO
374123	4	2	2	A	1	0	WACKER	JANINA
374124	5	2	2	A	1	0	MONTONE	TAD
374125	2	2	2	A	1	0	GANDHI	KATLYN
374126	5	2	2	A	1	0	YODES	ANDRIA
374127	4	2	2	A	1	0	SHIBBEFF	DANIELT

- 34. We can see some of the Algorithm tables that we deployed in the previous exercise. Open the **MPI_DVDXSTD** table. In this table you will see all the standardization functions that you created for the algorithm.

Standardization function					
IALLINT	STDFUNCCODE [VARCHAR(48)]	ATTR.CODE [VARCHAR(48)]	STDROLE1 [SMALLINT]	STDROLE2 [SMALLINT]	STDROLE1LABEL [VARCHAR(128)]
	IDENT1N	NATID	1	0	NONE
	ATTR	SEX	2	0	NONE
	PXNM	LGLNAME	3	0	Name Tokens
	PXNM	LGLNAME	4	0	Last Name token
	DATE1	BIRTHDT	5	0	Birth Date
	USZIP	HOMEADDR	6	0	NONE
	USADDR2	HOMEADDR	7	0	NONE
	PHONE1	HOMEOPHON	8	0	Home Phone
	PHONE1	MOBILEPHON	9	0	NONE

- 35. From the Standardization function, we move to the Comparison Roles. Open the **MPI_DVDXCMP** table. This table shows the link between the standardization function and the Comparison Roles. Notice that the last 2 standardization functions (for phones) go to the same Comparison Roles (9) as we defined in our algorithm.

I	DVDCODE [VARCHAR(48)]	DVSEQNO [SMALLINT]	STDROLE [SMALLINT]	CMPROLE [SMALLINT]	CMPROLELABEL [VARCHAR(128)]
	DVDPERSON	1	1	1	NONE
	DVDPERSON	2	2	2	NONE
	DVDPERSON	3	3	3	NONE
	DVDPERSON	4	4	4	NONE
	DVDPERSON	5	5	5	NONE
	DVDPERSON	6	6	6	NONE
	DVDPERSON	7	7	7	NONE
	DVDPERSON	8	8	8	NONE
	DVDPERSON	9	9	8	NONE

- ___ 36. For the Bucketing function, open the **MPI_DVDYBKT** table. In this table you'll find the Comparison Role map to the bucketing function.

DVDSEQNO [SMALLINT]	BKTFUNCCODE [VARCHAR(48)]	DVDGRP [SMALLINT]	CMPROLE [SMALLINT]	MINTOKENS [SMALLINT]	MAXTOKENS [SMALLINT]	GENFUNCCODE [VARCHAR(48)]
1	ATTR	1	1	1	1	SORTED
2	PXNM	2	3	1	2	EQMETA
3	PXNM	3	4	1	1	EQMETA
4	DATE	2	5	0	1	DTY4MM
5	ATTR	3	6	1	1	ASIS
5	ATTR	4	8	1	1	SORTED

- ___ 37. For the Bucket configuration, open the **MPI_DVDXBKT** table. In this table you'll find the 4 buckets that we defined in our algorithm.

DVDGRP [SMALLINT]	DVDGRPLABEL [VARCHAR(128)]	BKTROLE [BIGINT]	MINNWAY [SMALLINT]	MAXNWAY [SMALLINT]	MAXBKTFREQ [INTEGER]
1	NATLID	1	1	1	0
2	1 Name token + DOB	2	2	3	1000
3	Last Name token + ZIP	3	2	2	0
4	Phone (Home or Mobile)	4	1	1	0

- ___ 38. For our Comparison Functions, open the **MPI_CMPSPEC** table. This table connects the comparison roles to the comparison functions that we defined in the algorithm.

CMFUNKCODE [VARCHAR(48)]	CMPSPECODE [VARCHAR(48)]	ISRI [VARCHAR(4)]	CMPROLE1 [SMALLINT]	CMPROLE2 [SMALLINT]	CMPROLE3 [SMALLINT]
DR1D1B	NATLID	Y	1	0	0
EQVD	SEX	N	2	0	0
QXNM	NAME	N	3	0	0
DATE	DOB	N	5	0	0
AXP	AXP	N	7	8	0

- ___ 39. When we loaded the data into the InfoSphere MDM, the data would have gone through the standardization and bucketing. Open the **MPI_MEMCMPD** table to see the derived data from the standardization.

STD	MPI_DVDXCMP	MPI_DVDXBKT	MPI_DVDYBKT	MPI_CMPSPEC	MPI_MEMCMPD
BIGINT	SRCRECNO [SMALLINT]	CMPSEQNO [SMALLINT]	CMPVAL [VARCHAR(1020)]		
1	1		217824877^M^BROSE:PERRY:O..III^BROSE:::,^19430922^85217^N-5520:S-BUCHANAN:S-ST:.S-A		
1	1		321808574^M^TEEPLE:DANIAL:E..^TEEPLE:::,^19521110^85283^N-30829:S-SIXTEENTH:S-ST:.S-TI		
1	1		312212702^F^HOWSE:ADELAIDE:W..^HOWSE:::,^19290624^N-14553:S-TRIANON:S-PLZ:.S-PHO		
1	1		217581311^F^CORNELL:DIANNR:R..^CORNELL:::,^19000226^85217^N-25298:S-BENTON:S-ST:.S-A		
1	1		366929687^M^BLAYLOCK:A..^BLAYLOCK:::,^20060902^85318^N-16724:S-CARROLLTON:S-ST:.S-		
1	1		244571710^M^SCHRAMEK:GREGORY::,^SCHRAMEK:::,^19871020^85340^N-25694:S-CUMBERLAND		
1	1		240971960^M^WOLD:L..III^WOLD:::,^20060503^85219^N-3872:S-OLIVER:S-ST:.S-APACHE:S-JU		
1	1		203992948^F^BLASH:AILENE:A..^BLASH:::,^19210425^N-25621:S-FLOW:S-CT:.S-PHOENIX:S-AZ		

- 40. To see the buckets, open the **MPI_MEMBKTD**. The Bucket Hash won't mean anything to you by looking at it, however if you went through the table, you would find that some of the members are tied to the same bucket hash which means they belong to the same bucket and will be compared.

MEMRECNO [BIGINT]	SRCRECNO [SMALLINT]	BKTHASH [BIGINT]
156302	1	3231749844540364177
156302	1	2756343778986405013
156302	1	2775862724386657855
156302	1	4058249364839743773
156302	1	5706636600850225490
156302	1	5523824746154945159
156303	1	1612128873005708557
156303	1	3105415786586344941
156303	1	2317185105493555382
156303	1	3306033667858839616
156303	1	2374409101578215520
156303	1	3001813289623380262
156303	1	2873403548916321145
156303	1	3504094778535701585
156303	1	4803085451758838870
156303	1	4745522043994609175
156304	1	1377356059638141911
156304	1	2713008153695969233
156304	1	3446402512810464321
156304	1	3446377224043015468
156304	1	2585438920364552118
156304	1	2866122169605918728
156304	1	3395452466957435471
156304	1	3395440372329525150
156304	1	2505626898908762956
.	.	.

End of exercise

Exercise 4. Bucket Analysis

What this exercise is about

This exercise covers running some analysis on our buckets and fixing our algorithms based on our analysis.

What you should be able to do

At the end of this exercise, you should be able to:

- Running Bucket Analysis
- Modifying the algorithm to solve large buckets and members belonging to no buckets.

Introduction

In this exercise we will use the data we loading in the previous exercise to run a bucket analysis on our algorithm. It is important to run this step during any customization of the algorithm as buckets perform an important role in creating a subset of records we will be comparing.

A bucket that contain too many records will impact performance and buckets that do not contain two records that could potentially be matches could miss duplicates.

Requirements

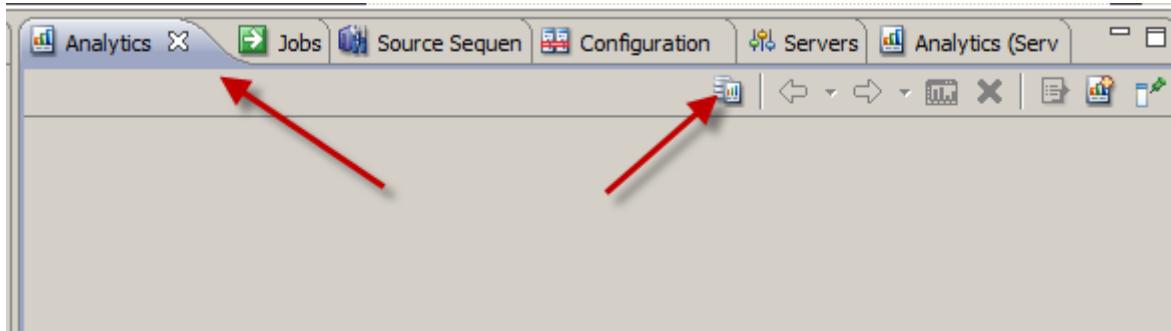
Each of these exercise depend on the previous exercise being complete. For this exercise you must have the data model and algorithms defined in the configuration file. The configuration must be deployed with sample data loading into the database.

Exercise instructions

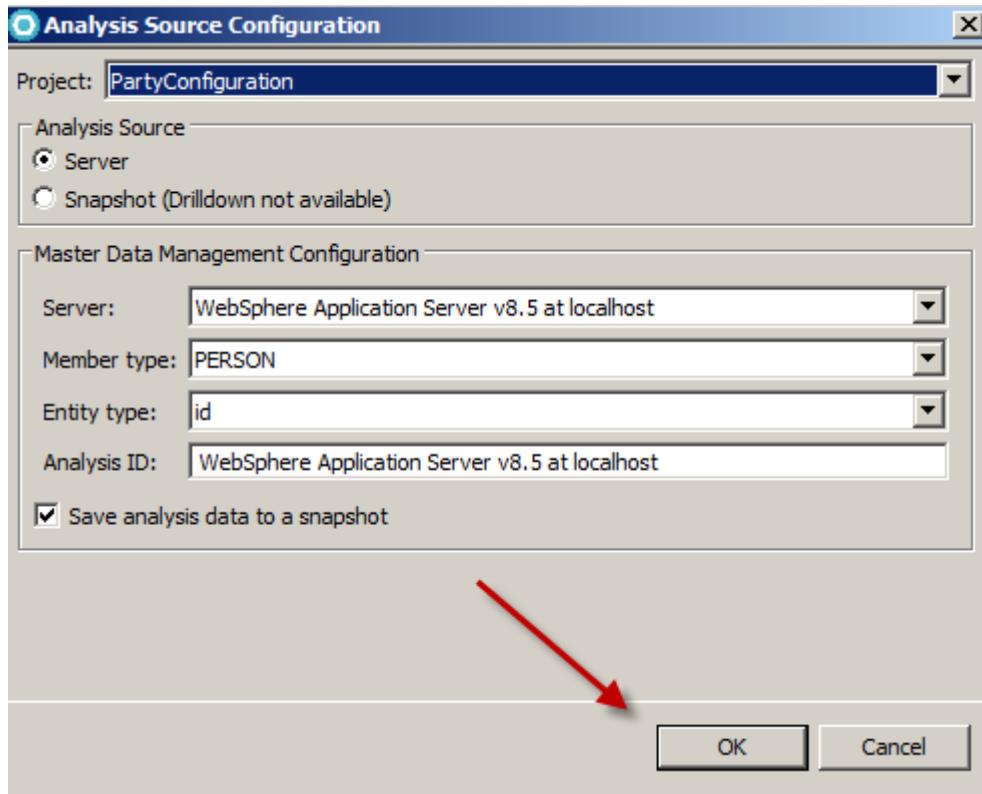
Part 1: Running Attribute Completeness

There are a number of analysis tools that we can use with the InfoSphere MDM, the first one we will look at is the Attribute Completeness report.

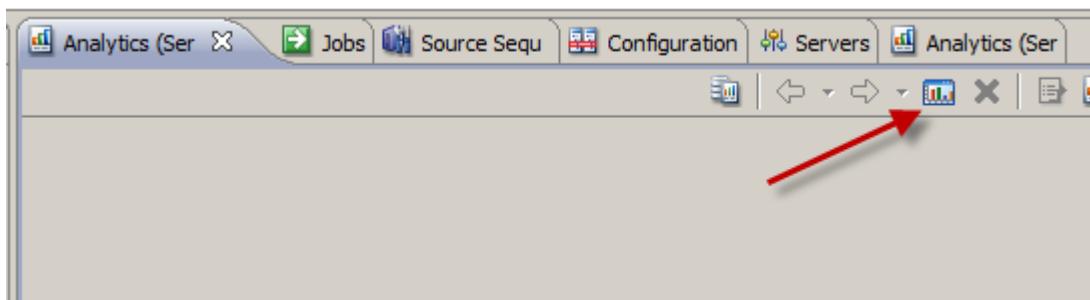
- ___ 1. Select the **Analytics** tab at the bottom of the RAD environment. Click the **Set the data source to use for the analysis view** button.



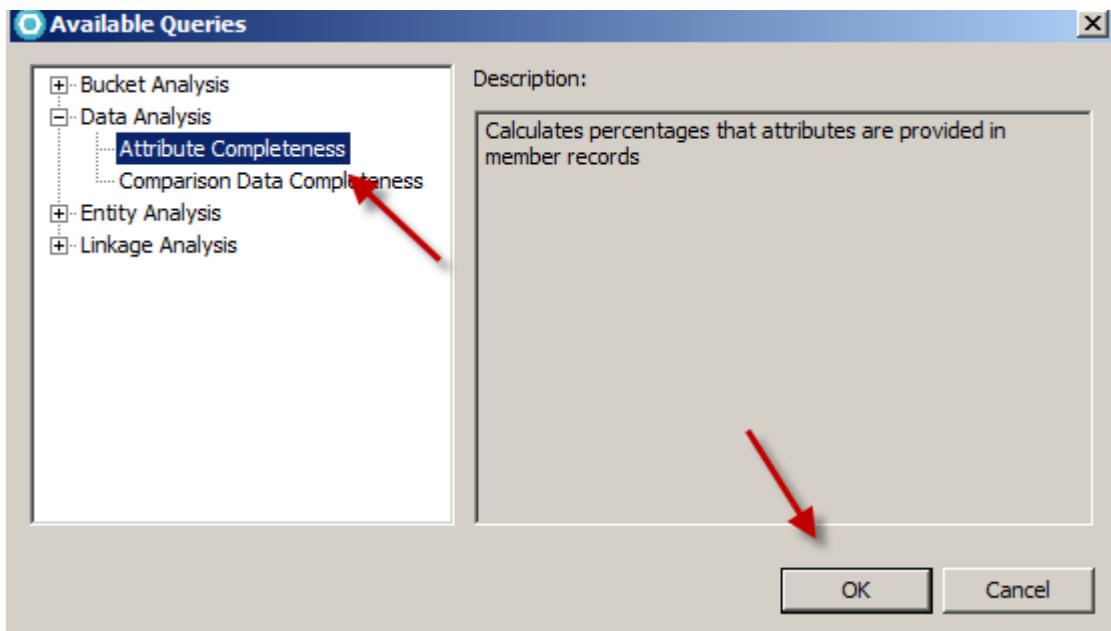
- ___ 2. Accept the default values for the source configuration and click the **OK** button.



- ___ 3. Now that the data source is configured, we can run some reports. Click the **Add a new query to the view for execution** button.



- ___ 4. Select **Data Analysis > Attribute Completeness** and click the **OK** button.



- ___ 5. Click the **Run Query** button.

	All Selected Sources	Archway (49%)	Bellwood (51%)
Birth Date	95	100	90
Gender	100	100	100
Home Address	91	100	83
Home Phone	95	100	90
Mobile Phone	80	96	65
Name	100	100	100
National ID Number	95	100	90

6. This report will check the percentage of complete attributes for each source, including eliminating the anonymous values that you added to your algorithm. You'll see in this report that the Archway source has more complete records and that the Mobile Phone is only 65% populated in the Bellwood source. (this could be valuable information when evaluating the sources for your composite view and also how you re-design your algorithms)

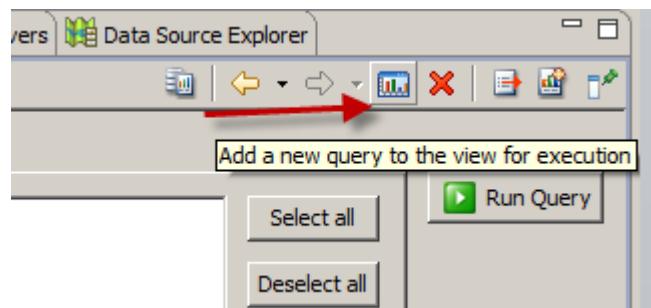
	All Selected Sources	Archway (49%)	Bellwood (51%)
Birth Date	95	100	90
Gender	100	100	100
Home Address	91	100	83
Home Phone	95	100	90
Mobile Phone	80	96	65
Name	100	100	100
National ID Number	95	100	90

Part 2: Running bucket analytics

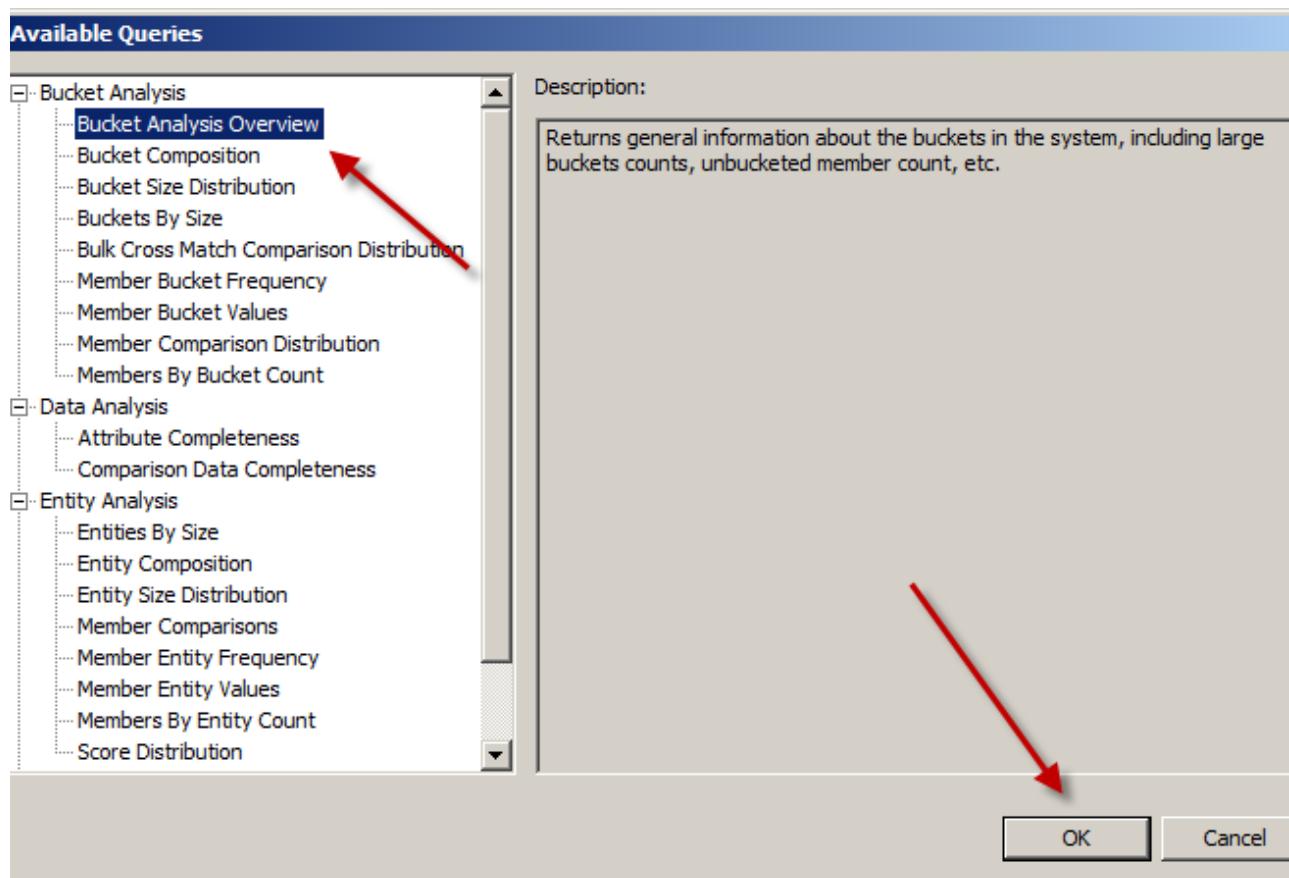
When we loaded the records into the Virtual MDM, the records would have been placed into our buckets that we defined and deployed to the Virtual MDM. We can now use this information to

analyze how well our buckets worked (e.g. did each record fall into one or more buckets, do we have buckets with only one record, do we have buckets with too many records).

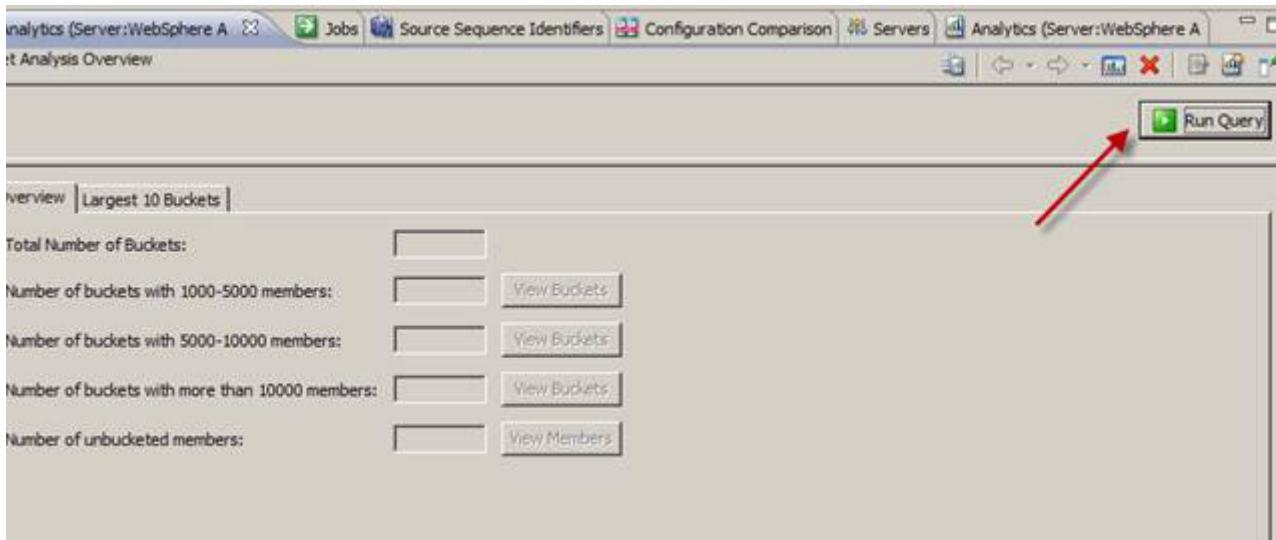
- ___ 1. Click the **Add a new query to the view for execution** button.



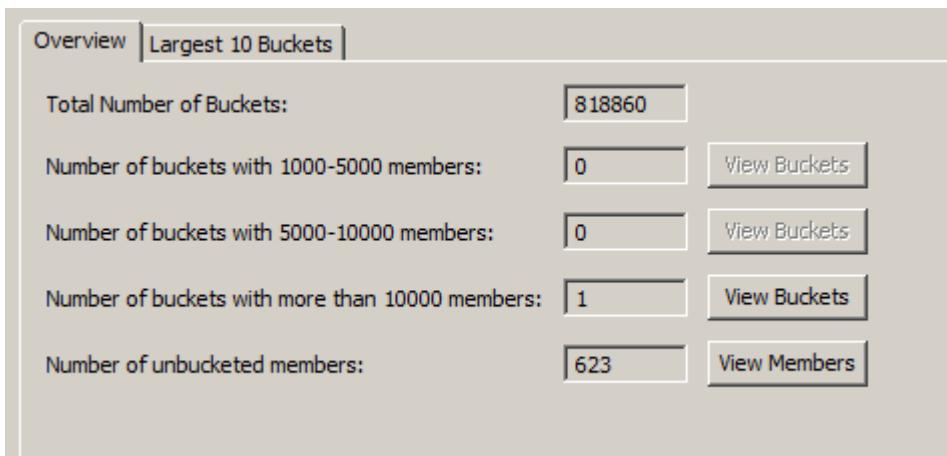
- ___ 2. Select **Bucket Analysis > Bucket Analysis Overview** and click the **OK** button



- ___ 3. This report will show us the number bucket and largest bucket details. Click the **Run Query** button.



- 4. In this report, you will see that we have one bucket that contains over 10000 members (this is a concern as this will impact performance) and that 623 member do not even belong to a bucket (meaning they are not compared to any members and cannot be searched.)



- 5. Click the **View Buckets** button to see the details of the bucket that contains over 10000 members.

The screenshot shows a user interface for managing buckets. At the top, there are two tabs: 'Overview' and 'Largest 10 Buckets'. The 'Overview' tab is selected. Below the tabs, there are five data rows:

- Total Number of Buckets: 818860
- Number of buckets with 1000-5000 members: 0
- Number of buckets with 5000-10000 members: 0
- Number of buckets with more than 10000 members: 1
- Number of unbucketed members: 623

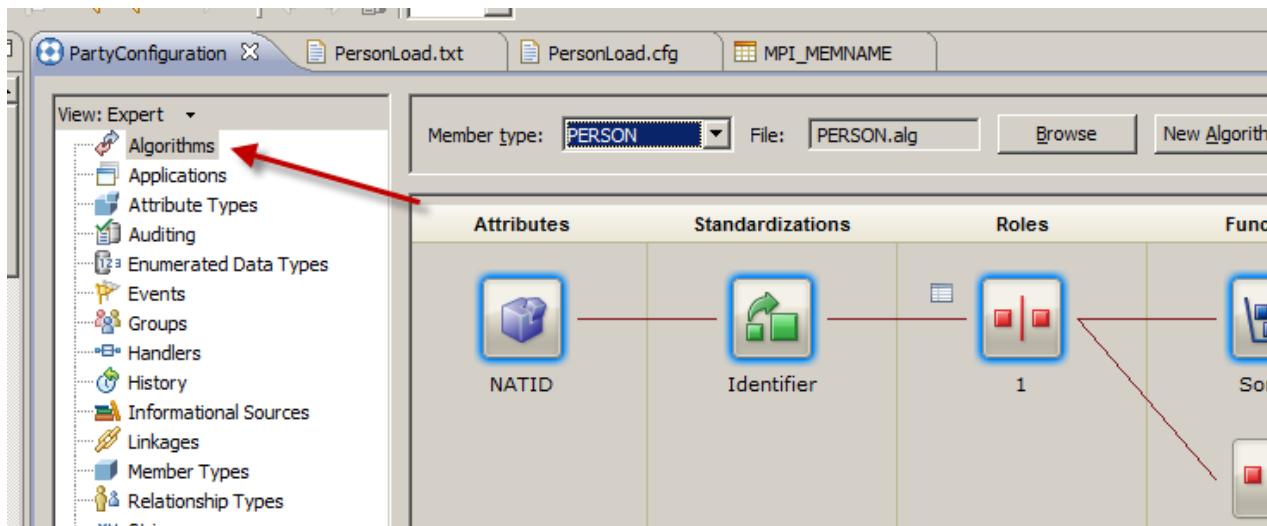
Each row has a 'View Buckets' button to its right. A red arrow points to the 'View Buckets' button for the row where 'Number of buckets with more than 10000 members' is 1.

- 6. We can see that the value that caught all these member in the bucket is 9999999 (obviously an anonymous value). It references **Bucket Role 4**, this is defined in our algorithm.

The screenshot shows a 'Query Parameters' section at the top with three input fields: 'Minimum bucket size' set to 10000, 'Maximum bucket size' set to 0, and 'Maximum # of results to return' set to 0. Below this, a message '(Found 1 buckets)' is displayed. A table follows, showing the results of the query:

Number of Members	Bucket Hash	Bucket Role	Bucket Value
13534	5163446155826240785	4	9999999

- 7. To see the bucket function that is causing this issue, from the **PartyConfiguration** file, select the **Algorithms** tab.

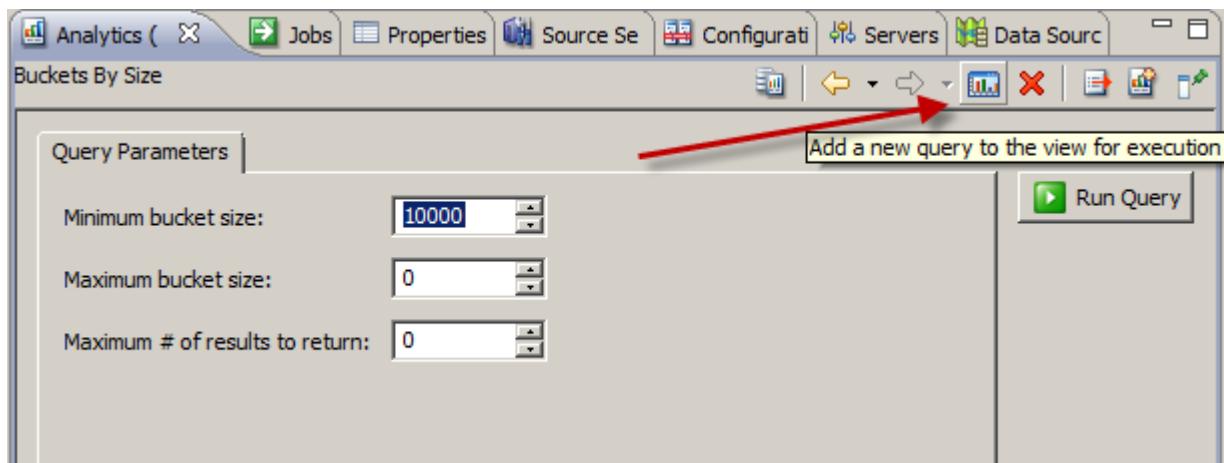


8. Select the Bucketing Group icons in the algorithm and using the RAD Properties window find the **Bucketing Group 4**.

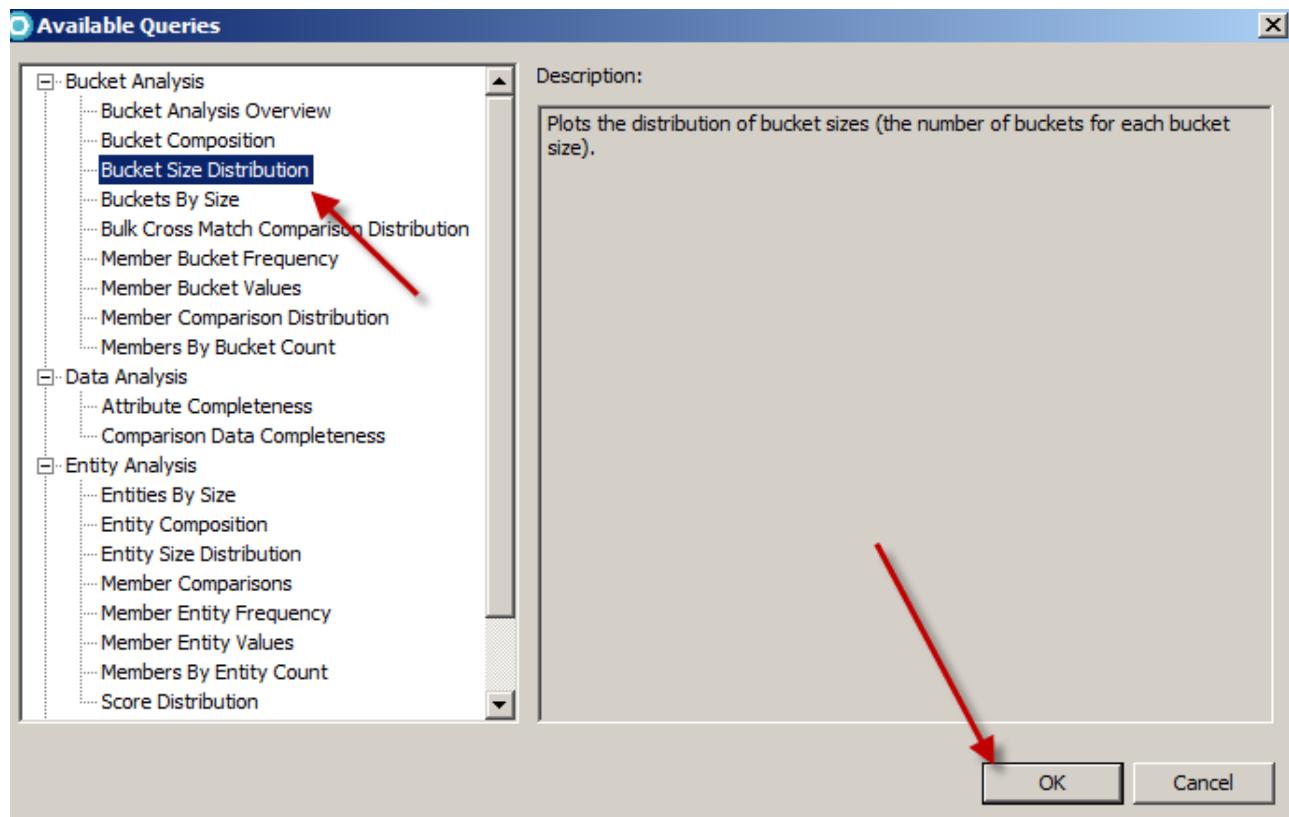
Property	Value
Bucketing role	4
Derivation group	4
Description	Phone (Home or Mobile) to build
Label	Phone (Home or Mobile)
Maximum attribute tokens	1
Maximum bucket size	0
Minimum attribute tokens	1

We know from this information, that many members have 9999999 as their Home or Mobile phone number (definitely an anonymous value)

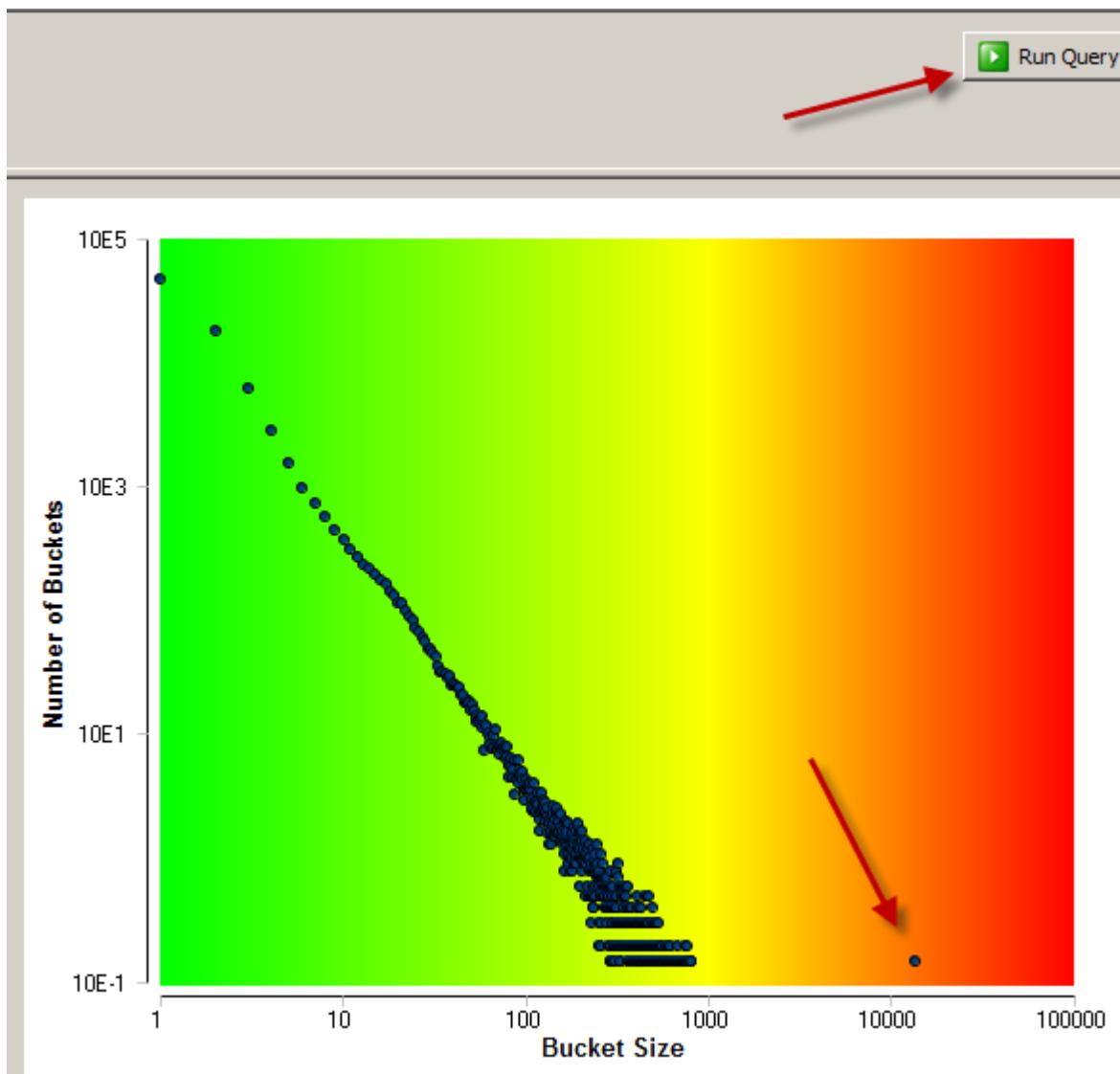
9. To see the large bucket information in a different visual, go back to the Analytics window and click the **Add a new query** to the view for execution button.



- ___ 10. Select **Bucket Analysis > Bucket Size Distribution** and click the **OK** button.



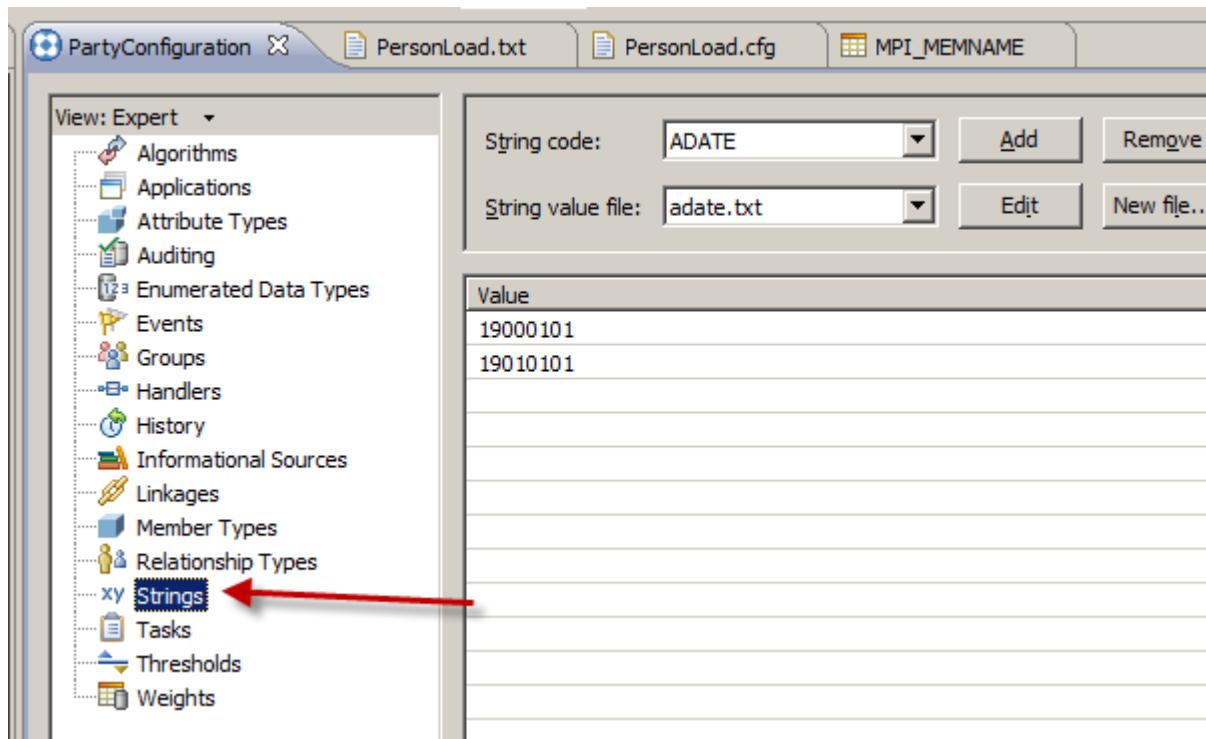
- ___ 11. Click the **Run Query** button. Notice the distribution with one bucket way off to the right.
Any bucket in the red is a concern.



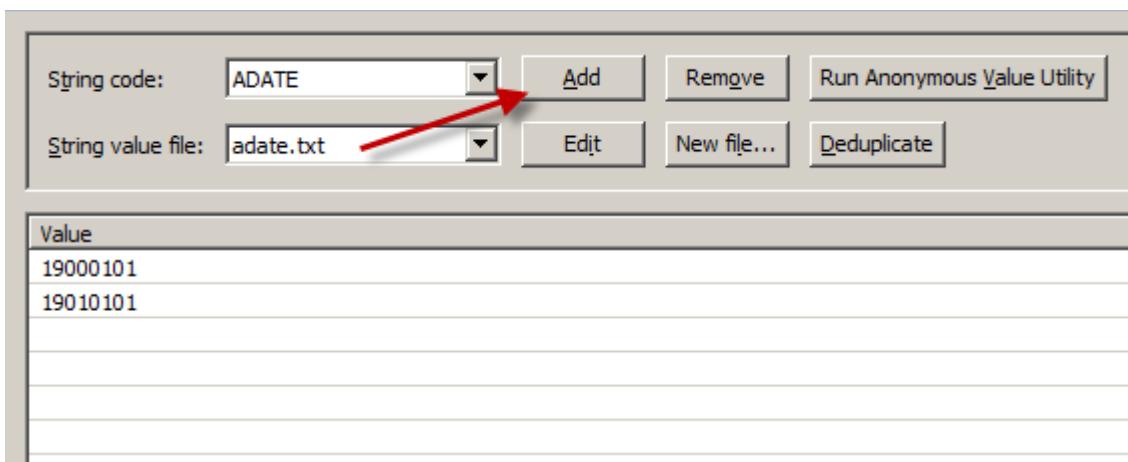
Part 3: Configuration modification

With our current configuration, there is one bucket with more than 10000 members. When we inspected the bucket we further noticed that there are anonymous phone numbers of 9999999 associated with the bucket. Now we are going to make some configuration changes to solve this anonymous phone problem.

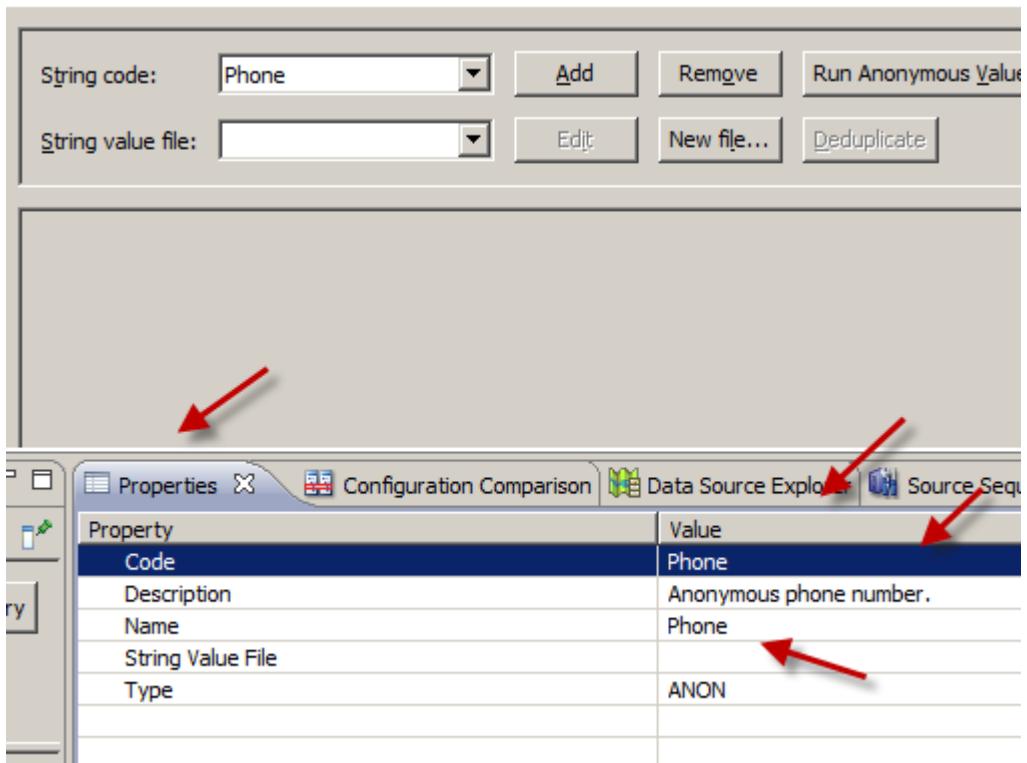
1. Back in the **PartyConfiguration**, select the **strings** tab.



- 2. Click the **Add** button to the right of the String code field to add a new String code.



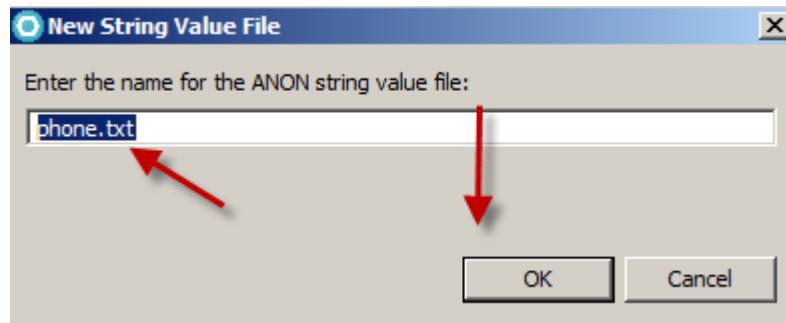
- 3. Open the RAD **Properties** window and enter the following values:
- Code / Name:** Phone.
 - Description:** Anonymous phone number.
 - Type:** ANON.



- 4. In the Editor pane, click the **New file...** button to the right of the String value file field to create a new String file for these new Phone anonymous values



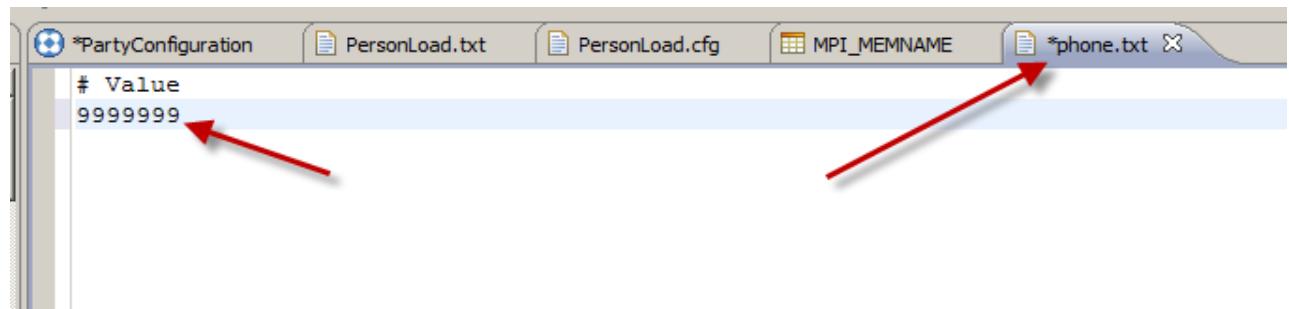
- 5. Enter **phone.txt** as the file name and click the **OK** button



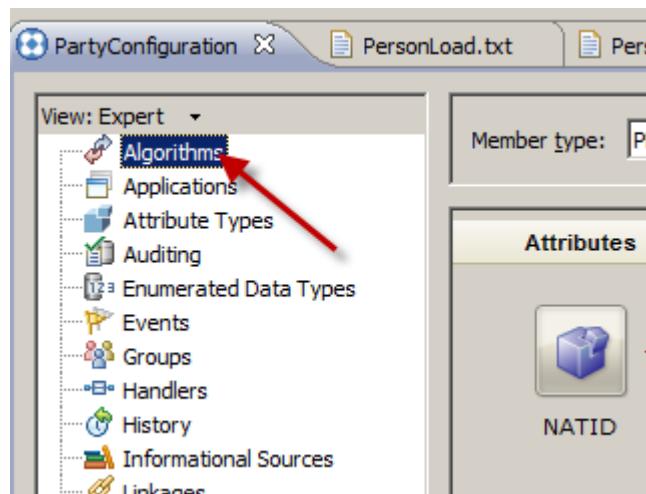
- 6. Click the **Edit** button to the right of the String value file field to open a text editor.



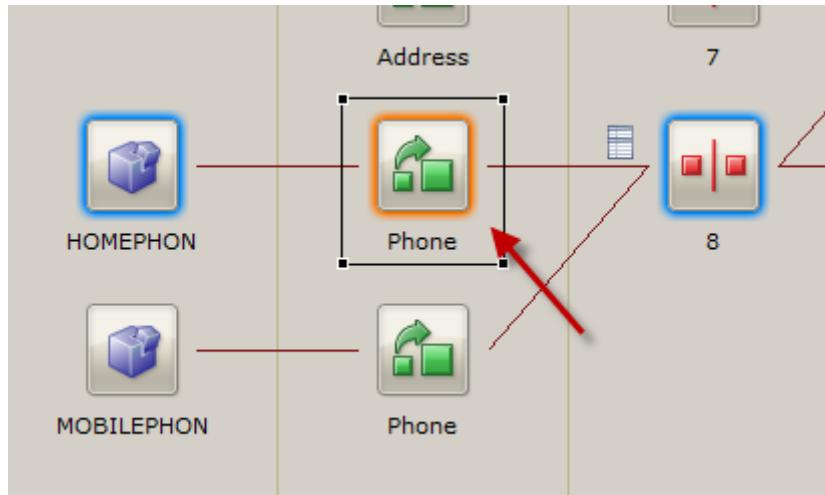
- 7. Add **9999999** (7x9) to the file. Save the file (CTRL+S).



- 8. Go back to the PartyConfiguration file and select the **Algorithms** tab.



- 9. Click the **Phone** icon in the Standardizations column (2nd column) next to the HOMEPHON attribute.

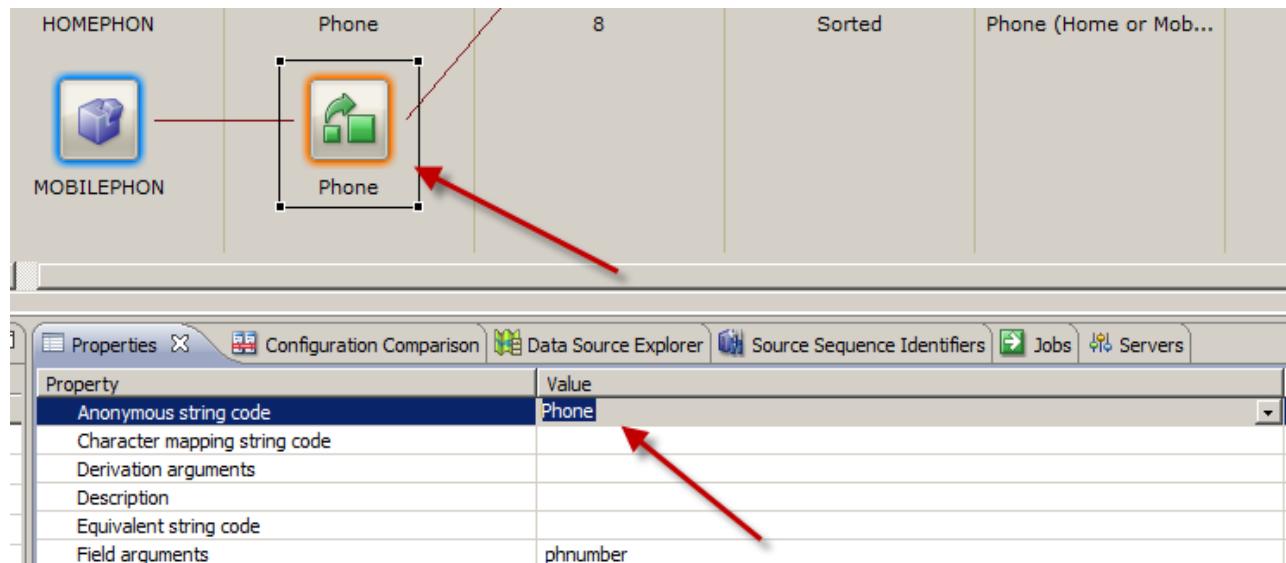


10. Open the RAD Properties window and change the Anonymous string code to **Phone**.

The screenshot shows the Address component in the InfoSphere MDM interface. It contains two attributes: HOMEPHON and MOBILEPHON, each with a Phone standardization function attached. A red arrow points to the Phone function under the HOMEPHON attribute. Below the component, the RAD Properties window is open, showing the 'Anonymous string code' property set to 'Phone'. Another red arrow points to this property.

Property	Value
Anonymous string code	Phone
Character mapping string code	
Derivation arguments	
Description	
Equivalent string code	
Field arguments	phonenumber
Primary standardization role	8

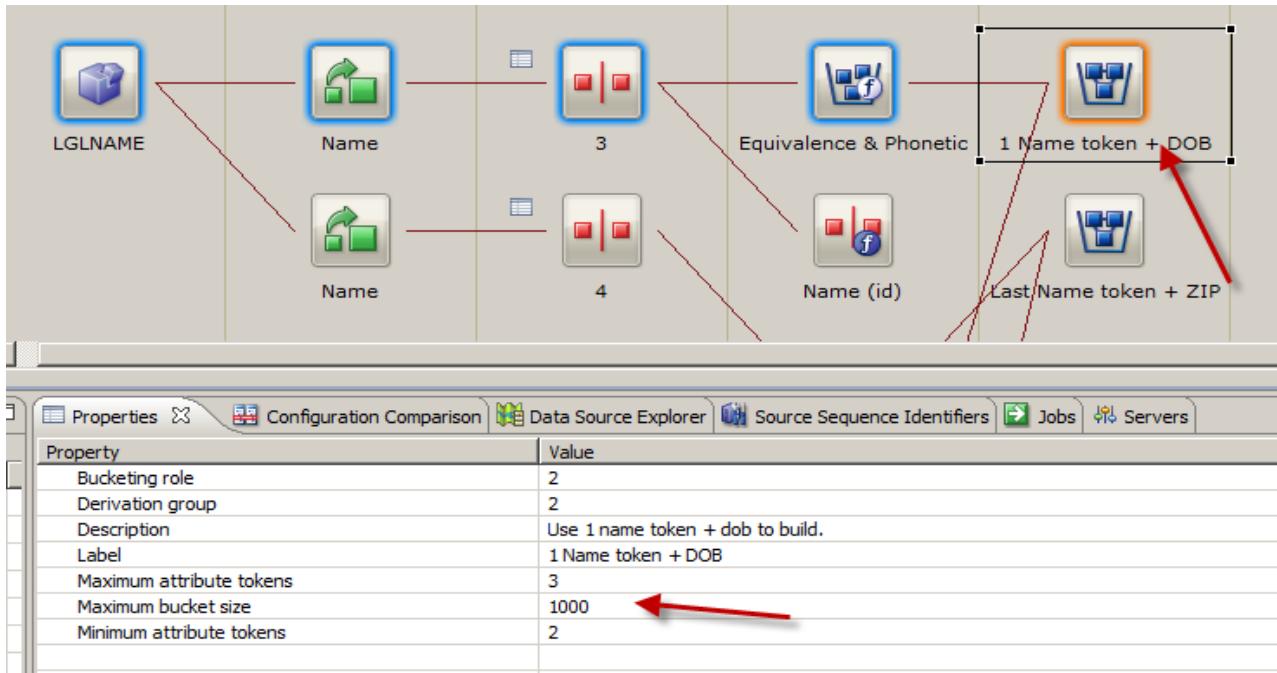
11. Repeat the previous steps for the Phone Standardization function attached to the **MOBILEPHON** attribute.



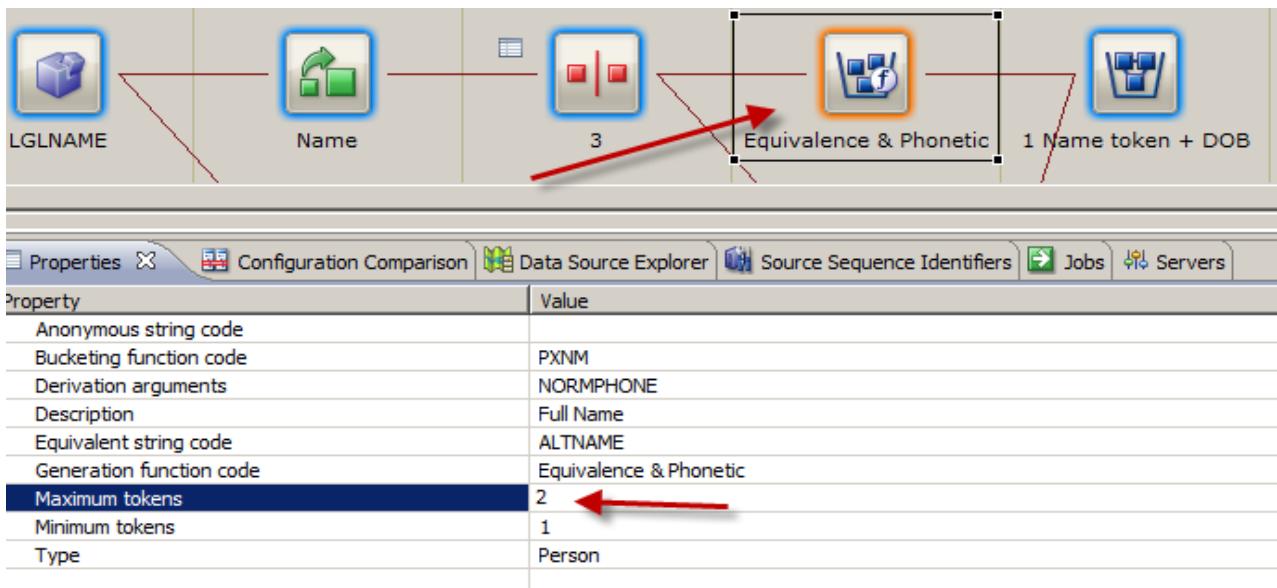
12. Save the configuration change (CTRL+S)

With our current configuration, there are 623 unbucketed members. Upon further investigation, these unbucketed members do not have phone number, national ID or DOB. Next we are going to make some configuration change to solve this problem.

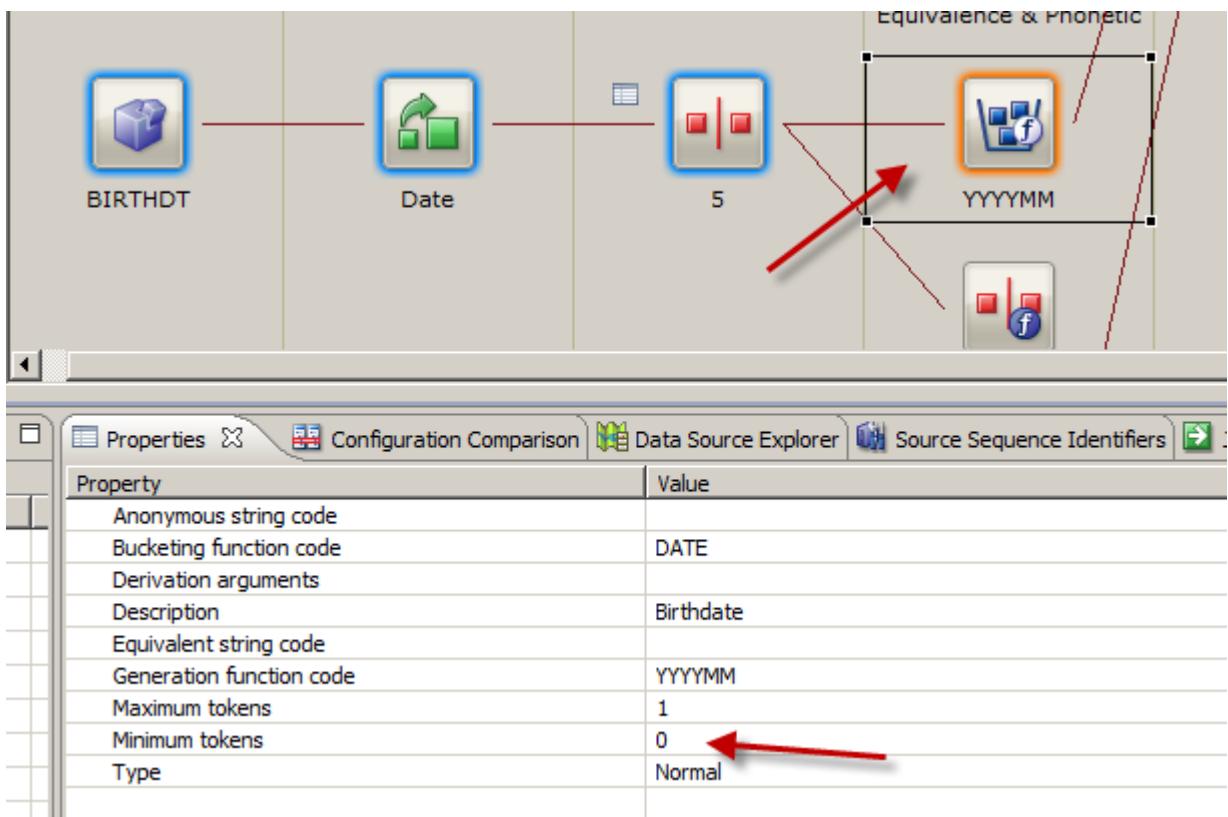
13. Select the **Name + DOB** bucket group in your algorithm and using the RAD **Properties** window, change the following values:
- Maximum attribute tokens:** 3,
 - Maximum bucket size:** 1000
 - Minimum attribute tokens:** 2.



14. Change the **Max Tokens** to **2** for the **Equivalence & Phonetic** bucketing function for the LGLNAME



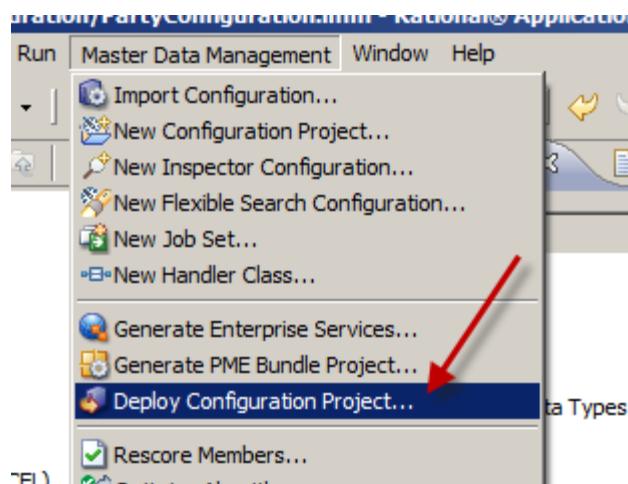
15. Change the **Min Tokens** to **0** for the **YYYYMM** bucketing function for the BIRTHDT.



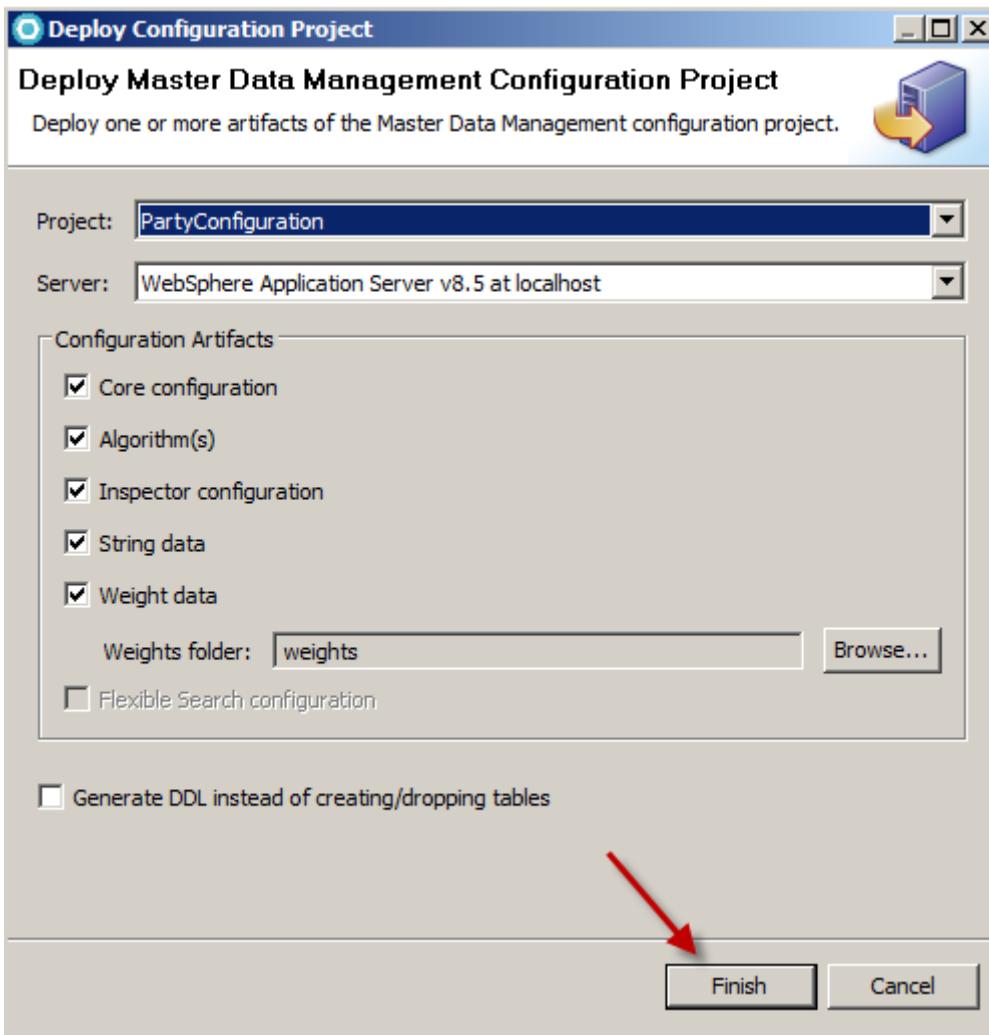
16. Save your changes (CTRL+S)

What did we just change? We changed the **1 Name token + DOB bucket** to really be a **(1 or 2) name tokens and DOB or a 2 Name token**. It's the 2 Name token buckets that will capture Members that do not have a DOB in their record.

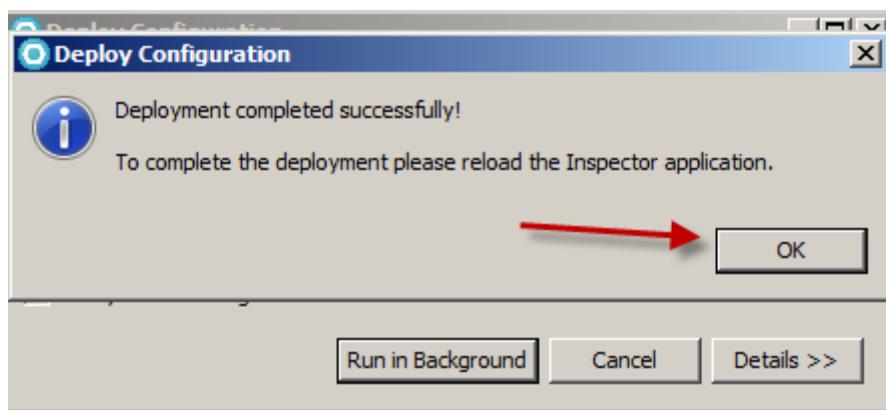
17. To see our changes we need to redeploy our configuration and rerun our job that loaded the database (so the records are bucketed again). From the RAD file menu, select **Master Data Management > Deploy Configuration**.



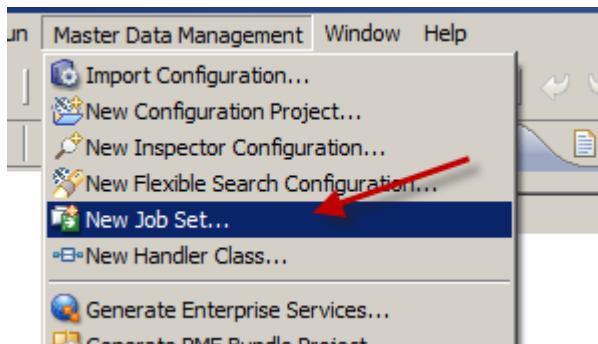
18. Accept the defaults and click the **Finish** button.



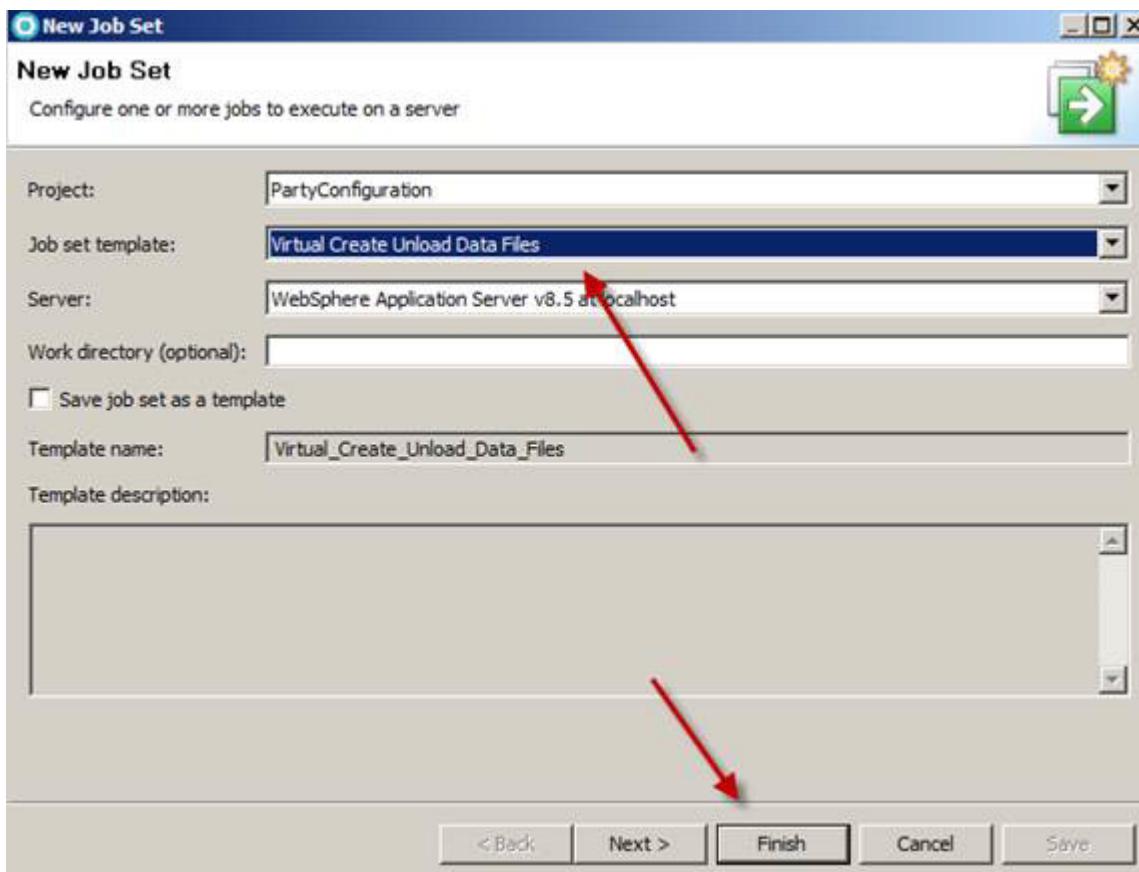
- ___ 19. Once the deployment is complete, click the **OK** button.



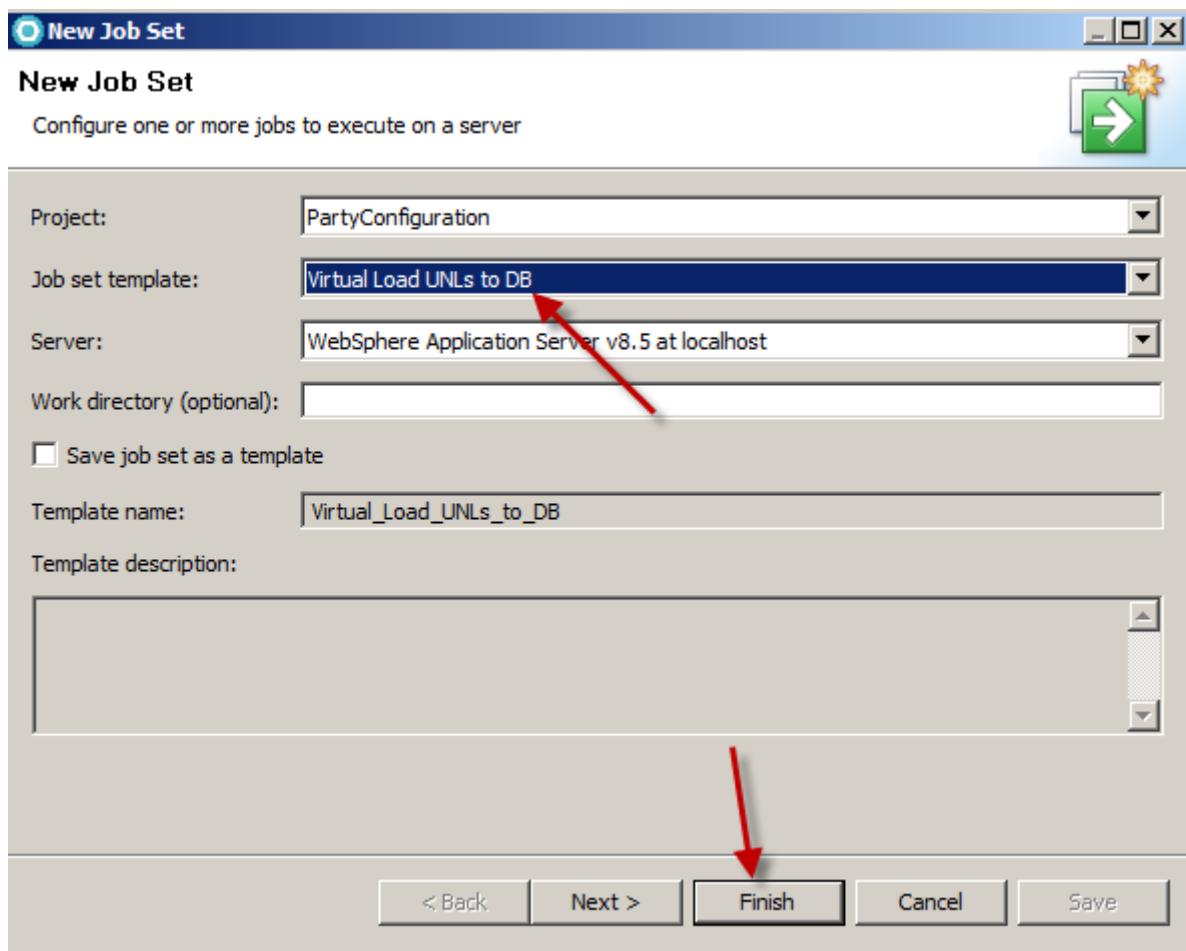
- ___ 20. Next we need to redeploy our members. From the RAD file menu select **Master Data Management > Run Job ...**



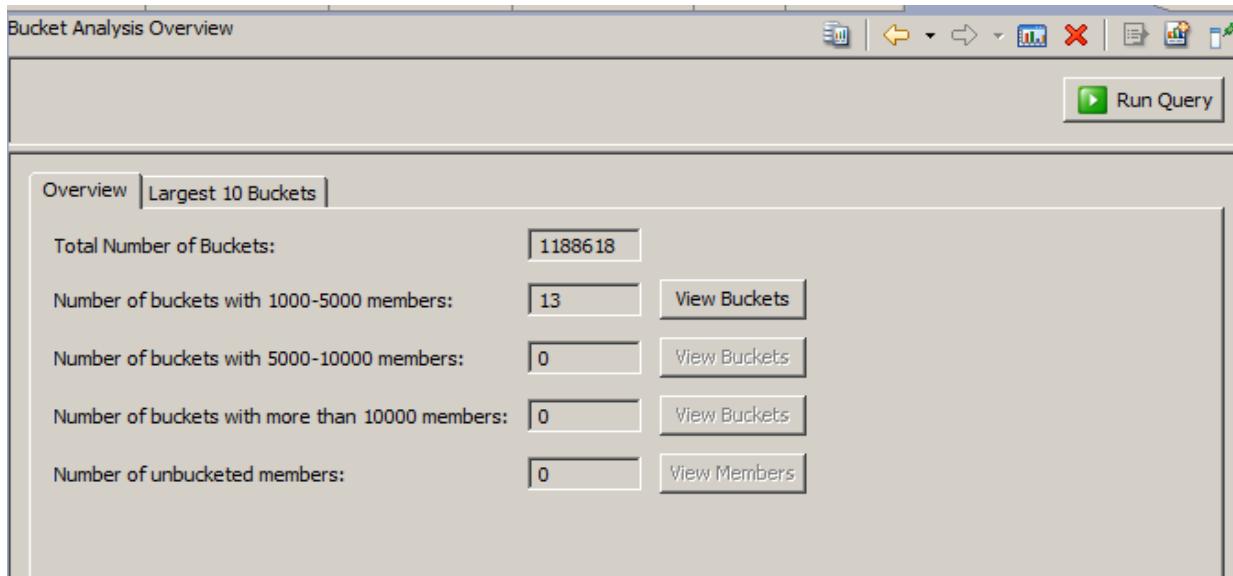
- 21. Select **Virtual Create Unload Data Files** for the Job set template and click the **Finish** button.



- 22. Once the job is complete (you can check the jobs window). Run a Job set again but this time select **Virtual Load UNLs to DB** for the Job set template and click the **Finish** button.



23. Once the 2nd Job is complete (check the Jobs window), open the **Analytics** window again and re-run the **Bucket Analysis Overview** query.



Congratulation, you have used the bucket analysis to evaluate and improve our algorithm.

End of exercise

Exercise 5. Weight Generation

What this exercise is about

This exercise covers generating and reviewing weights.

What you should be able to do

At the end of this exercise, you should be able to:

- Generate weights
- Deploy weights
- Reviewing weights

Introduction

The weight generation process is an integrated utility that goes through multiple steps to measure the frequency of individual values in the database and then assigns weights to those values with the most common values weighing less and rare values weighing more. The weight generation process creates unload files which will later be loaded into the database.

Requirements

Each of these exercise depend on the previous exercise being complete. For this exercise you must have the data model and algorithms defined in the configuration file. The configuration must be deployed with sample data loading into the database.

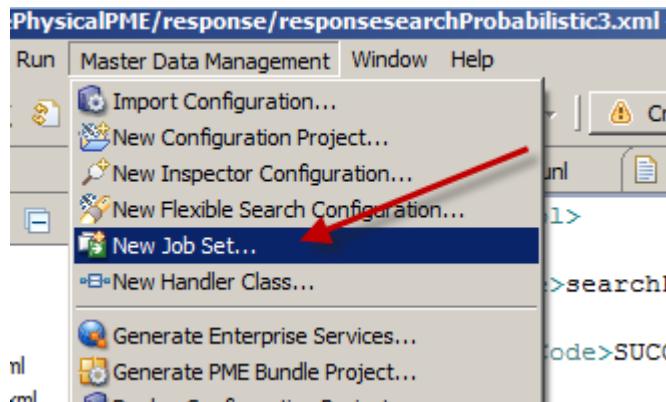
Exercise instructions

Part 1: Generating weights

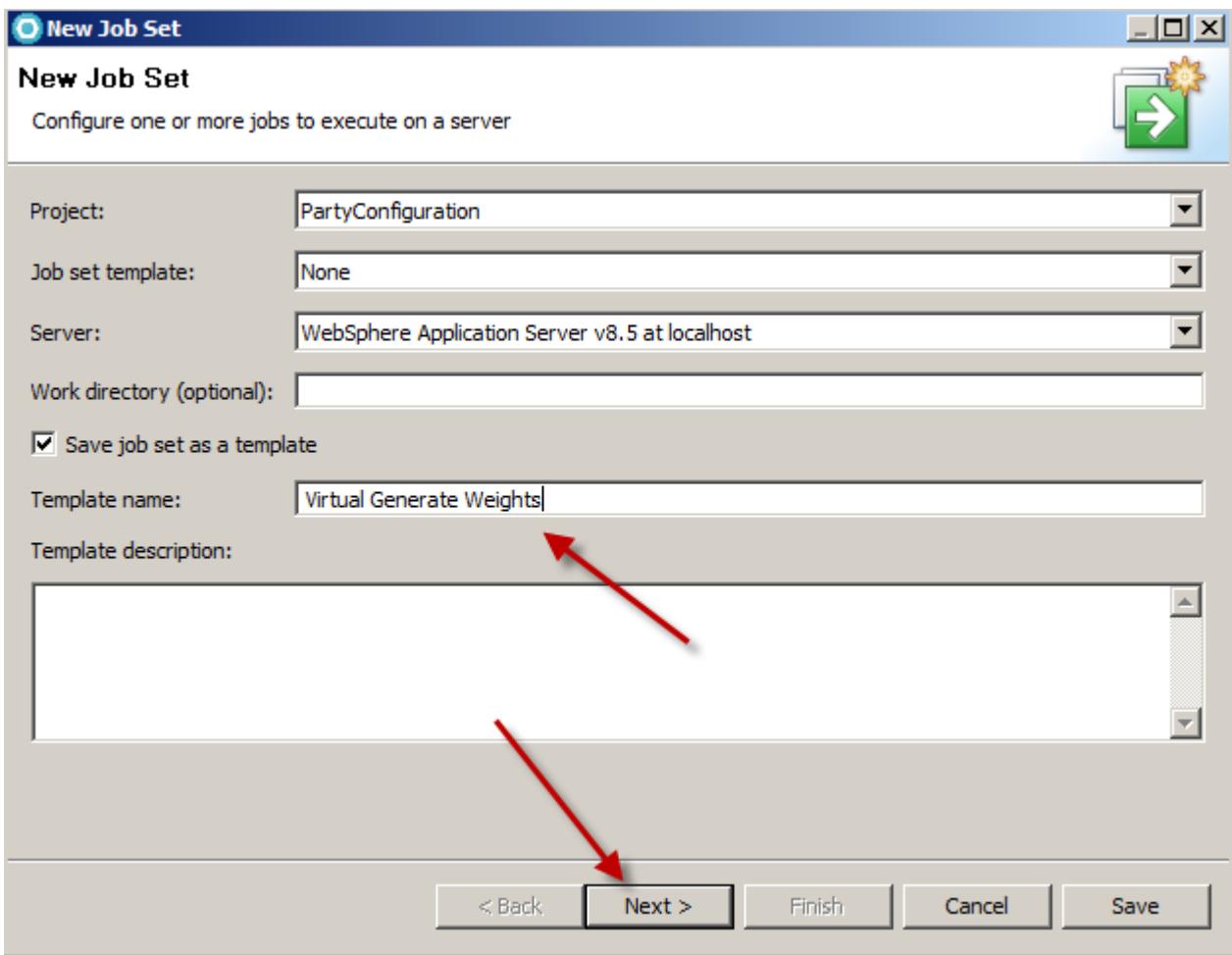
Even though our members are now being bucketed, we do not have weights for our comparison functions.

To generate weights there is a service that we can use.

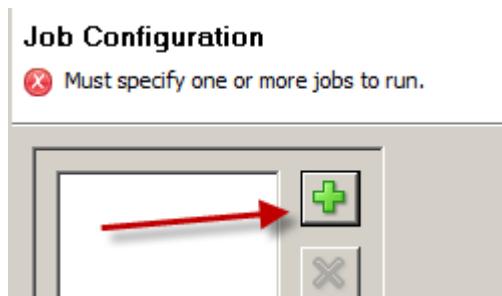
- 1. From the RAD file menu, select **Master Data Management > New Job Set...**



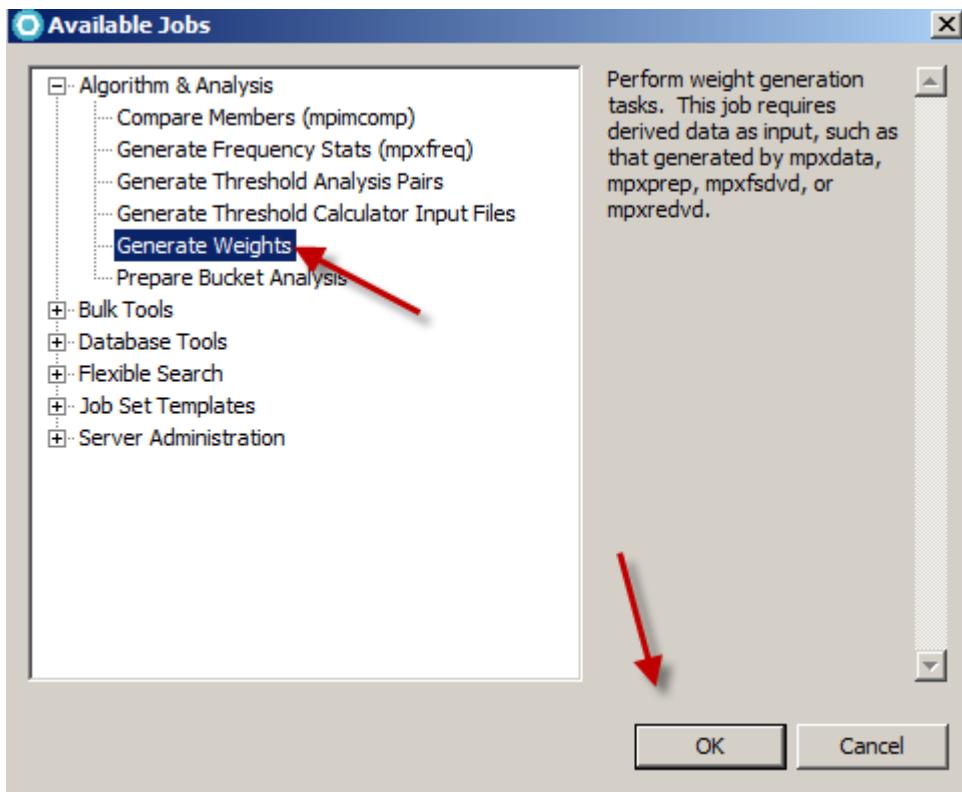
- 2. Select the **Save job set as a template** checkbox and enter **Virtual Generate Weights** as the Template name. Click the **Next >** button when complete.



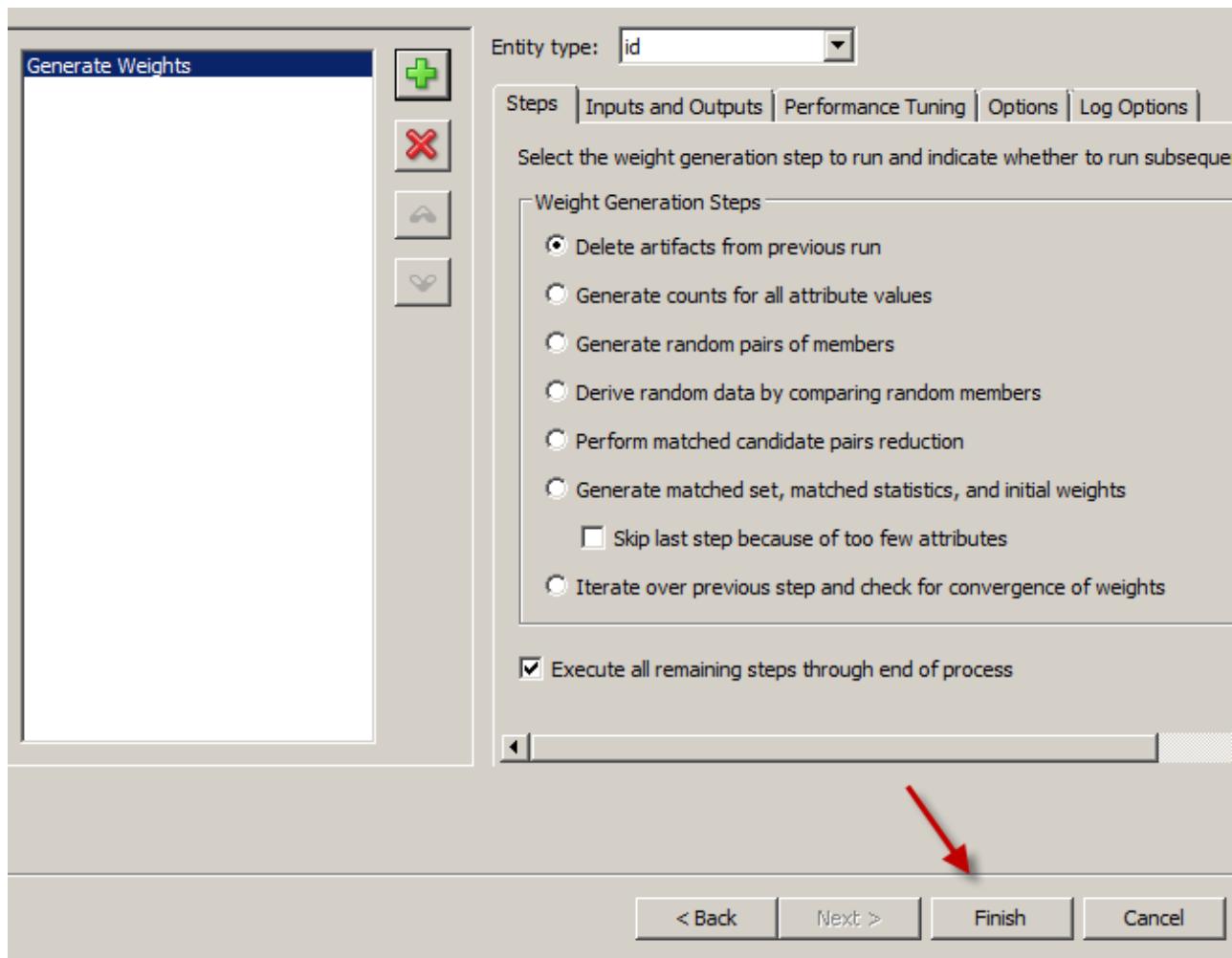
3. Click the green plus (+) button to add a new job.



4. Select **Algorithms & Analysis > Generate Weights** and click the **OK** button.

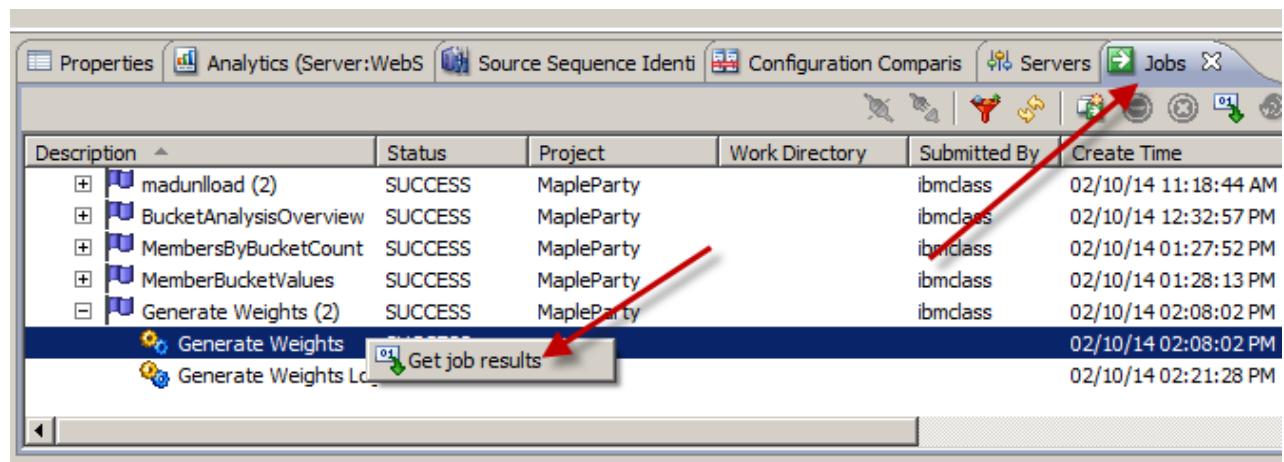


- ___ 5. Click the **Finish** button.

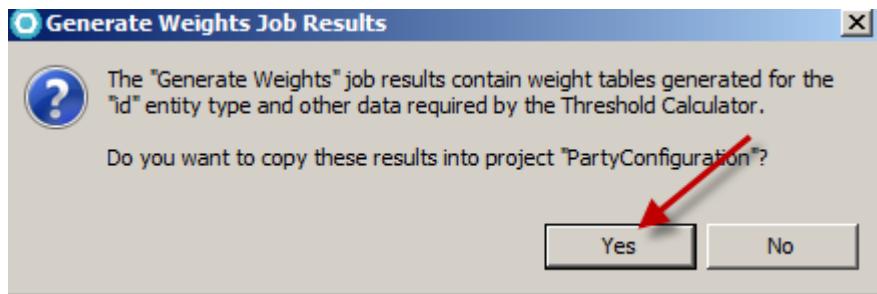


This job can take up to 20 min to complete (time to take a break), you can check the status under the Jobs window.

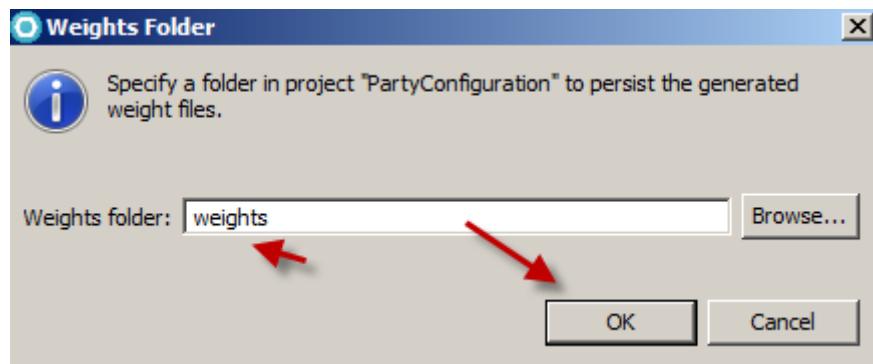
6. Once the job is complete, open the Jobs window, select the last **Generate Weights** job segment, right click and select **Get job results**.



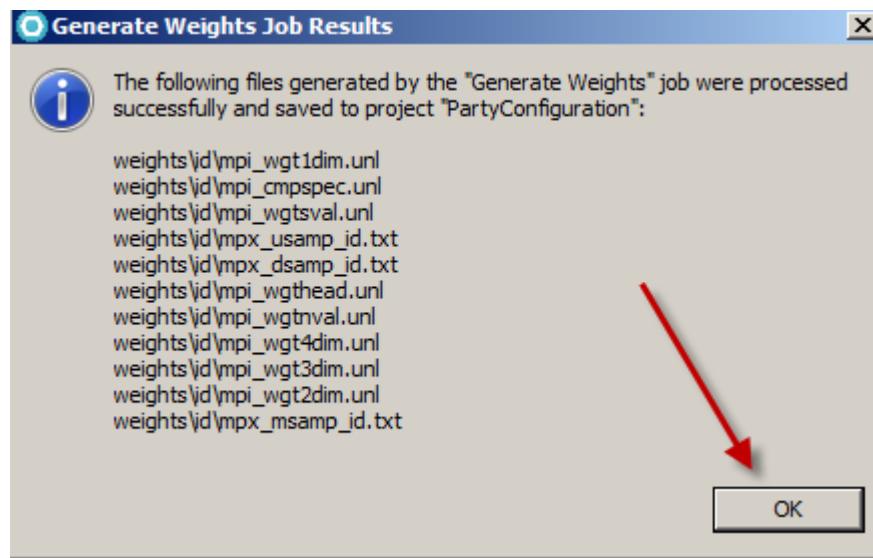
- 7. You will get a pop up asking whether you want to copy the weights that we generated. Click the **Yes** button.



- 8. Select the **weights** folder and click the **OK** button.



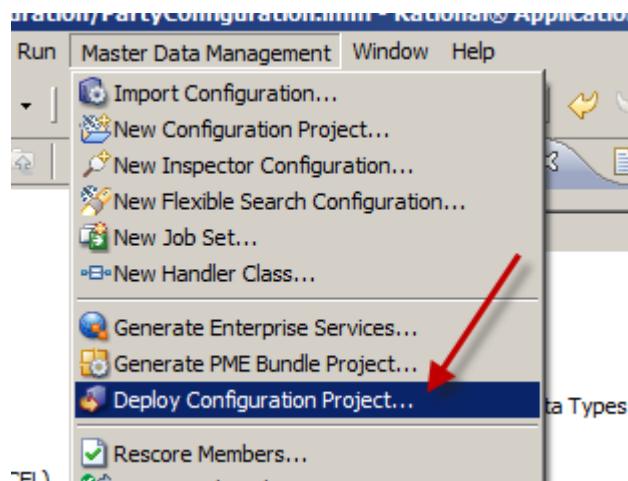
- 9. Once the results have been copied, click the **OK** button.



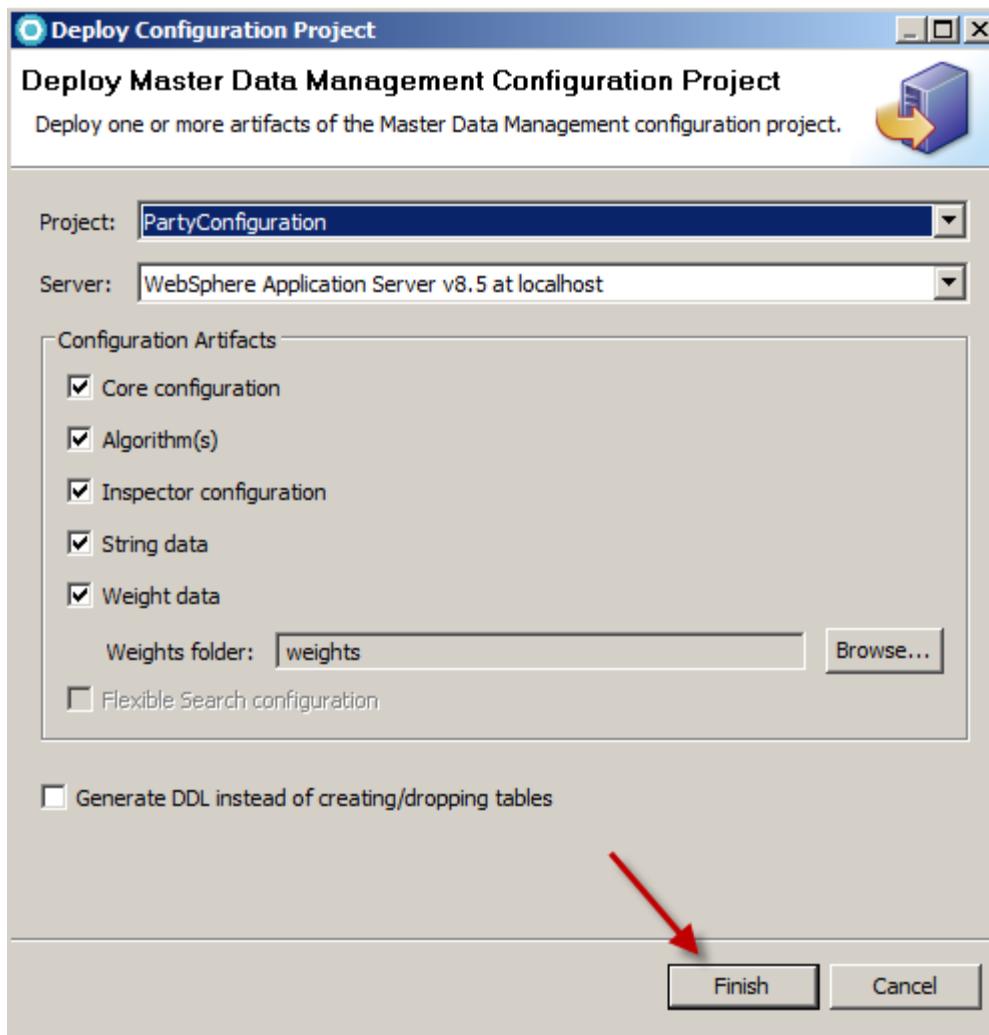
Part 2: Redeploy the Configuration

When you deploy the configuration, one of the assets that get deployed is our weights, therefore since we modified our weights, we need to redeploy our configuration.

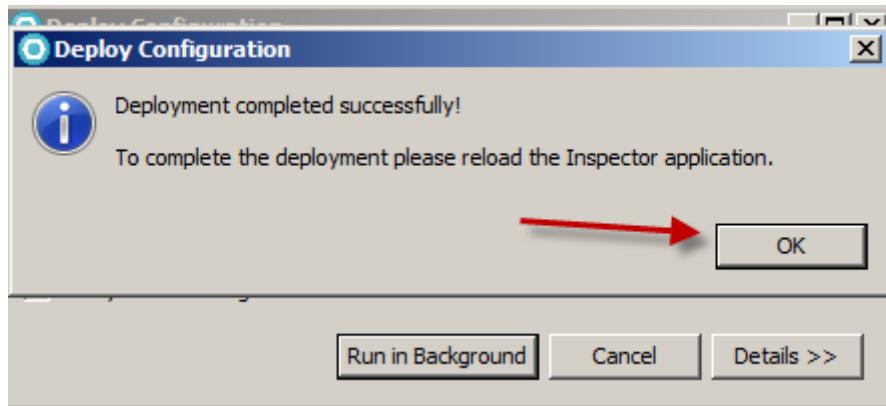
- 1. From the RAD file menu, select **Master Data Management > Deploy Configuration**.



- 2. Accept the defaults and click the **Finish** button.

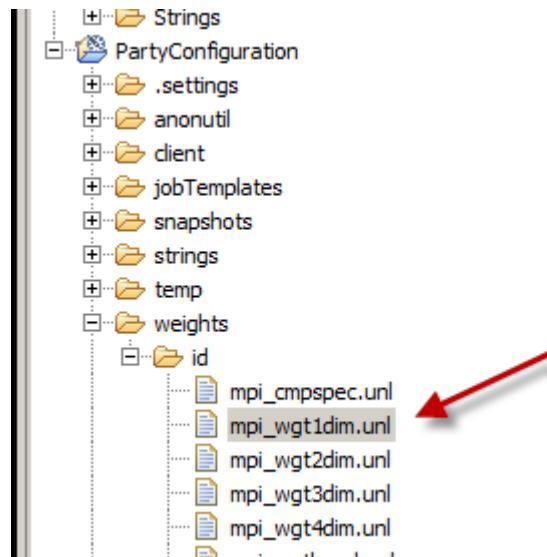


- 3. Once the deployment is complete, click the **OK** button.



Part 3: Viewing the weights

- 1. To see the weights that were generated, open the **mpi_wgt1dim.unl** found under the **PartyConfiguration > weights > id** folder.



- 2. Look at the lines that include the **CMPID-NATLID**. Since the algorithm uses an edit distance comparison function, you will notice that the numbers from 1 (a perfect match) to 10 (9 edits) decrease in the score that is given during comparison.

```

99|99|A|CMPID-NATLID-DIST|0|0|
99|99|A|CMPID-NATLID-DIST|1|571|
99|99|A|CMPID-NATLID-DIST|2|388|
99|99|A|CMPID-NATLID-DIST|3|314|
99|99|A|CMPID-NATLID-DIST|4|177|
99|99|A|CMPID-NATLID-DIST|5|32|
99|99|A|CMPID-NATLID-DIST|6|-91|
99|99|A|CMPID-NATLID-DIST|7|-203|
99|99|A|CMPID-NATLID-DIST|8|-251|
99|99|A|CMPID-NATLID-DIST|9|-290|
99|99|A|CMPID-NATLID-DIST|10|-352|
99|99|A|CMPID-DOB-DIST|0|0|
99|99|A|CMPID-DOB-DIST|1|0|
99|99|A|CMPID-DOB-DIST|2|46|
99|99|A|CMPID-DOB-DIST|3|-129|
99|99|A|CMPID-DOB-DIST|4|-308|

```

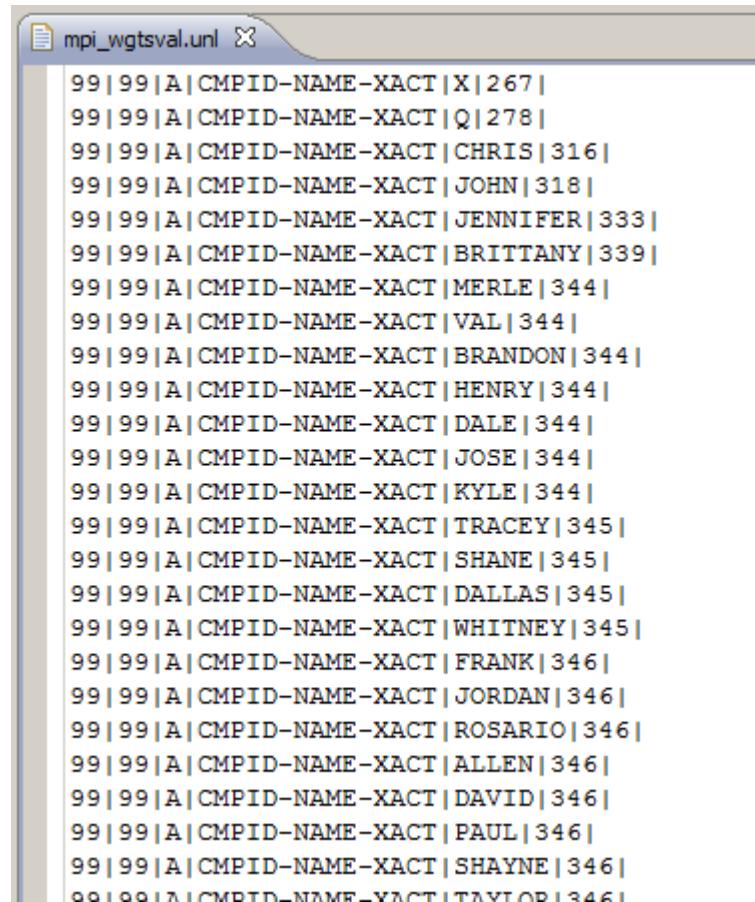
3. To see the values of the 2 dimensional address scores, open the **mpi_wgt2dim.unl** file. Here you notice as I go from the top left hand corner of the values (a perfect match between phone number and address) to the bottom right corner of the values (a non match between the phone number and address) the score becomes less.

```

99|99|A|CMPID-AXP-2DIM|0|0|448|215|63|-79|-178|-230|-378|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|1|477|561|375|334|309|286|225|164|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|2|330|405|272|212|179|164|164|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|3|290|363|246|202|164|164|164|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|4|270|343|229|190|164|164|164|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|5|155|164|164|164|164|164|164|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|6|155|164|164|164|164|164|164|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|7|149|164|164|164|164|164|150|139|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|8|128|164|164|164|164|123|111|111|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|9|77|164|164|144|114|74|58|46|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|10|-64|164|120|24|-53|-68|-86|-100|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|11|-78|164|114|-4|-64|-72|-92|-115|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|12|-146|164|111|-12|-87|-156|-172|-182|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|13|-156|164|90|-25|-108|-162|-185|-191|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|14|-189|164|81|-56|-138|-187|-217|-226|0|0|0|0|0|0|0|0|
99|99|A|CMPID-AXP-2DIM|15|-321|156|6|-134|-246|-326|-348|-361|0|0|0|0|0|0|0|0|

```

4. For our names, we use a frequency evaluation. Open the **mpi_wgtsval.unl**. In this file you file scores that are provided for each name as well as nicknames, min max scores, etc. This is all based on how unique a name is.



The screenshot shows a Windows Notepad window with the title bar 'mpi_wgtsval.unl'. The content of the window is a list of names and their corresponding CMPID values, separated by vertical bars. The list starts with '99|99|A|CMPID-NAME-XACT|X|267|' and ends with '00|00|A|CMPID-NAME-XACT|TAYTOR|346|'. The text is in a standard black font on a white background.

```
99|99|A|CMPID-NAME-XACT|X|267|
99|99|A|CMPID-NAME-XACT|Q|278|
99|99|A|CMPID-NAME-XACT|CHRIS|316|
99|99|A|CMPID-NAME-XACT|JOHN|318|
99|99|A|CMPID-NAME-XACT|JENNIFER|333|
99|99|A|CMPID-NAME-XACT|BRITTANY|339|
99|99|A|CMPID-NAME-XACT|MERLE|344|
99|99|A|CMPID-NAME-XACT|VAL|344|
99|99|A|CMPID-NAME-XACT|BRANDON|344|
99|99|A|CMPID-NAME-XACT|HENRY|344|
99|99|A|CMPID-NAME-XACT|DALE|344|
99|99|A|CMPID-NAME-XACT|JOSE|344|
99|99|A|CMPID-NAME-XACT|KYLE|344|
99|99|A|CMPID-NAME-XACT|TRACEY|345|
99|99|A|CMPID-NAME-XACT|SHANE|345|
99|99|A|CMPID-NAME-XACT|DALLAS|345|
99|99|A|CMPID-NAME-XACT|WHITNEY|345|
99|99|A|CMPID-NAME-XACT|FRANK|346|
99|99|A|CMPID-NAME-XACT|JORDAN|346|
99|99|A|CMPID-NAME-XACT|ROSARIO|346|
99|99|A|CMPID-NAME-XACT|ALLEN|346|
99|99|A|CMPID-NAME-XACT|DAVID|346|
99|99|A|CMPID-NAME-XACT|PAUL|346|
99|99|A|CMPID-NAME-XACT|SHAYNE|346|
00|00|A|CMPID-NAME-XACT|TAYTOR|346|
```

End of exercise

Exercise 6a.Bulk Cross Load

What this exercise is about

This exercise covers bulk cross matching.

What you should be able to do

At the end of this exercise, you should be able to:

- Run a bulk cross match and load the entity/linkage data

Introduction

Although we have loaded sample data into MDM and they have been bucketed, not one member has been compare at this point. To compare our members for a bulk load, we must run the bulk cross match (BXM) process.

The bulk cross match (BXM) is a process that allows you to compare and link thousands of records per second. The BXM is most commonly performed in the initial stage of the implementation and again right before the system goes live. The BXM process is made up of two primary jobs; Compare Members in Bulk (mpxcomp) and Link Entities (mpxlink). The BXM process creates entities so after running the compare and link, the entity data will need to be loaded into the database.

Requirements

Each of these exercise depend on the previous exercise being complete. For this exercise you must have the data model and algorithms defined in the configuration file. The configuration must be deployed with sample data loading into the database and the weights must be generated and deployed.

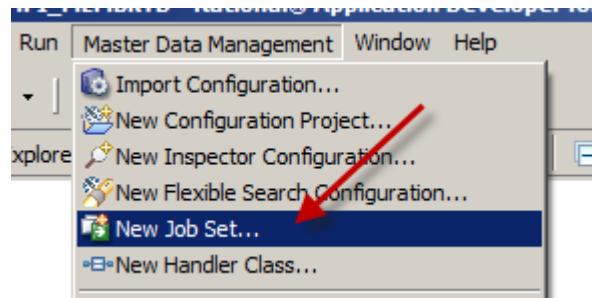
Exercise instructions

Part 1: Run a bulk cross match and load entity data

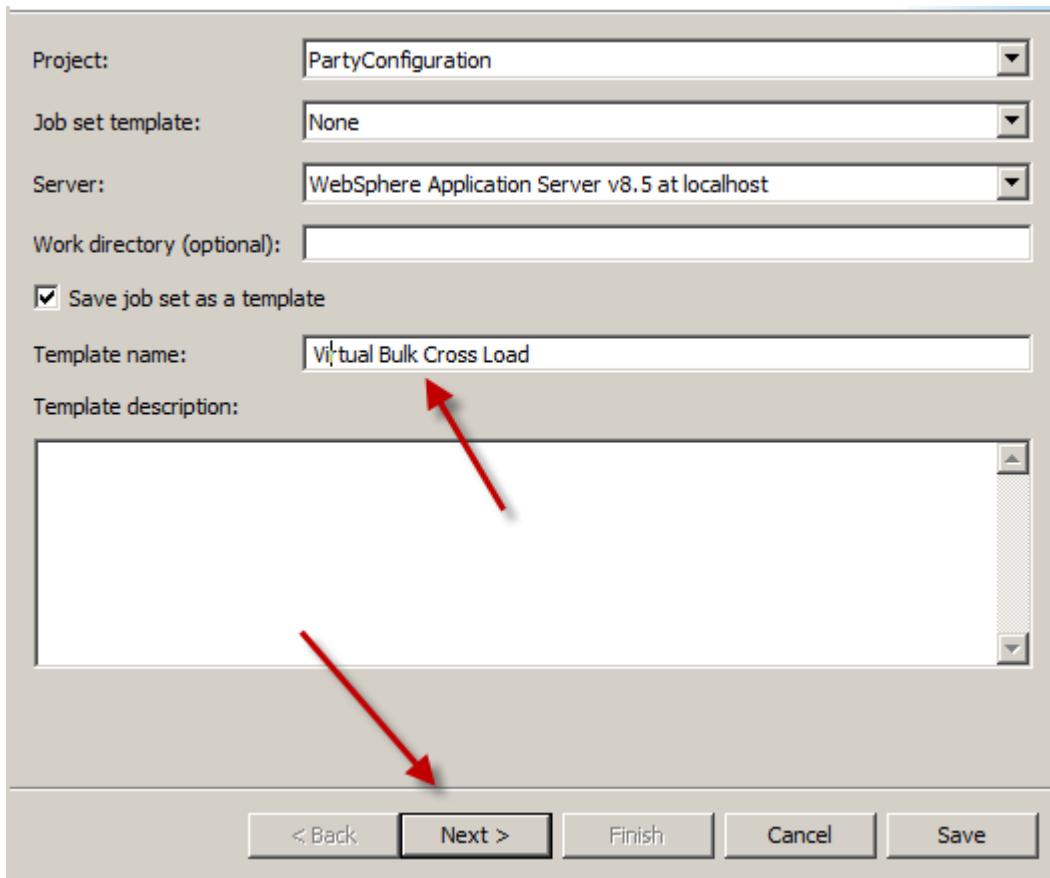
For the Bulk Cross Match Load we will create a job that will run 3 tasks:

- Compare Members in Bulk (mpxcomp)
- Link Entities (mpxlink)
- Load UNLs to DB (madunlload)

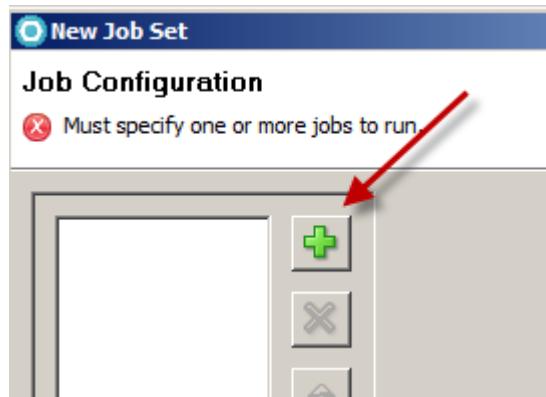
___ 1. From the RAD file menu, select **Master Data Management > New Job Set...**



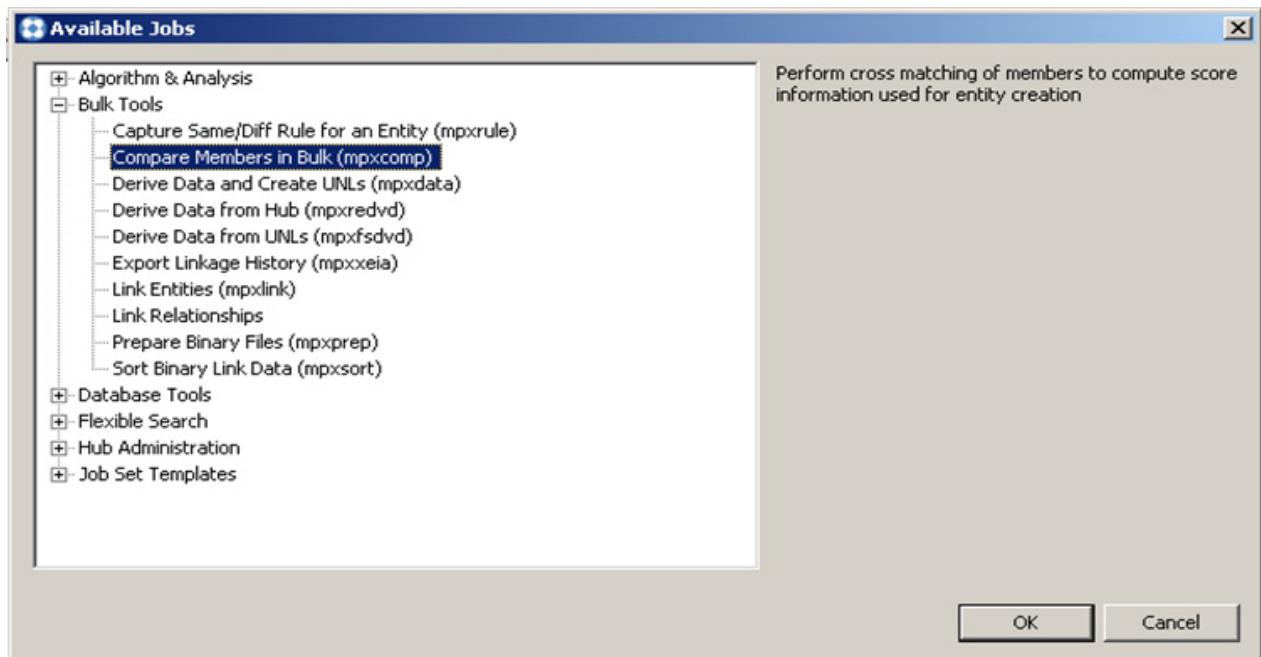
___ 2. Enter **Virtual Bulk Cross Load** as the job set template name and click the **Next >** button.



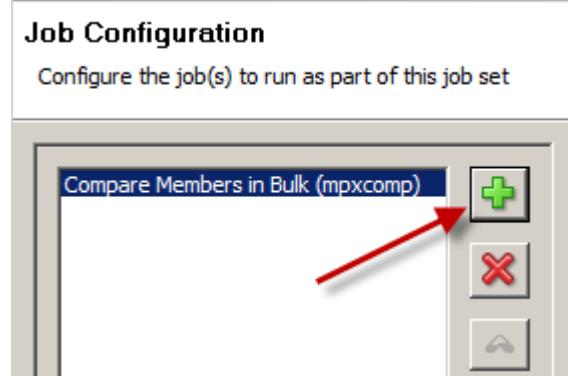
___ 3. Click the **Add Job** (the green plus sign) button.



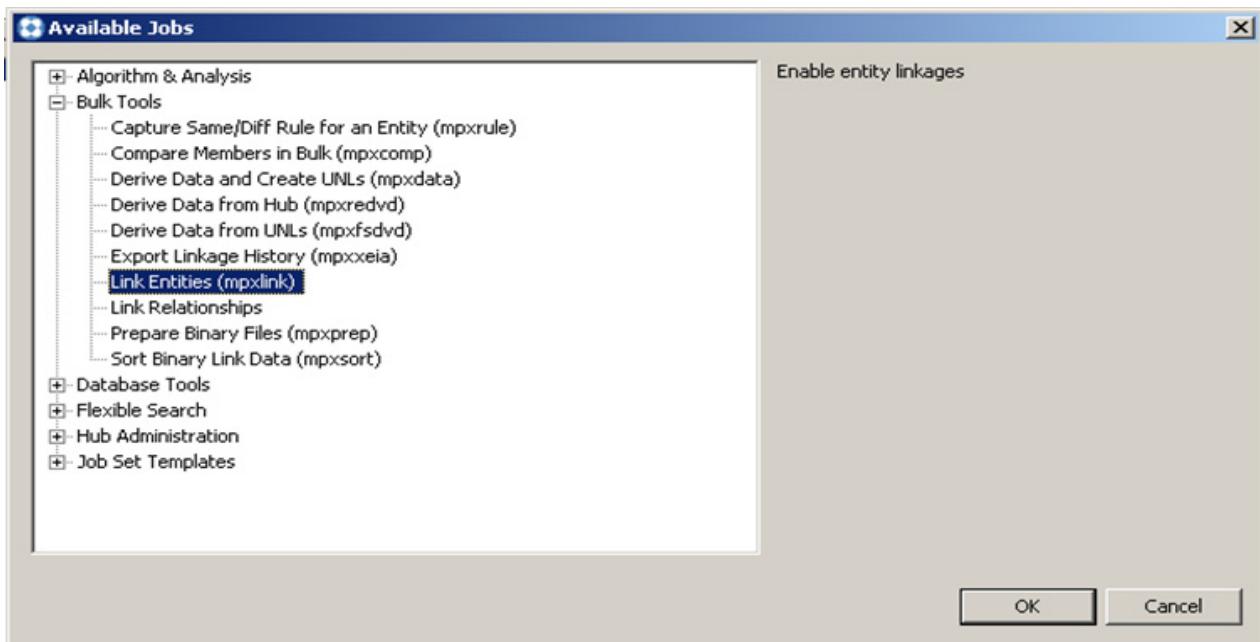
- ___ 4. Expand Bulk tools, select **Compare Members in Bulk (mpxcomp)** and click the **OK**.



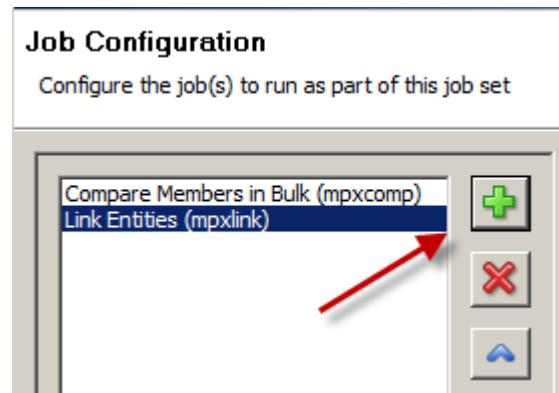
- ___ 5. Click the **Add Job** (the green plus sign) button.



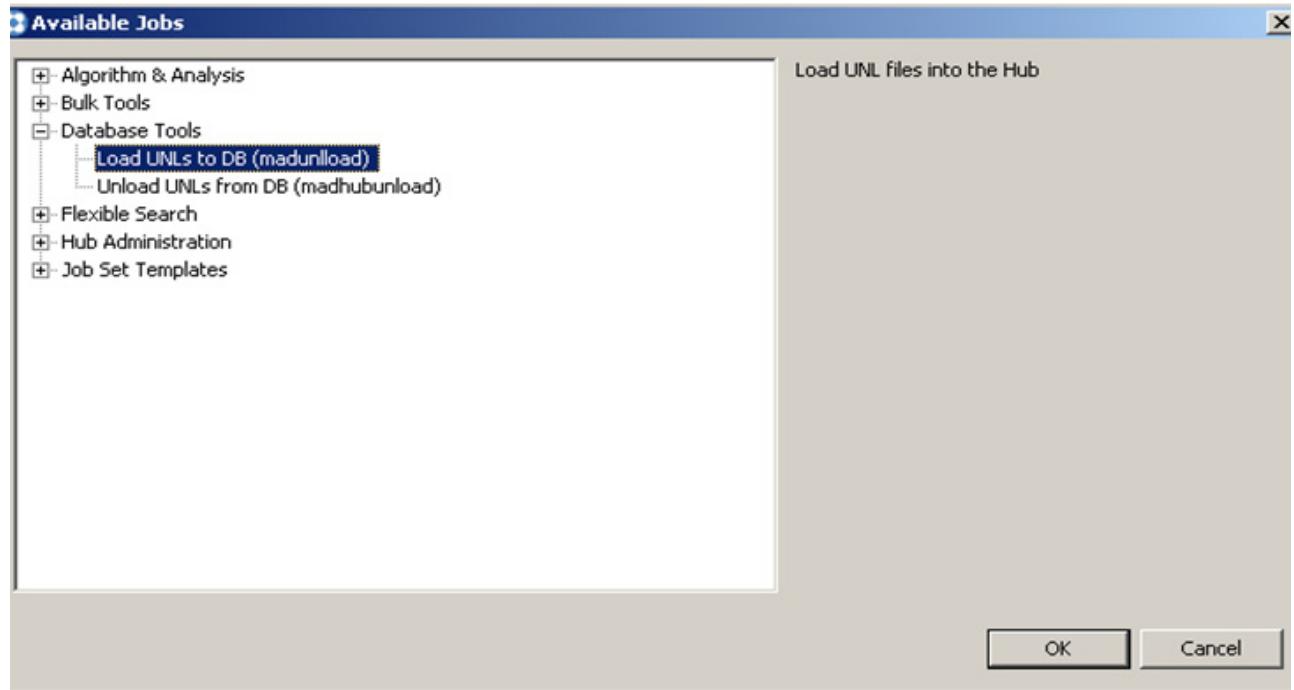
- ___ 6. Expand Bulk Tools, select **Link Entities (mpxlink)** and click the **OK** button.



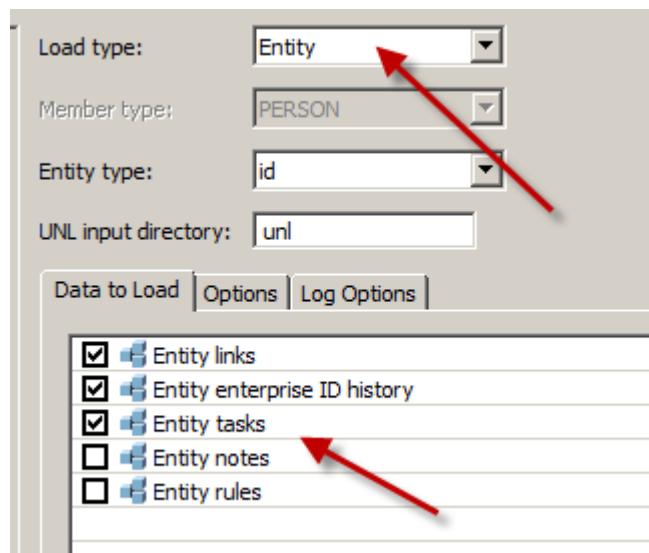
- ___ 7. Click the **Add Job** (the green plus sign) button.



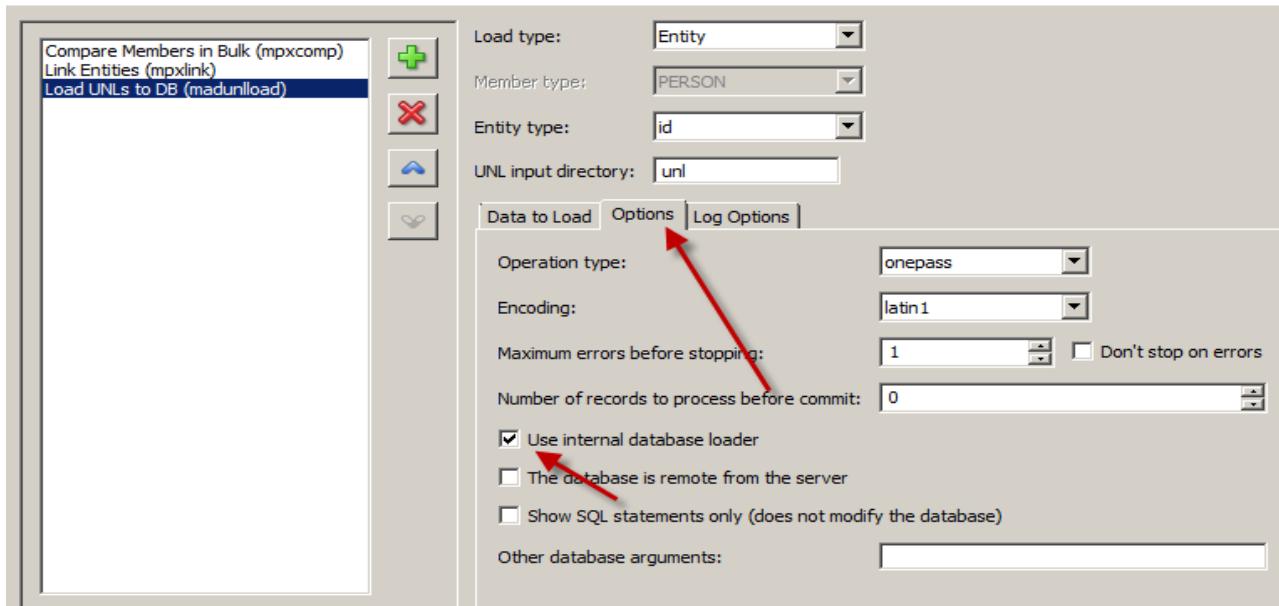
- ___ 8. Expand Database Tools, select **Load UNLs to DB (madunlload)** and click the **OK** button.



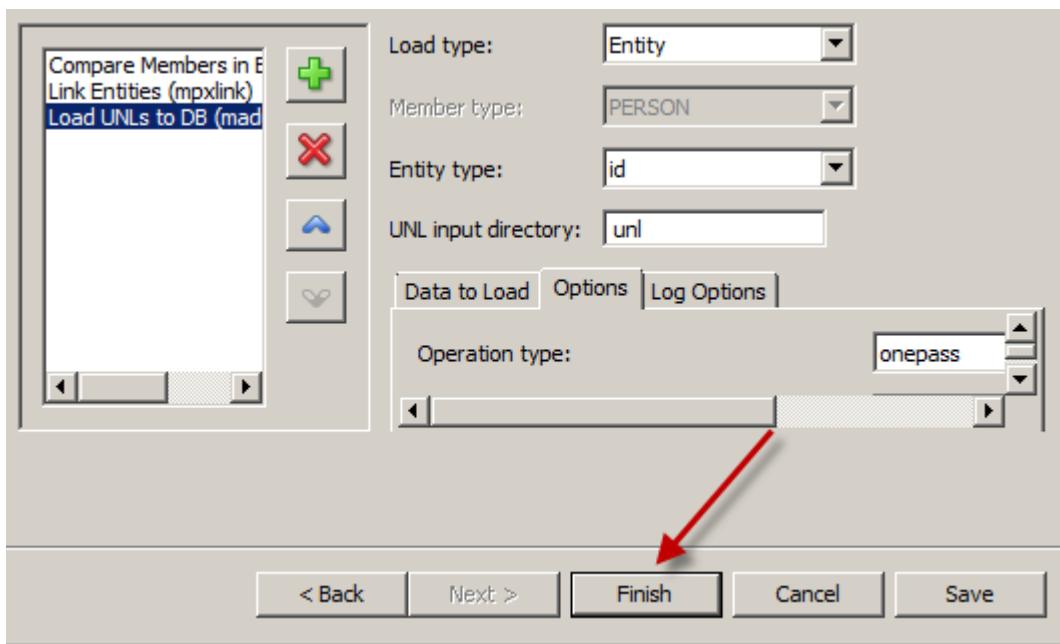
- 9. Select **Entity** in the Load type dropdown and ensure that only **Entity links**, **Entity enterprise ID history** and **Entity tasks** are selected for Data to Load.



- 10. Select the Options tab and check the **Use internal database loader** checkbox.



- 11. Click the **Finish** button to start running the bulk cross match job set. (This job can take up to 20 min)



- 12. Open the RAD Jobs window and expand the **mpxcomp** job set that is running to view status.

Description	Status	Project	Work Directory	Subm
+ madownload (2)	SUCCESS	PartyConfiguration		ibmc
+ BucketAnalysisOverview	SUCCESS	PartyConfiguration		ibmc
+ Generate Weights (2)	SUCCESS	PartyConfiguration		ibmc
+ Deploy Configuration Proj	SUCCESS	PartyConfiguration		ibmc
+ Deploy Configuration Proj	SUCCESS	PartyConfiguration		ibmc
+ Generate Weights (2)	SUCCESS	PartyConfiguration		ibmc
+ Deploy Configuration Proj	SUCCESS	PartyConfiguration		ibmc
- mpxcomp (3)	STARTED	PartyConfiguration		ibmc
mpxcomp	STARTED			
mpxlink	QUEUED			
madownload	QUEUED			

13. Verify that the Bulk Cross Match job set has completed.

Description	Status	Project	Work Directory	Subm
+ Deploy Configuration Proj	SUCCESS	PartyConfiguration		ibmda
+ Deploy Configuration Proj	SUCCESS	PartyConfiguration		ibmda
+ Generate Weights (2)	SUCCESS	PartyConfiguration		ibmda
+ Deploy Configuration Proj	SUCCESS	PartyConfiguration		ibmda
- mpxcomp (6)	SUCCESS	PartyConfiguration		ibmda
mpxcomp	SUCCESS			
mpxlink	SUCCESS			
madownload	SUCCESS			
mpxcomp Log	SUCCESS			
mpxlink Log	SUCCESS			
madownload Log	SUCCESS			

We should now have Entities in our MDM implementation.

End of exercise

Exercise 6b.Pair Manager and Threshold Calculations

What this exercise is about

This exercise covers how to use the IBM® Master Data Management Pair Manager to review sample pairs to determine system thresholds.

What you should be able to do

At the end of this exercise, you should be able to:

- Generate and review Threshold Analysis sample pairs
- Use Pair Manager to review Threshold Analysis sample pairs
- Set thresholds using Threshold Calculator
- Re-deploy the configuration
- Re-run Bulk Cross Match

Introduction

Pair Manager is a stand-alone tool that reads the sample pair file created from the Threshold Analysis Pair Generation job. The Threshold Analysis Pair Generation job supports dividing the generated sample pairs into multiple.xls files. This tool is primarily used during the implementation phase to speed the process of sample-pair evaluation.

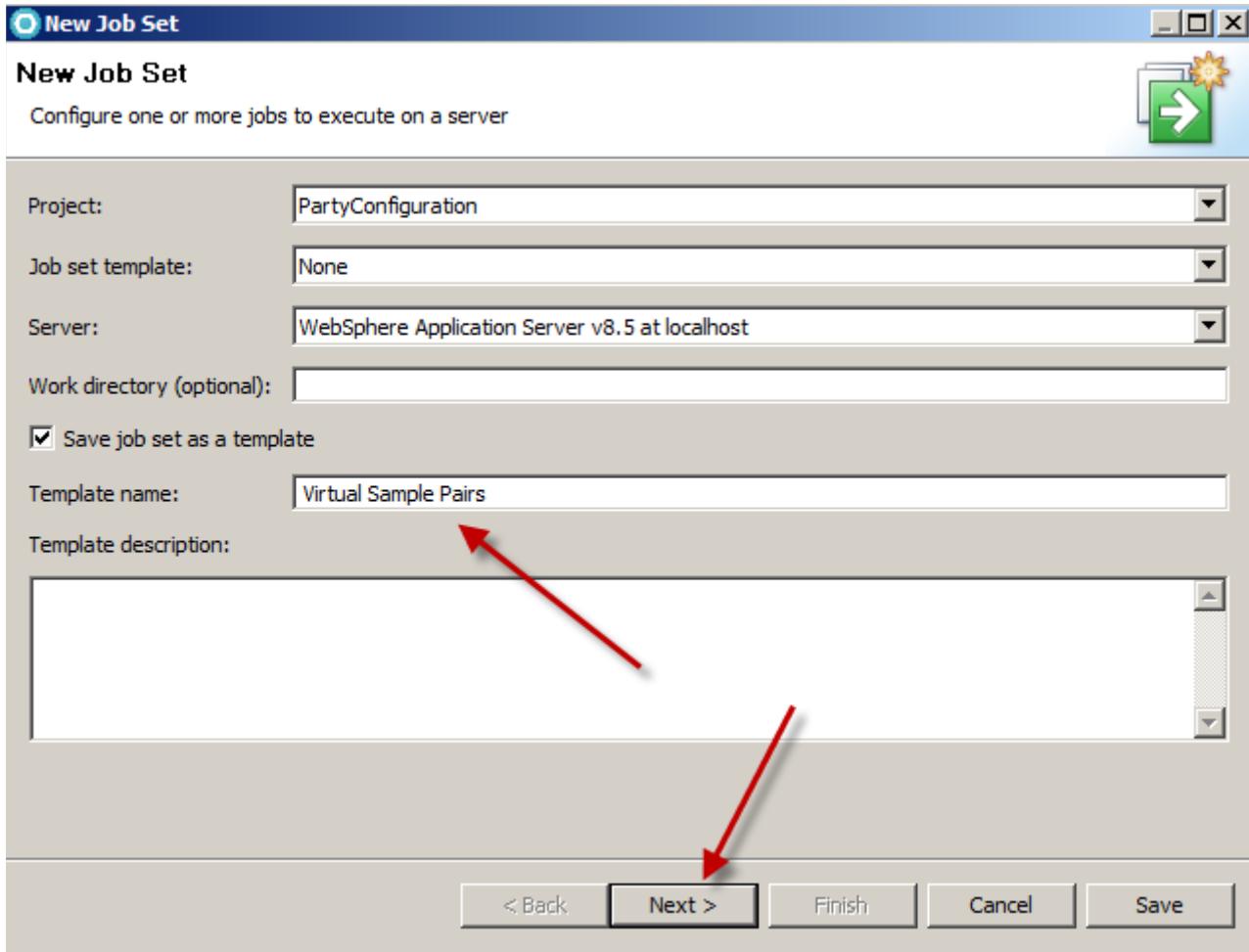
Requirements

Each of these exercise depend on the previous exercise being complete. For this exercise you must have the data model and algorithms defined in the configuration file. The configuration must be deployed with sample data loading into the database. The weights must be generated and a bulk cross match and load process must have been run on the sample data.

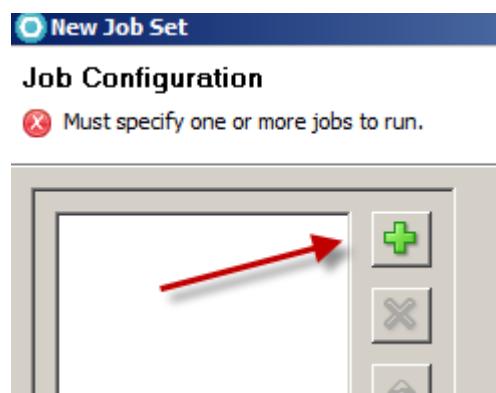
Exercise instructions

Part 1: Generate and review Threshold Analysis sample pairs

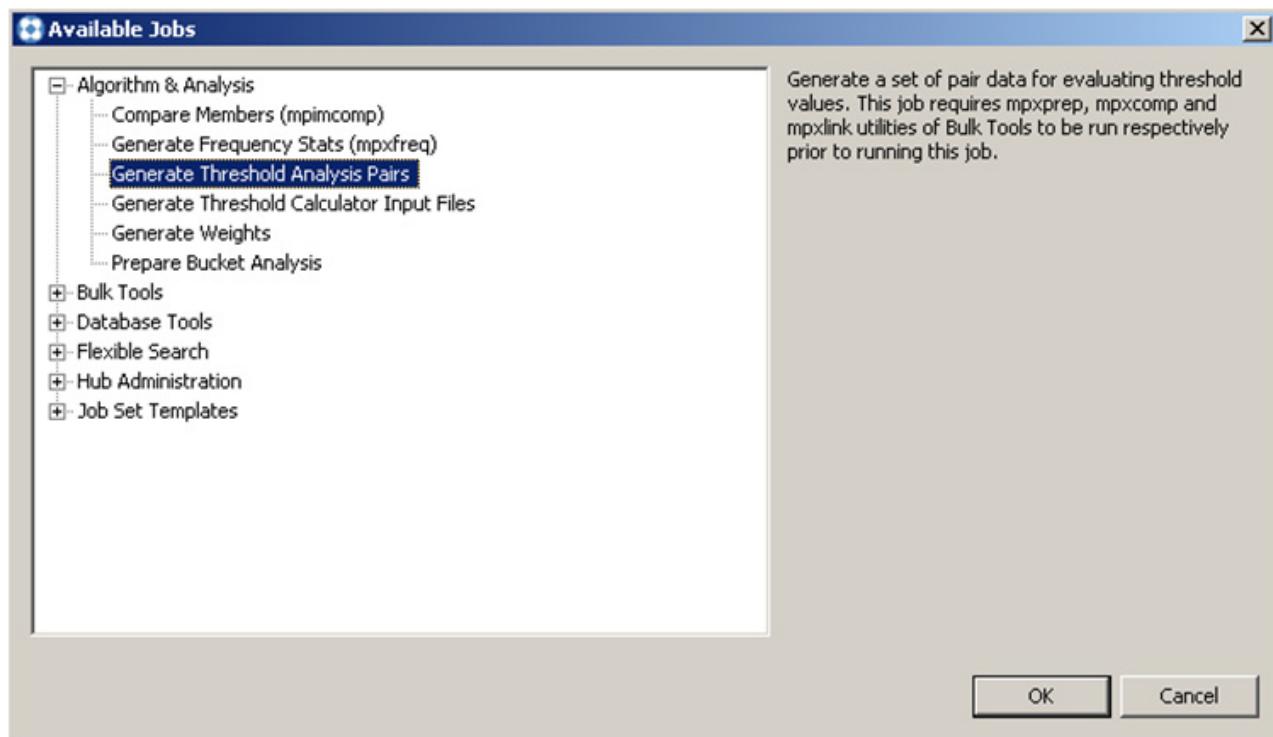
- ___ 1. From the RAD file menu, select **Master Data Management > New Job Set...**
- ___ 2. Enter **Virtual Sample Pairs** as the job set template name and click the **Next >** button.



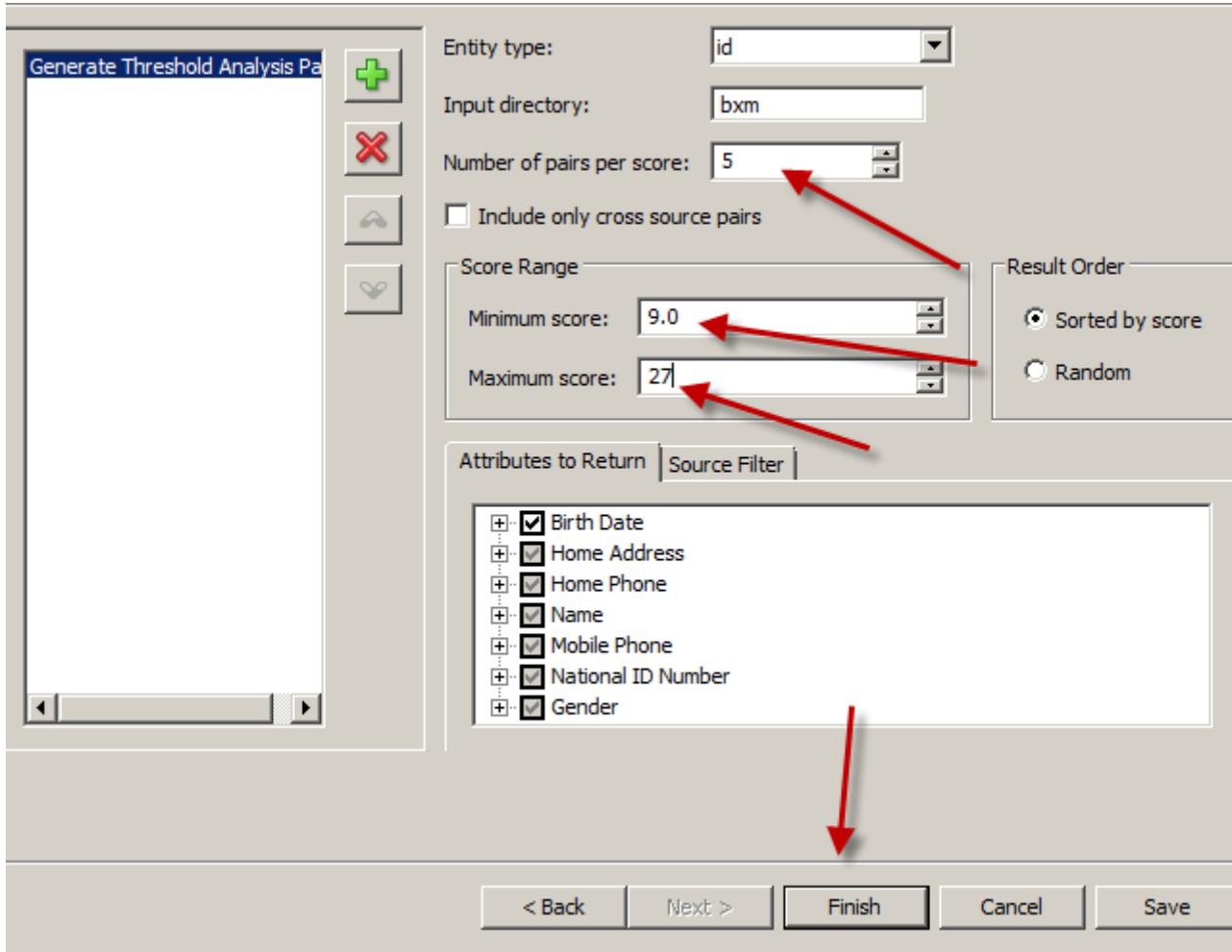
- ___ 3. Click the **Add Job** (the green plus sign) button.



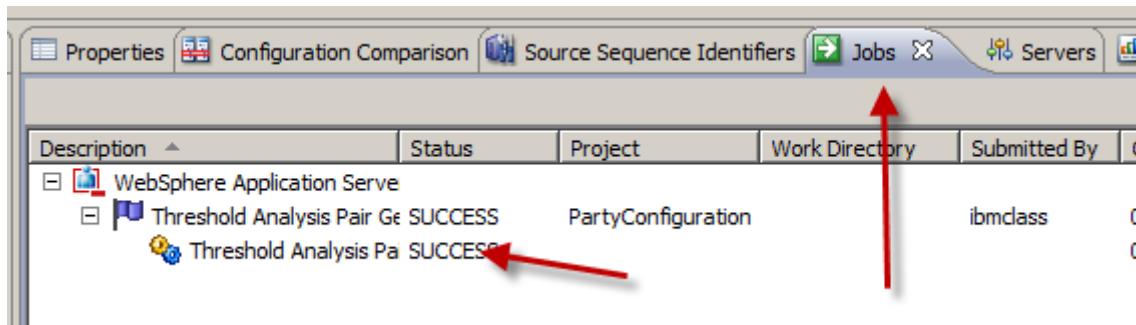
- ___ 4. Expand Algorithm & Analysis, select **Generate Threshold Analysis Pairs** and click the **OK** button.



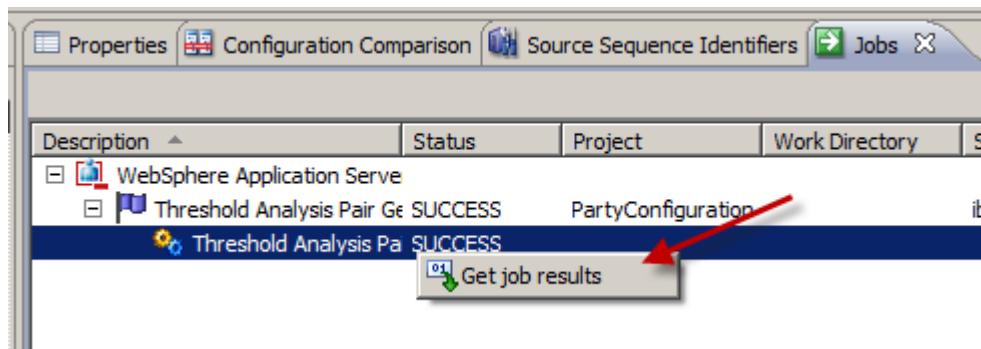
- ___ 5. Change the following values and click on the **Finish** button:
- ___ a. *Number of pairs per score: 5.*
 - ___ b. *Minimum Score: 9.0.*
 - ___ c. *Maximum Score: 27.0.*



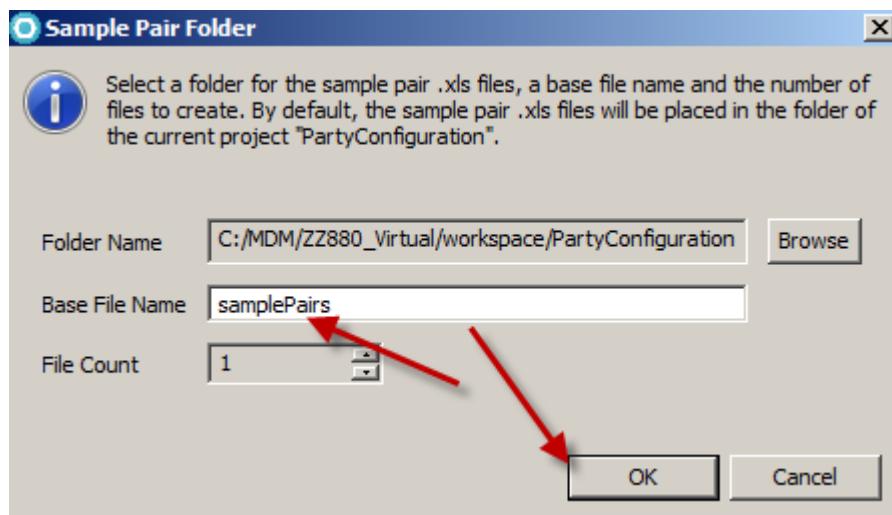
6. Validate that the job has started by verifying in the **Jobs** window.



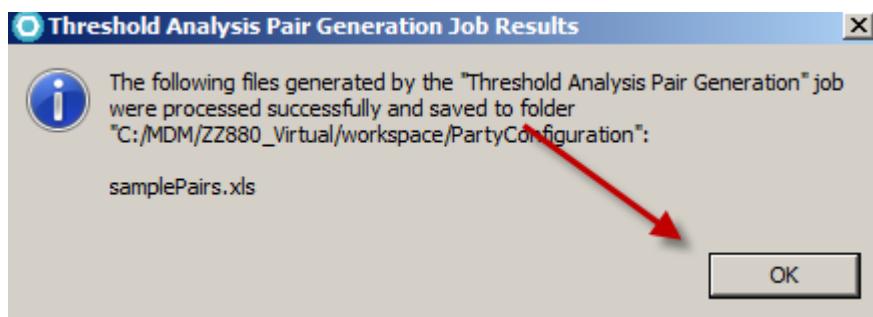
7. When job completes, right click on **Threshold Analysis Pair Generation** and select **Get job results**.



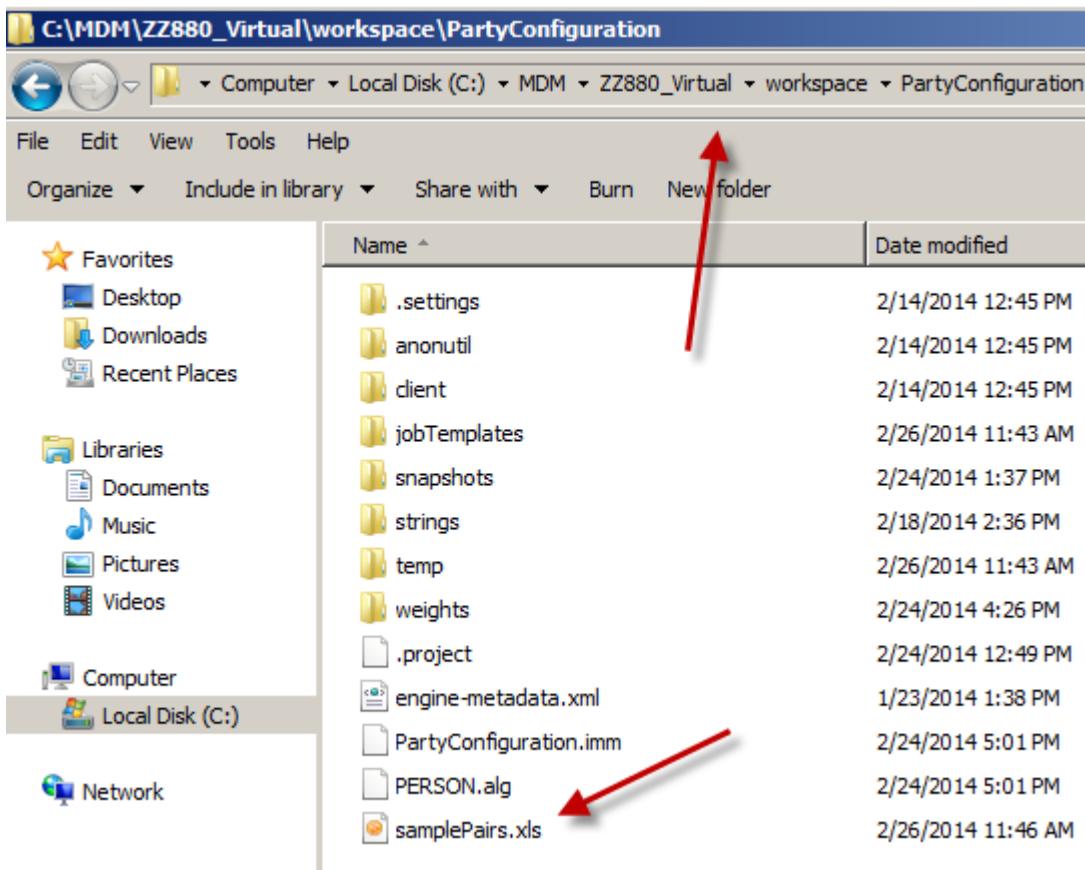
- 8. Click **OK** to use **samplePairs** as the Base File Name and to put the file in the default Folder Name.



- 9. Click **OK** to accept / complete the job and copy the samplePairs.xls file into your project.



- 10. Open a windows explorer and navigate to
C:\MDM\ZZ880_Virtual\workspace\PartyConfiguration. Open the **samplePairs.xls** spreadsheet.

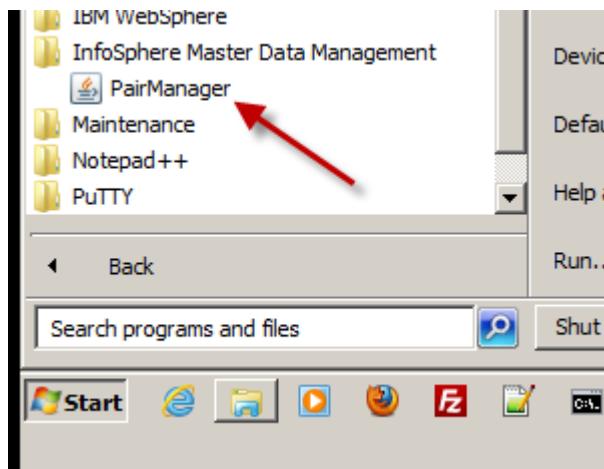


11. Inside this file, you'll find pairs of member that are to be compared. This manual comparison will help the InfoSphere MDM learn which threshold values should be in place. You can go through the spreadsheet and change column A to Yes, No or Maybe, or we can use the Pair Manager.

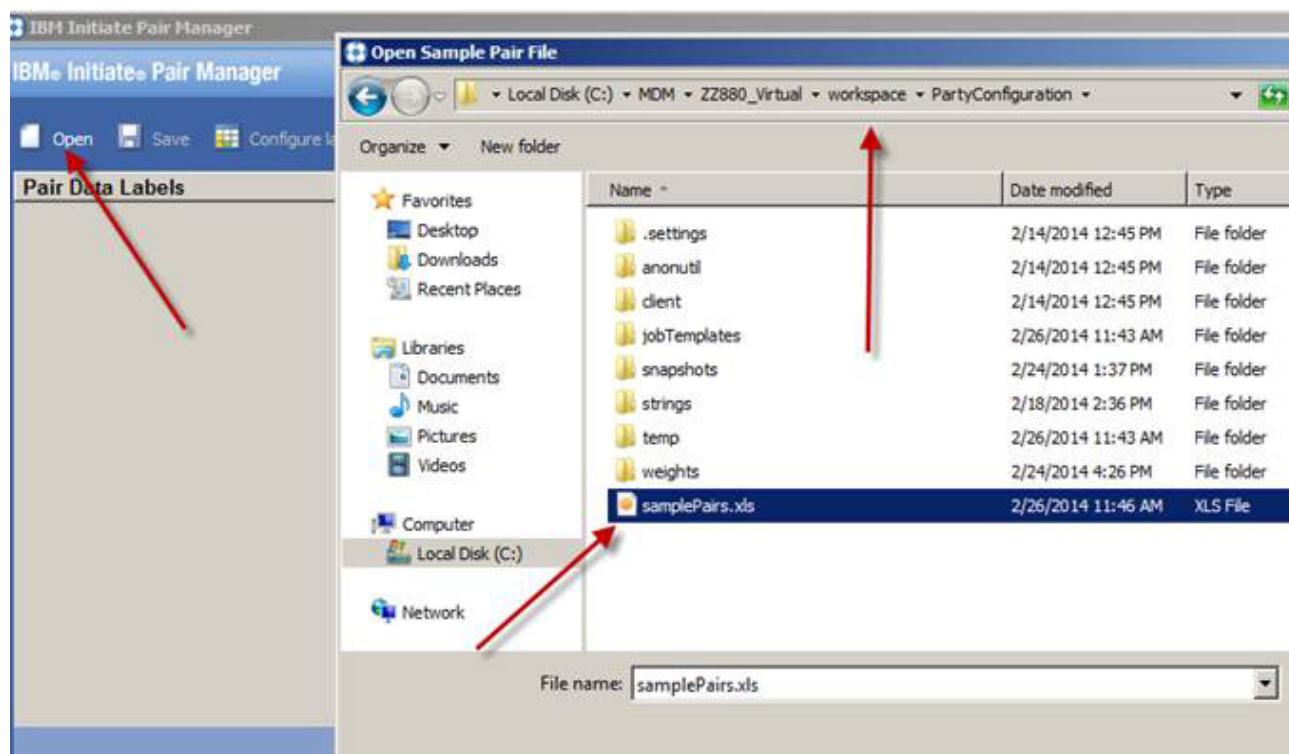
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1	Same?	Pair number	Score	Original score	memrecno	Source	Has more data?	Birth Date	Home Add1	Home Add2	Home Add3	Home Add4	Home Add5	Home Add6	Home Add7
2	Yes	1	90	90	117580	Archway	FALSE	1900-05-30	PHOENIX	AZ	220 BROOKLYN AVE.				85016
3		1	90	90	399633	Bellwood	FALSE								
4	Yes	2	90	90	298959	Bellwood	FALSE	1897-03-24	GLENDALE	AZ	30725 BASIN ST.				85302
5	No	2	90	90	372914	Bellwood	FALSE								
6	Maybe	3	90	90	133170	Archway	FALSE	1901-01-01	TEMPE	AZ	10646 HELENA ST.				85280
7		3	90	90	403739	Bellwood	FALSE								
8		4	90	90	274979	Bellwood	FALSE	1890-07-01	PHOENIX	AZ	9871 DUELS ST.				85021
9		4	90	90	441124	Bellwood	FALSE								
10		5	90	90	5366	Archway	FALSE	1900-04-19	GLENDALE	AZ	31179 BENJAMIN ST.				85308
11		5	90	90	370574	Bellwood	FALSE	1900-04-19							
12		6	91	91	161246	Archway	FALSE	1976-02-01	QUEEN CRE	AZ	24444 BELLECHASSE ST.				85242
13		6	91	91	411728	Bellwood	FALSE								
14		7	91	91	213768	Archway	FALSE	1939-02-09	PHOENIX	AZ	22605 SCOTLAND ROAD				85006
15		7	91	91	425171	Bellwood	FALSE								
16		8	91	91	61609	Archway	FALSE	1899-02-23	PHOENIX	AZ	31324 HAGAN AVE.				85026
17		9	91	91	284000	Bellwood	FALSE	1900-02-22							

Part 2: Review Threshold Analysis sample pairs

1. Start the Pair Manager from the Windows Start menu. Select **Start > All programs > InfoSphere Master Data Management > Pair Manager**



2. Click the **Open** icon and navigate to C:\MDM\ZZ880_Virtual\workspace\PartyConfiguration



3. Select **samplePairs.xls** and click **Open** to display the first row of sample pairs.

Pair Data Labels	Member #1	Member #2
Score	90	90
Original score	90	90
memrecno	117580	399633
Source	Archway	Bellwood
Birth Date	dateVal	1900-05-30
Home Address	city	PHOENIX
	state	AZ
	stLine1	220 BROOKLYN AVE.
	stLine2	
	stLine3	
	stLine4	
	zipCode	85016
Home Phone	phNumber	4802765328
Name	onmFirst	KIZZY MOYR

___ 4. Review the sample pair set and indicate:

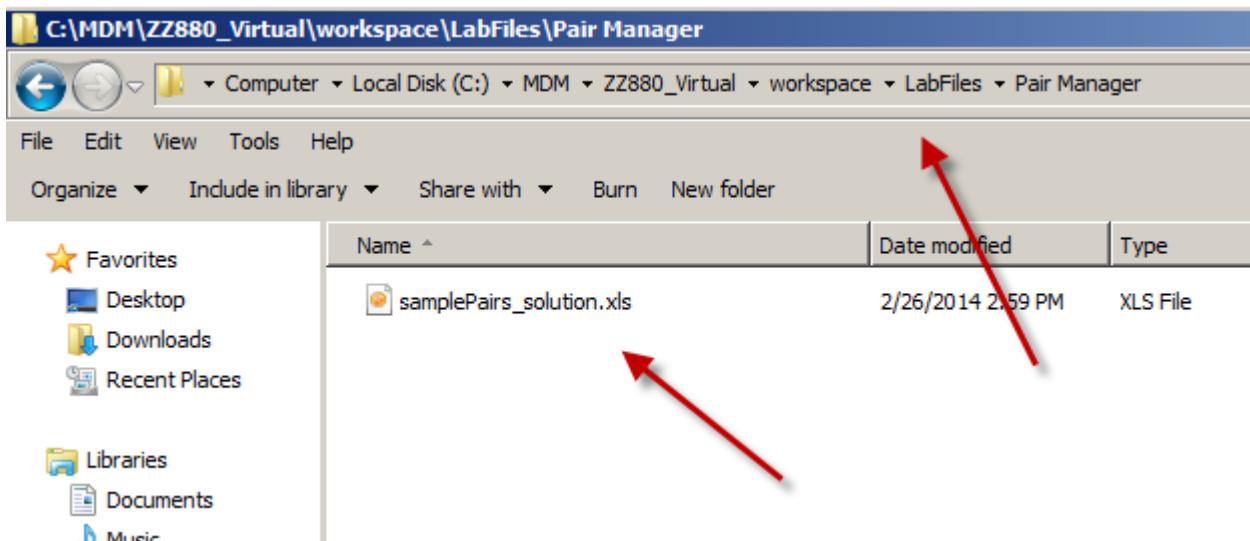
- **Yes** , green checkmark to say that they are a matching pair.
- **No** , red line through circle to say that they are NOT a matching pair.
- **Maybe** , yellow question mark to say that there is not enough information for you to decide if they are a matching pair or not.

(NOTE: you don't have to review all of them, we have a solution file for you)

___ 5. Once you feel comfortable with the Pair Manager and how it works, save the file, you will notice that it also produces an xls file. (The xls file is exactly the same as the one we loaded with the first column populated during your reviews)

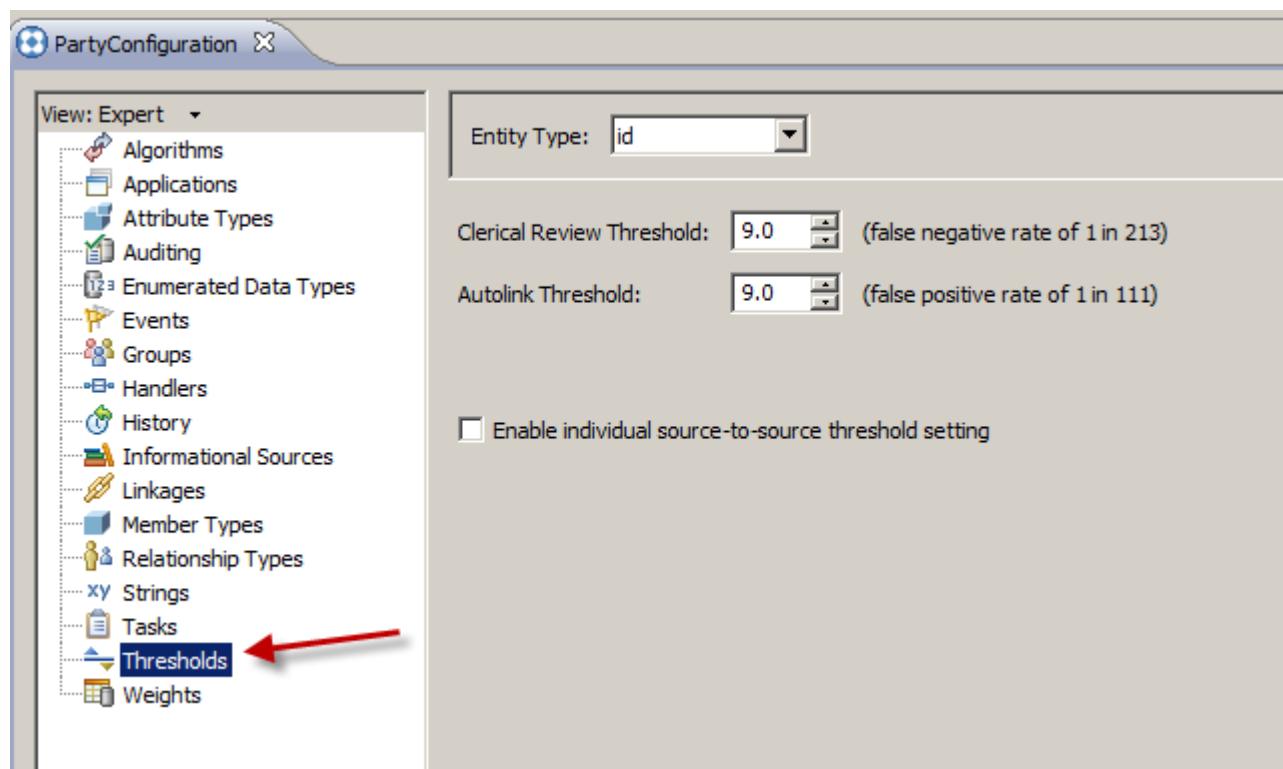
We have provided you with a solution file that can be found under

C:\MDM\ZZ880_Virtual\workspace\LabFiles\PairManager\samplePairs_solution.xls. We will use this file for our Threshold Calculations,

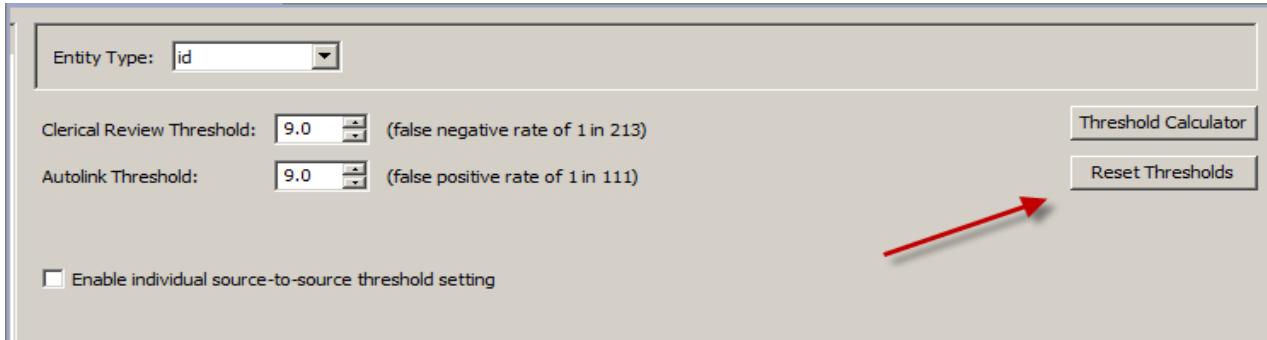


Part 3: Setting new thresholds using Threshold Calculator

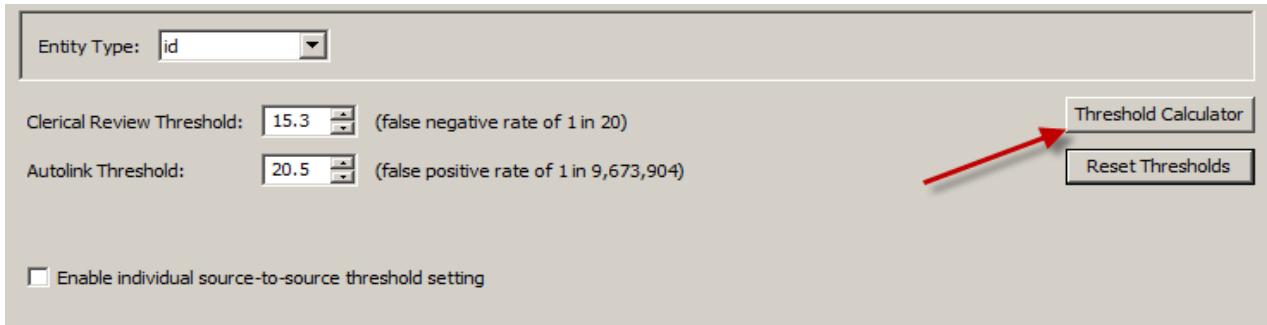
- Back in the RAD environment, open the PartyConfiguration.imm file and select the **Thresholds** tab.



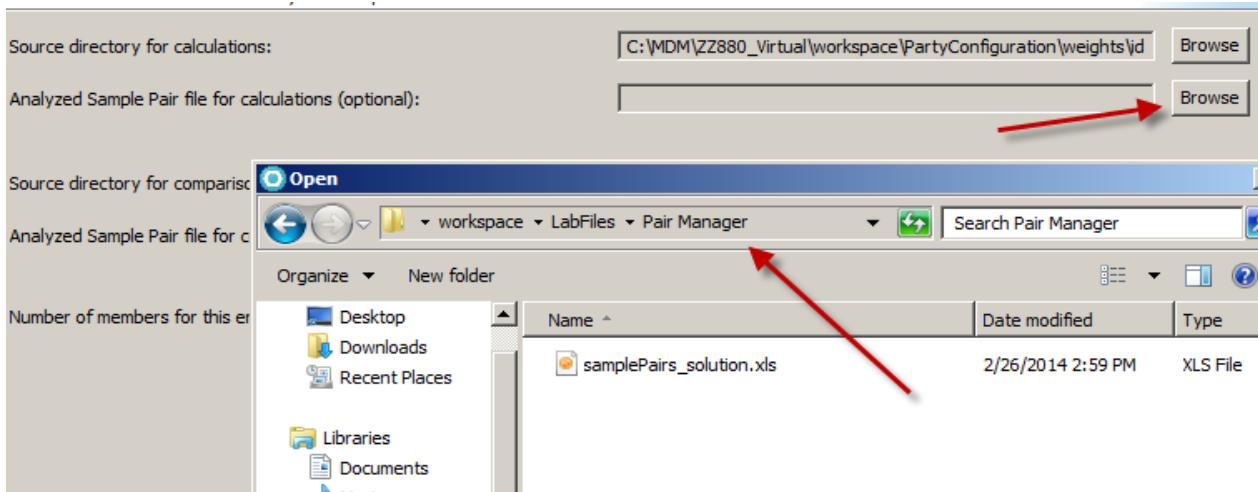
- Click the **Reset Thresholds** button to first let the system determine what the thresholds should be set to without input.



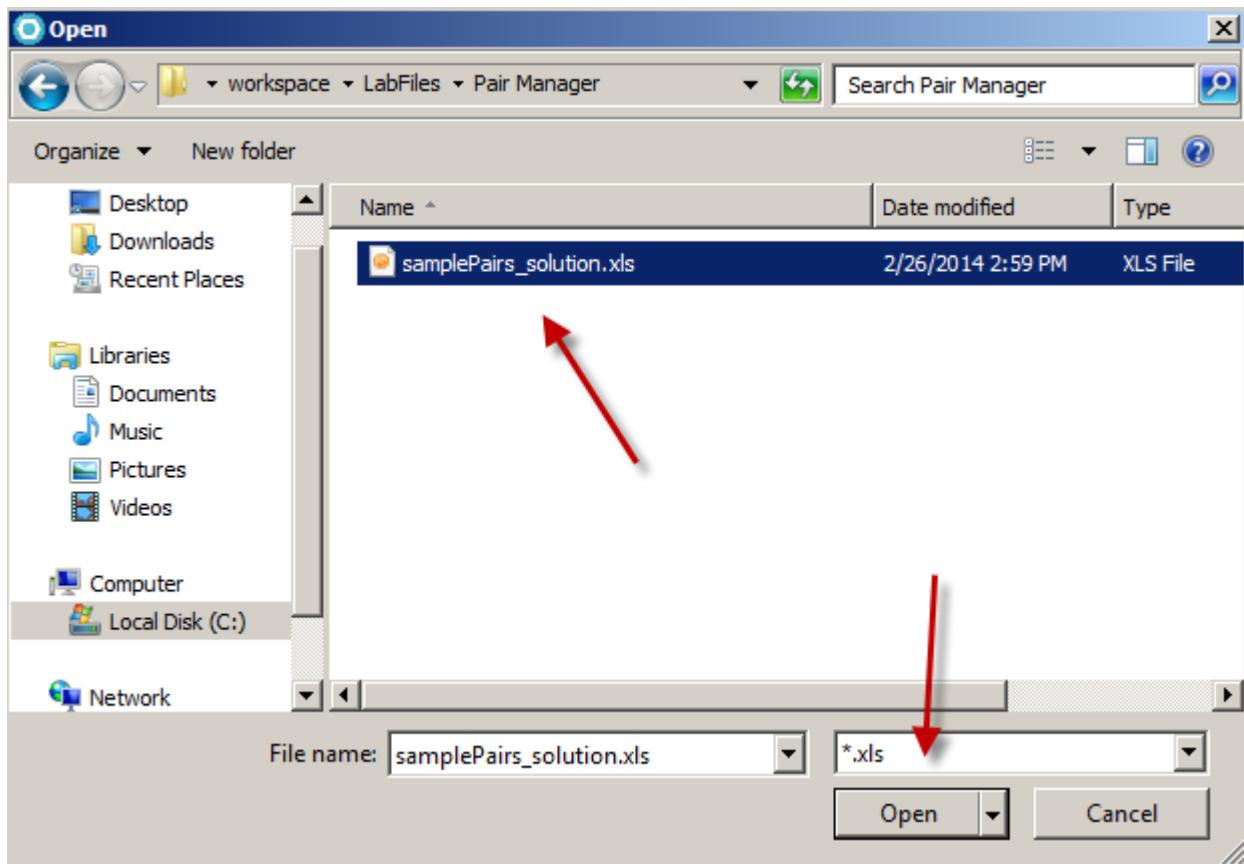
- ___ 3. Click the **Threshold Calculator** button.



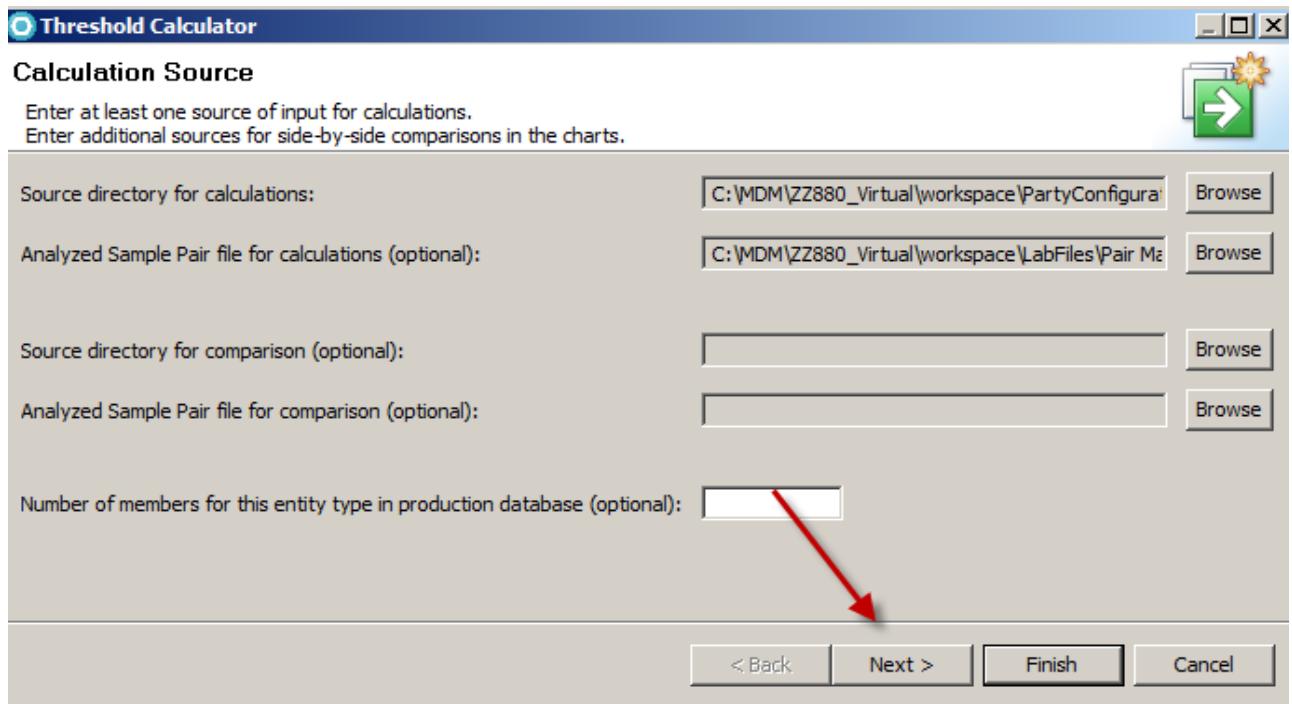
- ___ 4. Click the **Browse** button to change property **Analyzed Sample Pair file for calculations (optional)**: navigate to the directory C:\MDM\ZZ880_Virtual\workspace\LabFiles\Pair Manager

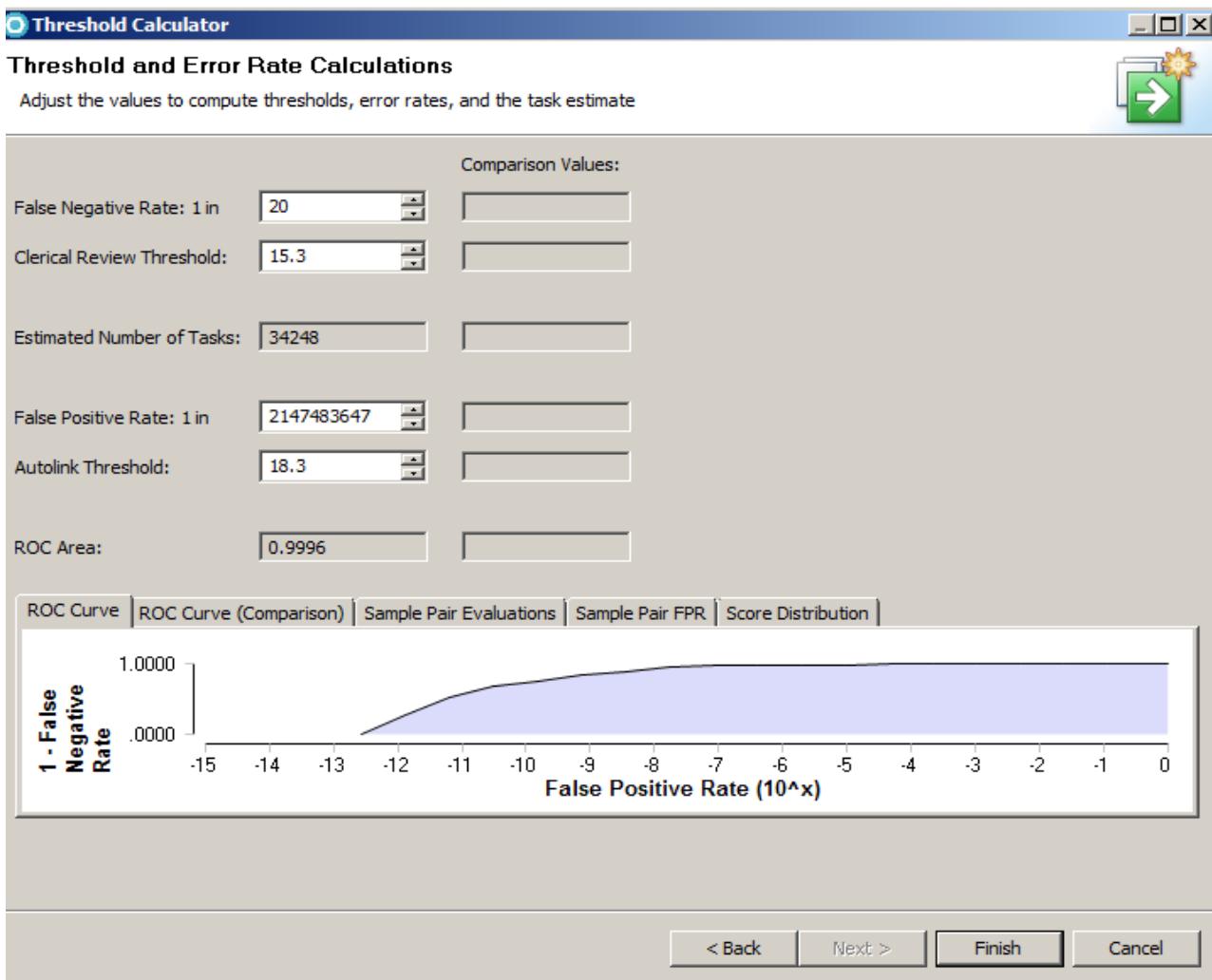


- ___ 5. Select **samplePairs_solution.xls** and click **Open**.

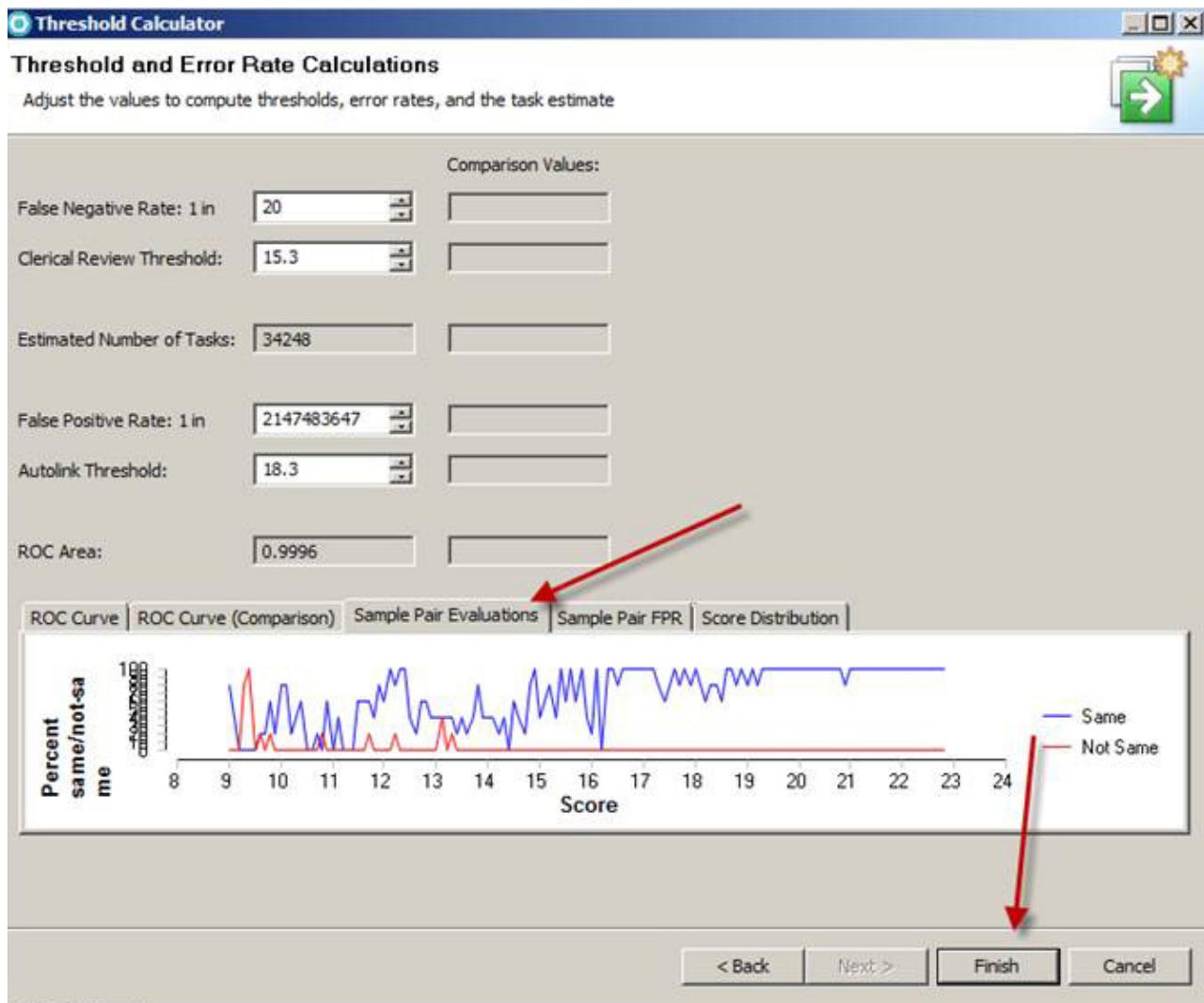


- 6. Click the **Next >** button to display the ROC (Receiver Operating Characteristics) curve.

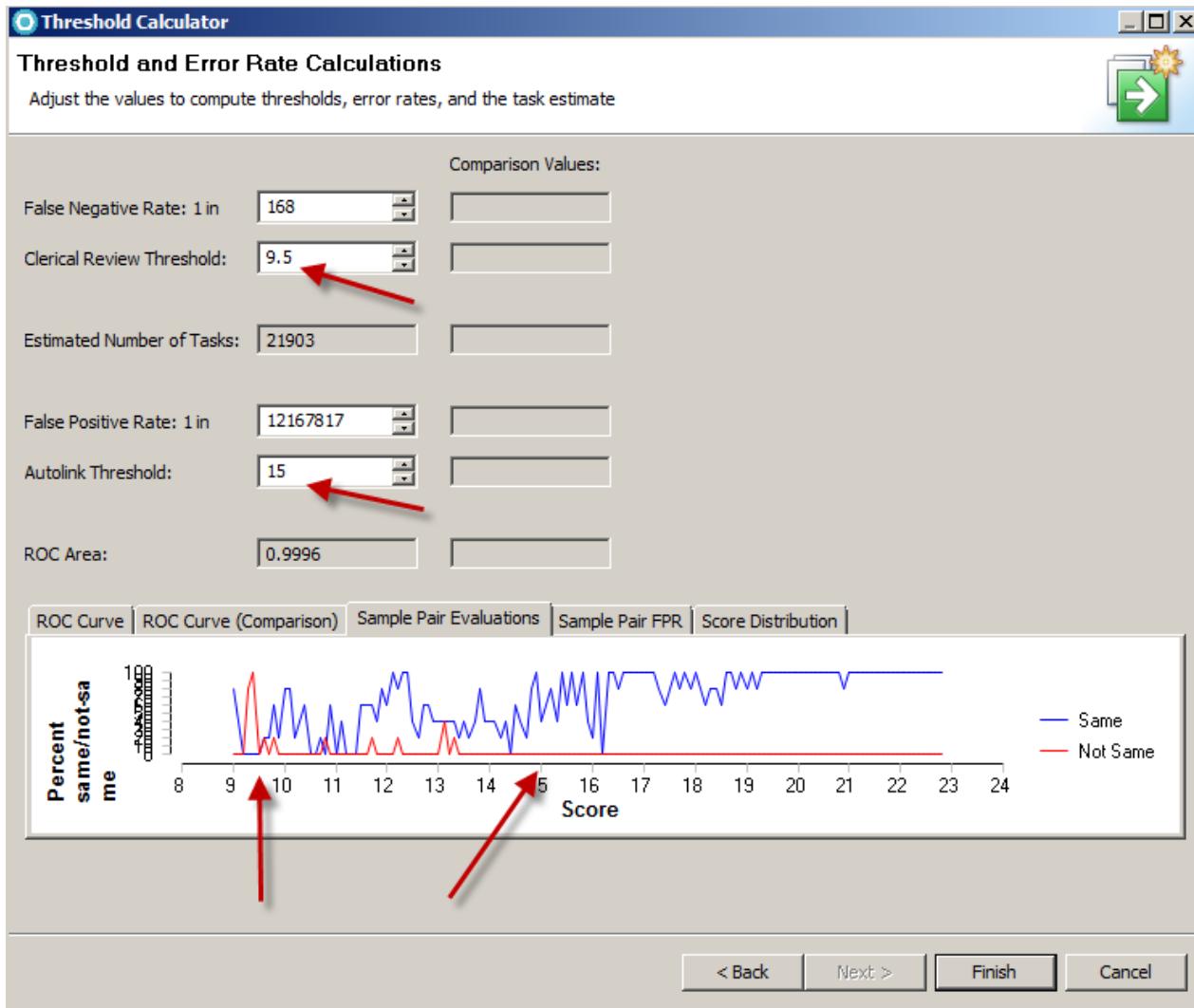




7. Select the **Sample Pair Evaluations** tab to view the results of the Pair Manager output.



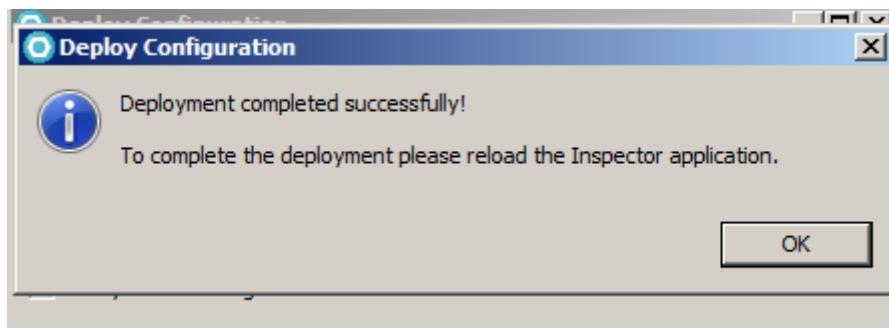
8. These graphs allow you to evaluate what would be a good threshold based on the sample data. Based on a quick analysis, it looks like between 9.5 and 15 there are a mix of 'Same' and 'Not Same', so we would like to keep that range as our Clerical Review. Set the property **Clerical Review Threshold** to 9.5 and the **Autolink Threshold** to 15 and click on the **Finish** button to set the thresholds.



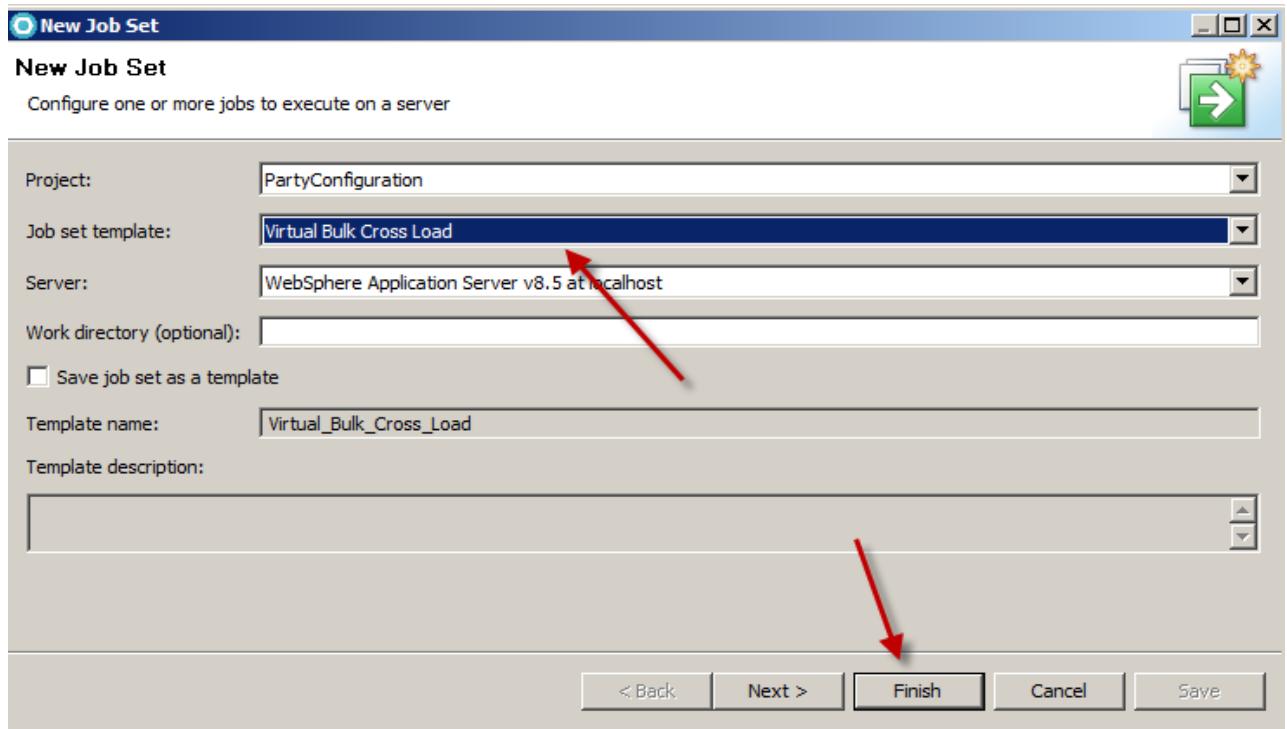
___ 9. Save the project (CTRL+S)

Part 4: Re-deploy the configuration

- ___ 1. Redeploy the configuration (from the RAD menu, select **Master Data Management > Deploy Hub Configuration....**)



- ___ 2. Re-run the Virtual Cross Load job (from the RAD menu, select **Master Data Management > New Job Set**)



End of exercise

Exercise 6c. Testing Our Algorithms

What this exercise is about

This exercise covers viewing our Entities using the reporting tools available in the workbench.

What you should be able to do

At the end of this exercise, you should be able to:

- View an Entity distribution report
- View the Member comparison data of an Entity
- View the Entity Member breakdown

Introduction

Now that we have created our thresholds and re-ran the bulk cross load, we can view how Entities that were linked and created in our solution. We will use the analysis section of the workbench similar to our Bucket analysis exercise.

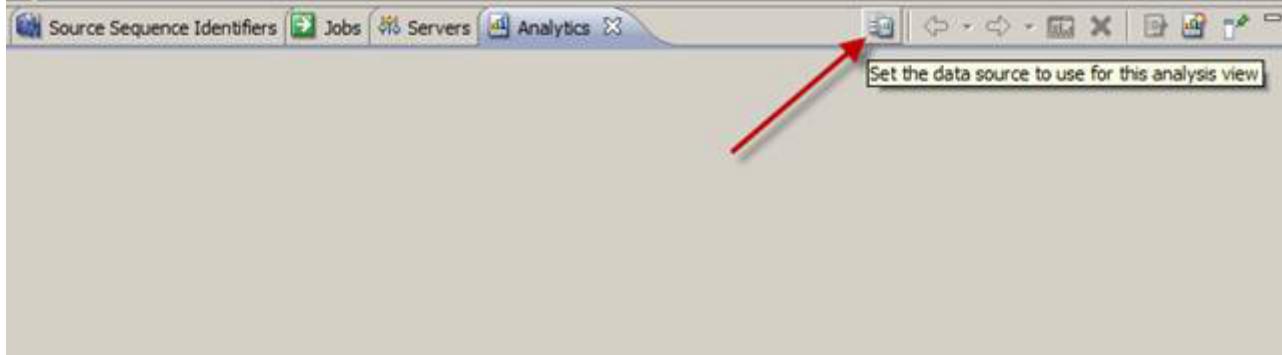
Requirements

Each of these exercise depend on the previous exercise being complete. For this exercise you must have the data model and algorithms defined in the configuration file. The configuration must be deployed with sample data loading into the database. The weights must be generated and a bulk cross match and load process must have been run on the sample data.

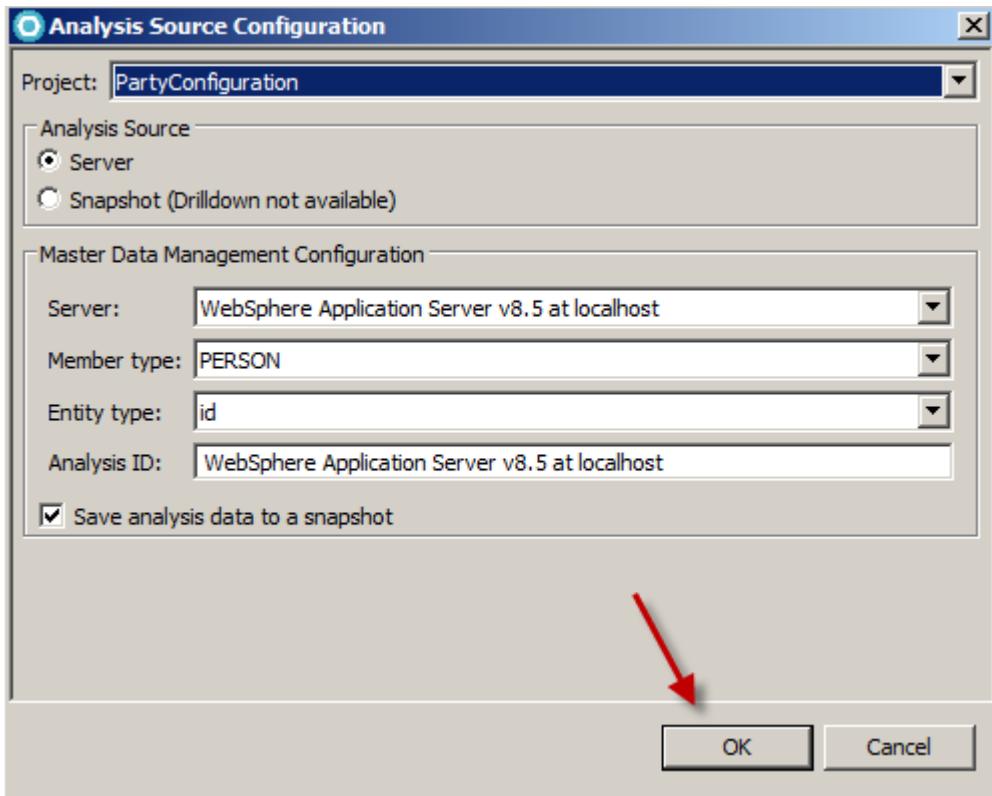
Exercise instructions

Part 1: Entity analysis overview

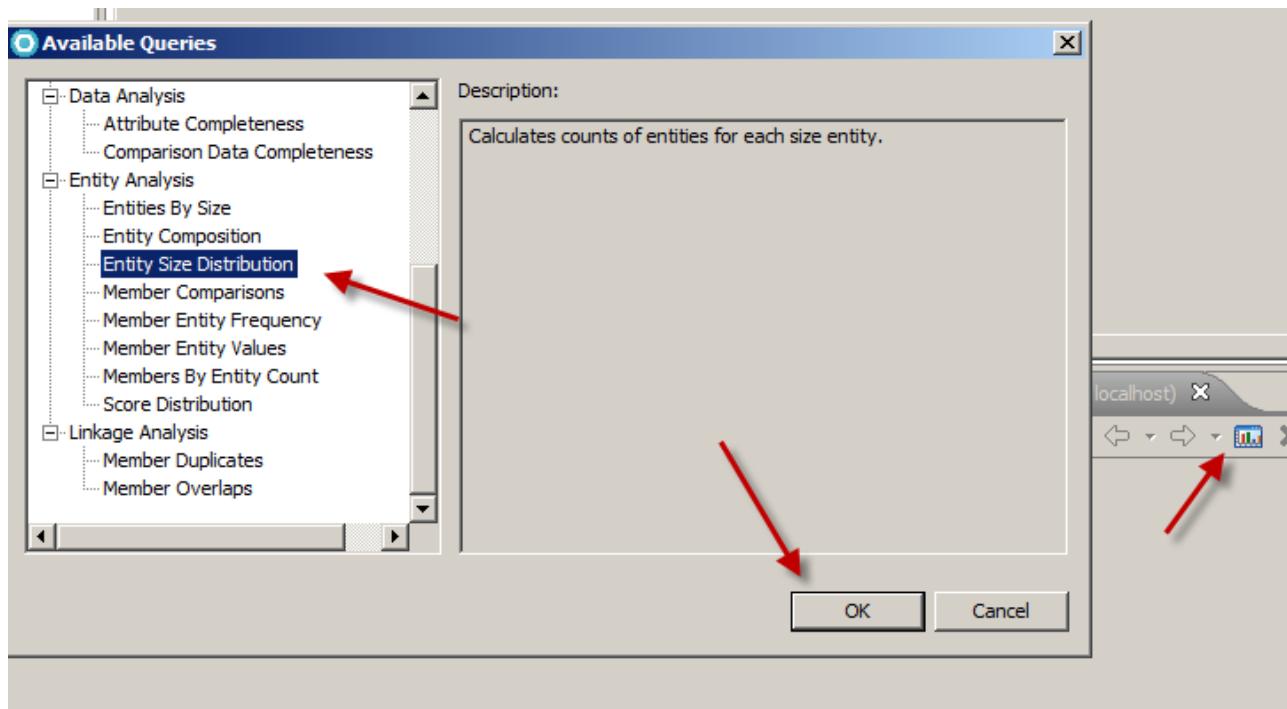
- 1. Open the **Analytics** window in RAD and if not already set, click on the **Set the data source to use for this analysis view** button.



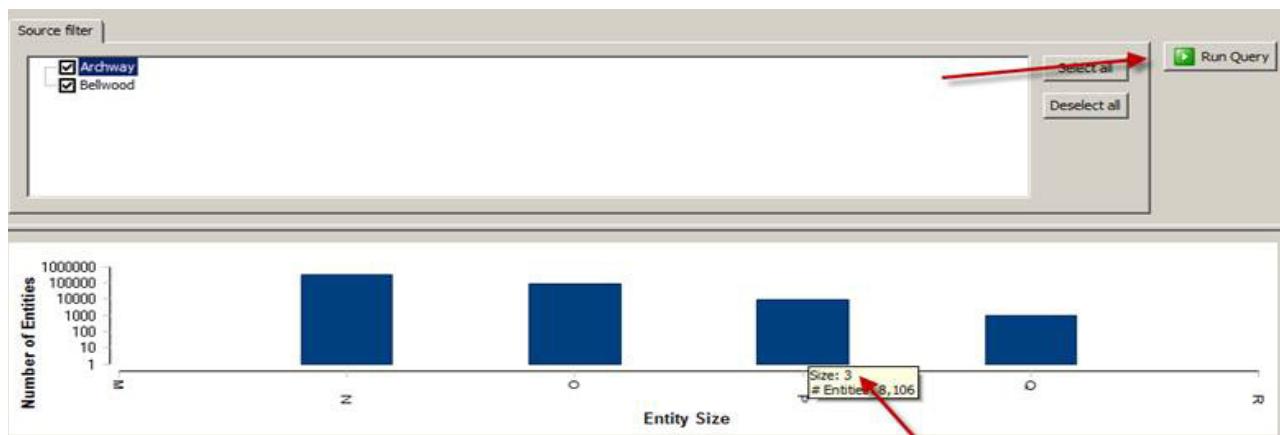
- 2. Accept the default values and click the **OK** button



- 3. Click the Add Query button and select the **Entity Size Distribution** query found under the Entity Analysis folder. Click the **OK** button.

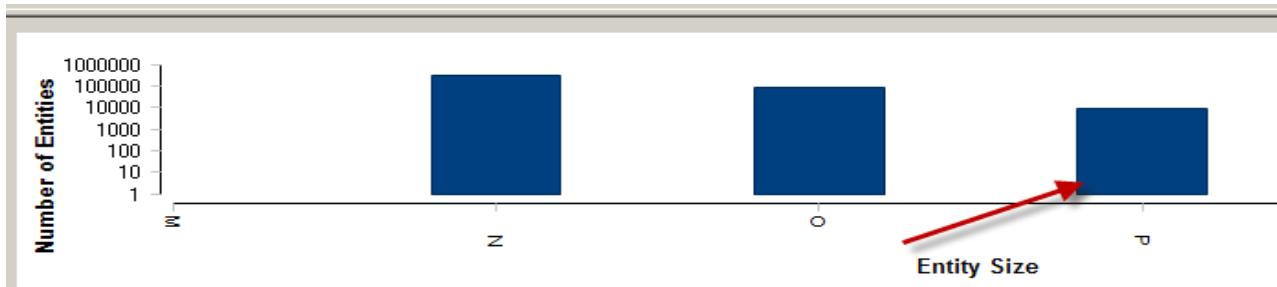


4. Click the **Run Query** button



From this report you can see that we have a number of entities made up of 1 Member, 2 Members, 3 Members and 4 Members. The Entities with 2, 3, and 4 have been created based on our algorithm and have the comparison score > 15. You can drive down the query by clicking on one of the bars to see the details.

5. Click on the 3rd bar in the chart.



6. Select **Entity 27** and click the **View Entity** button.

(Found 8106 entities)

Number of Members	Entity ID
3	27
3	28
3	31
3	34
3	41
3	48
3	49
2	50

From this screen you can see the multiple members that make up the Entity 27

Query Parameters |

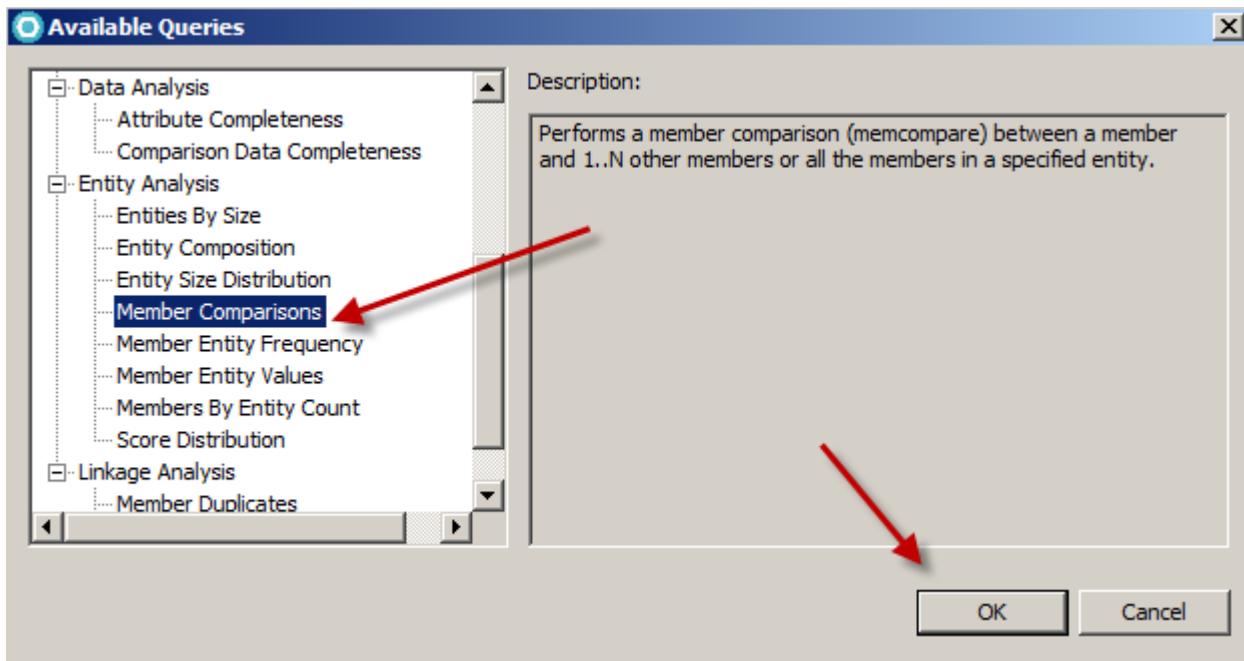
Entity ID:

(Found 3 members)

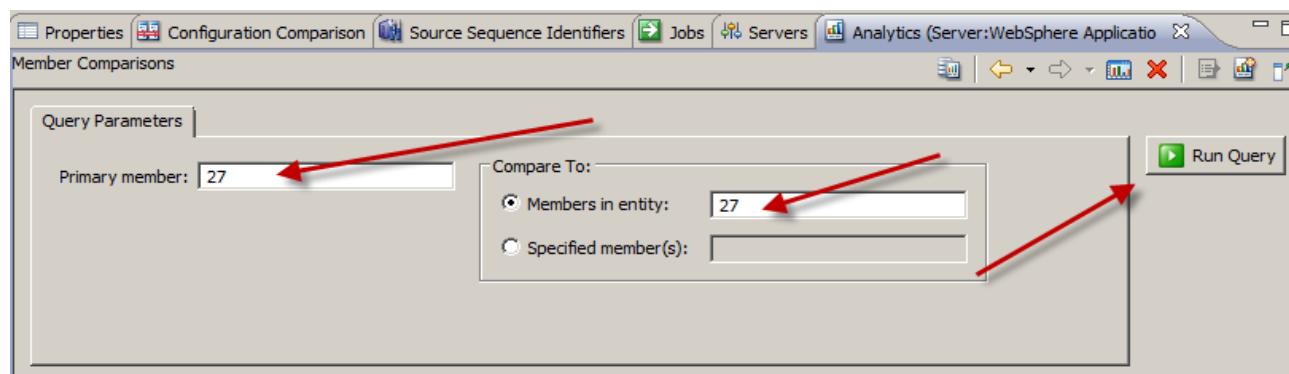
Member	Source	1	2	3	4	5	6	7
27	1	271276576	F	DREWES:LUIS:E..	DREWES::..	19421024	85309	N-25232:S-BOL
136786	1	271276576	F	DREWES:LUIS:E..	DREWES::..	19421024	85309	N-25232:S-BOL
404692	2	271276576	F	DREWES:LUIS:E..	DREWES::..	19421024		

Part 2: Member comparison

1. Click the Add Query icon again, but this time select Member Comparisons from the Entity Analysis, folder. Click the **OK** button.



2. Change the **Primary Member** and **Members in entity** fields to **27** (a Member from the Entity we saw in the last part). Click the **Run Query** button.



Here you can see the breakdown of the match between the member 27 and the other members in the Entity.

...	Score	cmpspeccode	Proband Value	Candidate Value	Match Type
		AXP	25232 BOLTON ST LUKE AFB	25232 BOLTON ST LUKE AFB	E
22.7					
21.6		NATLID	271276576	271276576	E
		SEX	F	F	E
		NAME	DREWES LUIS E	DREWES LUIS E	E
		DOB	19421024	19421024	E
		AXP	2783292	2783292	P

3. To see where the scores come from, open the **mpi_wgt1dim.unl** file from the **PartyConfiguration > weights > id** folder in RAD. In this file you'll find a match on the National ID provides a score of 5.71.

99|99|A|CMPID-NATLID-DIST|0|0|
99|99|A|CMPID-NATLID-DIST|1|571| 1
99|99|A|CMPID-NATLID-DIST|2|388|
99|99|A|CMPID-NATLID-DIST|3|314|
99|99|A|CMPID-NATLID-DIST|4|177|
99|99|A|CMPID-NATLID-DIST|5|32|
99|99|A|CMPID-NATLID-DIST|6|-88|
99|99|A|CMPID-NATLID-DIST|7|-203|
99|99|A|CMPID-NATLID-DIST|8|-251|
99|99|A|CMPID-NATLID-DIST|9|-290|
99|99|A|CMPID-NATLID-DIST|10|-352|
99|99|A|CMPID-NATLID-DIST|11|0|

- ___ 4. Under the **mpi_wgtsval.unl**, you'll find an agreement weight for the gender provide 0.30 value to the score.

99|99|A|CMPID-SEX-XACT|M|30|
99|99|A|CMPID-SEX-XACT|a|30| a
99|99|A|CMPID-SEX-XACT|d|-392| d

- ___ 5. For the name we match on all values DREWES (3.89), LUIS (3.53) and E (1.44). However since there is a **__FULLNAME_MAXWGT** or 5.83, we only receive 5.83 + (3x20) for the positional bonus = 6.43

99|99|A|CMPID-NAME-XACT|LUIS|354|
99|99|A|CMPID-NAME-XACT|MARTY|354|
99|99|A|CMPID-NAME-XACT|MELVIN|354|
99|99|A|CMPID-NAME-XACT|a|389| a
99|99|A|CMPID-NAME-XACT|d|-102| d
99|99|A|CMPID-NAME-PARM|__FULLNAME_MAXWGT|583| 583
99|99|A|CMPID-NAME-PARM|__INITIAL_ADJWGT|202|
99|99|A|CMPID-NAME-PARM|__INITIAL_MINWGT|50|
99|99|A|CMPID-NAME-PARM|__INITIAL_MAXWGT|389|
99|99|A|CMPID-NAME-PARM|__PHONETIC_ADJWGT|18|

- ___ 6. Under the DOB, if we open the **mpi_wgtnval.unl**, we'll find that we get a score of 4.74 for matching the year 1942.

Exercise 7a. Reset the MDM database

What this exercise is about

In this exercise, we will reset the MDM database. The reason to reset the database is that we will be working with a new algorithm using the Virtual MDM in order to configure the Physical PME algorithm. Since the previous configuration has added values and new tables, we will reset to have an empty database as a starting point. NOTE: the project with the configuration for the Virtual MDM will still exist and can be redeployed again if that is needed (this will not destroy any of our previous work)

What you should be able to do

At the end of this exercise, you should be able to:

- Reset the MDM database

Introduction

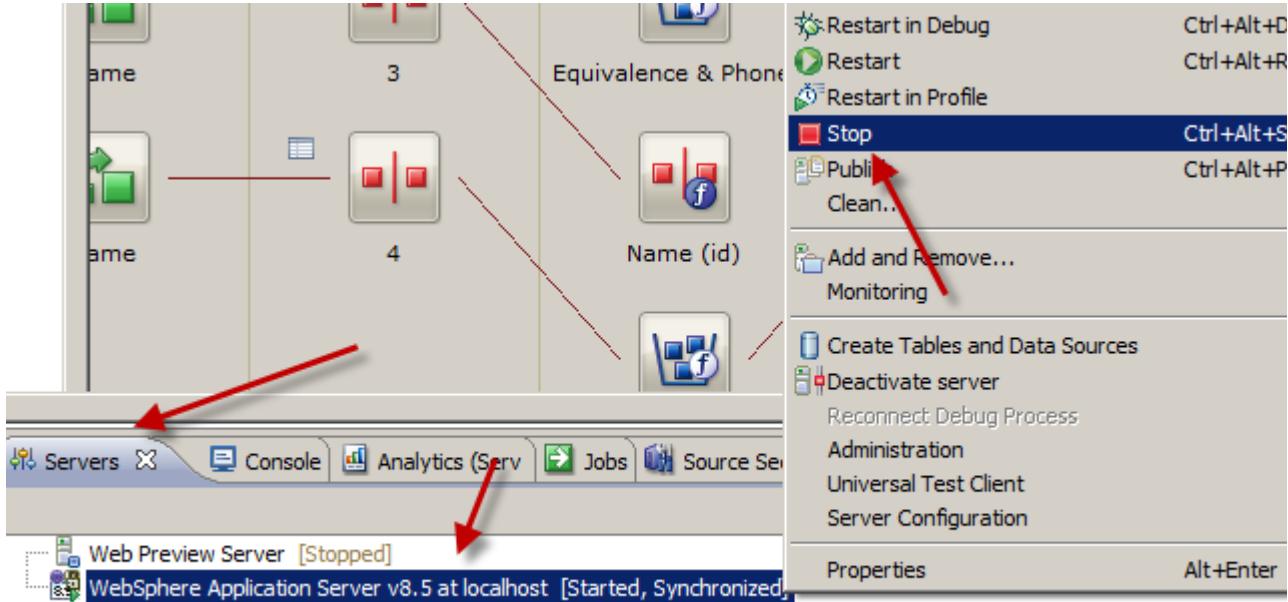
The MDM workbench provides the capabilities to reset the database and server back to its original state after installation. Since we have not modified the Application Server (e.g. deploying new OSGi bundles), we will only need to reset the database to get a clean environment.

Exercise instructions

Part 1: Stop the WAS server

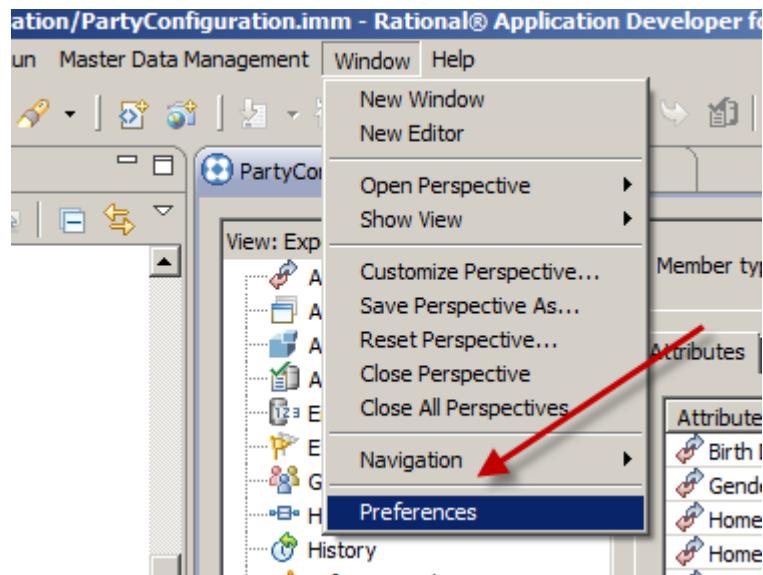
Before we reset the database we need to stop the WAS server so that there are no open connections to the database.

- Under the Servers tab in RAD, right click on the WebSphere Application Server and select Stop.

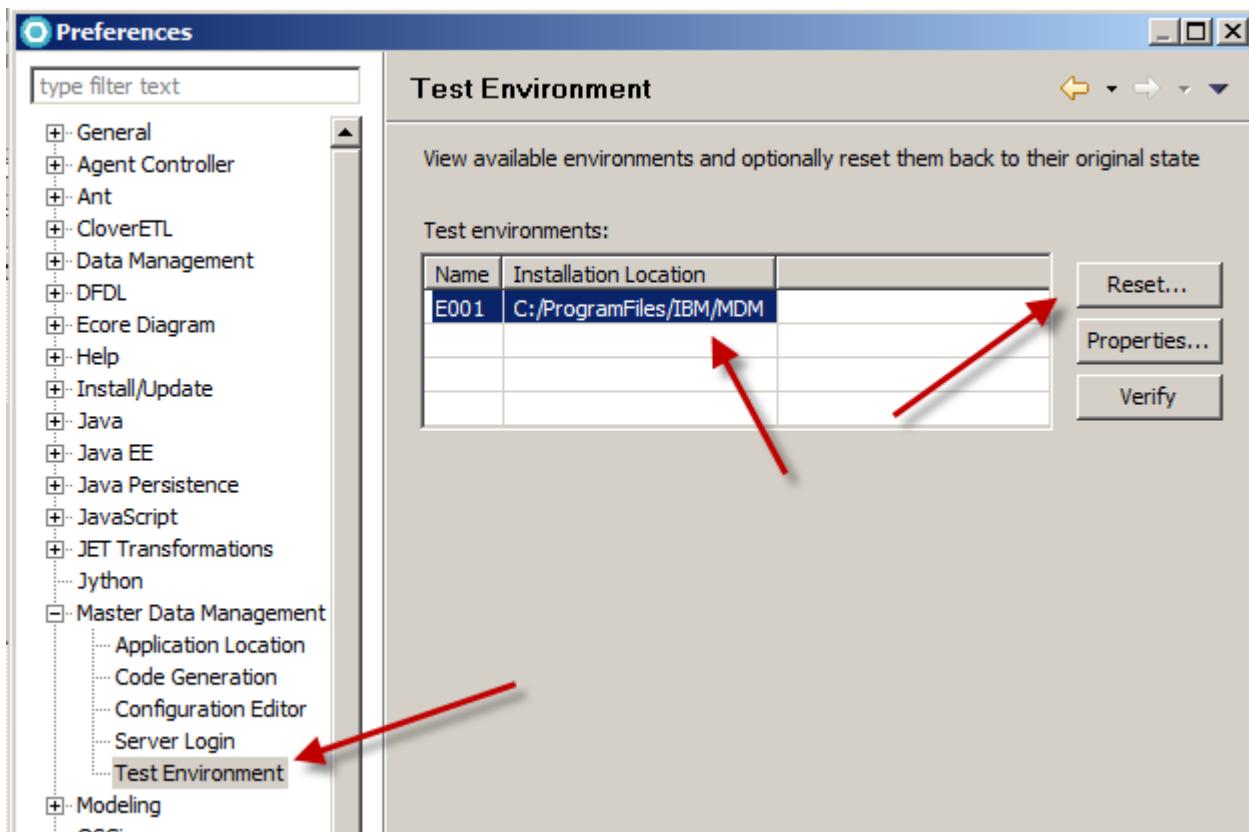


Part 2: Database reset

- Inside the RAD environment select **Window > Preferences**



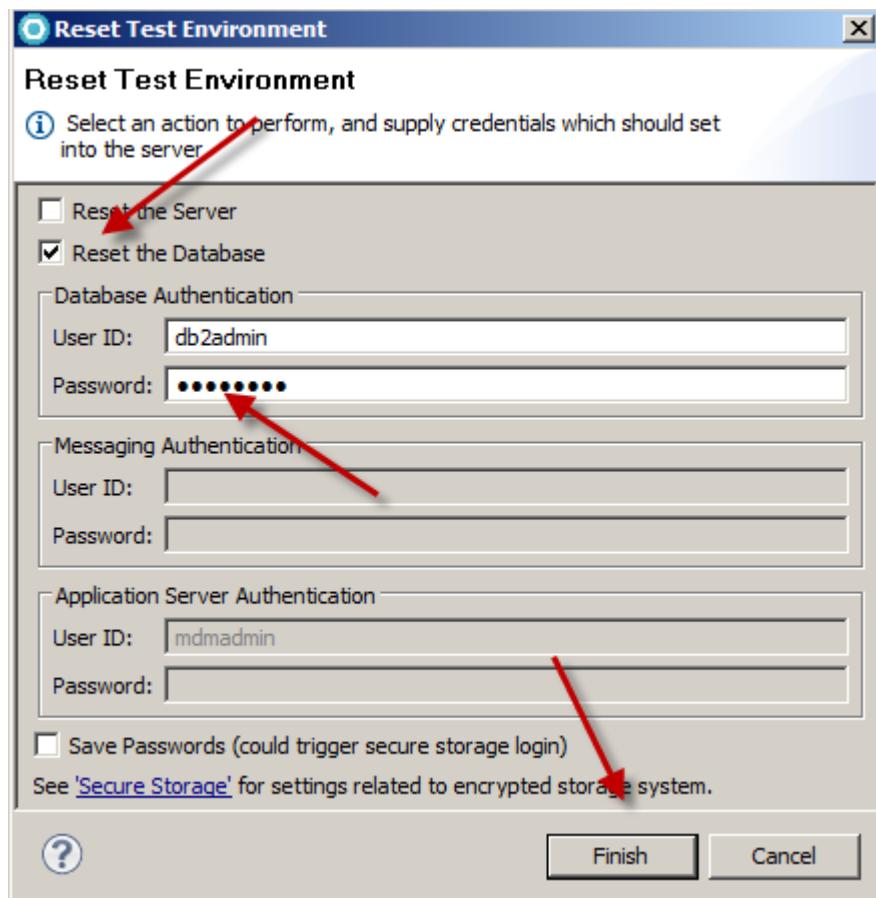
2. Navigate to **Master Data Management > Test Environment**. Select the **E001** environment and click the **Reset ...** button.



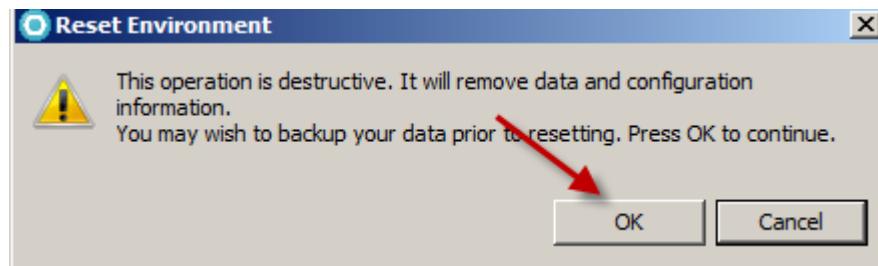
3. Enter **mdmadmin** for the Password and click the **OK** button.



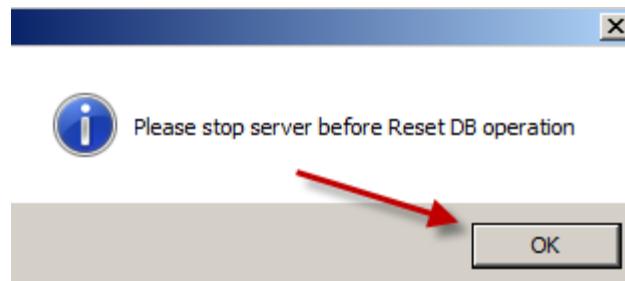
- 4. Select the **Reset the Database** checkbox, enter **db3Admin** for the database password and click the **Finish** button.



- 5. Click the **OK** button on the warning popup.



- ___ 6. Click the **OK** button on the stop server warning.

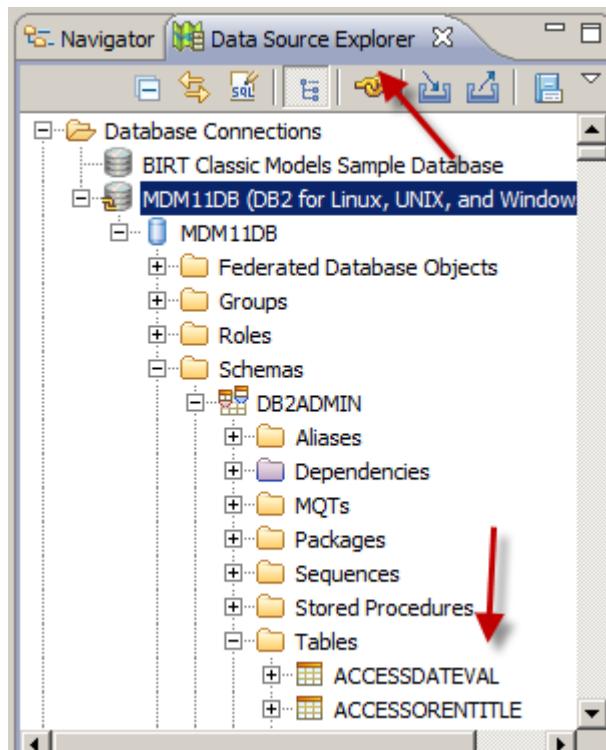


The reset will take ~5-10 min. Wait until it is complete until you move on to the next part of the exercise.

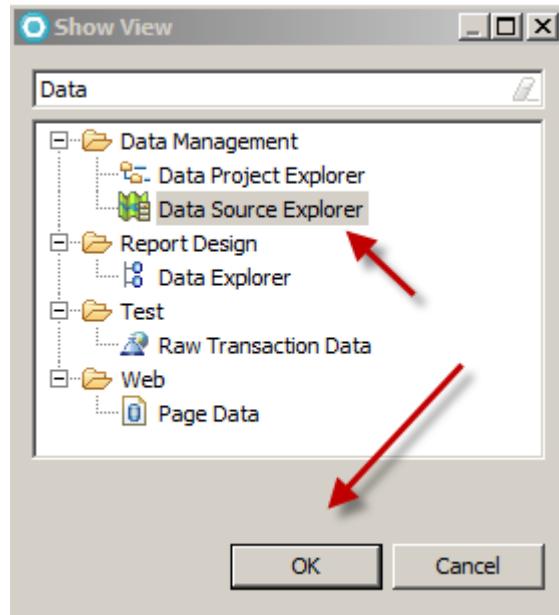
Part 3: Setting the multi timezone settings

Once the database is done resetting, we will need to change one of the configuration in the CONFIGELEMENT table that is not set during the reset (Multi-time zone feature)

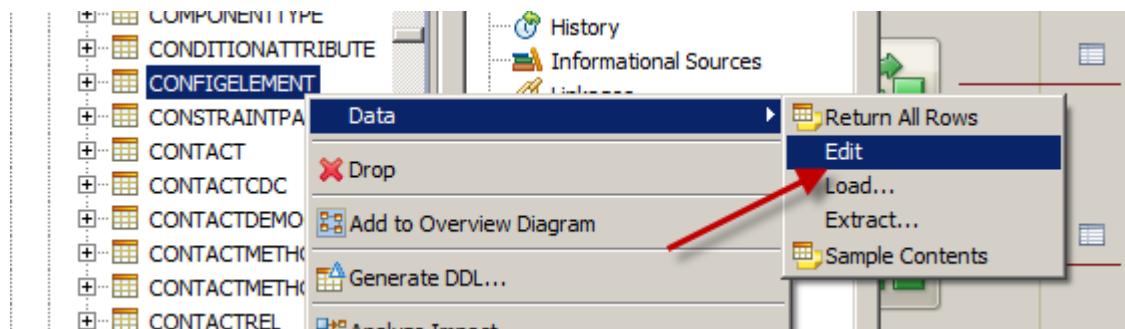
- ___ 1. Open the MDM11DB database (under the Data Source Explorer window) and navigate to the **CONFIGELEMENT** table.

**Note**

If the Data Source Explorer window is not available, open the view by select **Window > Show View > Other ...** and selecting the **Data Source Explorer** view.



2. Right click on the **CONFIGELEMENT** table and select **Data > Edit**



- ___ 3. Navigate to the **/IBM/DWLCommonServices/MultiTimeZoneDeployment/enabled** (element id = 1275) configuration and change the VALUE to **false**. Save the changes.

PartyConfiguration				
MENT_ID [BIGINT]	DEPLOYMENT_ID [BIGINT]	NAME [VARCHAR(220)]	VALUE [VARCHAR(1000)]	VALUE_DEFAULT [VARCHAR(1000)]
'2	1000	/IBM/DWLCommonSe...		false
'3	1000	/IBM/DWLCommonSe...		true
'4	1000	/IBM/DWLCommonSe...		false
'5	1000	/IBM/DWLCommonSe...	false	\${db.mtz.enabled}
'6	1000	/IBM/DWLCommonSe...		\${db.mtz.tz}
'7	1000	/IBM/DWLCommonSe...		1
'8	1000	/IBM/DWLCommonSe...		-
'9	1000	/IBM/DWLCommonSe...		false
'0	1000	/IBM/DWLCommonSe...		false
'2	1000	/IBM/Hybrid/PersistE...		ORG
'3	1000	/IBM/Hybrid/PersistE...		A
'4	1000	/IBM/DWLCommonSe...		

- ___ 4. Save the changes.

We now have a fresh database to perform our configuration for the Physical MDM.

- ___ 5. Close the RAD environment, we will be using a new workspace for our next exercise.

End of exercise

Exercise 7b.Running Probabilistic Search with the Physical PME

What this exercise is about

In this exercise, we will add a party to the InfoSphere MDM physical module, we will then perform a number of probabilistic search to see the weight results based on different search criteria.

What you should be able to do

At the end of this exercise, you should be able to:

- Add a new Party to the physical MDM module
- Perform a probabilistic search for the party

Introduction

In the Physical PME exercises we will examine the default PME configuration and how this produces matches for the Physical Parties stored in the InfoSphere MDM.

In our high level scenario, we will work with the assumption that Driving License and the Display Name are 2 fields that our organization would like to search and match on in addition to the other default fields. These fields (Driving License and Display Name are not currently included in the default PME configuration and therefore throughout the Physical PME exercises we will customize the algorithms and configuration to includes our new search and match criteria.

In this first exercise, we will examine the output of the default PME configuration.

Requirements

- Typical Installation of the InfoSphere MDM Developers workstation

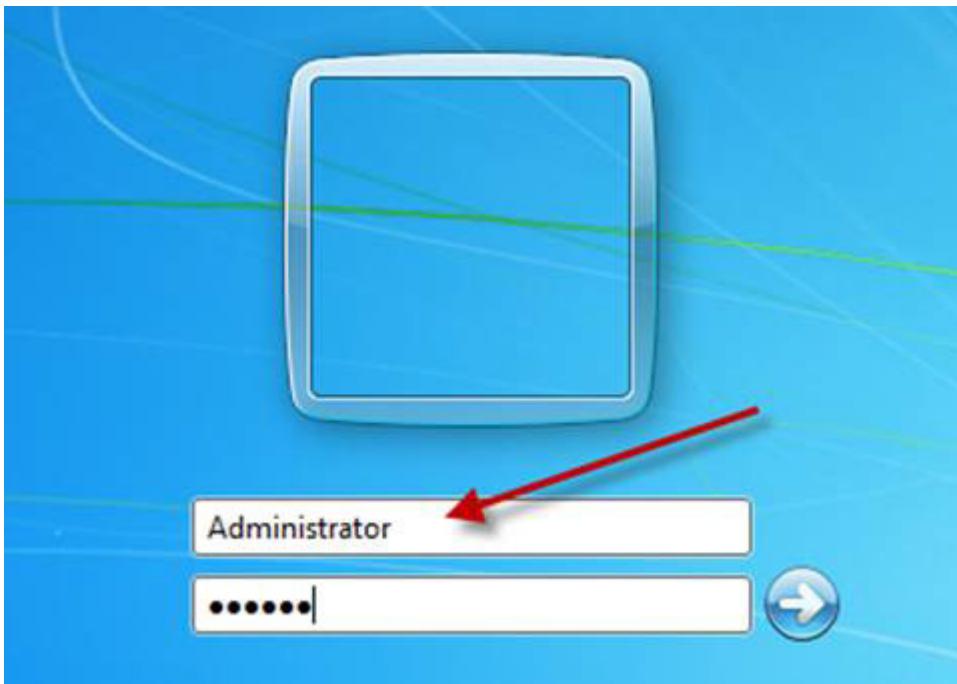
Exercise instructions

Part 1: Setup

To setup this exercise, we will need a brand new MDM RAD workspace.

If you ran the Virtual PME exercises on the image, close the ZZ880 Virtual RAD workspace

- ___ 1. Login to the Windows image using the following credentials:
 - ___ a. Username:**Administrator**
 - ___ b. Password: **passw0rd**



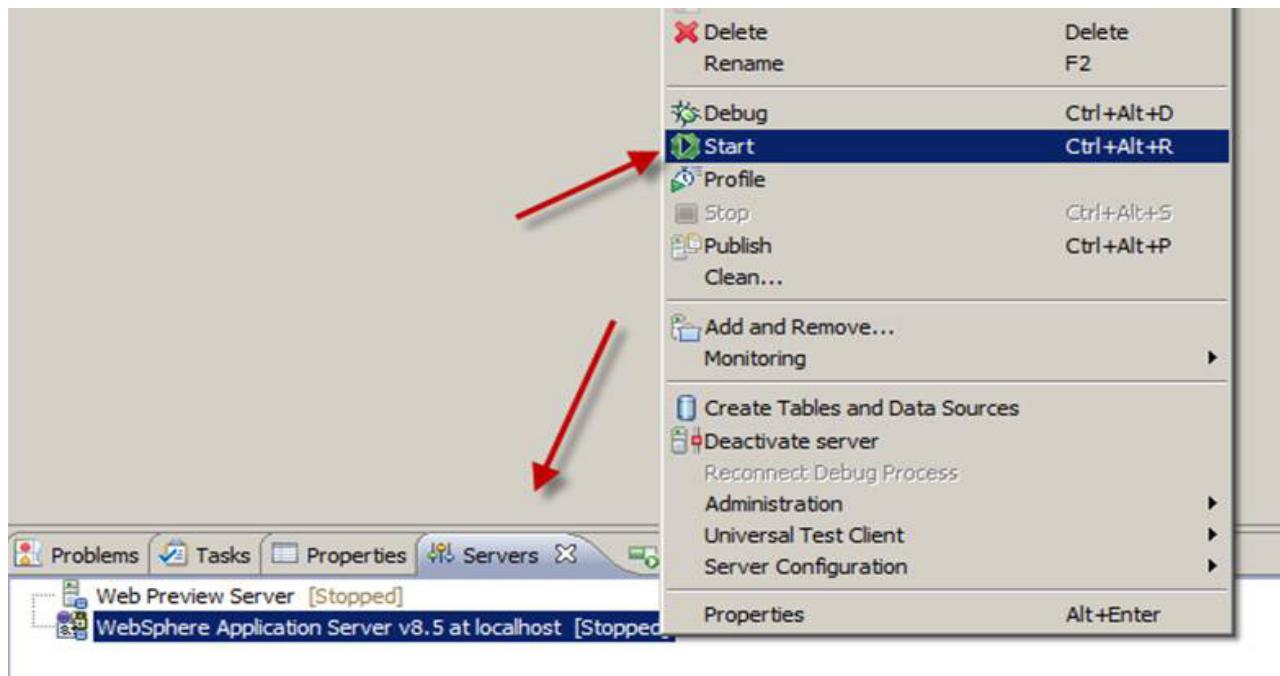
- ___ 2. On the desktop, you'll find the link to the **ZZ880 Physical Workspace**. Double click (right click and select Run) on the link.



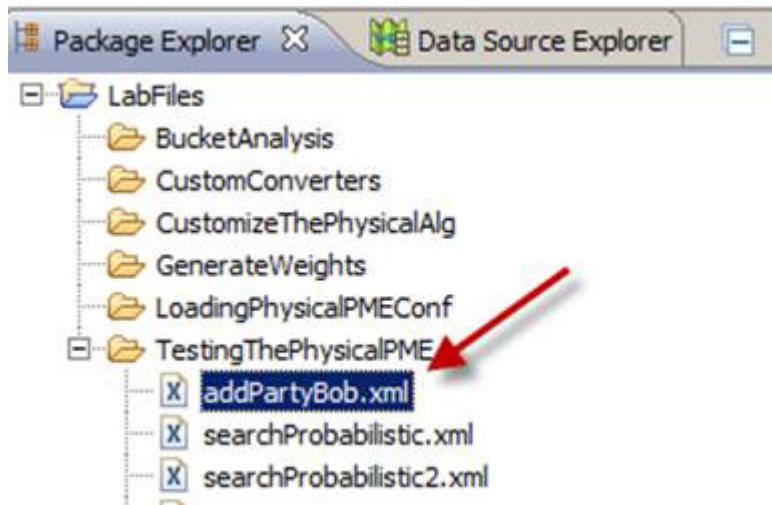
Part 2: Adding the Physical MDM record

To start off, we will add a party to the InfoSphere MDM physical module.

- ___ 1. If the server is not yet started, under the Servers tab (bottom of the RAD environment), right click on the **WebSphere Application Server** and select **Start**. (Starting the server will take 5-10 minutes). You can continue on the next couple steps while waiting for the server to start.

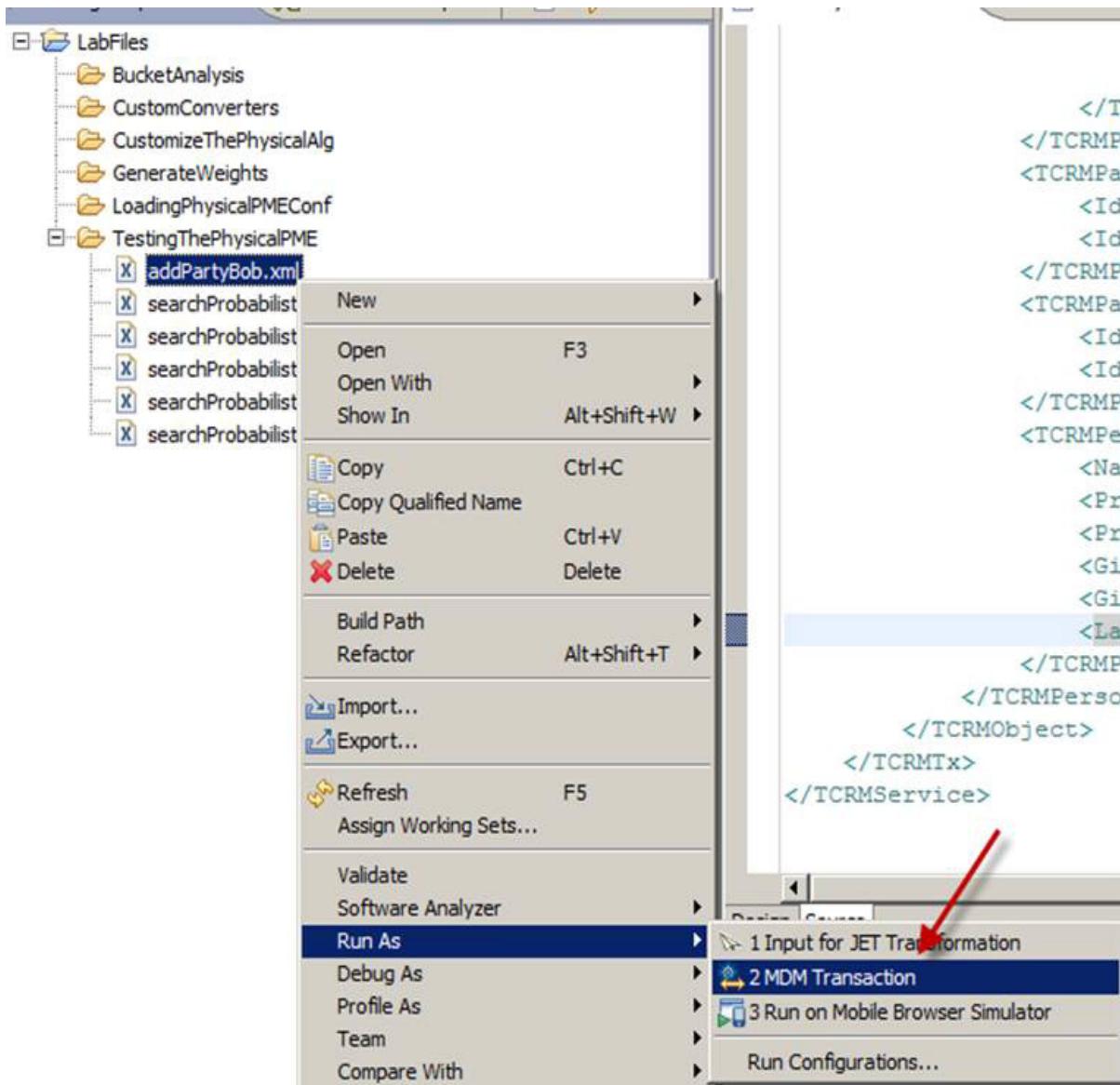


- ___ 2. Inside the **LabFiles** project, under the **TestingThePhysicalPME** folder, double click on the **addPartyBob.xml** to open the file.

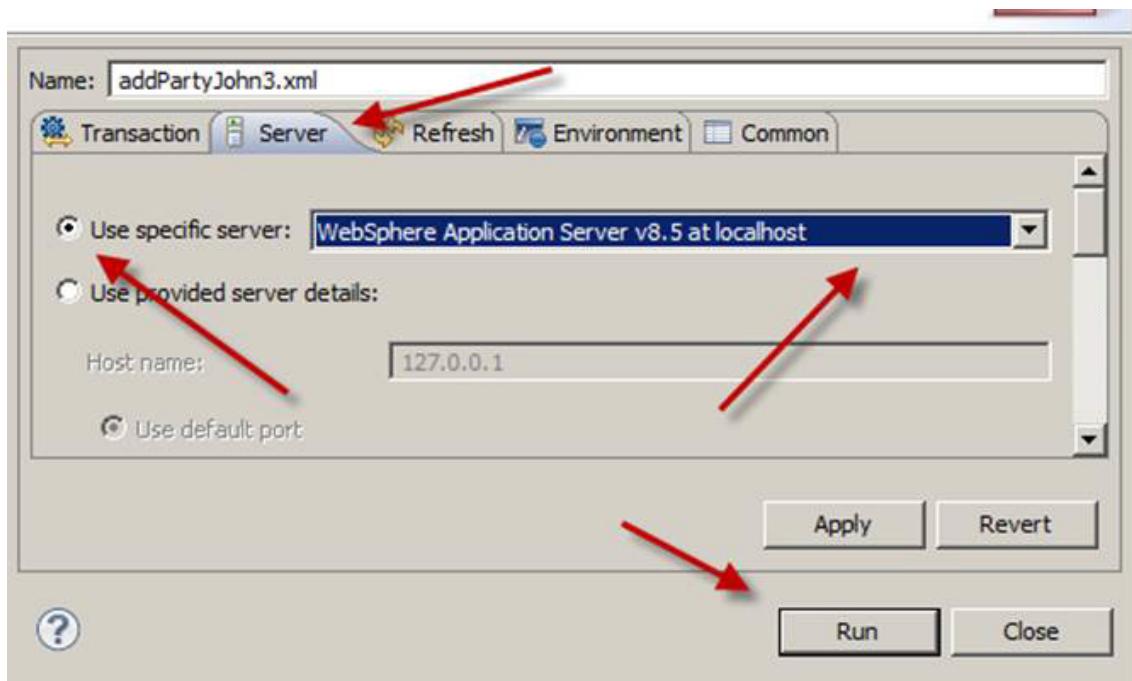


- ___ 3. Look through the XML that we are going to run against the InfoSphere MDM. You'll find the following attributes:
- ___ a. Display Name: **bjones79**
 - ___ b. Gender: **M**
 - ___ c. Birth Date: **1979-03-10**
 - ___ d. Address: **123 First Street, New York, NY, 92342**
 - ___ e. SSN: **436363438**
 - ___ f. Phone Number: **613-274-1245**

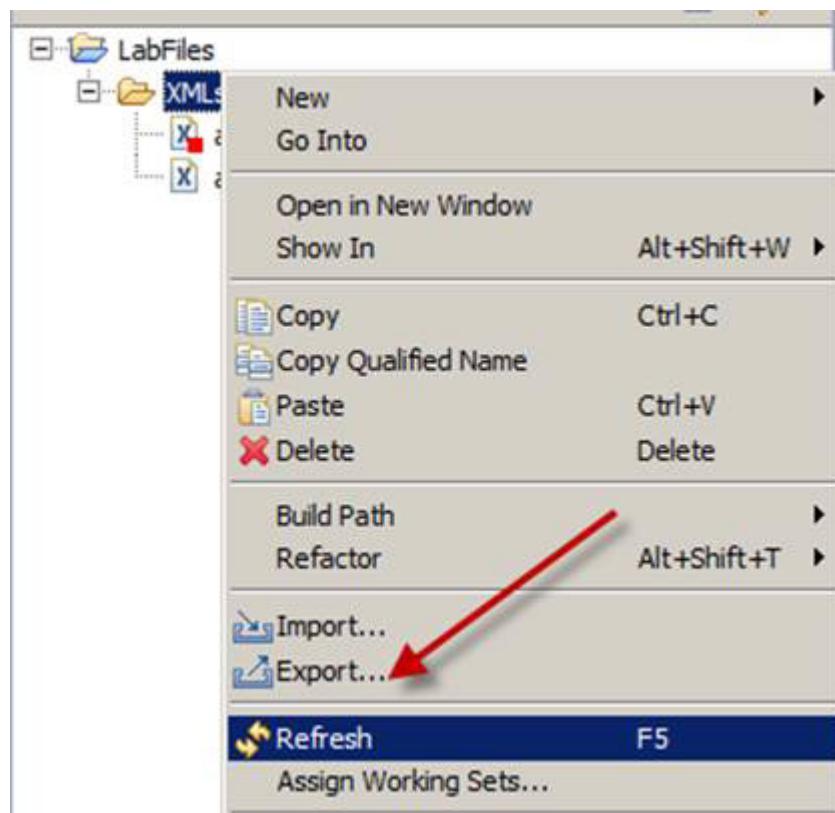
- __ g. Driver's License: **RJ99790310**
 - __ h. Legal Name: **Mr Robert M Jones**
4. To run the XML, right click on the XML and select **Run As > MDM Transactions**



5. Under the Edit Configuration, select the **Server** tab, select the **Use specific server** and select **WebSphere Application Server**. Click the **Run** button.



6. After the service has run against the MDM, right click on the **TestingThePhysicalPME** folder and select **Refresh**.





Troubleshooting

If you found that no response folder is found and there was an error with running the XML, try stopping the server and restarting under the Servers tab.

- 7. You'll find a **response** folder under the **TestingThePhysicalPME** folder. Inside the response folder you'll find the **responseaddPartyBob.xml**. Open the file and look for **SUCCESS** under the **resultCode** tag.

```
<?xml version="1.0" encoding="UTF-8"?>
<TCRMService xmlns="http://www.ibm.com/mdm/schema" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://www.ibm.com/mdm/schema/TCRMService" xmlns:ns="http://www.ibm.com/mdm/schema/TCRMService/ResponseControl">
  <ResponseControl>
    <resultCode>SUCCESS</resultCode>
    <ServiceTime>11172</ServiceTime>
    <DWLControl>
      <requesterName>mdmadmin</requesterName>
      <requesterLanguage>100</requesterLanguage>
      <requesterLocale>en</requesterLocale>
      <userRole>mdm_admin</userRole>
      <requestID>10026142</requestID>
    </DWLControl>
  </ResponseControl>
</TCRMService>
```

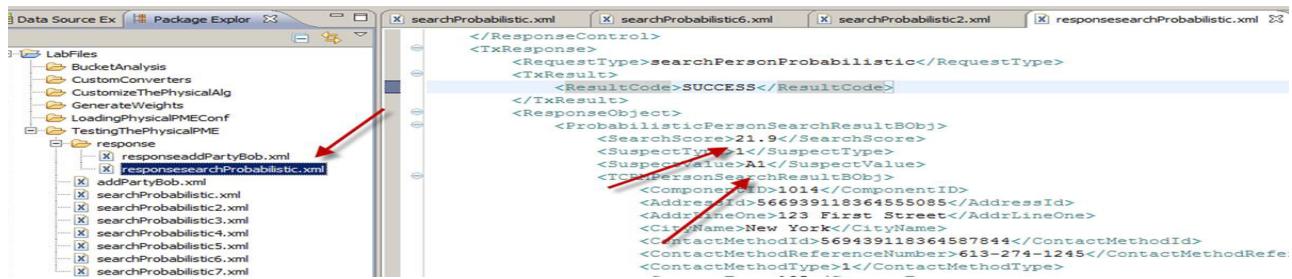
Part 3: Running the Probabilistic Searches

Now that we've been able to add a party to the Physical MDM module, let's run a few search using the PME to see what scores our solutions produce.

- 1. Under the **TestingThePhysicalPME** folder you'll find the **searchProbabilistic.xml**. Open this file in the RAD environment.

```
<TCRMTxType>searchPersonProbabilistic</TCRMTxType>
<TCRMTxObject>ProbabilisticPersonSearchBObj</TCRMTxObject>
<TCRMObject>
  <ProbabilisticPersonSearchBObj>
    <MinScore>0</MinScore>
    <ICRMPersonBObj>
      <DisplayName>bbjones79</DisplayName>
      <BirthDate>1979-03-10</BirthDate>
      <GenderType>M</GenderType>
    </ICRMPersonBObj>
    <ICRMPartyAddressBObj>
      <PartyId>56963911184348556</PartyId>
      <AddressUsageType>1</AddressUsageType>
      <AddressUsageValue>Primary Residence</AddressUs...
    </ICRMPartyAddressBObj>
    <AddressLineOne>123 First Street</AddressLi...
    <City>New York</City>
    <ZipPostalCode>92342</ZipPostalCode>
    <ProvinceStateType>37</ProvinceStateType><!-- USA -->
    <CountryType>185</CountryType>
  </ProbabilisticPersonSearchBObj>
</TCRMObject>
```

- 2. In this file you'll find the probabilistic search using all the attributes that we used in the **addPerson**. Notice that we also specified the **MinScore** to 0 to make sure that the party will be returned regardless of the score. Run the XML against the InfoSphere, (see previous steps on how to run the service).
- 3. Open the response file. In the response you'll find a **SearchScore** of **21.9** which equals an **A1** match (automatic match in Physical MDM).



4. Run the following XML (take a look at what you are searching and the results):

- __ a. searchProbabilistic2.xml
 - Search Criteria = _____
 - Score = _____
- __ b. searchProbabilistic3.xml
 - Search Criteria = _____
 - Score = _____
- __ c. searchProbabilistic4.xml
 - Search Criteria = _____
 - Score = _____
- __ d. searchProbabilistic5.xml
 - Search Criteria = _____
 - Score = _____
- __ e. searchProbabilistic6.xml
 - Search Criteria = _____
 - Score = _____
- __ f. searchProbabilistic7.xml
 - Search Criteria = _____
 - Score = _____
- __ g. searchProbabilistic8.xml
 - Search Criteria = _____
 - Score = _____

Notice that the last 2 searches produced no results. Search 6 did not find any records even though our DOB matched, and search 7 indicated that we did not provide enough search criteria even though our DisplayName and Driver's License matched. In our business scenario, the client would like to be able to search based on the DisplayName and Driver's License.

In the rest of the exercises we will examine the algorithms that produced these results and how we can modify the algorithms to include our DisplayName and Driver's License as search criteria.

End of exercise

Exercise 8a. Loading the PME Algorithm

What this exercise is about

In this exercise we will load the MDM Physical PME configuration and examine the default data model.

What you should be able to do

At the end of this exercise, you should be able to:

- Load the default Physical PME configuration into the RAD environment.
- Understand the default data model that is defined for the Physical PME configuration.
- Add a new Attribute to a Member Type

Introduction

In order to modify the Physical PME configuration (to include our new fields: Driving License and Display Name), we will first have to load the default configuration into our RAD environment.

In this exercise, we will load the default PME configuration into our RAD environment and update the data model for our attributes. The Driving License, although it is not included in the algorithm for matching or searching, is already defined. The DisplayName, which we will call the Username in the PME is not yet defined and will require some customizations.

This exercise will be a step in our customizations, since we will also need to modify the algorithm and weights (in the next units) to fully implement our customizations.

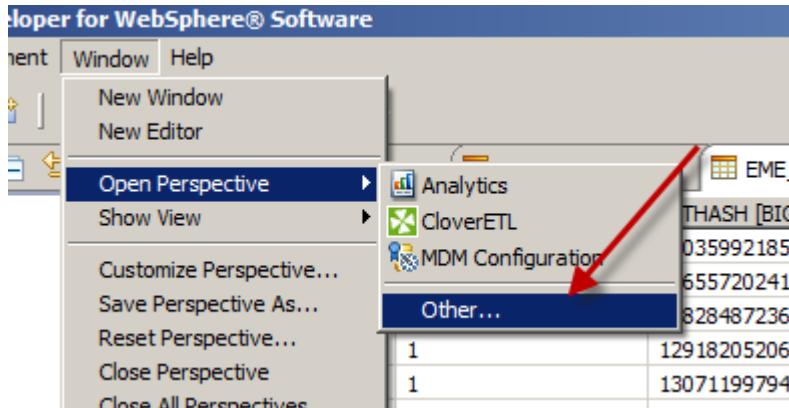
Requirements

- Access to the Internet
- Typical Installation of the InfoSphere MDM Developers workstation

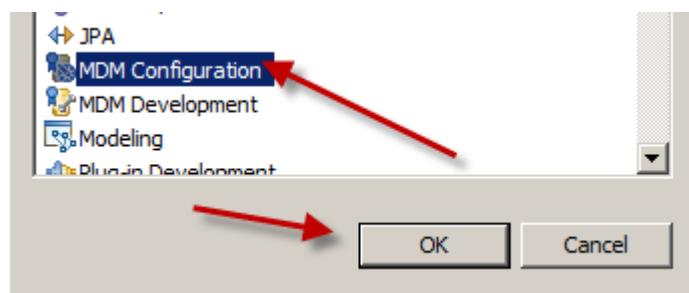
Exercise instructions

Part 1: Changing to the MDM Configuration perspective

- 1. In this exercise we will be working in the MDM Configuration. Inside the RAD environment select **Window > Open Perspective > Other ...**

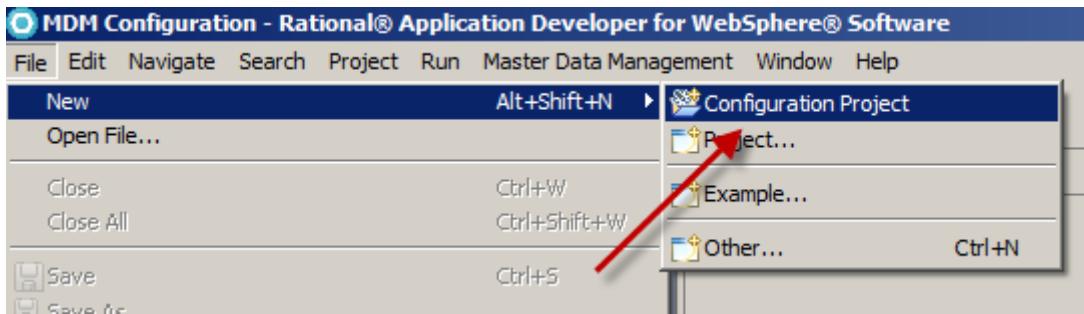


- 2. Select the **MDM Configuration** perspective and click the **OK** button.

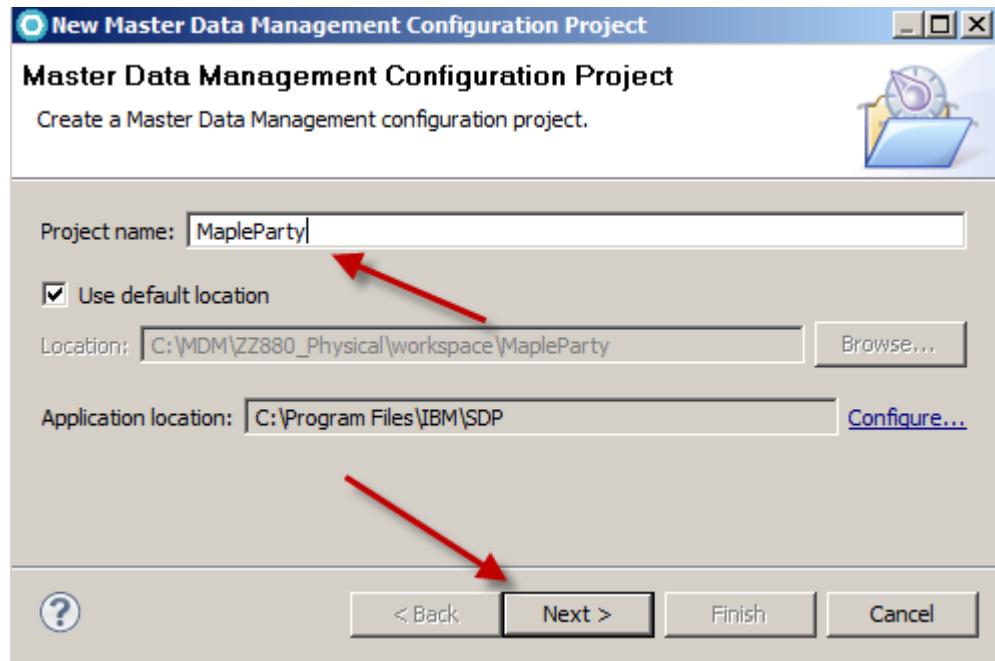


Part 2: Create the PME Configuration project

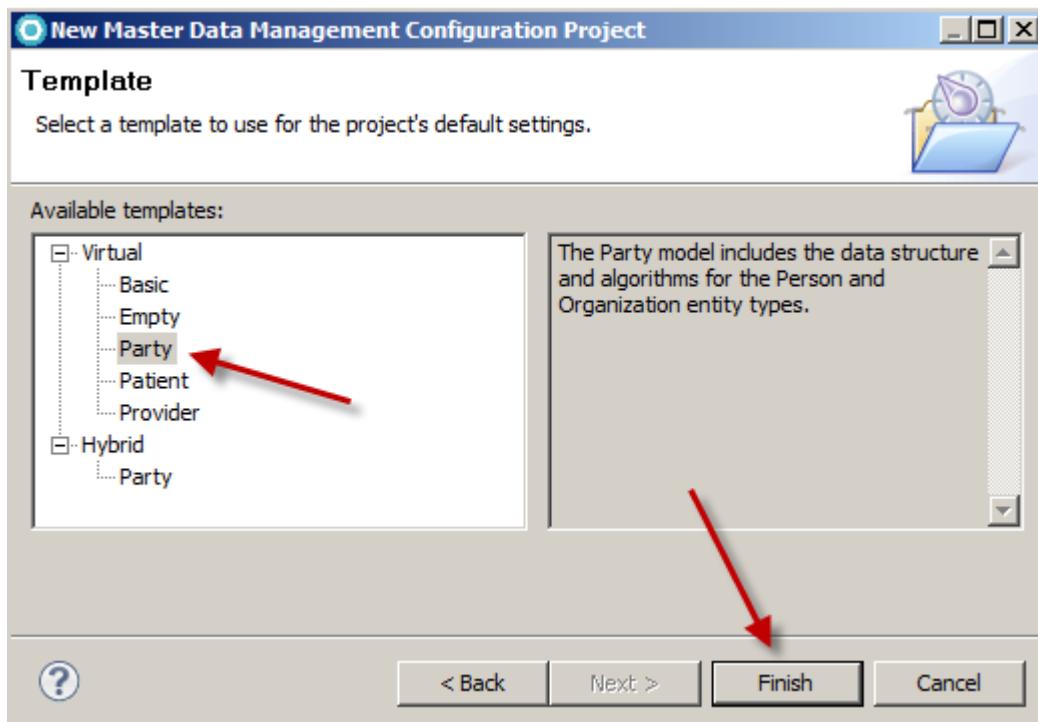
- 1. We will start by creating a new Configuration Project that will store the out of the box MDM Physical module PME configuration. From the RAD file menu, select **File > New > Configuration Project**.



- 2. Enter **MapleParty** for the Project name (this will be the name of the configuration deployed to the Physical MDM) and click the **Next >** button.



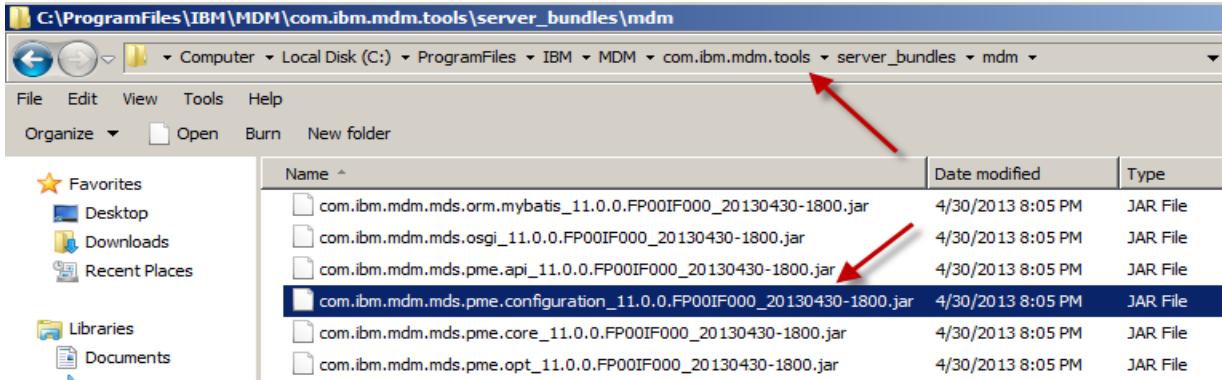
- 3. Select the **Party** template and click the **Finish** button. NOTE: the Party template will create an Organization and Person algorithms. The default Physical PME algorithms contain an organization and person algorithm.



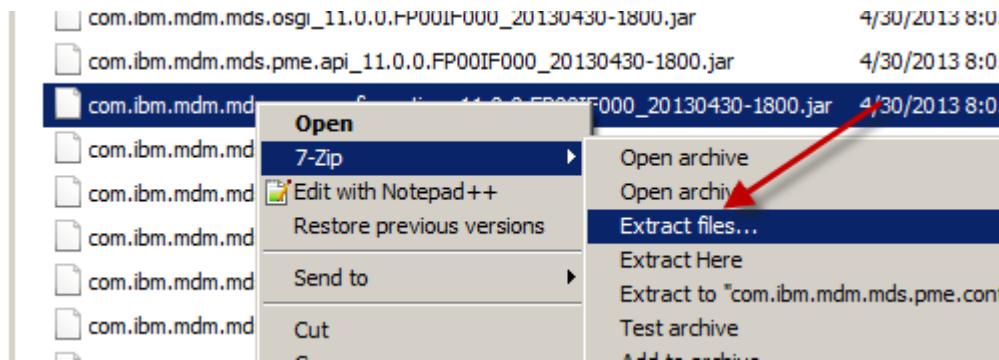
Part 3: Import the Default PME Configuration

In order to see the default Physical PME configuration, we will unzip the jar file containing the configuration and place the content inside our configuration project. The path and jar files that we will use come from the default installation of an MDM Development environment.

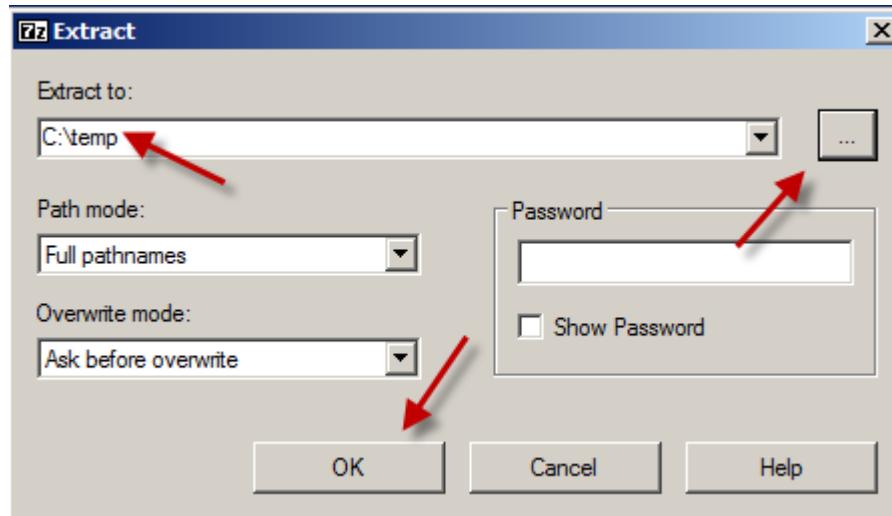
- 1. Inside the Windows environment, open a windows explorer and navigate to
C:\ProgramFiles\IBM\MDM\com.ibm.mdm.tools\server_bundles\mdm



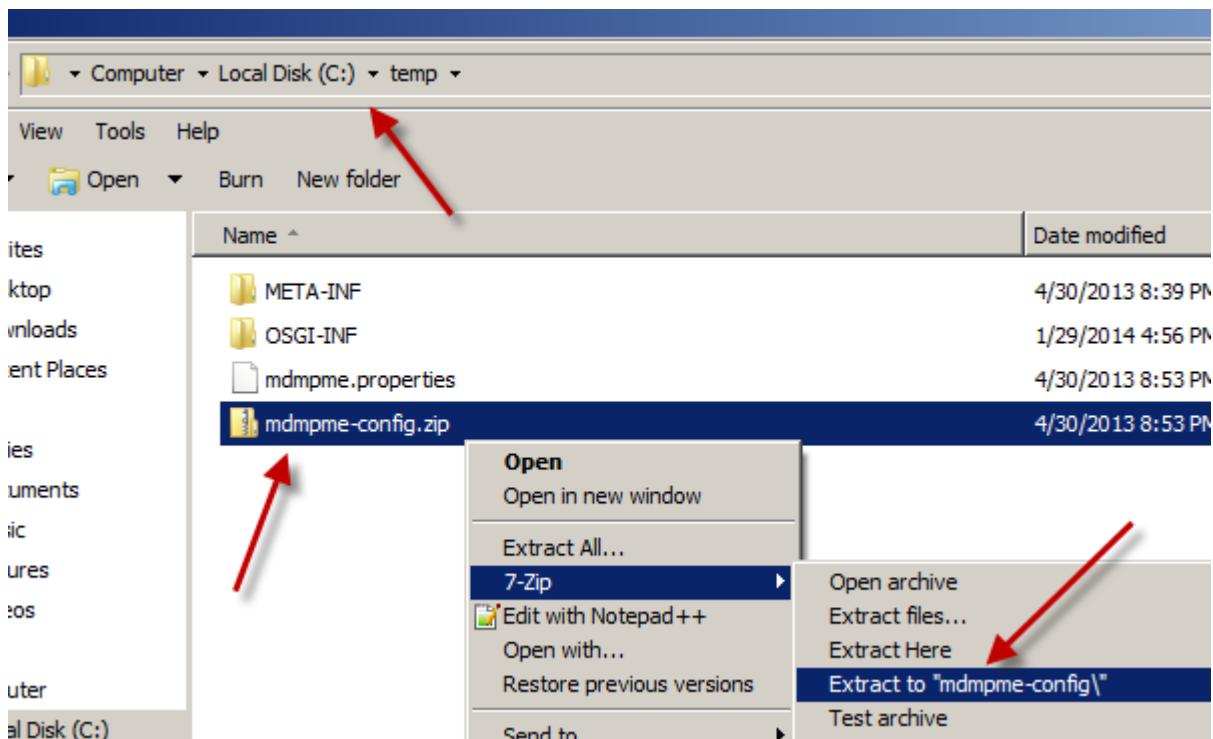
- 2. Right click on the
com.ibm.mdm.mds.pme.configuration_11.0.0.FP00IF000_20130430.jar file and select
7-Zip > Extract files...



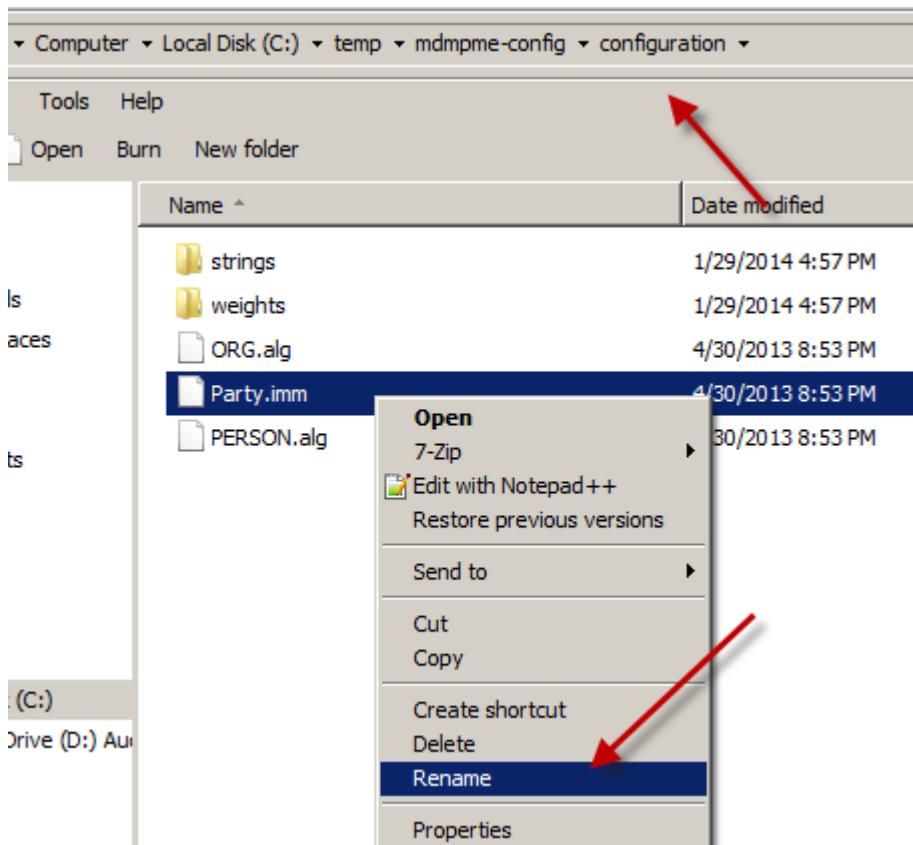
- 3. Enter **C:\temp** as the folder to Extract to and click the **OK** button.



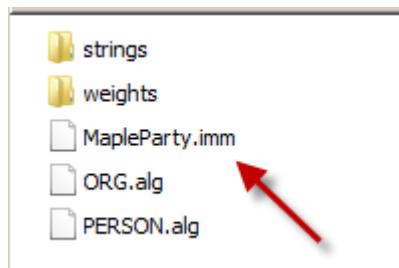
- 4. Navigate to the extracted files (under c:\temp), right click on the **mdmpme-config.zip** file and select **7-Zip > Extract to “mdmpme-config\”**



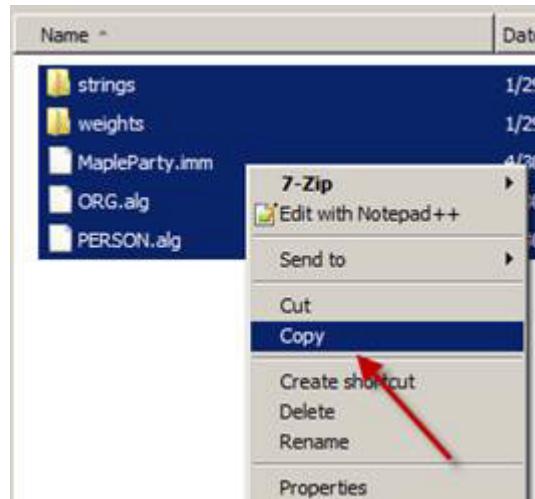
- 5. In this step we will change the name of the configuration file to match our project configuration name. Navigate to **C:\temp\mdmpme-config\configuration**. Right click on the **Party.imm** and select **Rename**.



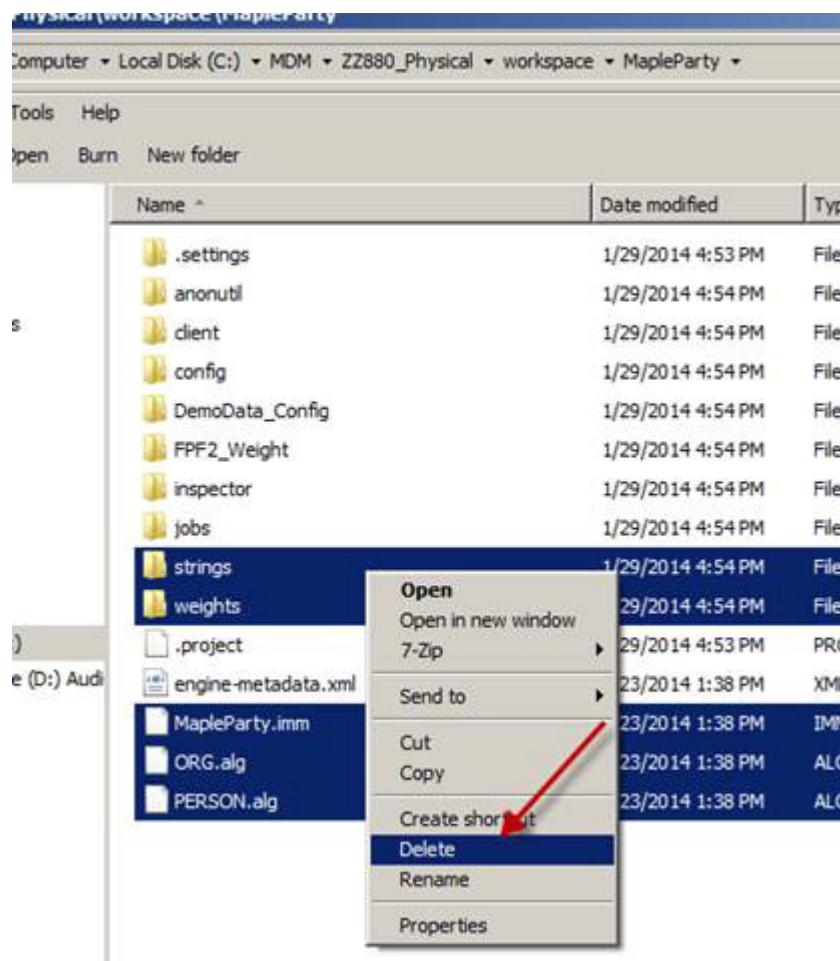
- ___ 6. Change the name of the file to **MapleParty.imm**. This should match the name of the Configuration Project that you defined in the RAD environment.



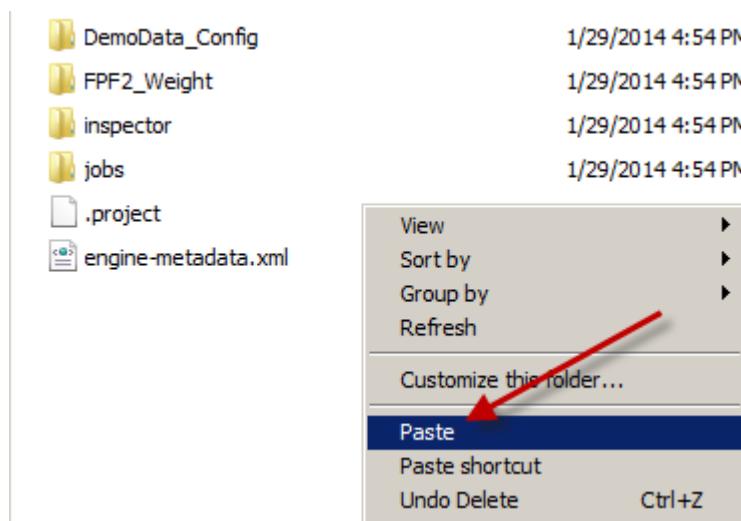
- ___ 7. Next, we will copy and replace the project that we created in the RAD environment with the default configuration. Copy all the content found under **C:\temp\mdmpme-config\configuration** by selecting all the content, right clicking and selecting **Copy**.



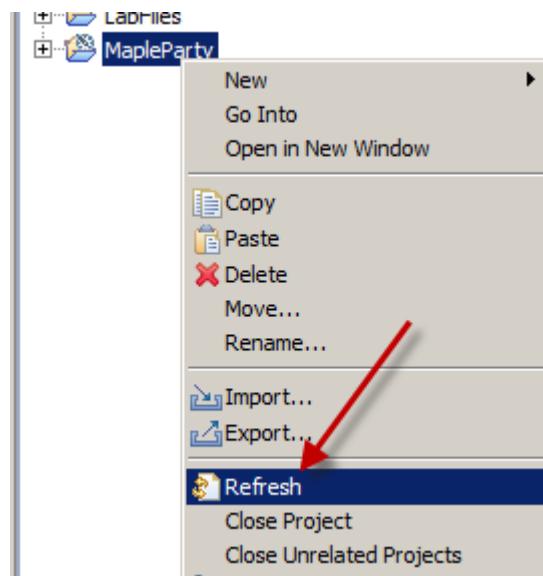
- 8. To replace the RAD project content, we will first delete the configuration created by RAD and paste our default configuration. Using the windows explorer, navigate to the RAD project that you created (**C:\MDM\ZZ880_Physical\workspace\MapleParty**). Select the **strings** folder, **weights** folder, **MapleParty.imm** file, **ORG.alg** file and **PERSON.alg** file. (NOTE: you can use the CRTL key to select multiple files). Right click on the selected files (and folders) and select **Delete**.



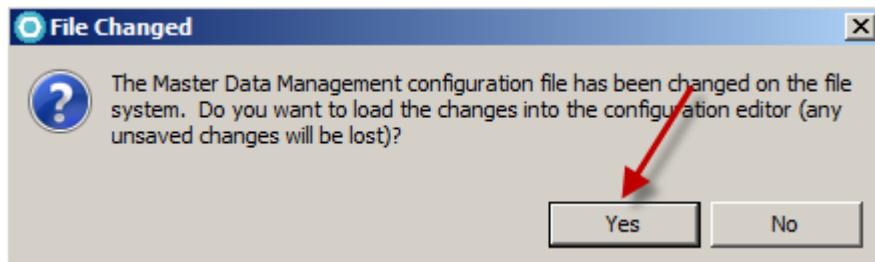
- 9. Once the files are deleted we can paste the files that we copied from the default configuration. Right click in the **MapleParty** project folder and select **Paste**.



- 10. Back in the RAD environment we can refresh our changes and see the new configuration. In the RAD environment, under the Project Explorer, right click on the **MapleParty** project and select **Refresh**.



- 11. Since the configuration files will be open in your RAD environment already, if RAD asks if you want to load the changes, click the **Yes** button.



- 12. The RAD environment will not refresh the open file properly as it was changed outside the environment, close the MapleParty window that is opened, when we open it in the next steps it will be refreshed properly.



Part 4: Examining the OOTB data model

In this part of the exercise we will examine the default data model defined for the Physical PME algorithm.

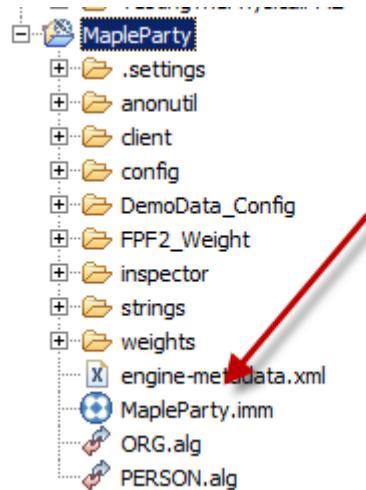
Not all attributes for a PERSON or ORGANIZATION in our configuration will be used in our algorithms, so the first thing we can do is look at the product documentation to see what is included.

- 1. Open an Internet Browser and navigate to

http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.0.0/com.ibm.mdmhs.dev.party.doc/concepts/c_understandingattributetypes.html. NOTE: you'll find a link on the desktop to the **Physical PME Attribute Types**. In the green boxes you'll find the attribute types and the yellow boxes you'll see the attribute fields that belong to these attribute types.

Address attribute	Person Name attribute	Birth Date attribute	Social Security Number attribute
addrline1	givenname1	val	idnum
addrline2	givenname2	Birth Date attribute type (PERBIRTHDATE)	Social Security Number attribute type (PERSSN)
city	lastname		
residencenum	Legal Name attribute type (PERLEGALNAME)		
provstate	Business Name attribute type (PERBUSNAME)		
postalcode	Nickname attribute type (PERNICKNAME)		
Primary Residence attribute type (PERPRIMERRES)	AKA Name attribute type (PERAKANAME)		
Other Residence attribute type (PEROTHERRES)	Maiden Name attribute type (PERMAIDENNAME)		
Business Address attribute type (PERBUSADDR)	Alias Name attribute type (PERALIASNAME)		
Mailing Address attribute type (PERMAILADDR)	Preferred Name attribute type (PERPREFNAME)		
Summer Residence attribute type (PERSUMMERRRES)	Previous Name attribute type (PERPREVNAME)		
Temporary Residence attribute type (PERTEMPADDR)			
Secondary Residence attribute type (PERSECONDRES)			

2. To see these attributes types, we will take a look at the configuration file. Inside the RAD environment, under the Project Explore, navigate under the MapleParty project and open the **MapleParty.imm** file.



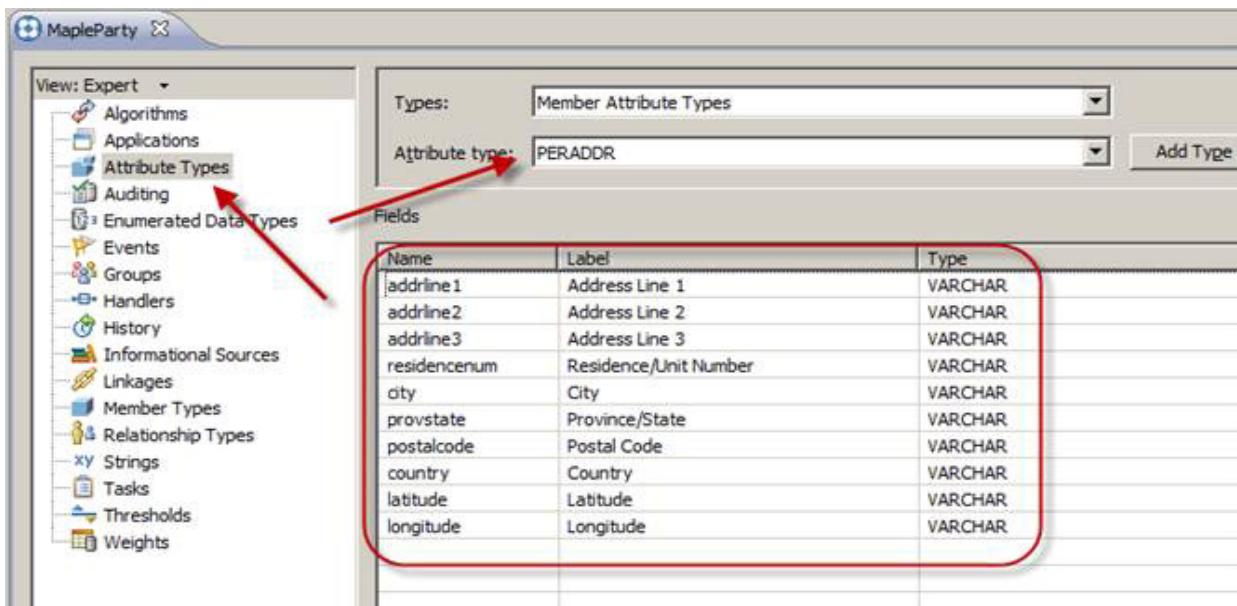
3. Select the **Member Types** table and change the Member type to **PERSON**.

Attribute	Code
Per AKA Name	PERAKANAME
Per Abilitec	PERABILITEC
Per Alias Name	PERALIASNAME
Per American Express	PERAMEX
Per Birth Certificate	PERBIRTHCERT
Per Business Address	PERBUSADDR
Per Business Email	PERBUSEMAIL
Per Business Name	PERBUSNAME
Per Business Phone	PERBUSPHONE
Per Cell Phone	PERCELLULAR
Per Cert Deposit	PERCERTDEPST

4. Click the **Type** heading to sort the Attribute Types. At the top, you'll find the **PERADDR** types, (These attributes will be mapped to the PersonAddress on the Physical module). This includes the following Attributes:
- Person Primary Residence
 - Person Other Residence
 - Person Business Address
 - Person Mailing Address
 - Person Summer Residence
 - Person Temporary Address
 - Person Second Residence

Attribute	Code	Type
Per Primary Residence	PERPRIMERRES	PERADDR
Per Other Residence	PEROTHERRES	PERADDR
Per Business Address	PERBUSADDR	PERADDR
Per Mailing Address	PERMAILADDR	PERADDR
Per Summer Residence	PERSUMMERRES	PERADDR
Per Temporary Address	PERTEMPADDR	PERADDR
Per Second Residence	PERSECONDRS	PERADDR
Per Checking Account	PERCHECKACCT	PERBANK
Per Savings Account	PERSAVINGACCT	PERBANK

5. To see the attribute fields of PERADDR type, select the **Attribute Types** tab and select the Attribute Type **PERADDR**. Under the fields you'll find addrline1, addrline2, city, etc.



6. Back under the **Member Types** tab, for the Person Member Type, you'll find the **Per DOB**. The **PERDATE** type includes a field called **val**.

Per Diners Club Card	PERDINERCLUB	PERBANKCARD
Per DOB	PERBIRTHDATE	PERDATE
Per Deceased Date	PERDECEASDTE	PERDATE

7. After the Per DOB, you'll find the **Per Gender**, this is of type **PERGENDER**. The **PERGENDER** type includes a field called **gender**.

Per Personal Email	PERPERSEMAIL	PEREMAIL
Per Gender	PERGENDER	PERGENDER
Per SSN	PERSSN	PERIDENT

8. After the Per Gender, you'll find the **PERIDENT** type. This includes the Person SSN, Person SIN and Person Driving License. Notice that the Person Driving License does not include a small icon to the left that the other attributes that we've mentioned includes. This is because it is not included in the Matching or Searching algorithms.

	PERGENDER	PERGENDER
Per Gender	PERSNN	PERIDENT
Per SSN	PERDRVRLICEN	PERIDENT
Per Driving License	PERBIRTHCERT	PERIDENT
Per Birth Certificate	PERTAXIDNUM	PERIDENT
Per Tax Identifier	PERTAXREGNUM	PERIDENT
Per Tax Reg Number	PERPASSPORT	PERIDENT
Per Passport	PERHEALTHCRD	PERIDENT
Per Health Card	PERSIN	PERIDENT
Per SIN	PERABILITEC	PERIDENT
Per Abilitec		

9. You'll also find the **PERNAME** type that includes the following Attributes:

- a. Person Legal Name
- b. Person Business Name
- c. Person Nickname

- d. Person AKA Name
- e. Person Alias Name
- f. Person Preferred Name
- g. Person Previous Name.

Per Marital Status	PERMARITAL	PERMARITAL
Per Legal Name	PERLEGALNAME	PERNAME
Per Business Name	PERBUSNAME	PERNAME
Per Nickname	PERNICKNAME	PERNAME
Per AKA Name	PERAKANAME	PERNAME
Per Maiden Name	PERMAIDENNAM	PERNAME
Per Alias Name	PERALIASNAME	PERNAME
Per Preferred Name	PERPREFNAME	PERNAME
Per Previous Name	PERPREVNAME	PERNAME

10. Next, the **PERPHONE** type that includes the following attributes:

- a. Person Home Phone
- b. Person Cell Phone
- c. Person Mobile Phone
- d. Person Business Phone

Per Previous Name	PERPREVNAME	PERNAME
Per Home Phone	PERHOMEPHONE	PERPHONE
Per Cell Phone	PERCELLULAR	PERPHONE
Per Mobile Phone	PERMOBILE	PERPHONE
Per Business Phone	PERBUSPHONE	PERPHONE
Per Fax	PERFAX	PERPHONE
Per Pager	PERPAGER	PERPHONE

What is missing from these attributes is the username. This is the field that we would like to search as the display name in the physical module.

Part 5: Adding a new Attribute

1. To add a new Attribute (username), click the **Add** button under the Member Types tab.

Attribute	Code	Type	
Per Driving License	PERDRVRLICEN	PERIDENT	<input type="button" value="Add"/>
Per Birth Certificate	PERBIRTHCERT	PERIDENT	<input type="button" value="Remove"/>
Per Tax Identifier	PERTAXIDNUM	PERIDENT	
Per Tax Reg Number	PERTAXREGNUM	PERIDENT	
Per Passport	PERPASSPORT	PERIDENT	
Per Health Card	PERHEALTHCRD	PERIDENT	
Per SIN	PERSIN	PERIDENT	
Per Abilitec	PERABILITEC	PERIDENT	
Per Marital Status	PERMARITAL	PERMARITAL	
Per Legal Name	PERLEGALNAME	PERNAME	



Troubleshooting

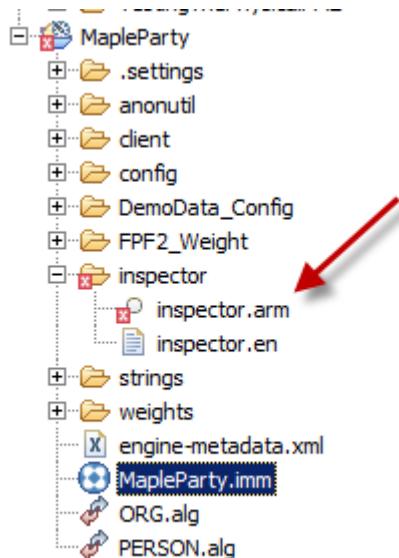
If a new row does not open, try closing the MapleParty.imm file and reopening the file. Because we copied this file outside of RAD, it may still need to be refreshed.

2. Enter the following information for our new Attribute:

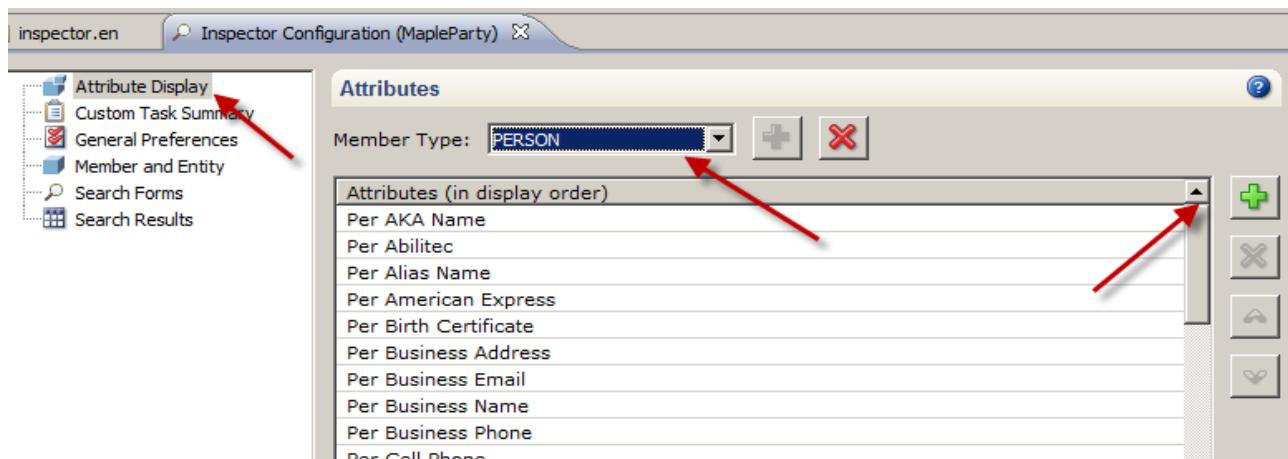
- a. Attribute: **Per Username**
- b. Code: **PERUSERNAME**
- c. Type: **PERIDENT** (this will contain a idnum field)

Attribute	Code	Type
Per Username	PERUSERNAME	PERIDENT
Per AKA Name	PERAKANAME	PERNAME
Per Abilitec	PERABILITEC	PERIDENT
Per Alias Name	PERALIASNAME	PERNAME
Per American Express	PFRAMFX	PFRANKCARD

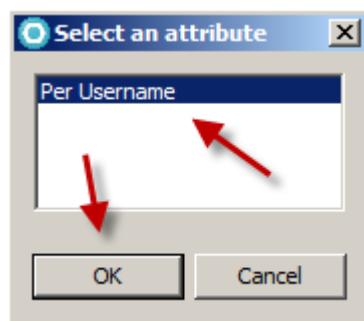
3. Save the Configuration file (you may need to click away from the Per Username row).
 4. Once the file is saved, you'll notice an error in your workspace. Under the Package Explorer, navigate to **MapleParty > inspector** and open the **inspector.arm** file.



5. Under the **Attribute Display** tab, select the **PERSON** Member Type and click the green plus button beside the Attributes.



- ___ 6. Select our new **Per Username** attribute and click the **OK** button



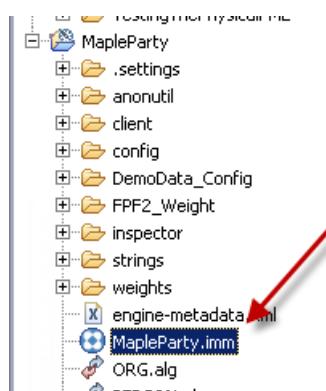
- ___ 7. Save the file (CTRL+S)

In the next exercises we will add the Per Username and Per Driving License to the Algorithms.

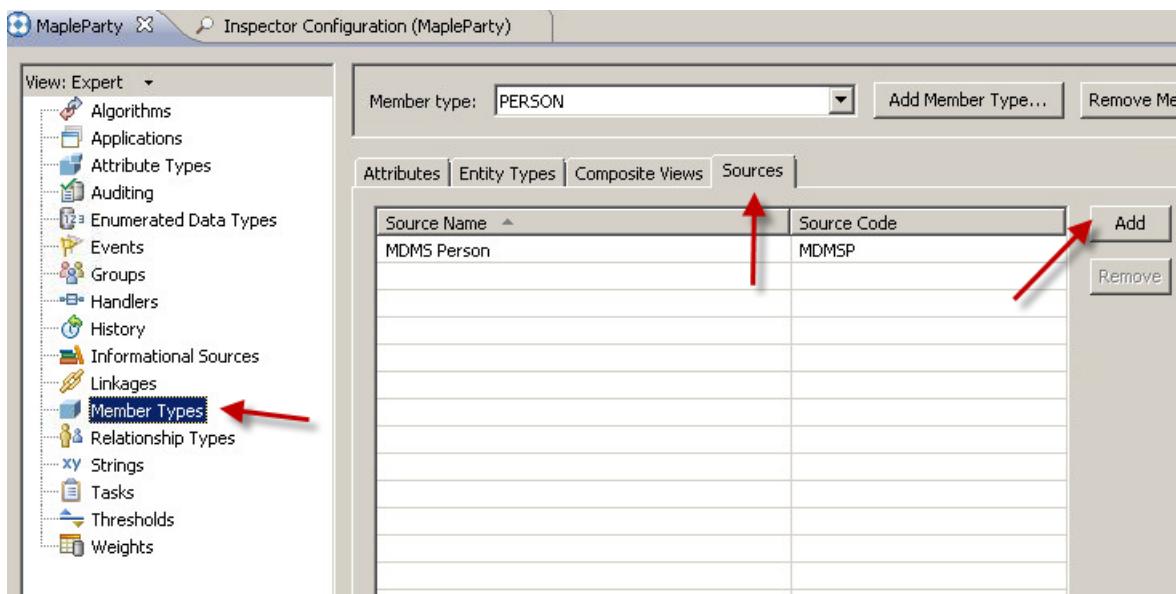
Part 6: Adding a new Source

In our exercise we will be loading a sample data set that comes from 2 sources (MDMSP and MDMS2). This is used for bucket analysis, weight generation, analyzing our algorithm and setting thresholds. The default PME algorithm only defines one source for the Person entity (MDMSP), we will define the other.

- ___ 1. Open the PartyConfiguration.imm file found under the MapleParty project.



- __ 2. Under the **Member Types** link, select the **Sources** tab and click the **Add** button.



- __ 3. Under the RAD Properties window, change the following values:

- __ a. Physical code: **MDMS2**
- __ b. Source code: **MDMS2**
- __ c. Source name: **MDMS Person 2**

Property	Value
Default Entity Priority	100
Enable Review Identifier Task	No
Is virtual	No
Physical code	MDMS2
Potential overlay comparison code	
Potential overlay score	0
Record status	Active
Source code	MDMS2
Source name	MDMS Person 2
Srcrecno	290

- __ 4. Save the changes (CTRL+S)

End of exercise

Exercise 8b. Viewing the PME Derived Data

What this exercise is about

This exercise covers the Physical MDM tables that are used to store a party and the Physical PME tables that are used to store the derived data used for matching and searching. This exercise also examines the critical fields tables and how to customize the tables to synchronize additional fields with the PME.

What you should be able to do

At the end of this exercise, you should be able to:

- Understand how some of the Physical Tables are used to store a party
- Understand the PME tables and what is stored in each
- Understand how to configure the Critical Data to include new fields (to be passed to the PME)

Introduction

In this exercise we will view how our Party (that we added in a previous exercise) is stored in the InfoSphere MDM Physical tables and the PME tables used for matching and searching.

We will then see how the configuration decides which attribute from the Physical module is synchronize with the PME tables.

We will modify the physical configuration to include our 2 fields Driving License and Username (aka Display Name) but will not see the results until we modify the algorithm in the next unit.

Requirements

- The following Party should be loaded into the MDM Physical module:
 - Display Name: **bbjones79**
 - Gender: **M**
 - Birth Date: **1979-03-10**
 - Address: 123 First Street, New York, NY, 92342
 - SSN: 436363438
 - Phone Number: **613-274-1245**
 - Driver's License: **RJ99790310**

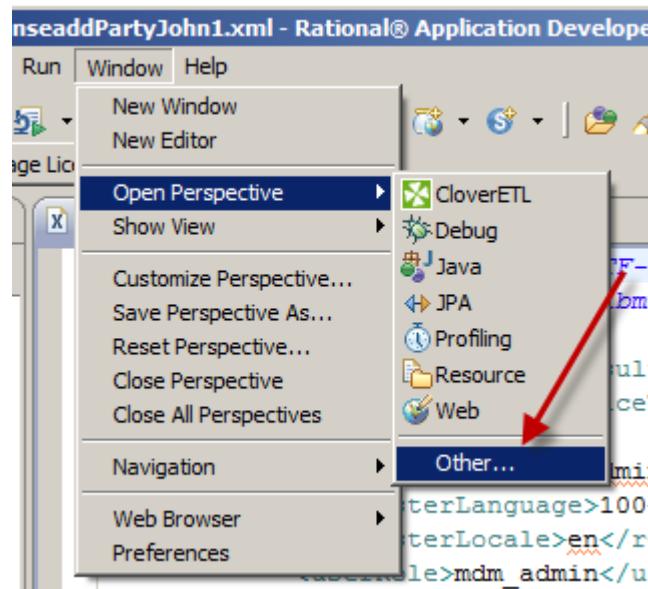
- Legal Name: Mr Robert M Jones
- Typical Installation of the InfoSphere MDM Developers workstation

Exercise instructions

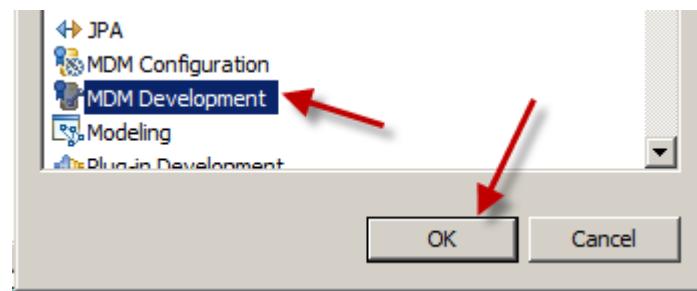
Part 1: Switch to the MDM Developer perspective

To start off, we will first switch to the MDM Development perspective, which contains the database explorer, in the RAD environment.

- 1. In the RAD File menu, select **Window > Open Perspective > Other ...**

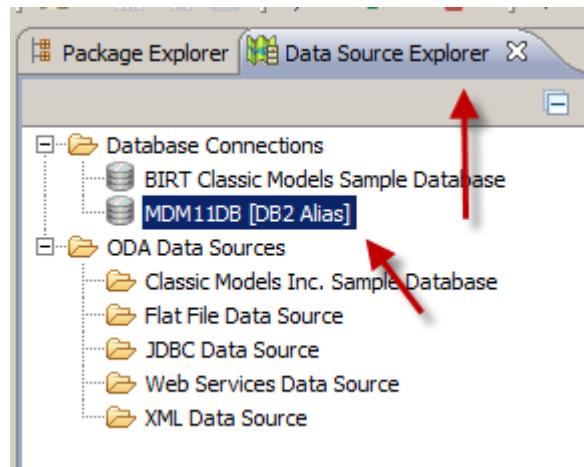


- 2. Select **MDM Development** and click the **OK** button.



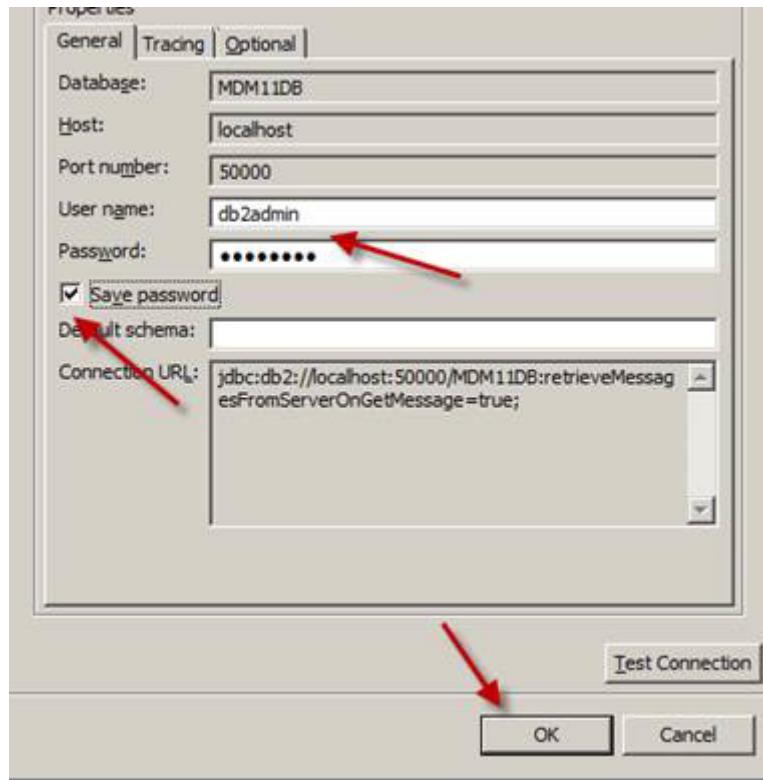
Part 2: Accessing the MDM 11 tables

- 1. Select the **Data Source Explorer** tab (top left hand corner) and double click the **MDM11DB (DB2 Alias)**

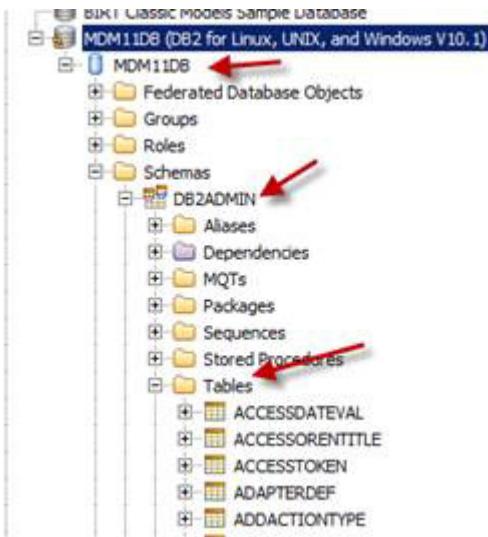


__ 2. Enter the following credentials and click the **OK** button:

- __ a. User name: **db2admin**
- __ b. Password: **db3Admin**



__ 3. Navigate to the MDM Tables (Found under **MDM11DB > Schemas > DB2ADMIN > Tables.**)



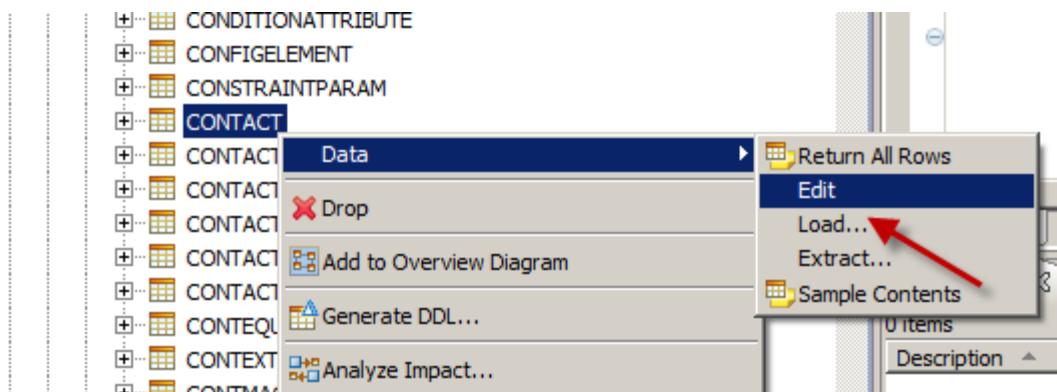
Part 3: View the derived data

In a previous exercise we added a Person to the Physical MDM. In this part of the exercise, we will see how this record is represented in the Physical tables and Physical PME tables.

The Person that we added had the following attributes:

- Display Name: **bjjones79**
- Gender: **M**
- Birth Date: **1979-03-10**
- Address: **123 First Street, New York, NY, 92342**
- SSN: **436363438**
- Phone Number: **613-274-1245**
- Driver's License: **RJ99790310**
- Legal Name: **Mr Robert M Jones**

— 1. Locate the **CONTACT** table, right click and select **Data > Edit**.



- 2. The last record is the Party we added in the previous exercise, take note of the PartyId as we will see it in other tables. Also notice the CONTACT_NAME, this is the displayname that we want to use for search purposes

CONT_ID [BIGINT]	ACCE...	PRE...	CRE...	I...	CONTACT_NAME [...]	P...
111111111	1	100	201...		Paint	P
311111111	1	100	201...		Active	O
911111111	1	100	201...		Vanessa Dana	P
840001	1	100	201...		Jim Story	P
568639118364403455	6	100	201...		bbjones79	P
<new row>						

- 3. Next, open the **IDENTIFIER** table. Here we will find 2 rows that are attached to the Party, the SSN (ID_TP_CD = 1) and Driving License (ID_TP_CD = 3).

IDENTIFIER_ID [BIGINT]	ID_S...	CONT_ID [BIGINT]	ID_TP_CD [BIGINT]	REF_NUM [VARCHAR(50)]
111111111	2	111111111	1	291292293
311111111	2	311111111	2	919293
567239118364530011		5686391183644...	1	436363438
565439118364531518		5686391183644...	3	RJ99790310
<new row>				

- 4. To see the rest of the record, open the **PERSON**, **ADDRESS**, and **CONTACT** tables.

CONTACT	IDENTIFIER	PERSON	ADDRESS	CONTACTMETHOD
1	613-274-1245		2014-01-31 10:54:05.877	mdmadmin

- 5. Now let's take a look at the Physical PME tables. When we added the record to the physical module, the PME would have been synchronized with the new record. Open the **EME_RECHEAD** table. Here you'll find our Party that was synchronized with the PME. Notice the RECNO column, we will see this value in the next tables.

RECNO [BIGINT]	SRCRECNO [SMALLINT]	SRCID [VARCHAR(240)]	TXNID [BIGINT]
1	1	111111111	0
2	289	311111111	0
3	1	911111111	0
4	1	840001	0
21	1	568639118364403455	0
<new row>			

- 6. To see the derived data that is used in Bucketing and Comparing the record (the data after it's been standardized), open the **EME_RECMPD** table. In this table, under the CMPVAL, you'll find the following fields separated by the “^” character. NOTE: this is similar to the **MPI_MEMCMPD** table on the Virtual module)
- a. Legal Name: **JONES:ROBERT:M**
 - b. Postal Code: **92342**
 - c. Address: **N-123:S-1ST:S-ST:.S-New:S-YORK:.N-92342**
 - d. SSN: **436363438**
 - e. DOB:**19790310**
 - f. Phone Number: **2741245**
 - g. Gender:**M**

TACT	IDENTIFIER	PERSON	ADDRESS	CONTACTMETHOD	EME_RECHEAD	EME_RECMPD
...	SRC...	C...	CMPVAL [VARCHAR(1020)]			
1	1	PAINT::: ^^N-211:N-12:S-PINTO:S-RD:.S-TORONTO:..^291292293^^19730901^F				
289	1	ACTIVE^^^^^919293				
1	1	DANA:VANESSA::: ^^^^^19420701^F				
1	1	STORY:JIM::: ^^^^^19340101^F				
1	1	JONES:ROBERT:M:: ^92342^N-123:S-1ST:S-ST:.S-New:S-YORK:.N-92342: ^436363438^2741245^19790310^M				
...						

Legal Name ZIP Address SSN Phone Number DOB Gender

Notice that the format of the data has been changed, we will see how and why the data has changed when we examine the algorithms in the next unit.

- 7. Next, open the **EME_RECBKTD** table. Here you'll find the various buckets that the record now belongs to. Notice the bucket name is a Hash representation so we don't necessarily know what each represent. But we will see the algorithms in another exercise that will give us some indications. Any record that shares the same buckethash will be compared to each other in the matching. NOTE: this is similar to the **MPI_MDMBKTD** table on the Virtual module.

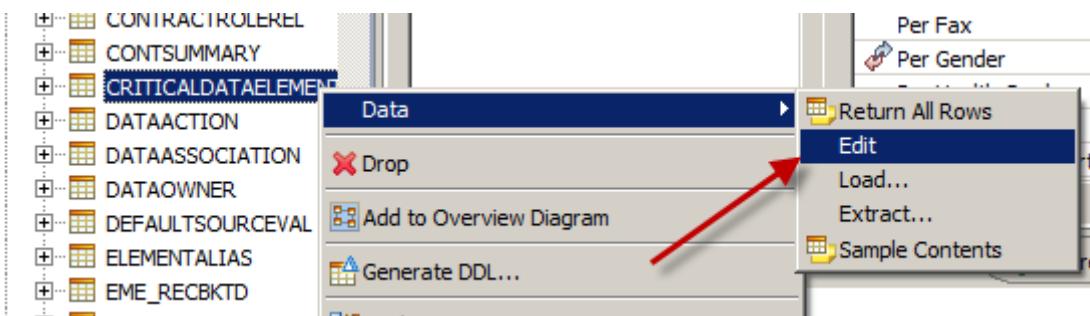
RECNO [BIGINT]	SRCRECNO [SMALLINT]	BKTHASH [BIGINT]
2	289	1203599218575...
4	1	1265572024182...
4	1	1291820520609...
3	1	1307119979499...
21	1	1309329165770...
21	1	1312050563622...
21	1	1348040768223...
21	1	1591859756411...
4	1	1814898121187...
21	1	2005639828896...
21	1	2095147090584...
21	1	2321562001288...
4	1	2659036718275...
21	1	3019938037141...
4	1	3082403975730...
21	1	3112776779420

Looking at the Derived data, you'll notice that the Driving License and Username are not included in the PME. We will change this in the next part of the exercise.

Part 4: Viewing the Critical Data tables

The critical data table indicates which attributes from the Physical Module is synchronize with the PME tables. In this part of the exercise we will update the tables to include our 2 attributes (Driving License and username)

- 1. Locate the **CRITICALDATAELEMENT** table inside the MDM tables, right click on the table and select **Data > Edit**.



- 2. Inside the CRITICALDATAELEMENT table you'll find the attributes that are being synchronize (**SYNCPURPOSE_TP_CD = 1**) with the PME tables and the attributes that trigger a Suspect Duplicate Processing (**SYNCPURPOSE_TP_CD = 2**)

C...	APPLICATION_ID	GROUP_NAME [V...]	ELEMENT_NAME [V...]	SYNPURPOSE_TP_CD...	ENTITY_TYPE [V...]	INSTANCE_PK [V...]
1	TCRM	Person	GenderType	1		
2	TCRM	Person	BirthDate	1		
3	TCRM	PersonName	GivenNameOne	1	NameUsageType	1
4	TCRM	PersonName	GivenNameTwo	1	NameUsageType	1
5	TCRM	PersonName	LastName	1	NameUsageType	1
6	TCRM	PersonName	GivenNameOne	1	NameUsageType	2
7	TCRM	PersonName	GivenNameTwo	1	NameUsageType	2
8	TCRM	PersonName	LastName	1	NameUsageType	2
9	TCRM	PersonName	GivenNameOne	1	NameUsageType	3

- 3. Find the row for the **Driving License** (GROUP_NAME = Party Identification, ELEMENT_TYPE = IdentificationType, INSTANCE_PK = 3). Change the **ACTIVE_IND** to Y and save the changes.

63	TCRM	PartyContactMe...	ContactMethodUsa...	1	ContactMethodU...	2	
64	TCRM	ContactMethod	ReferenceNumber	1			Y
69	TCRM	Person	DeceasedDate	1			N
70	TCRM	PartyIdentification	IdentificationNumber	1	IdentificationType	3	Y
71	TCRM	PartyIdentification	IdentificationNumber	1	IdentificationType	4	N
72	TCRM	PartyIdentification	IdentificationNumber	1	IdentificationType	6	N
73	TCRM	PartyIdentification	IdentificationNumber	1	IdentificationType	7	N
74	TCRM	PartyIdentification	IdentificationNumber	1	IdentificationType	8	N
75	TCRM	PartyIdentification	IdentificationNumber	1	IdentificationType	9	N

- 4. We've so far only changed the row that synchronizes the Driving License with the PME, we also want a change to the Driving License to trigger Suspect Duplicate Processing. Find the other row for the Driving License where the SYNPURPOSE_TP_CD = 2 and change the **ACTIVE_IND** to Y. Save the changes once complete.

C...	APPLICATION_ID	GROUP_NAME [V...]	ELEMENT_NAME [V...]	SYNPURPOSE_TP_CD...	ENTITY_TYPE [V...]	INSTANCE_PK [V...]	ACTIVE_IND [CHAR(1)]	ULTIM...
204	TCRM	PersonName	GenerationType	2			N	Perso...
205	TCRM	PersonName	PrefixType	2			N	Perso...
206	TCRM	Address	AddressLineThree	2			N	Perso...
207	TCRM	PartyIdentification	IdentificationNumber	2	IdentificationType	3	Y	Perso...
208	TCRM	PartyIdentification	IdentificationNumber	2	IdentificationType	4	N	Perso...

- 5. We also want to add a row for our new attribute "Per username". In the Physical MDM, this attribute has the Metadata name of Person.DisplayName. Right click in the **CRITICALDATAELEMENT** table and select **Insert Row**.

GROUP_NAME [...]	ELEMENT_NAME [V...]	SYNPURPOSE_TP_CD...	ENTITY_TYPE [V...]
Person	GenderType	1	
Person	BirthDate	1	
PersonName	GivenNameOne	1	
PersonName	GivenNameTwo	1	
PersonName	LastName	1	
PersonName	GivenNameOne	1	
PersonName	GivenNameTwo	1	
PersonName	LastName	1	
PersonName	GivenNameOne	1	



6. Enter the following values and save the table:
- __ a. CRITICAL_ELEMENT_ID = **880001**
 - __ b. APPLICATION = **TCRM**
 - __ c. GROUP_NAME = **Person**
 - __ d. ELEMENT_NAME = **DisplayName**
 - __ e. SYNCPURPOSE_TP_CD = **1**
 - __ f. ACTIVE_IND = **Y**
 - __ g. ULTIMATE_PARENT_GROUP_NAME = **Person**

MapleParty	CRITICALDATAELEMENT	MapleParty/MapleParty.imm [INT]	APPLICATION	GROUP_NAME	ELEMENT_NAME [V...]	SYNCPURPOSE_TP_CD	ENTITY_TYPE	INSTANCE_ID	ACTIVE_IND [CHAR(1)]	ULTIMATE_PARENT_GROUP_NAME
10032		TCRM	PersonName	LastName	2				Y	Person
10033		TCRM	TCRMPartyCDC...	CDCStatusType	2	CDCSta...	2		Y	TCRMMultiplePar...
10034		TCRM	TCRMMultiplePar...	ComponentID	2				Y	
10035		TCRM	PartyContactMe...	ContactMethodUsa...	2	Contact...	2		Y	Person
880001		TCRM	Person	DisplayName	1				Y	Person
<new row>										

7. Repeat the previous steps to add another row with the following values:
- __ a. CRITICAL_ELEMENT_ID = **880002**
 - __ b. APPLICATION = **TCRM**
 - __ c. GROUP_NAME = **Person**
 - __ d. ELEMENT_NAME = **DisplayName**
 - __ e. SYNCPURPOSE_TP_CD = **2**
 - __ f. ACTIVE_IND = **Y**
 - __ g. ULTIMATE_PARENT_GROUP_NAME = **Person**

MapleParty	CRITICALDATAELEMENT	MapleParty	CRITICAL_ELEMENT_ID [BIGINT]	APPLICATION	GROUP_NAME	ELEMENT_NAME [V...]	SYNCPURPOSE_TP_CD	ENTITY_TYPE	INSTANCE_ID	ACTIVE_IND [CHAR(1)]	ULTIMATE_PARENT_GROUP_NAME	LAST_UPDATED_BY
			880001	TCRM	Person	DisplayName	1			Y	Person	
			880002	TCRM	Person	DisplayName	2			Y	Person	
<new row>												

8. Save the changes (CTRL+S)

In order to see our changes reflected in the PME derived data tables, we will first need to modify the algorithms in the next exercise.

End of exercise

Exercise 9a. Customizing the Physical algorithms

What this exercise is about

This exercise covers the default Physical PME algorithms for the PERSON member type. It also goes through customizing the algorithm with new standardization's, buckets and comparisons to 2 new attributes.

What you should be able to do

At the end of this exercise, you should be able to:

- Understand the default Physical PME algorithm for the PERSON member type
- Modify an algorithm for a new attributes including standardization functions, bucketing and comparison functions.

Introduction

In this exercise, we will modify the default algorithm to include our new attributes (Driving License and Username). Both will include a Bucketing function and group which will allow us to search using just these attributes. Each will have a comparison function that we determine the match score based on the number of edits between the values.

Although we won't test out the algorithm in the exercise, this will allow us in the next exercise to test our Buckets and generates the weights for the comparison functions.

Requirements

- Typical Installation of the InfoSphere MDM Developers workstation
- Internet access

Exercise instructions

Before we make any modifications to the algorithms, let's take a look at the current out of the box algorithm.

1. Open the product documentation for the Physical Party Duplicate Suspect Processing Algorithms found at

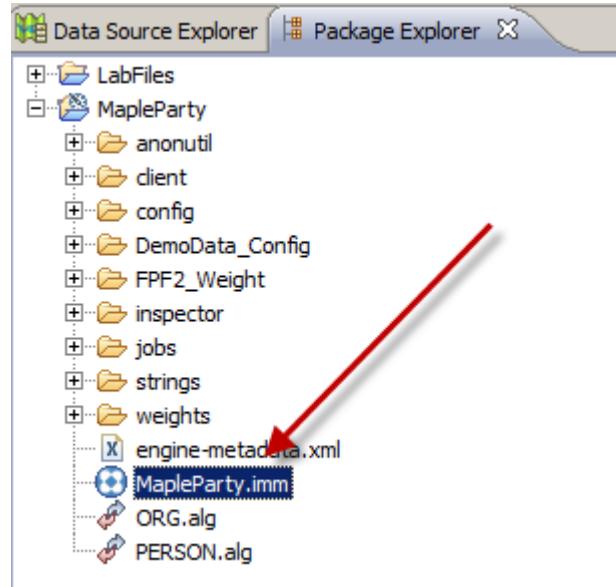
http://www-01.ibm.com/support/knowledgecenter/SSWSR9_11.0.0/com.ibm.mdmhs.de_v.party.doc/references/r_personalalgorithm.html. NOTE: there is a link on the desktop named **Physical PME Person Algorithm**

Table 1. Person algorithm (mdmnsperson)

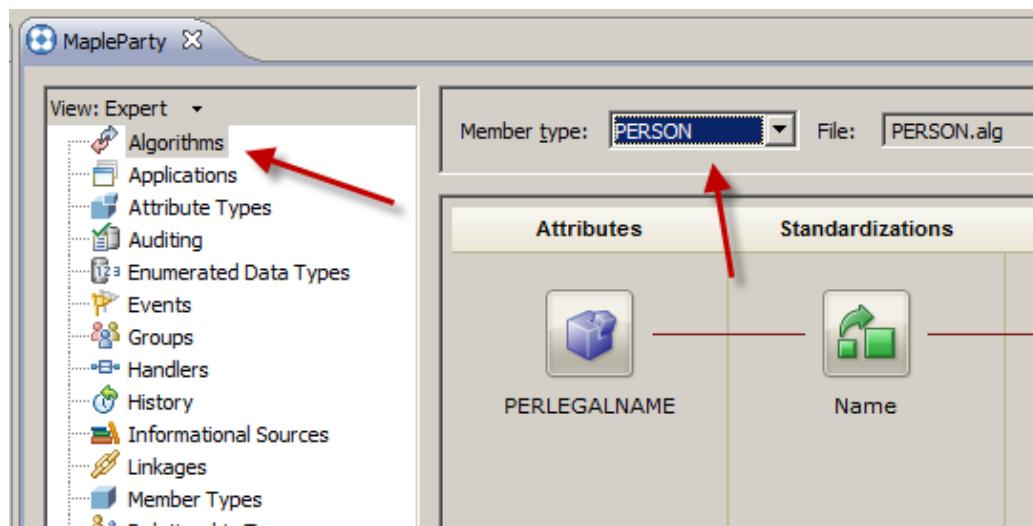
Attribute	Description
Name	The match score is based on the best match across any of the name attributes. For example, if the Legal Name from record one exactly matches the Alias Name from record two, the score from that comparison will be used for the Name score. Exotic name tokens will score higher than common tokens. For example, a match of "Xavier Q. Public" will score higher than a match of "John Doe".
Person Address and Phone (Combined)	The match score is based on the best match across any of the person address attributes, combined with the best match across any of the person phone attributes. For example, if comparing a person's primary residence and home phone number with another person's temporary residence and their mobile phone yields the highest match score, then those attributes will be used.
Business Address and Business Phone (Combined)	The match score is based on the comparison of the business addresses and business phone numbers between two person records. Partial scores may be given for non-exact matches or for situations where either a business address or business phone number is not available.
Birth Date	A straightforward comparison of birth dates between two person records. Partial scores may be given for non-exact matches.
Social Security Number	A straightforward comparison of one person's Social Security Number (SSN) with another person's SSN. Unlike with the MDM Classic Matching Engine, SSN numbers that differ by only one or two digits will still add to the match score, although less than a perfect match would.
Social Insurance Number	A straightforward comparison of one person's Social Insurance Number (SIN) with another person's SIN. Unlike with the MDM Classic Matching Engine, SIN numbers that differ by only one or two digits will still add to the match score, although less than a perfect match would.
Gender	A simple 'match' or 'no match' comparison, since there cannot be partial matches.

From the documentation, you'll find the description of the comparison algorithm used for each of the attribute types. Let's take a look at the algorithm to see how these descriptions match up to the configuration.

2. Open the **MapleParty.imm** configuration file found under the **MapleParty** project.



3. In the configuration file select the **Algorithms** tab. Under the Member type, select the **PERSON** from the dropdown list.



Take a look at the algorithms that are defined. In the following section we will break down each of the derived data, bucketing and comparison functions that are defined for the attributes.

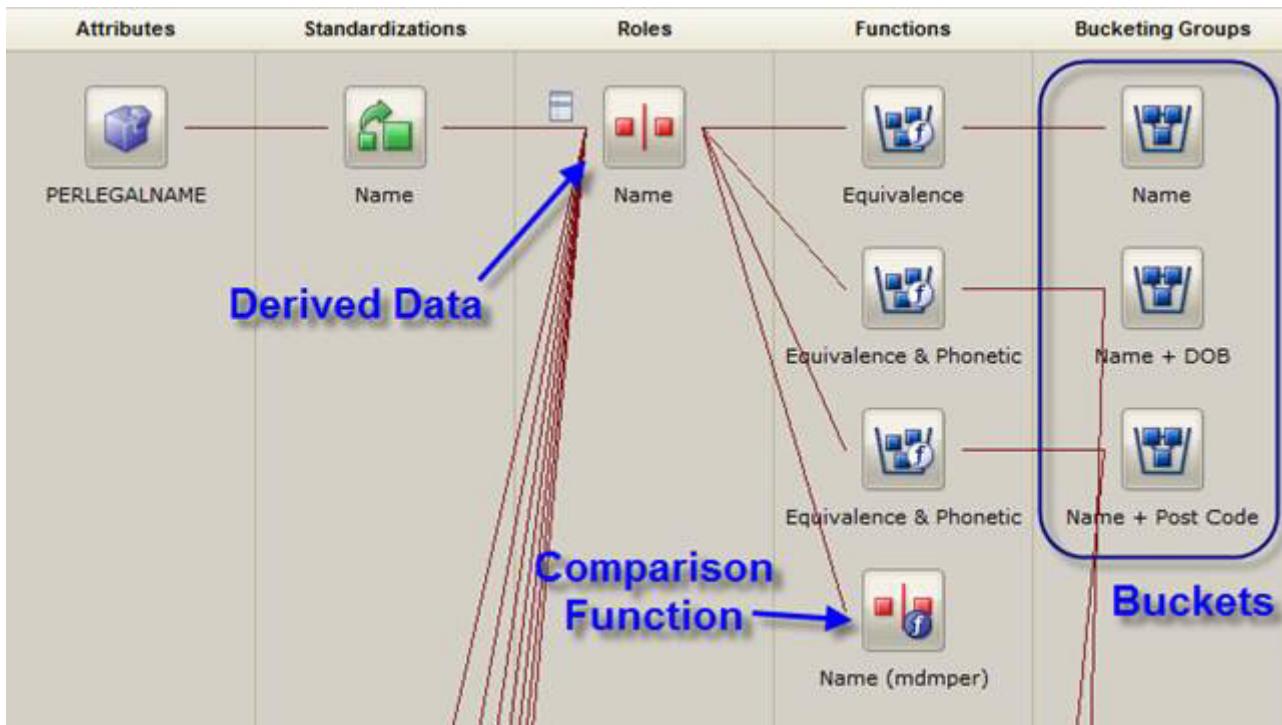
The first attribute we will look at is the **Name**.

Derived Data: The Name derived data is comprised of either Person Legal Name, Person Alias Name, Person Maiden Name, Person Preferred Name, Person AKA Name, Person Nickname, or Person Business Name.

Standardization: Each of the names go through a standardization function that eliminates anonymous values (e.g. Mother).

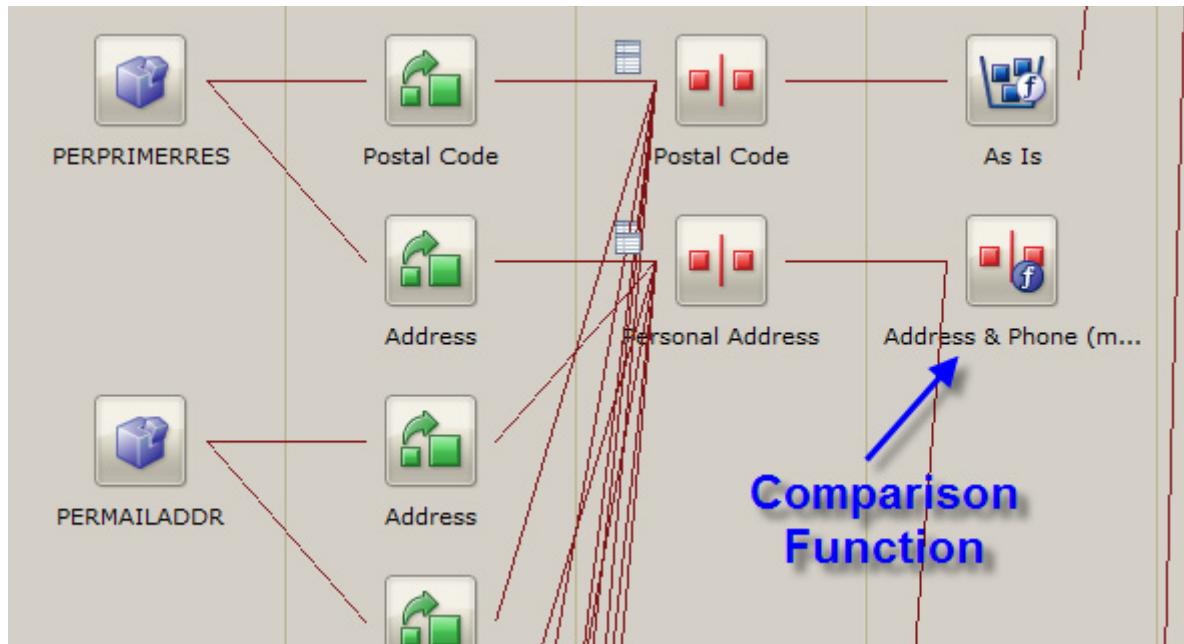
Buckets: There are 3 buckets defined for the Name derived data. The first one is an exact match on 2 of the tokens from the name (e.g. givenname1, lastname). The second bucket is a combination between the Phonetic version of the Name and Match of the DOB. The third Bucket is a combination of the Phonetic version of the Name and the Postal Code (or ZIP code).

Comparison: The comparison of the name uses the QXNM Metaphone function. This will produce a value that is added to the overall match score between records that belong to the same bucket.

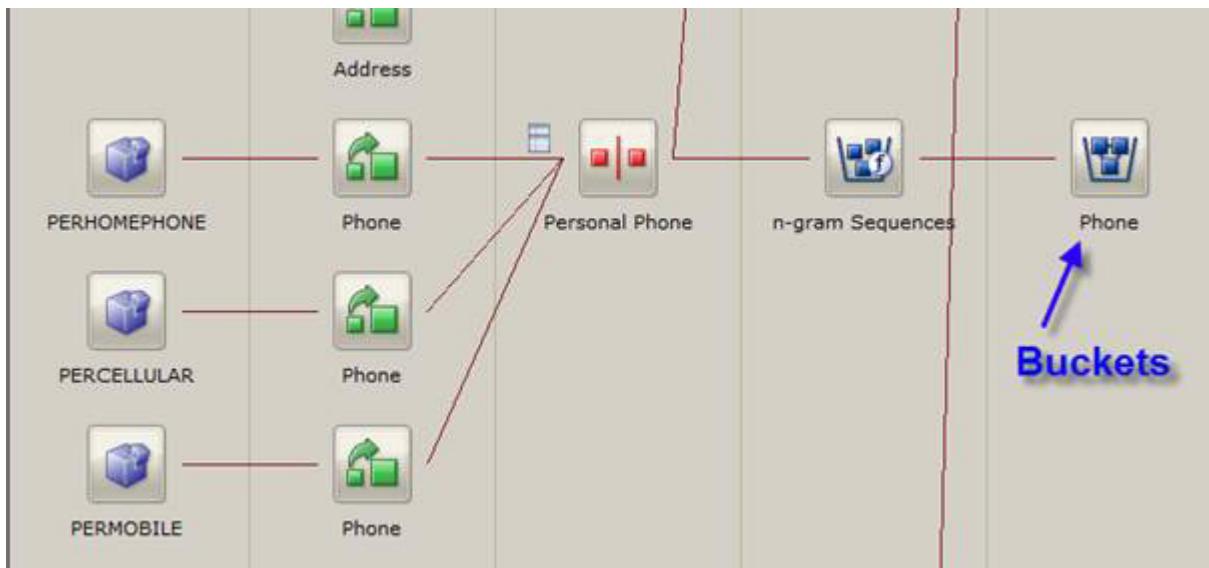


The next area we will look at is the **Address and Phone** Comparison. In the case of the Address and Phone there are no buckets, this means that if only the address matches, there will not be a score as it will not be compared to any other record (only records from the same bucket are compared to each other).

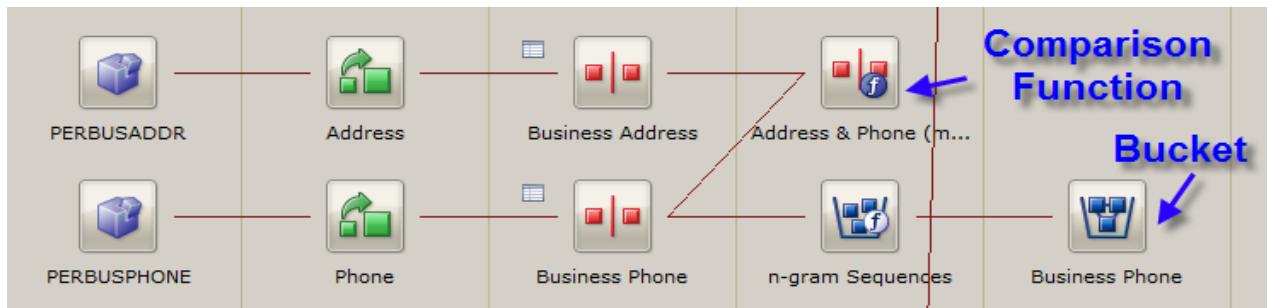
The Address and Phone uses a 2 dimensional weight that combines the comparison of the Address and the comparison of the phone to come up with a value. We will look at the weights in the next unit.



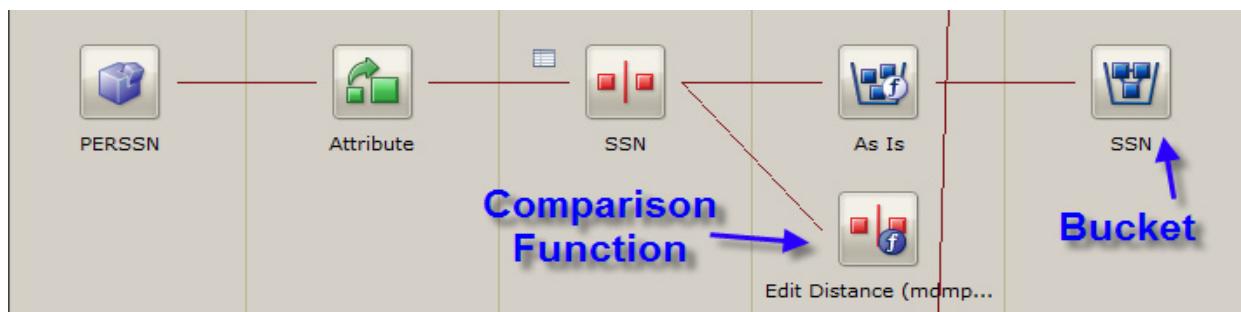
Next in the Algorithm, you will find a **Phone** Bucket. This uses a n-gram Sequences bucketing function (using 6 as the N value) which allows values that contain 6 digits of the phone number to belong to the same bucket.



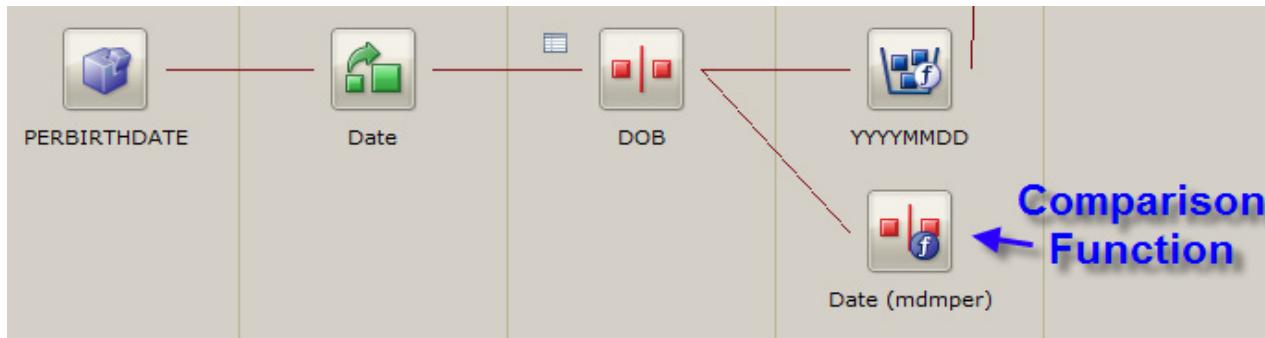
The Business Address and Phone is similar to the Personal Address and Phone, there is one bucket for the Business Phone using the n-gram Sequences (where $n = 6$) and a comparison using a 2 dimensional weight between the Business Address and Phone.



The SSN also has a Bucket for any SSN that matches and has a comparison function that uses edit distance to calculate the score.



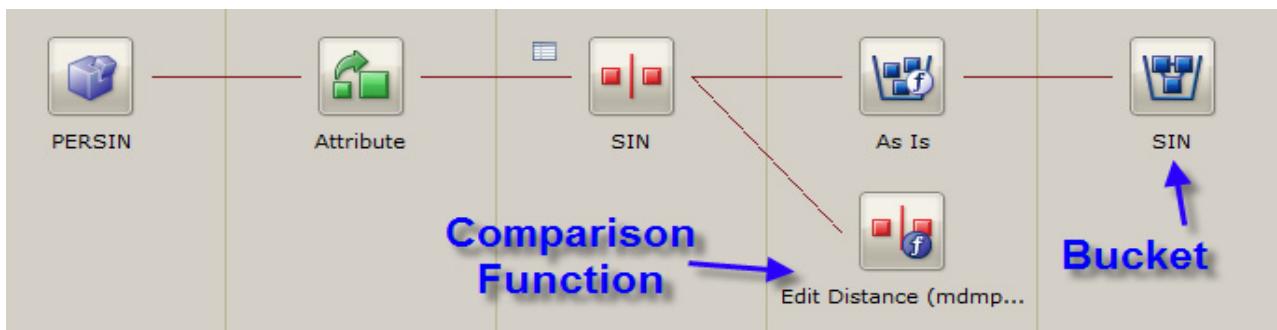
The Person birth date is used in one of the Name buckets (e.g. match a name token and DOB). If two members belong to the same bucket that will be compared using the DATE comparison, that will give some value if the day, month or year match.



The Person gender will not have any buckets as they would be too large, however they will be used in the Equivalency comparison that provides a value if the gender matches. The score is based on the frequency of the gender. For example, if there are less males than females in the sample set, then matching a male would provide a slightly higher score than matching a female.



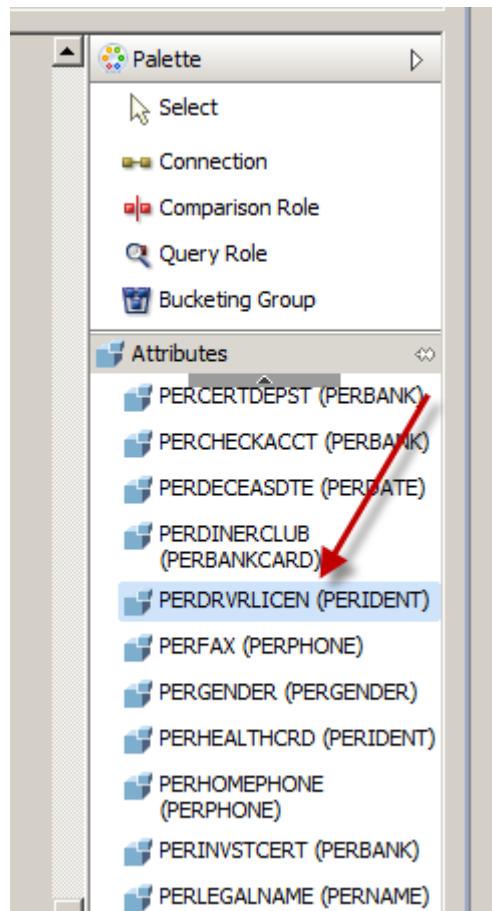
Similar to the Person SSN, the Person SIN also has a Bucket for any SIN that matches and has a comparison function that uses edit distance to calculate the score.



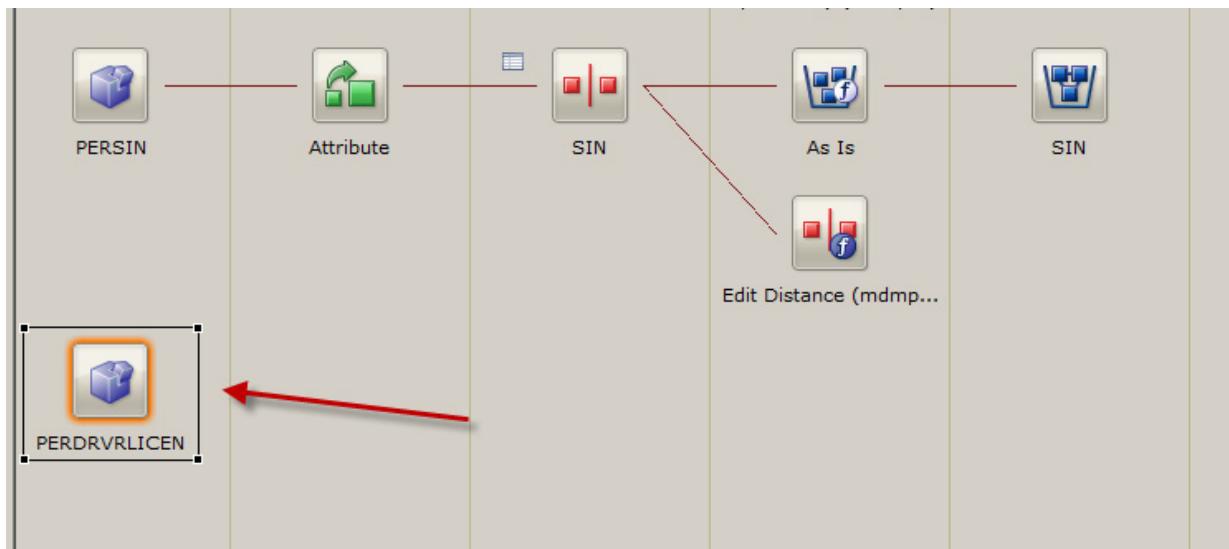
Part 1: Adding the Driving License to the Algorithm

Now that we know the default algorithm, we can modify it to meet our requirements (using the Person Username and Driving License. These 2 attribute are not currently part of the algorithm.

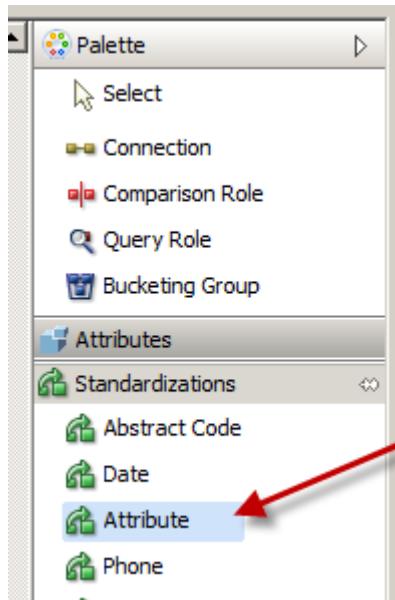
- 1. On the right hand side of the algorithm, you'll find a **Palette** that contains **Attributes**. Under the attributes tab, select the **PERDRVRLICEN (PERIDENT)**



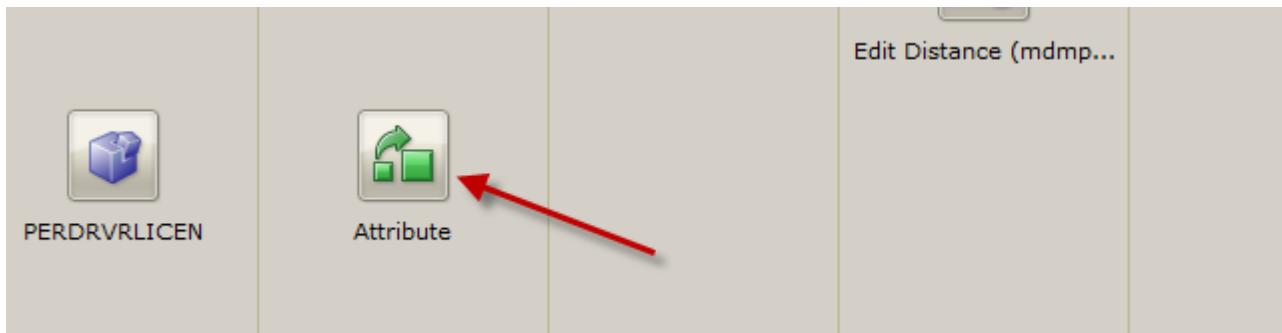
2. Place the **PERDRVRLICEN** attribute at the bottom of the algorithm (under the PERSIN) in the first column.



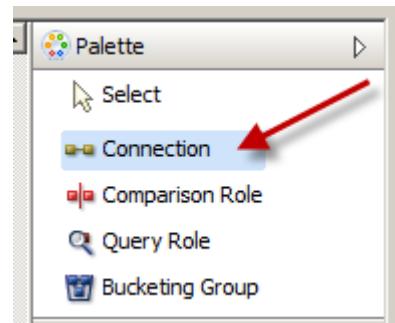
3. Back under the Palette, under the **Standardizations**, select the **Attribute** standardization function.



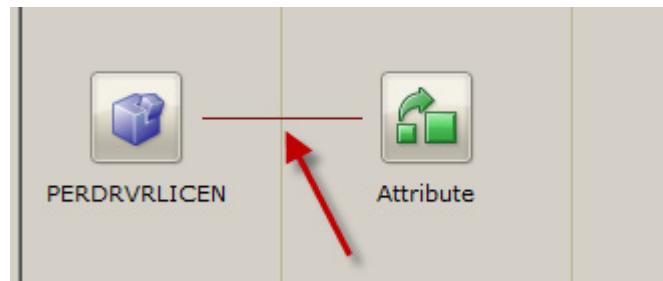
- ___ 4. Place the **Attribute** standardization function in the second column beside the PERDRVRLICEN



- ___ 5. Back under the Palette, select the **Connection** widget.



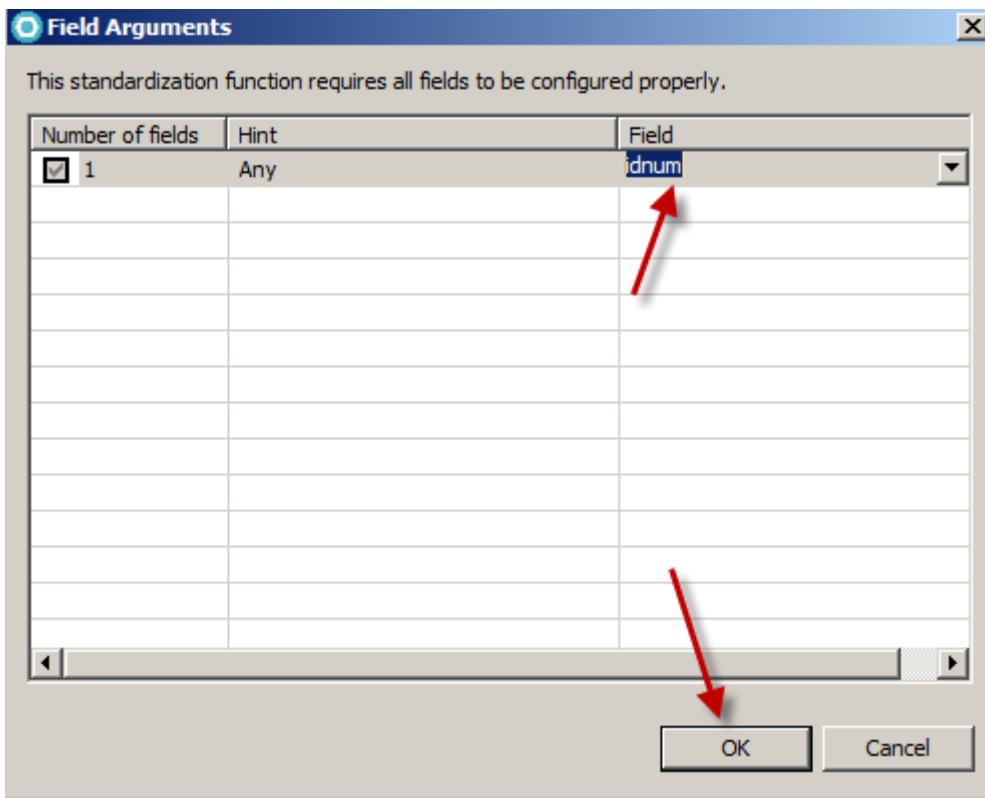
- ___ 6. Connect the **PERDRVRLICEN** to the **Attribute** standardization function using the Connection tool.



7. Select the Attributes standardization function, under the Properties window in RAD, click the ... button beside the **Field arguments**.

Property	Value
Anonymous string code	
Character mapping string code	
Derivation arguments	
Description	
Equivalent string code	
Field arguments	
Primary standardization role	30
Primary standardization role label	AI
Record status filter	ATTR
Standardization function code	
Type	Alphanumeric

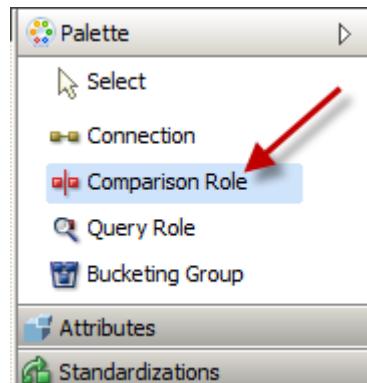
8. Under the Field column for the 1st row, select **idnum** from the dropdown and click the **OK** button.



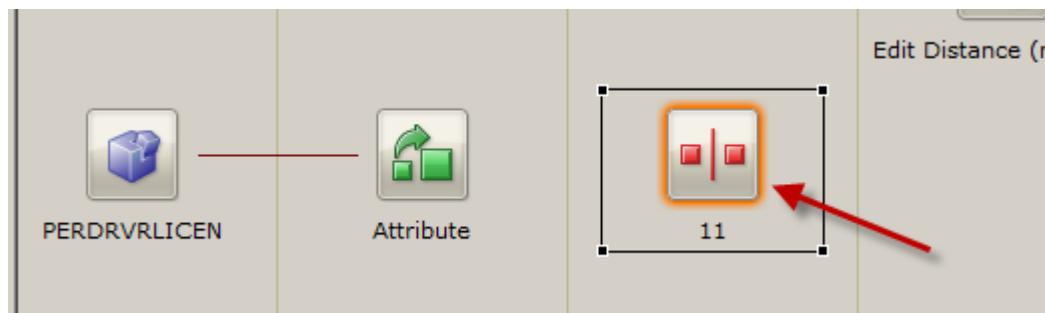
- ___ 9. Under the **Anonymous string code**, select **SSN**. We will use the SSN Anonymous values, which will prevent comparisons of 111111111, 99999999, etc. In a real implementation you might create your own anonymous values that may differ from the SSN anonymous values.

Property	Value
Anonymous string code	SSN
Character mapping string code	
Derivation arguments	
Description	
Equivalent string code	
Field arguments	
Primary standardization role	
Primary standardization role label	
Record status filter	
Standardization function code	
Type	

- ___ 10. Back under the Palette, select the **Comparison Role** tool.



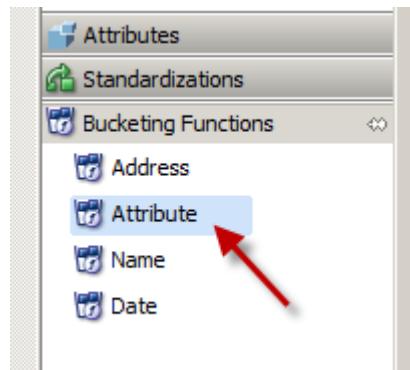
11. Place the Comparison Role in the **3rd** column in the PERDRVRLICEN row.



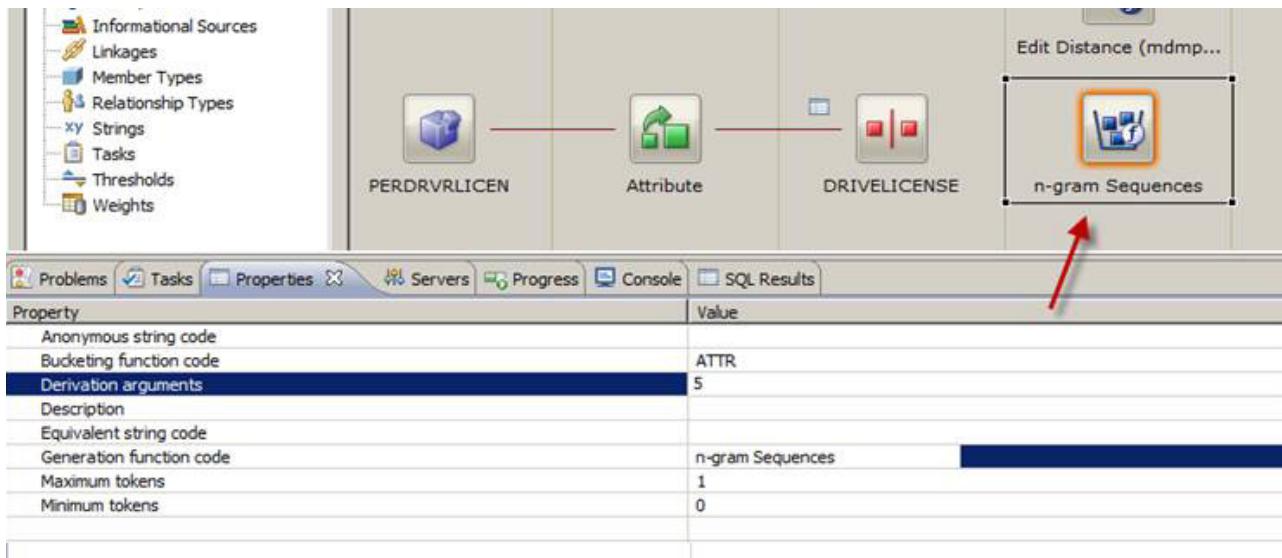
12. Select the Comparison role, under the **Properties** tab, enter **DRIVELICENSE** for the label.

Property	Value
Comparison role	11
Description	
Label	DRIVELICENSE

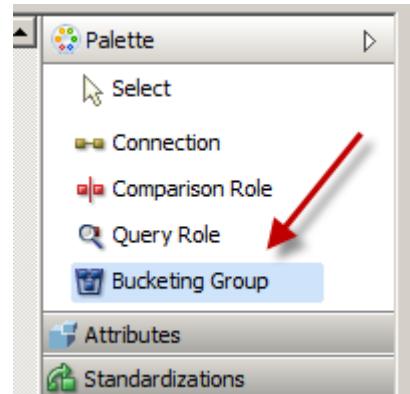
13. Under the Bucketing Functions, select the **Attribute** function.



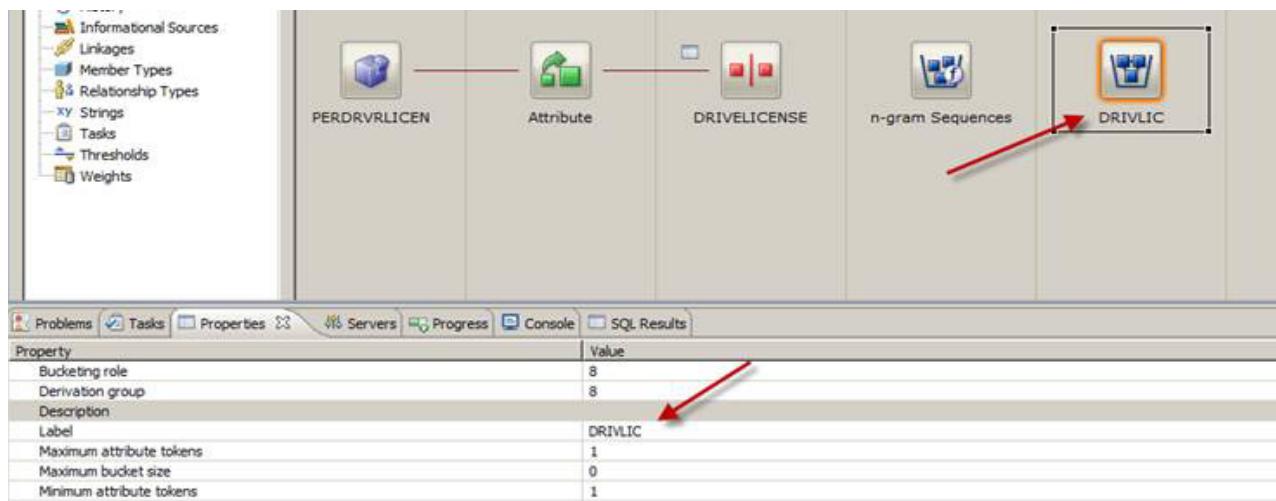
- ___ 14. Place the **Attribute** Bucketing function in the PERDRVRLICEN row in the 4th column.
Select the bucketing function and change the following values in the Properties window:
- ___ a. Generation function code: **n-gram Sequences**
 - ___ b. Derivation argument: **5**



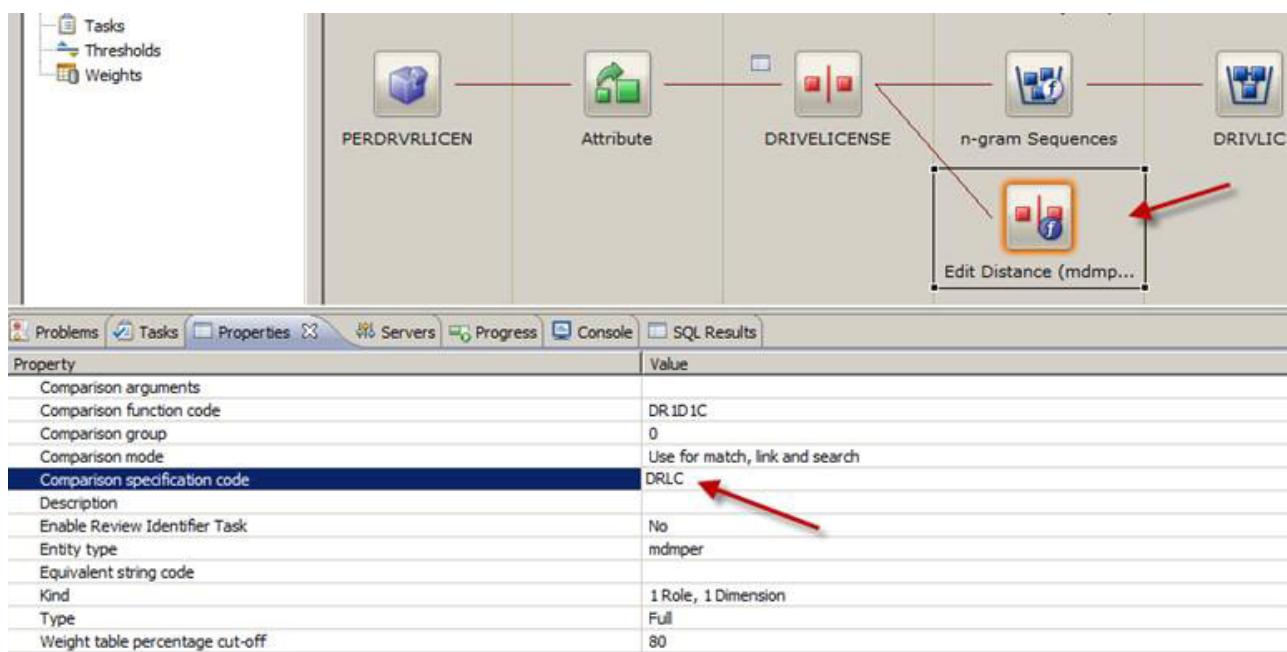
- ___ 15. Under the Palette, select the **Bucketing Group** tool.



- ___ 16. Place the Bucketing Group in the 5th column of the PERDRVRLICEN row. Under the properties tab, change the Label to **DRIVLIC**



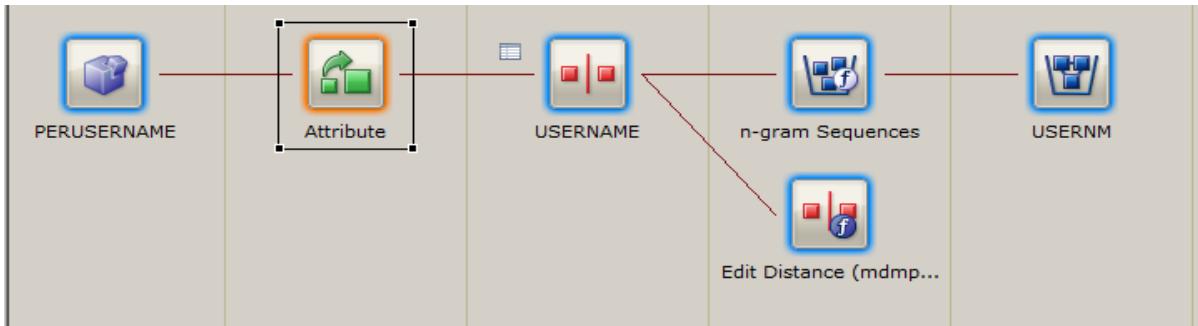
- 17. Using the **Connection** tool, connect the **DRIVELICENSE** comparison role to the **bucketing function** and the **bucketing function** to the bucketing group **DRIVLIC**.
- 18. From the Palette, select the **Edit Distance** tool from the **Comparison Functions** heading. Place the Edit Distance function in the 4th column under the PERDRVRLICEN row. Change the **Comparison specification code** to **DRLC** under the properties tab.



Part 2: Creating the Username Algorithm

- 1. Repeat the previous steps to create another piece of the algorithm for the Username, with the following configuration
 - a. Attribute: **PERUSERNAME**
 - b. Standardization: **Attribute**
 - Field arguments: **idnum**

- __ c. Comparison Role Label: **USERNAME**
- __ d. Bucketing Function: **n-gram Sequences**
 - Derivation arguments: **5**
- __ e. Bucketing Group Label: **USERNM**
- __ f. Comparison Function: **Edit Distance**
 - Comparison Specification code: **USRNM**



- __ 2. Save the configuration when complete (CTRL+S)

Although we are finished modifying the algorithm, we have not yet generated any weights for our comparison function or decided whether the buckets that we created will not create a problem (e.g. too many comparison or too little comparisons)

End of exercise

Exercise 9b.Bucket Analysis

What this exercise is about

This exercise covers loading some sample data and running the bucket analysis.

What you should be able to do

At the end of this exercise, you should be able to:

- Generate unloaded data files that can be used for Bucket Analysis and Weight generation.
- Running Bucket Analysis

Introduction

In this exercise we will use some sample data to run a bucket analysis on our algorithm. It is important to run this step during any customization of the algorithm as buckets perform an important role in creating a subset of records we will be comparing.

A bucket that contains too many records will impact performance and algorithm buckets that do not contain two records that could potentially be matches could miss duplicates.

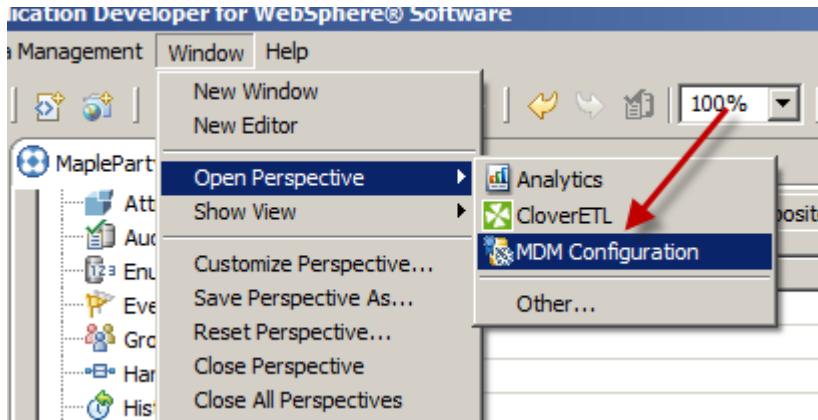
Requirements

Each of these exercise depend on the previous exercise being complete. For this exercise you must have the Physical MDM PME configuration imported into the workbench

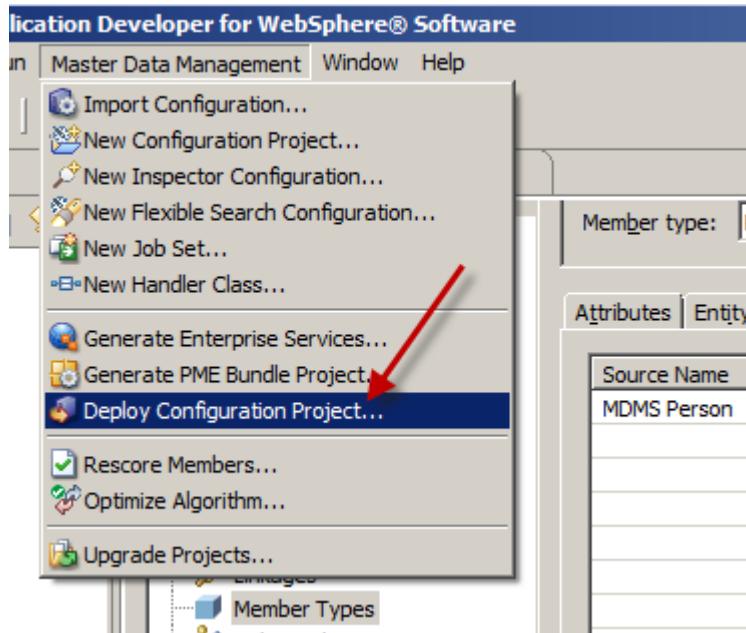
Exercise instructions

Part 1: Deploying the MDM Configuration

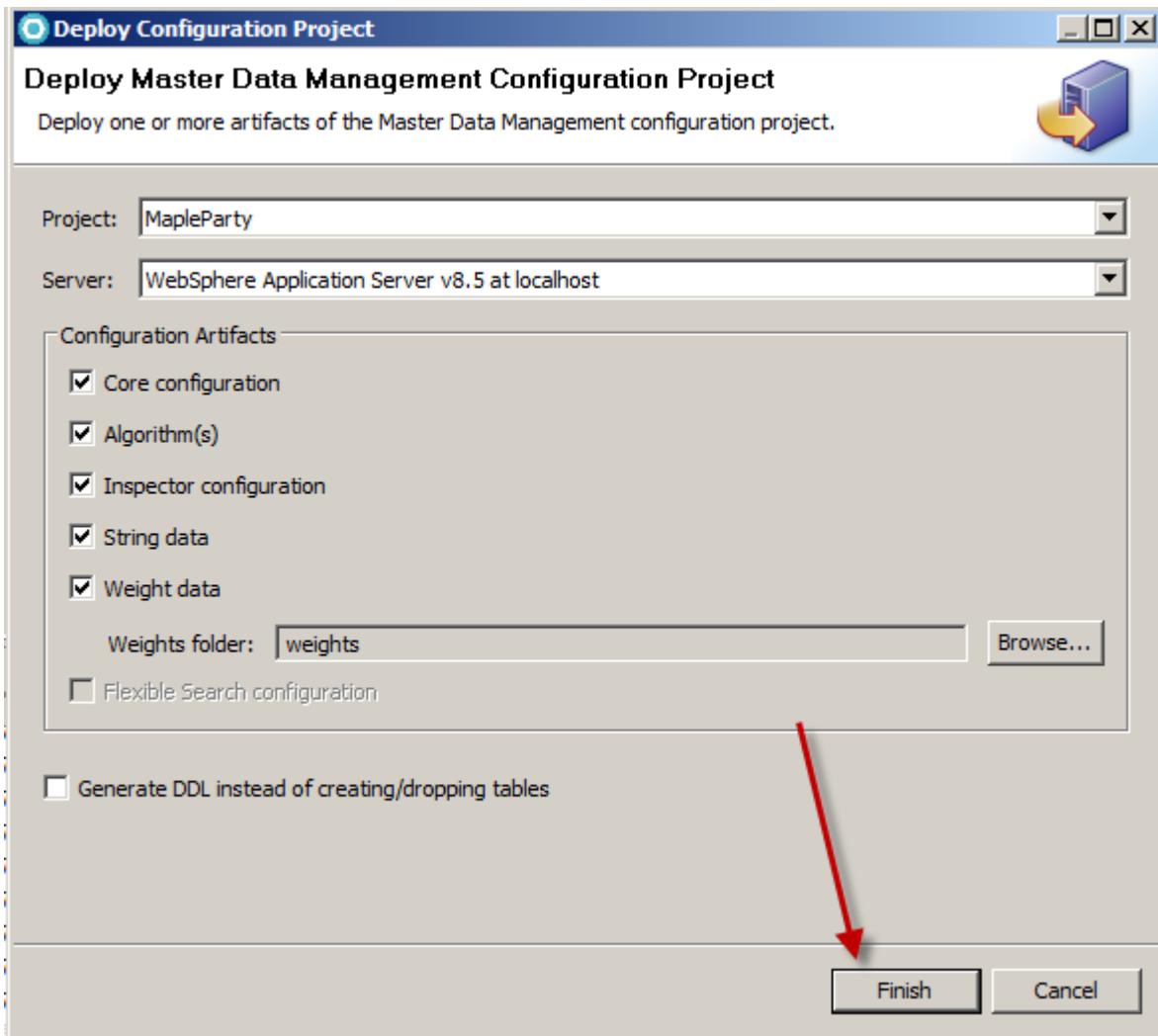
- 1. We will be working in the MDM Configuration perspective in this exercise, to switch to the perspective, in the RAD File menu, select **Window > Open Perspective > MDM Configuration**



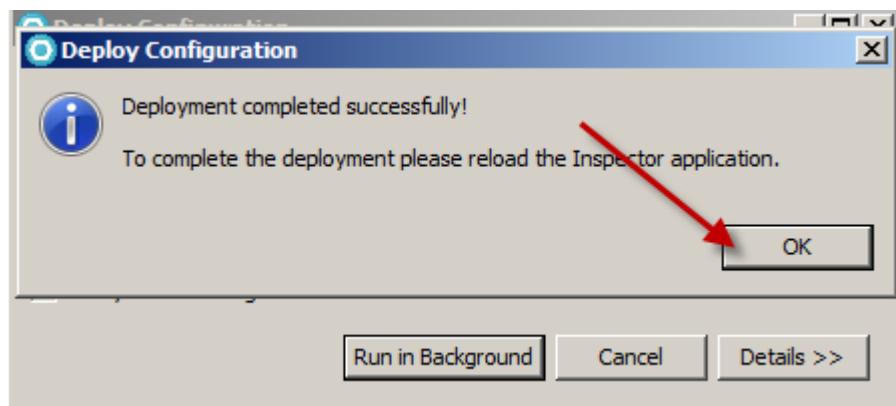
- 2. Before we can run the analysis, we need to deploy the algorithm to the InfoSphere MDM. From the RAD file menu, select **Master Data Management > Deploy Configuration Project ...** (NOTE: you need to be in the MDM Configuration perspective to see this menu option.)



- 3. Leave all Configuration Artifacts selected and click the **Finish** button.



- 4. The deploying will take several minutes, once it's complete click the **OK** button.



Part 2: Deploying Sample Data

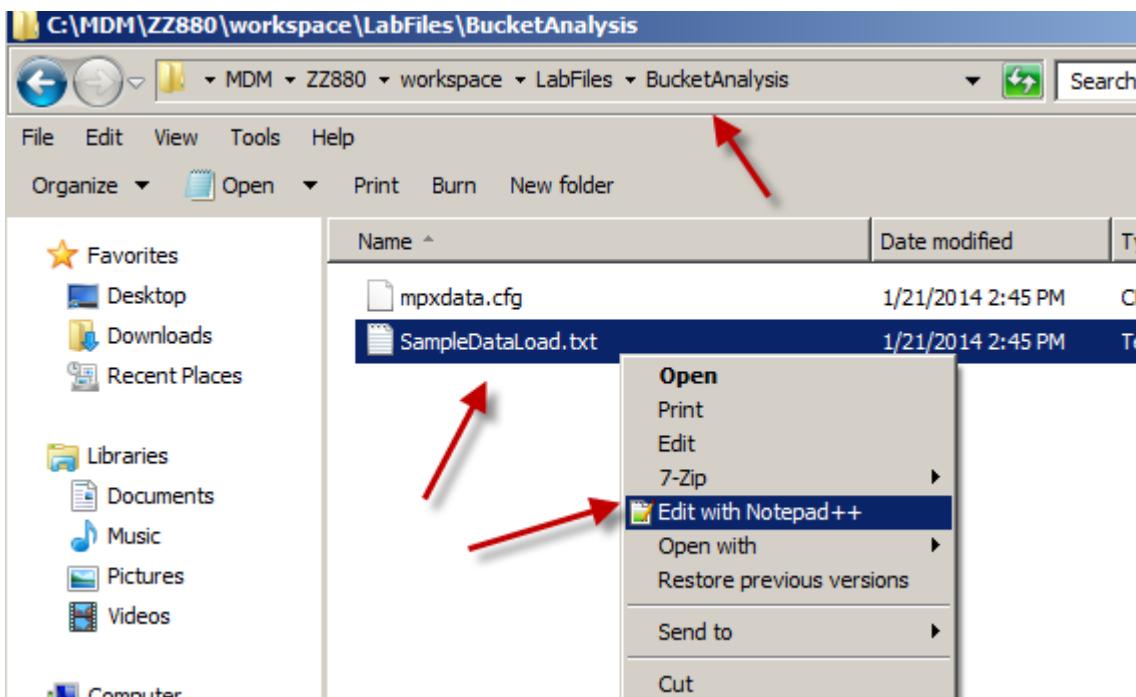
Now that the configuration (including the algorithms) is deployed, in order to test our buckets, we need to load some sample data. Luckily we've already produced a data load file for you that will populate the InfoSphere MDM with sample data that will test our algorithms.



Note

When working with the algorithms, bucket analysis, weight generations, etc. You will want to have a sample data load of at least 250,000 records to provide proper analysis and weight generation. The higher the number the better and the sample set should be representative of the data that will be used in production.

1. To see the sample data file that we will use in this exercise, open a windows file explorer and navigate to **C:\MDM\ZZ880_Physical\workspace\LabFiles\BucketAnalysis**. Right click on the **SampleDataLoad.txt** and select **Edit with Notepad++**. (NOTE: this file could be produced by any ETL tool, such as DataStage from a source database)



2. In the **SampleDataLoad.txt** you'll find **465,142** records delimited by a | character.

```

1 MDMSP|435202|M|BROSE|PERRY|O|III|1943-09-22|5520 BUCHANAN ST.|APACHE JUNCTION|AZ|85260
2 MDMS2|435203|M|TEEPLE|DANIAL|E||1952-11-10|30829 SIXTEENTH ST.|TEMPE|AZ|85281
3 MDMSP|435204|F|HOWSE|ADELAIDE|W||1929-06-24|14553 TRIANON PLAZA|PHOENIX|AZ|85013
4 MDMS2|435205|F|CORNELL|DIANN|R||1900-02-26|25298 BENTON ST.|APACHE JUNCTION|AZ|85260
5 MDMS2|435206|M|BLAYLOCK|BABYBOY|A||2006-09-02|16724 CARROLLTON ST.|GLENDALE|AZ|85024
6 MDMSP|435207|M|SCHRAMEK|GREGORY|||1987-10-20|25694 CUMBERLAND STREET|LITCHFIELD PARK|AZ|85048
7 MDMSP|435208|M|WOLD|BABYBOY|L|III|2006-05-03|3872 OLIVER ST.|APACHE JUNCTION|AZ|85013
8 MDMS2|435209|F|BLASH|AILENE|A||1921-04-25|25621 FLOW COURT|PHOENIX|AZ|00000
9 MDMS2|435213|F|GOREE|RAY|R||1945-03-09|16773 HENRY STREET|TEMPE|AZ|85287|480
10 MDMS2|435214|M|LOAIZA|WARREN|A|II|2001-03-06|2527 ADAMSVILLE ROAD|MESA|AZ|85221
11 MDMSP|435217|M|JUHASZ|JULES|||1952-06-25|12616 PILIE ST.|PHOENIX|AZ|00000|||
12 MDMSP|435218|M|GARLICK|SEBASTIAN|R||1926-06-09|10402 DUNCAN STREET|GLENDALE|AZ|85025
13 MDMS2|435219|M|WESTRICK|BABYBOY|S||1900-03-16|6569 RANELAGH STREET|PHOENIX|AZ|85013
14 MDMSP|435220|M|EUSTICE|BABYBOY|S||2007-03-17|16290 KENILWORTH ST.|GLENDALE|AZ|85025
15 MDMS2|435221|M|CROZIER|BABYBOY|O||1909-07-01|13569 AVENUE "A".|PHOENIX|AZ|85021

```

- 3. To find out which each column in the delimited file represents, open the **mpxdata.cfg** file found under the same directory. This file maps the field position to the data model attribute, we will use this file when loading our data.

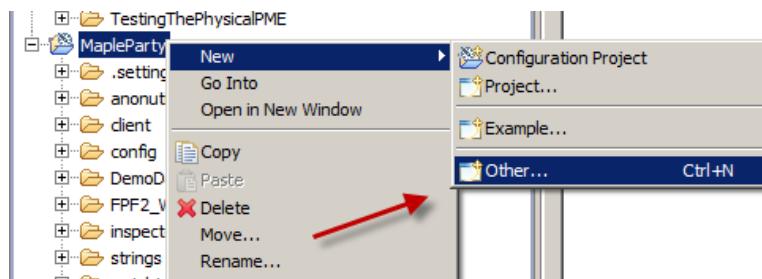
```

1 MEMHEAD 1 1 NA SetString srcCode
2 MEMHEAD 1 2 BL SetString memidnum
3 PERGENDER 1 3 BL SetString gender
4 PERLEGALNAME 1 4 BL SetString lastname
5 PERLEGALNAME 1 5 BL SetString givenname1
6 PERLEGALNAME 1 6 BL SetString givenname2
7 PERLEGALNAME 1 7 BL SetString suffix
8 PERBIRTHDATE 1 8 BL SetDate_Y4MD val
9 PERPRIMERRES 1 9 BL SetString addrline1
10 PERPRIMERRES 1 10 BL SetString city
11 PERPRIMERRES 1 11 BL SetString provstate
12 PERPRIMERRES 1 12 ZX SetString postalcode
13 PERHOMEPHONE 1 13 ZL SetString refnum
14 PERBUSPHONE 1 14 ZL SetString refnum
15 PERSSN 1 15 BL SetString idnum
16 PERDRVRLICEN 1 16 TR SetString idnum
17 PERBUSADDR 1 17 TR SetString addrline1
18 PERBUSADDR 1 18 TR SetString city
19 PERBUSADDR 1 19 TR SetString provstate
20 PERBUSADDR 1 20 TR SetString postalcode
21

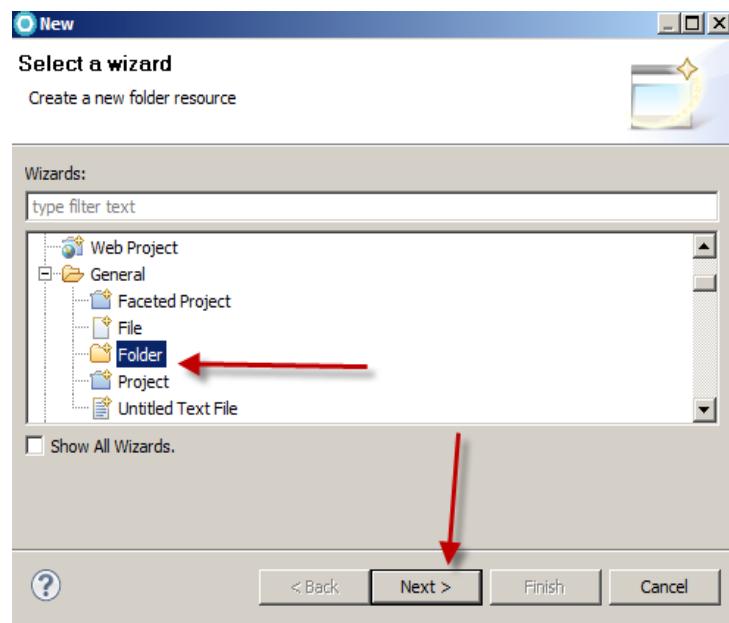
```

Field Position

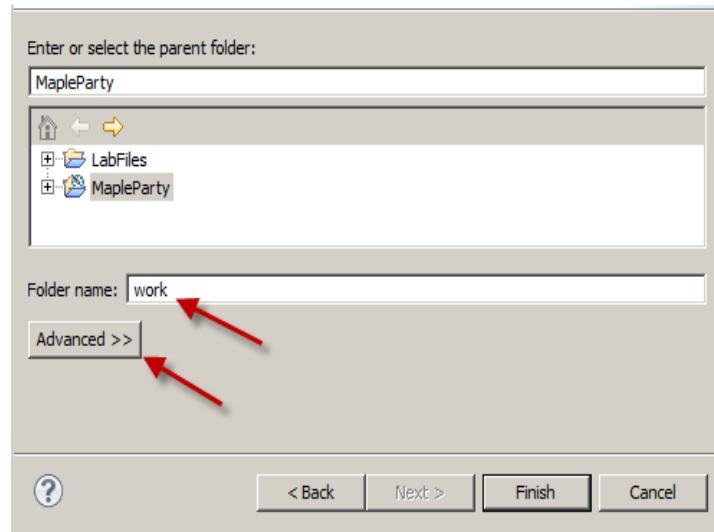
- 4. To load our data into the InfoSphere MDM, we will create a folder that is linked to the work directory of the MDM implementation. Right click on the **MapleParty** project and select **New > Other ...**



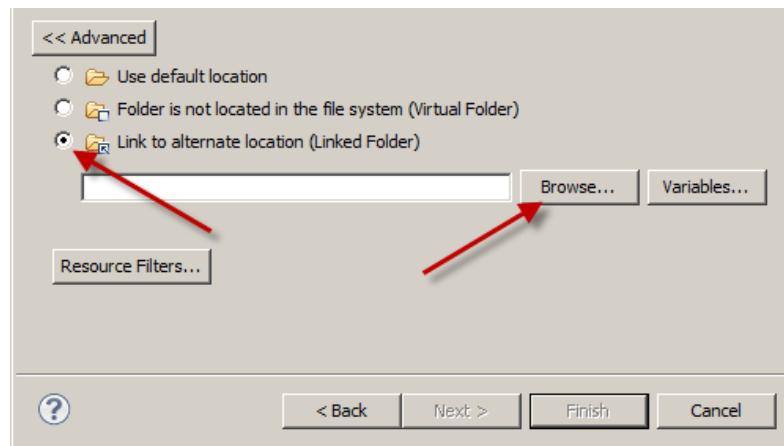
5. Select **General > Folder** and click the **Next >** button.



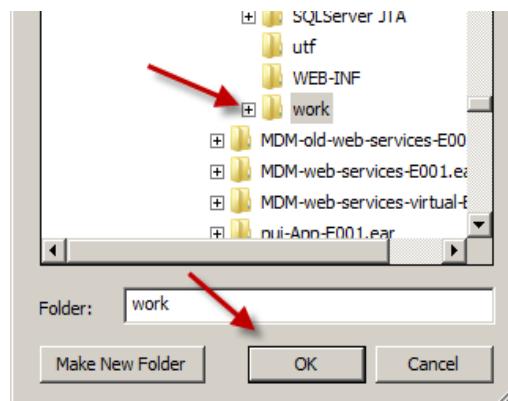
6. Enter **work** for the folder name and click the **Advanced >>** button.



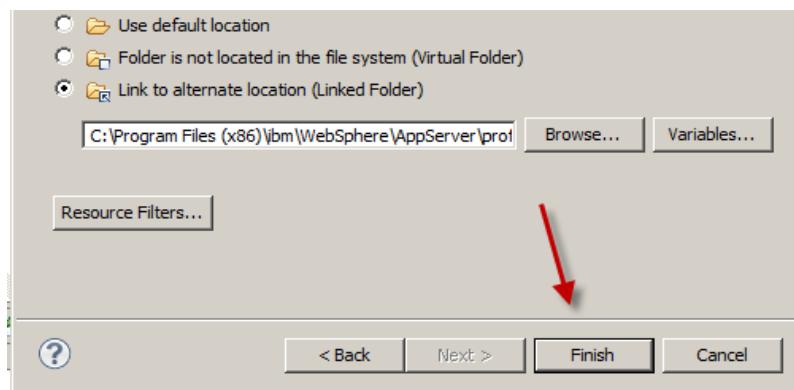
7. Select the **Link to alternate location (Linked Folder)** checkbox and click the **Browse** button.



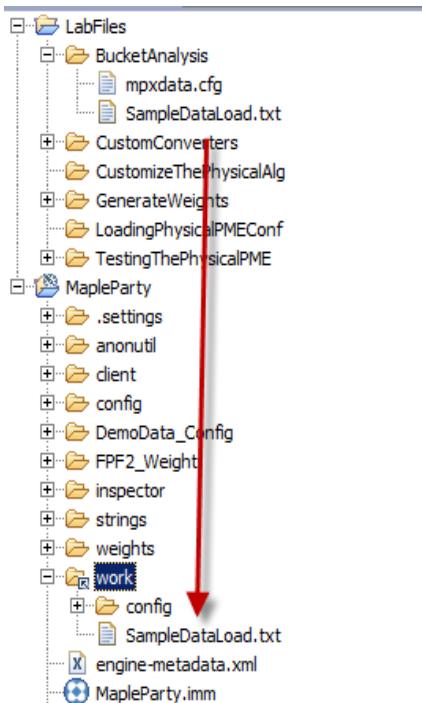
- 8. Navigate to: **C:\Program Files (x86)\ibm\WebSphere\AppServer\profiles\AppSrv01\installedApps\mdmCell\MDM-native-E001.ear\native.war\work\MapleParty\work** and click the **OK** button. (NOTE: there are 2 work folders in the path)



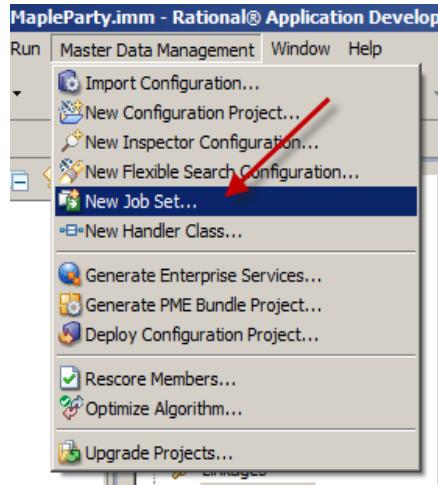
- 9. Click the **Finish** button to link the directory to your Workbench project.



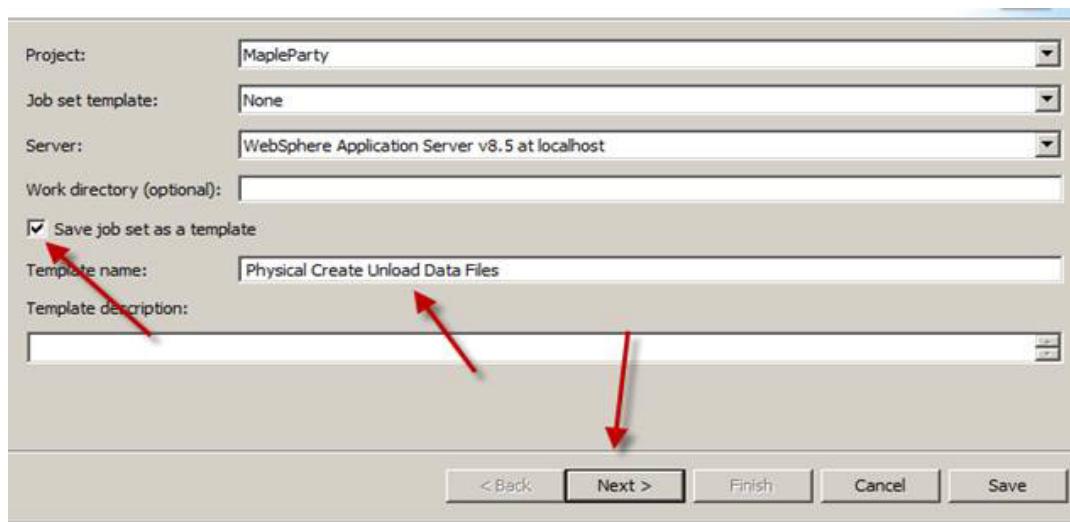
- 10. In order for MDM to access our data load, it must exist in the work path. In RAD, copy the **SampleDataLoad.txt** file into our work folder.



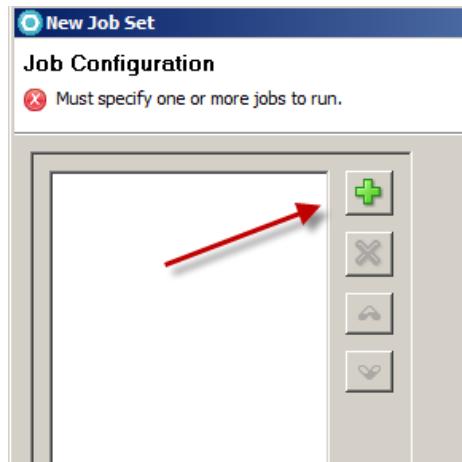
- 11. To load our data into the InfoSphere MDM, we will first create Unload files (these are files that are divided into a similar structure as the database tables. To create these files from our sample data file, we can run a create UNL job. From the RAD file menu, select **Master Data Management > New Job Set...**



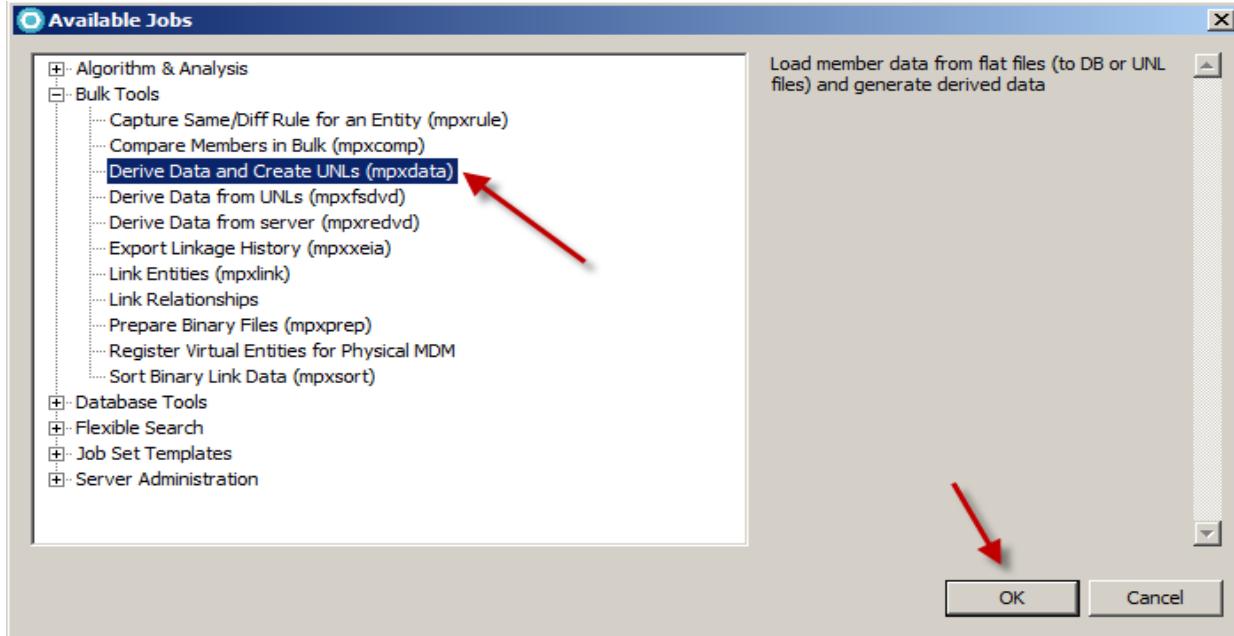
- 12. We will want to run this job a number of times, so we will save the job as a template. Select the **Save job as a template** checkbox and enter **Physical Create Unload Data Files** as the Template name, click the **Next >** button when complete.



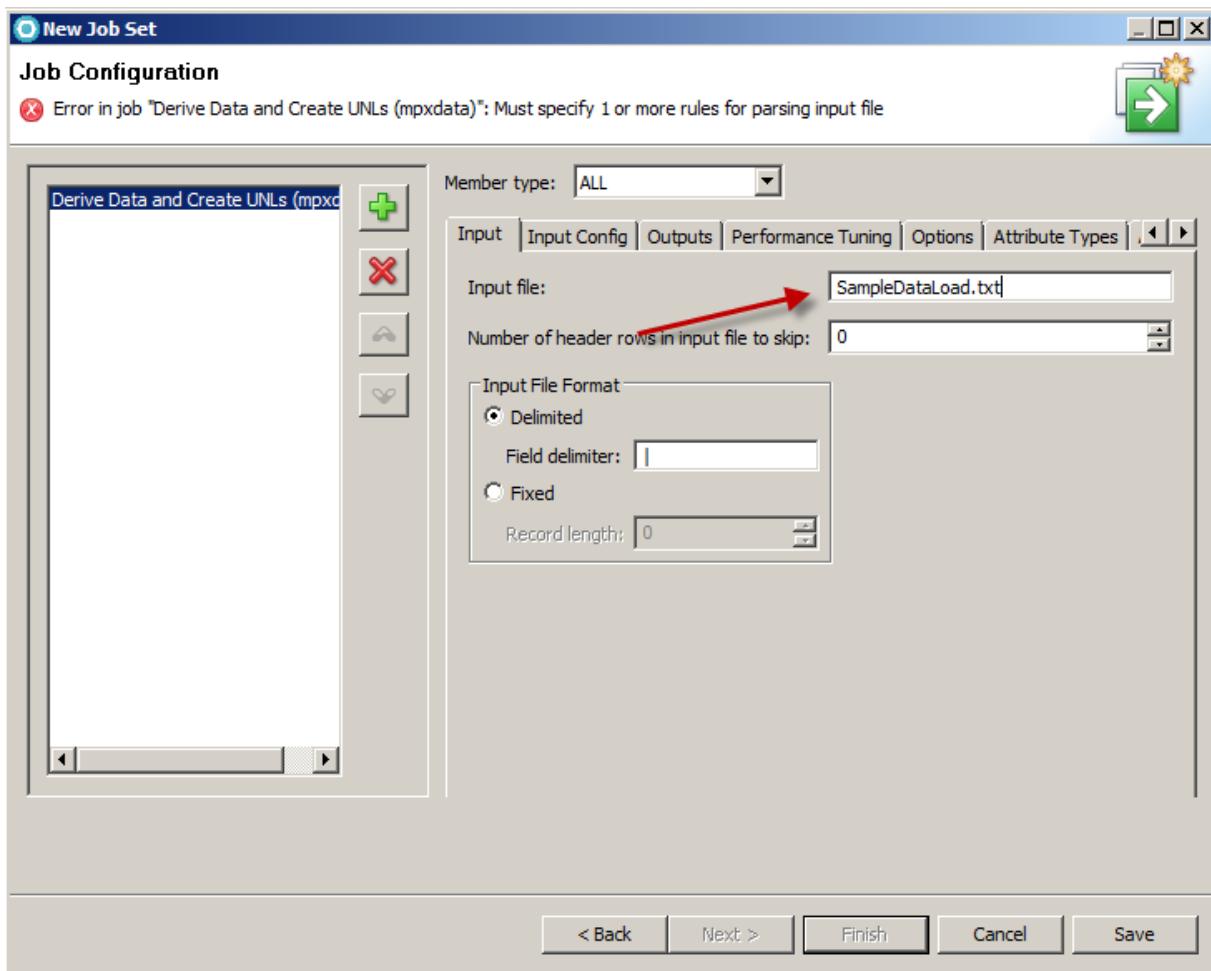
- ___ 13. Inside the New Job Set window, click the green plus button to add a new job task.



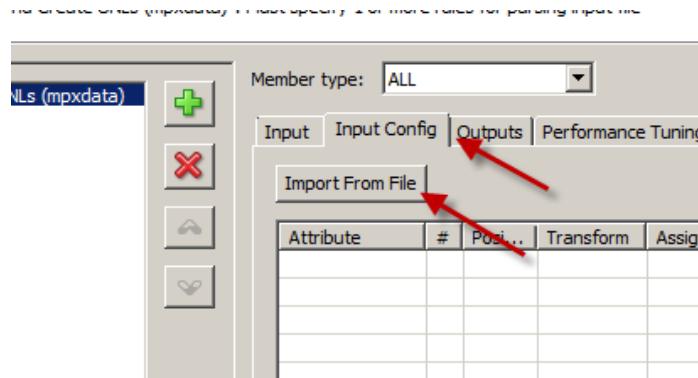
- ___ 14. Select Bulk Tools > Derive Data and Create UNLs (mpxdata) and click the OK button.



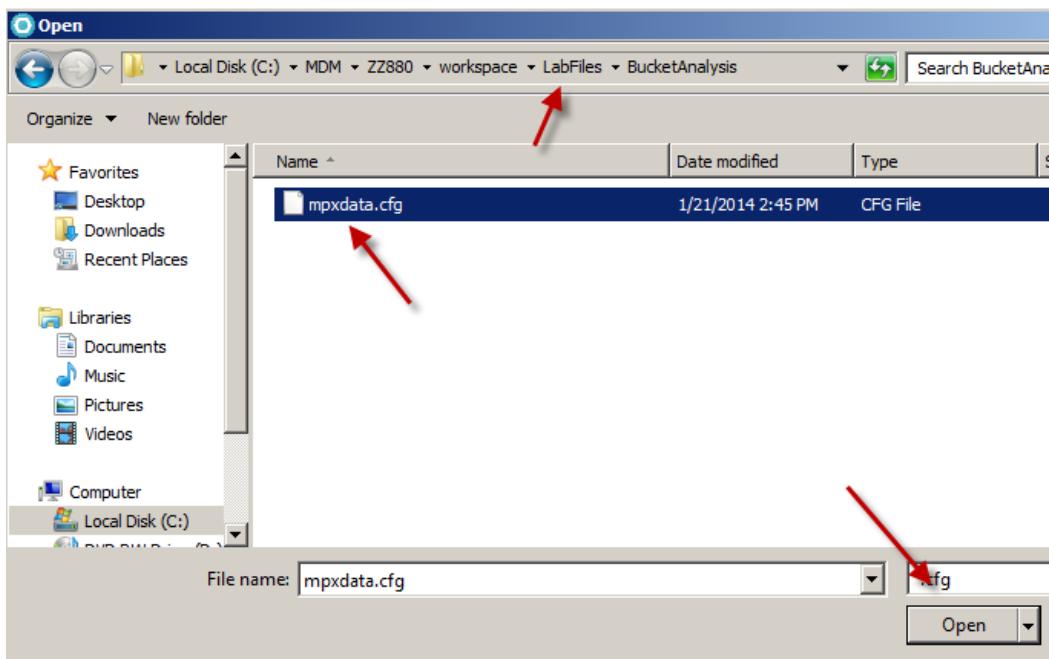
15. Under the Input tab, enter **SampleDataLoad.txt** for the Input file.



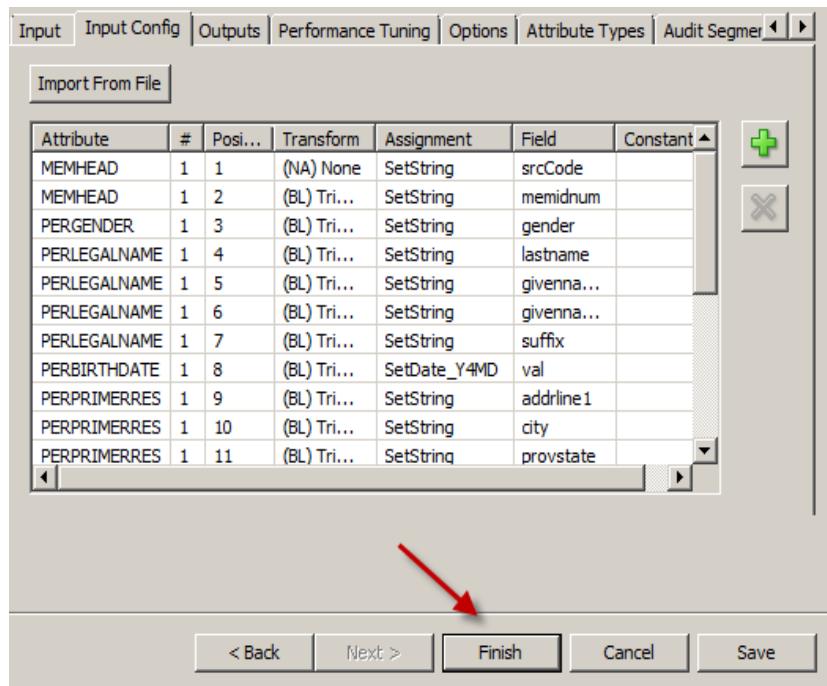
16. Select the Input Config tab and click the Import From File button.



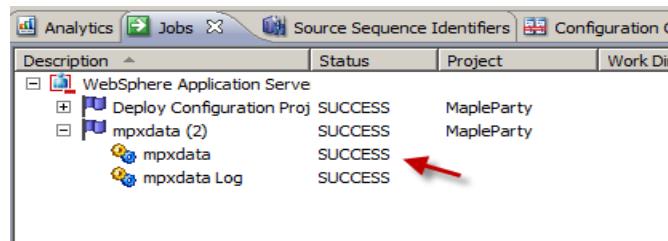
- 17. Navigate to C:\MDM\ZZ880_Physical\workspace\LabFiles\BucketAnalysis. Select the mpxdata.cfg file and click the Open button.



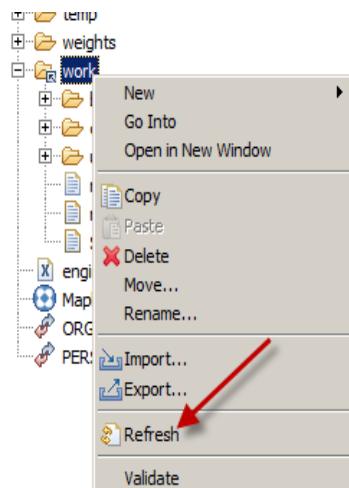
- 18. This will import the file that we saw earlier. In this view you can see what each column in the configuration file represented. Click the **Finish** button to run the job.



- ___ 19. The job will take a few minutes to run, to see the status, open the **Jobs** tab in RAD and view the status of the last job. Wait until the status changes to SUCCESS (NOTE: you may need to refresh the view from time to time using the refresh button.)



- ___ 20. Once the job is complete you can check to see if it created the UNL files by navigating to the **work** directory (that we created in our RAD workspace) under the unl folder. Right click on the **work** folder and select **Refresh**.



- 21. Under the unl folder you will find the unl files. To see the data that would be loading into the various table, open any of the unl files.

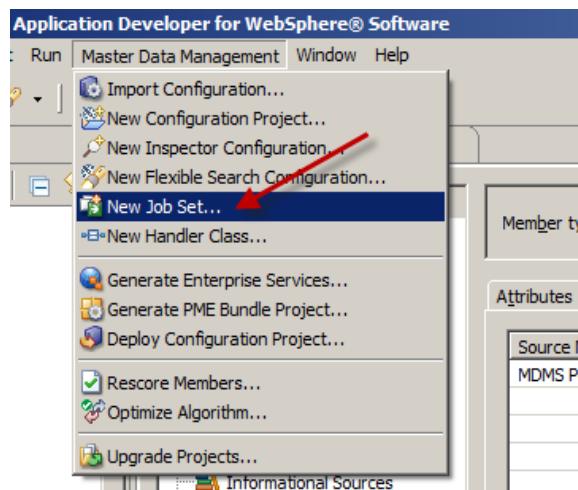


Note

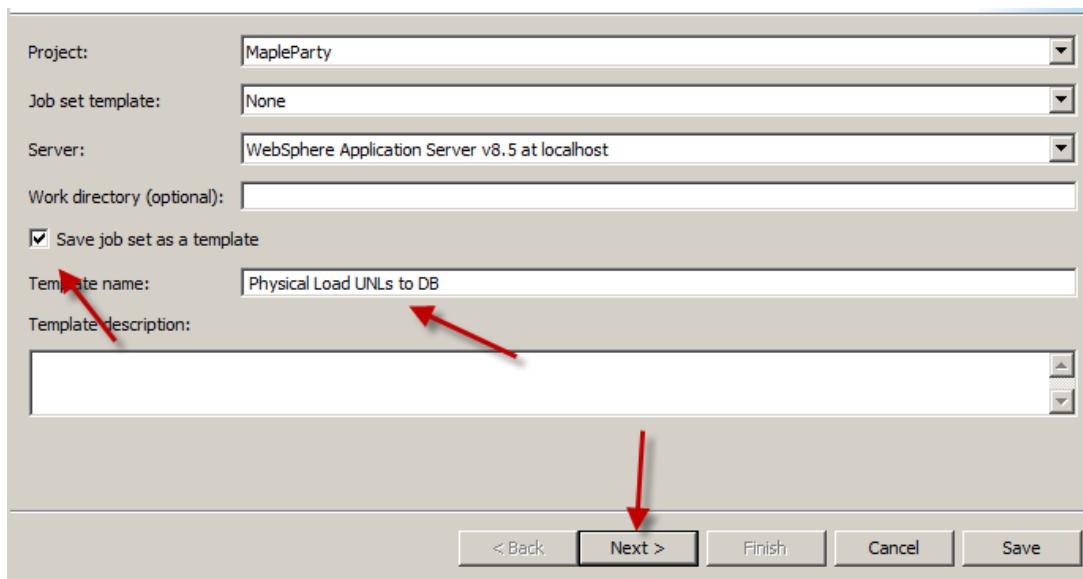
Although we are configuring the algorithm for the Physical MDM, while we are working with the Configuration we will be using data on the Virtual MDM. All the mpi_ tables are for the virtual module and will be used during configuration. Once we are happy with the configuration we will deploy to the Physical MDM and use the algorithm with the Physical tables.

	CRITICALDATAELEMENT	Inspector Configuration (MapleParty)	mpi_pername.unl
1	8 2 2 A 1 0 BROSE PERRY O I I I		
2	8 2 2 A 1 0 TEEPLE DANIAL E		
3	8 2 2 A 1 0 HOWSE ADELAIDE W		
4	8 2 2 A 1 0 CORNETT DIANN R		
5	8 2 2 A 1 0 BLAYLOCK BABYBOY A		
6	8 2 2 A 1 0 SCHRAMEK GREGORY		
7	8 2 2 A 1 0 WOLD BABYBOY L I I I		
8	8 2 2 A 1 0 BLASH AILENE A		
9	8 2 2 A 1 0 GOREE RAZ R		
10	8 2 2 A 1 0 LOAIZA WARREN A I I		
11	8 2 2 A 1 0 JUHASZI JULES		
12	8 2 2 A 1 0 GARLICK SEBASTIAN R		
13	8 2 2 A 1 0 WESTRICK BABYBOY S		
14	8 2 2 A 1 0 EUSTICE BABYBOY S		
15	8 2 2 A 1 0 CROZIER BABYBOY O		
16	8 2 2 A 1 0 SERNA KRISTIAN R		
17	8 2 2 A 1 0 HAGIN BABYBOY G		
18	8 2 2 A 1 0 HIGGENBOTHAM BETHANY G		
19	7 2 2 A 1 0 ROMAN TIMMY M		
20	8 2 2 A 1 0 WITHERSPOON JACKSON T		
21	8 2 2 A 1 0 SHARIF MELLIE		
22	8 2 2 A 1 0 SAKAI BABYBOY K I I I		
23	8 2 2 A 1 0 BURDINE BABYBOY R I I I		
24	8 2 2 A 1 0 MOOERS BABYBOY O		
25	8 2 2 A 1 0 VERDEJO BABYGIRL R		
26	8 2 2 A 1 0 VANEATON BABYBOY N		
27	8 2 2 A 1 0 DREWES LUIS E		
28	8 2 2 A 1 0 SHIRLEY EARLEAN I		
29	8 2 2 A 1 0 OLESEN NATALYA R		
30	8 2 2 A 1 0 CHESSER BABYBOY E		
31	8 2 2 A 1 0 KOZIK BABYGIRL		
32	8 2 2 A 1 0 WIND BABYBOY N J R		
33	8 2 2 A 1 0 HUGO DELORIS G		
34	8 2 2 A 1 0 MATTES CHRIS T		
35	8 2 2 A 1 0 MATTY DORETHEA T		
36	8 2 2 A 1 0 DURGIN BABYBOY R		
37	8 2 2 A 1 0 LUCKY MIKE C		

- 22. Next we will load the UNL files into the InfoSphere MDM database. To do this we will run another job. From the RAD file menu, select **Master Data Management > New Job Set...**



- 23. Select the **Save job set as a template** checkbox and name the template **Physical Load UNLs to DB**. Click the **Next >** button.

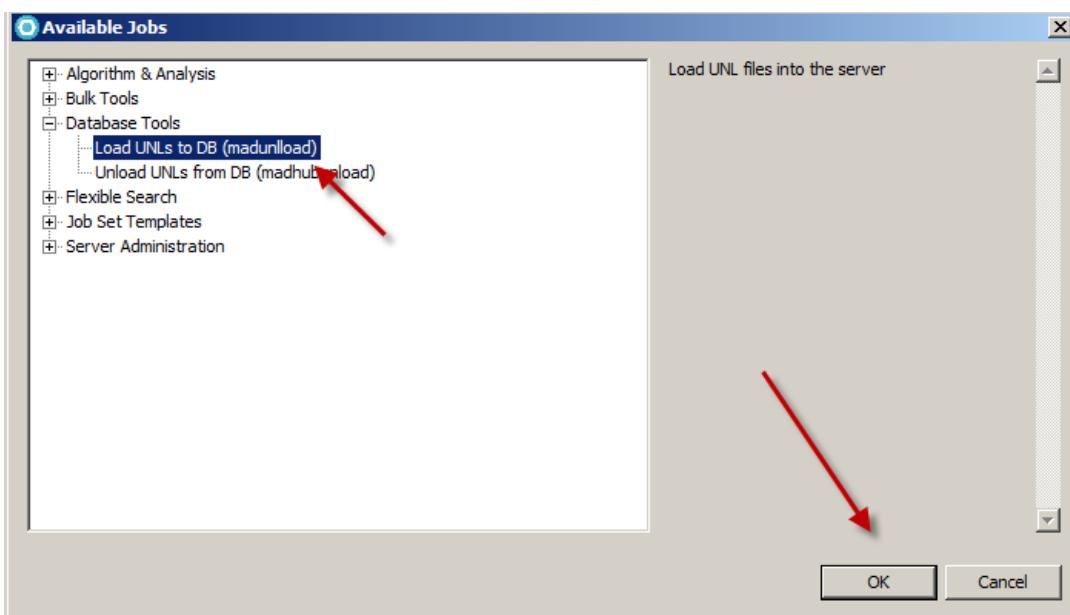


Project: MapleParty
Job set template: None
Server: WebSphere Application Server v8.5 at localhost
Work directory (optional):
 Save job set as a template
Template name: Physical Load UNLs to DB
Template description:
Next > Back Finish Cancel Save

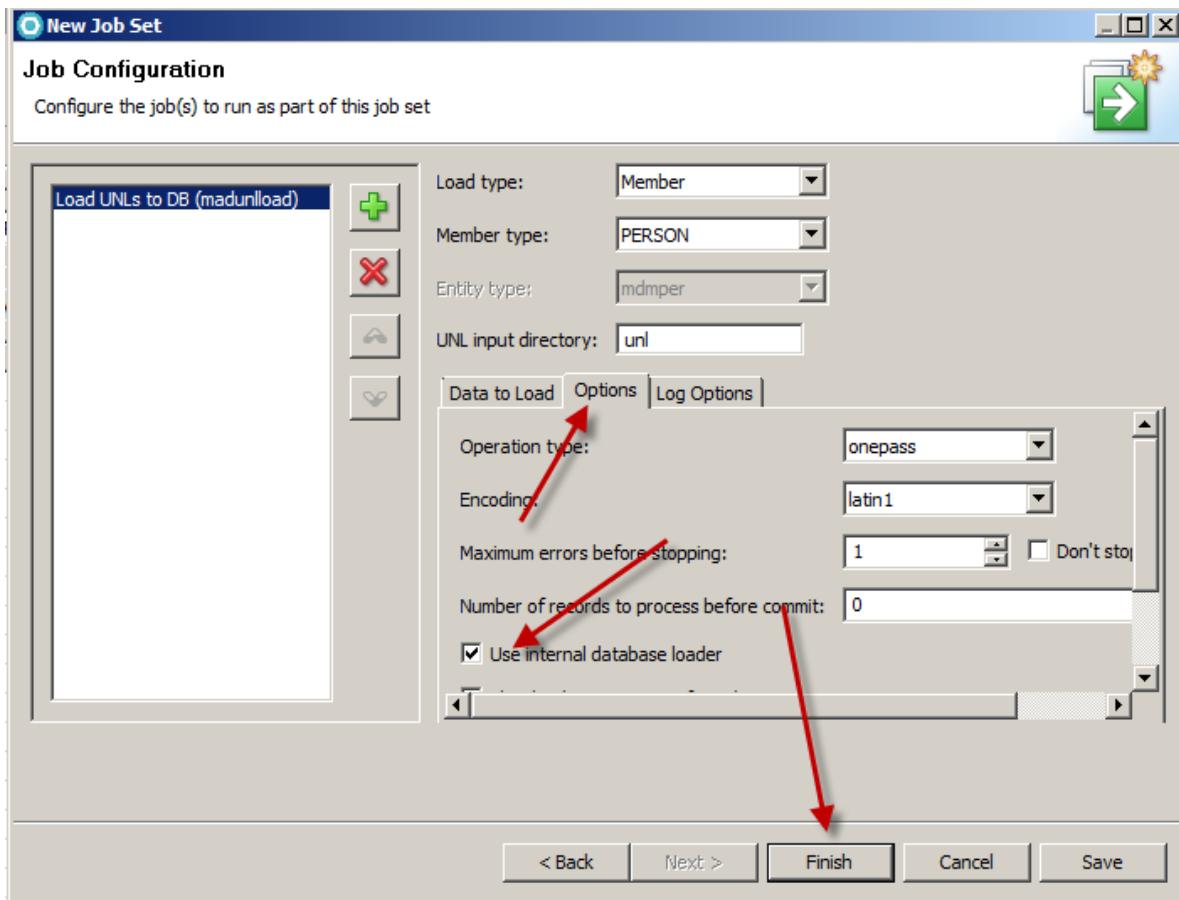
- 24. Inside the New Job Set window, click the green plus button.



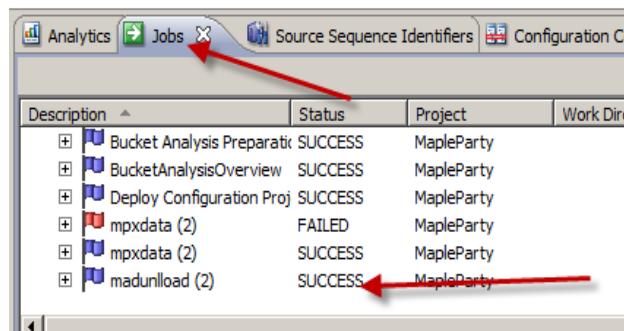
___ 25. Select **Database Tools > Load UNLs to DB (madunload)** and click the **OK** button.



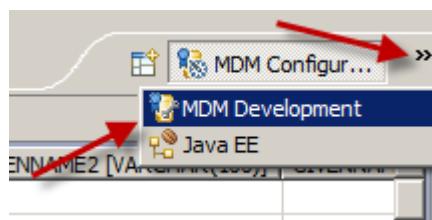
___ 26. Under the **Options** tab, select the **Use internal database loader** checkbox and click the **Finish** button



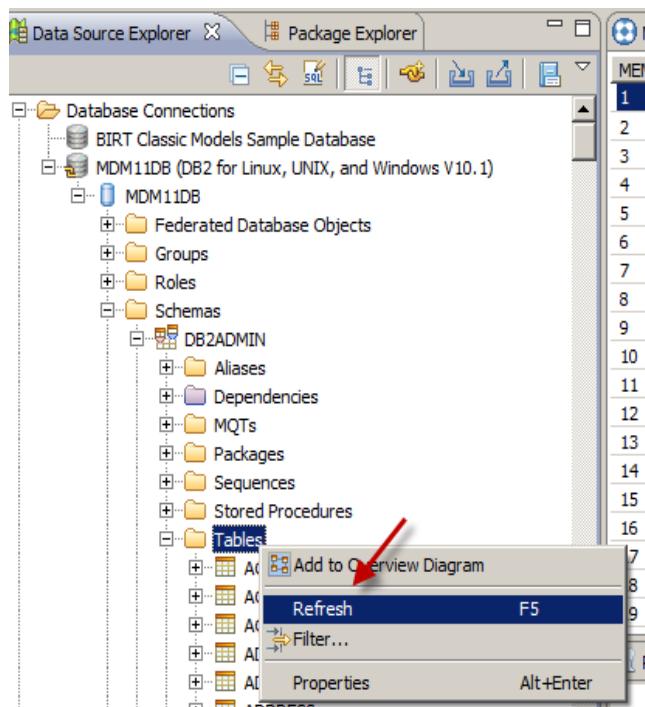
- ___ 27. This job will take a little longer than the previous one as it is loading all the records (~500000 records) into the database, you can check the status under the **Jobs** tab in RAD.



- ___ 28. Once the job is complete, we can check the Virtual database table to see if the data was populated. To see the database, switch back to the **MDM Development** perspective (top right hand corner of the RAD environment provides a shortcut)



29. Under the Data Source Explorer, navigate to the Tables folder (under **MDM11DB > MDM11DB > Schemas > DB2ADMIN**). Right click on the **Tables** folder and select **Refresh** (the configuration deployment contained new tables for the Person Member).



30. Open the **MPI_PERNAMES** table to see if our records have been populated. If they are empty, there was an error loading the records (The Console tab in the RAD environment should provide more details if there was an error).

RECNO [BIGINT]	MEMSEQNO [SMALLINT]	CALDRECHNO [BIGINT]	MAUDRECNO [BIGINT]	RECSTAT [CHAR(1)]	ATTRRECCNO [SMALLINT]	ASAIIDNO [SMALLINT]	LASTNAME [VARCHAR(120)]	GIVENNAME1 [VARCHAR(100)]
42	2	2	2	A	1	0	GAMBALE	JESSE
43	2	2	2	A	1	0	YOUNGQUIST	EMERY
44	2	2	2	A	1	0	PARMELEE	MARLIN
45	2	2	2	A	1	0	DOETSCH	ALAINA
46	2	2	2	A	1	0	SCHULTHEISS	Yael
47	2	2	2	A	1	0	DEITSCH	JAQUELINE
48	2	2	2	A	1	0	KLEINER	JORDON
49	2	2	2	A	1	0	VANHOY	BETTYANN
50	2	2	2	A	1	0	NOBSE	DALLAS
51	2	2	2	A	1	0	BULLS	BRADLEY
52	2	2	2	A	1	0	SCHLATER	JENNIFER
53	2	2	2	A	1	0	STERRY	MARCUS



Note

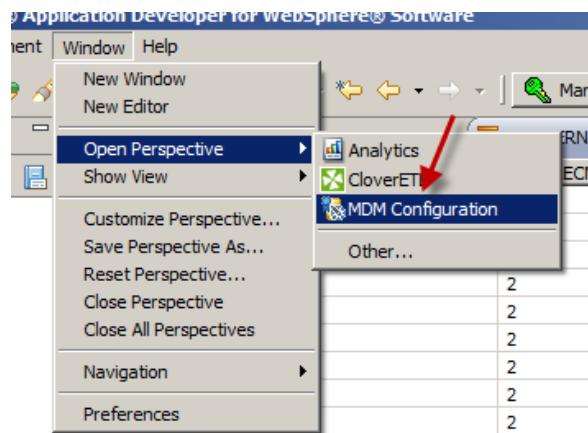
We are loading data into the Virtual MDM only to help configure the PME configuration. Ultimately this configuration and algorithms will be run against records that are persisted in the Physical MDM).

Part 3: Running bucket analytics

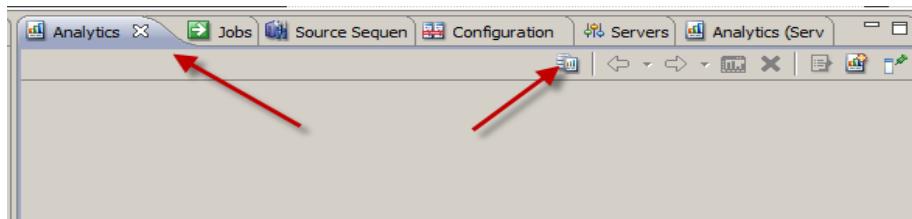
When we loaded the records into the Virtual MDM, the records would have been placed into our buckets that we defined and deployed to the Virtual MDM. We can now use this information to

analyze how well our buckets worked (e.g. did each record fall into one or more buckets, do we have buckets with only one record, do we have buckets with too many records).

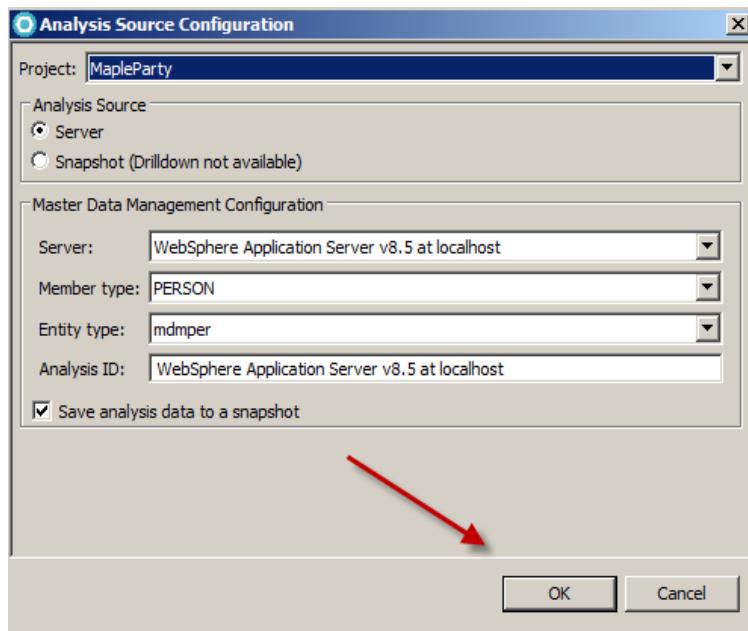
- 1. To run our bucket analytics, we first need to switch back to the MDM Configuration perspective. Switch to the MDM Configuration perspective by selecting **Window > Open Perspective > MDM Configuration** in the RAD file menu.



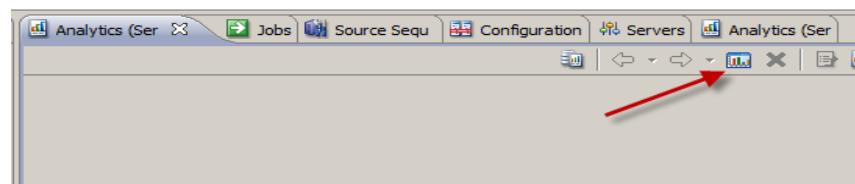
- 2. Select the **Analytics** tab at the bottom of the RAD environment. Click the **Set the data source to use for the analysis view** button.



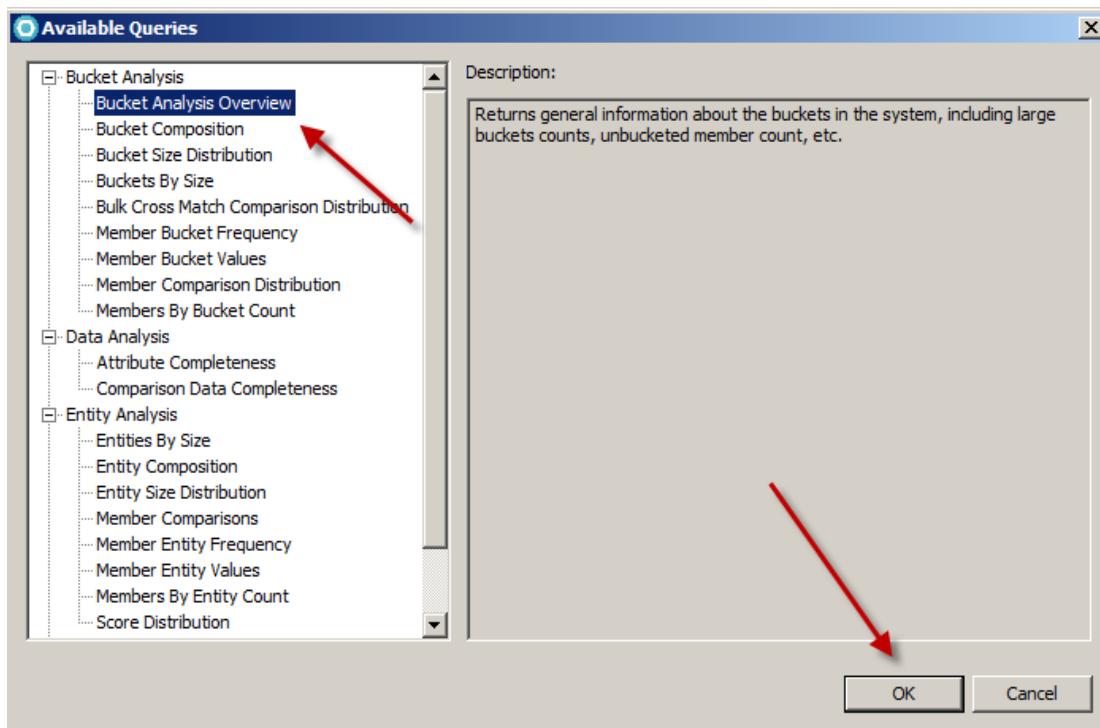
- 3. Accept the default values for the source configuration and click the **OK** button.



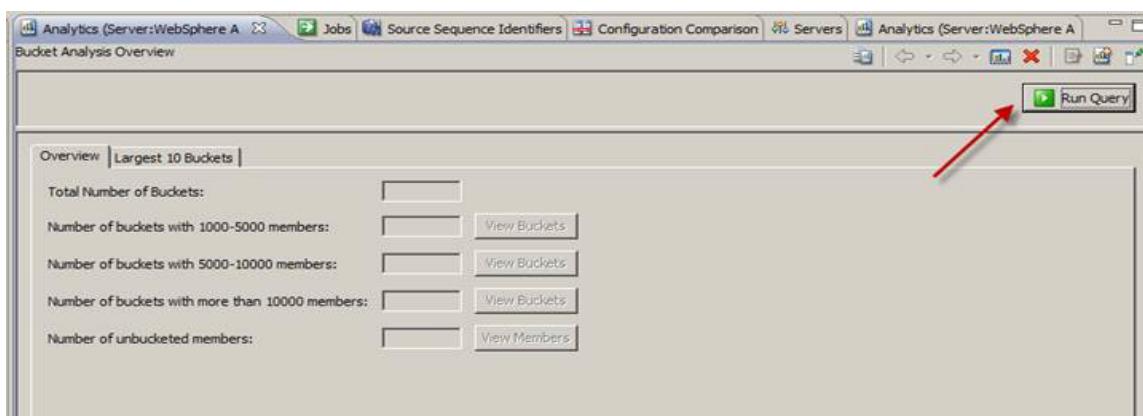
- 4. Now that the data source is configured, we can run some analysis on our buckets. Click the **Add a new query to the view for execution** button.



- 5. Select **Bucket Analysis > Bucket Analysis Overview** and click the **OK** button



- 6. This report will show us the number bucket and largest bucket details. Click the **Run Query** button.



7. In this report, you will see that we have a number of buckets that are between 1000-5000 members as well as 2 buckets that have over 10000 buckets (this is a concern as this will impact performance).

Overview Largest 10 Buckets	
Total Number of Buckets:	3266283
Number of buckets with 1000-5000 members:	37
Number of buckets with 5000-10000 members:	0
Number of buckets with more than 10000 members:	2

8. Click the **Largest 10 Buckets** tab to see the details of the largest buckets. Here we can see that one of the buckets created by Bucket Role 8 contains 23956 members and another bucket created by Bucket Role 4 contains 13534 records. The Bucket Values (99999 and 999999) also look as though they may be anonymous values. To see the details of the bucket, select the first row and click on the **View Bucket** button.

Number of Members	Bucket Hash	Bucket Role	Bucket Value	
23956	-9147867811652094305	8	99999	View Bucket
13534	5117546426236787730	4	999999	View Algorithm
1901	-8434208790018000537	8	MM999	
1901	-8544913409080044245	8	M9999	
1759	-8347012817387628441	8	CC999	
1759	-8918237386613954259	8	C9999	
1750	-8823632683028761051	8	LL999	

9. From this report of the bucket, we can see the members that belong to this bucket. Select the first row (member 19) and click the **View Member** button.

Query Parameters

Bucket hash: -9147867811652094305

Maximum # of results to return: 1000

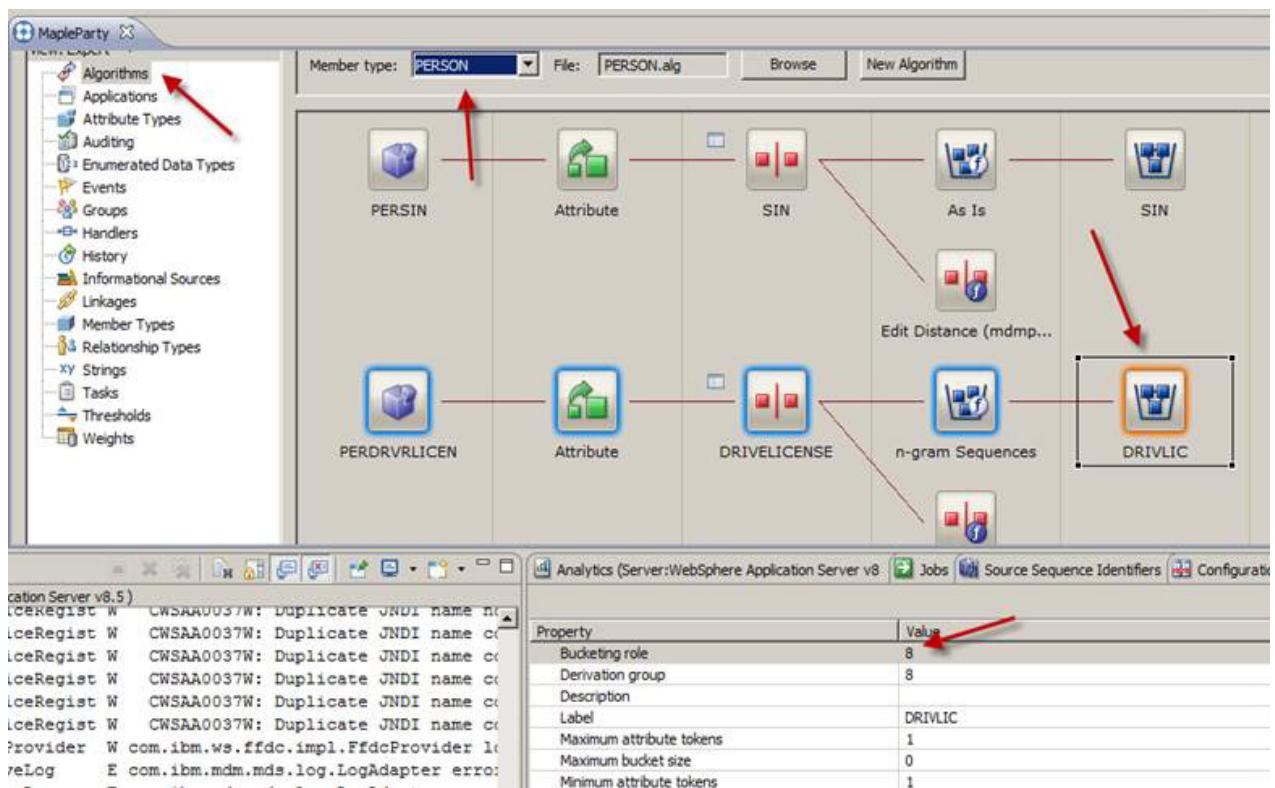
(Found 1000 members)

Member #	Bucket Role	Bucket Value	
19	8	99999	View Member
41	8	99999	View Algorithm
61298	8	99999	
199228	8	99999	
369189	8	99999	
369192	8	99999	
369196	8	99999	

10. This member belongs to 3 buckets created by Bucket Role 8 and each look as though they are anonymous values.

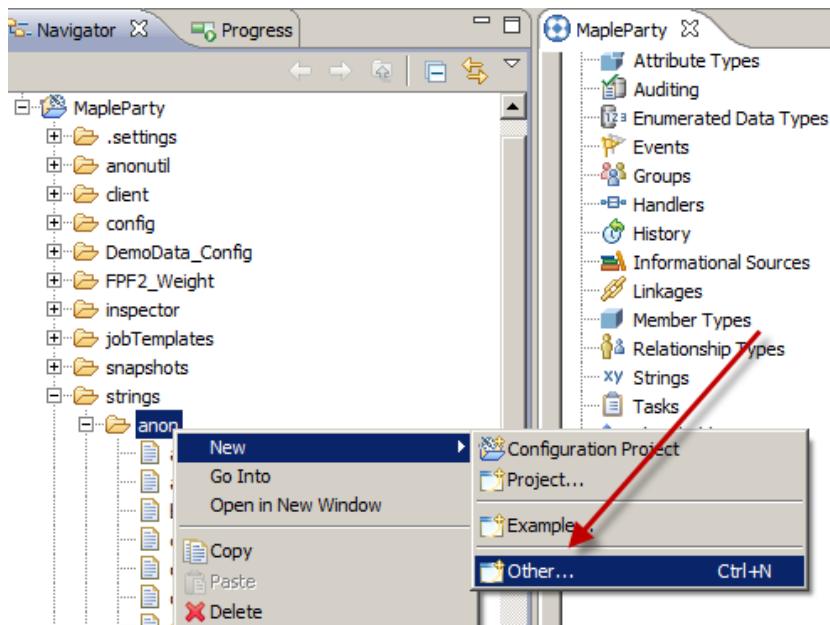
5057183508378818447	288667	4
4867331032954224885	886676	4
-8393828589103162459	TT999	8
-8338838960496521128	T9999	8
-9147867811652094305	99999	8

11. To see what which bucket has caused these values to be grouped together, go back to the **Algorithms** tab (inside the MapleParty configuration file). Find the **Bucketing role 8** by selecting the Bucketing Groups and looking at the Properties window in RAD. Also look for Bucketing role 4 as this was also a large bucket with another anonymous value.

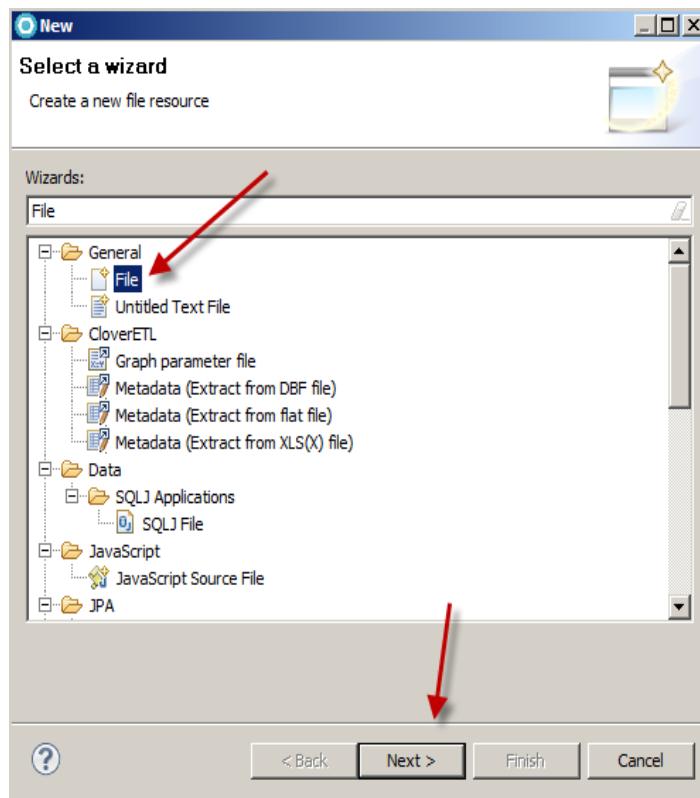


We know from this information, that many members have a sub value of 99999 as their Driver License (definitely an anonymous value)

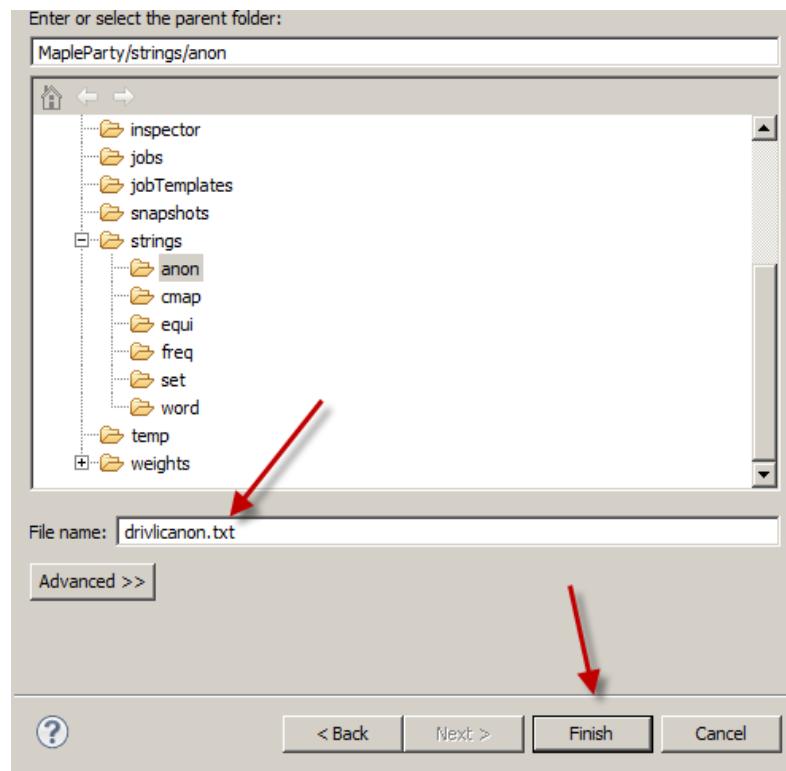
12. To stop these values from becoming a bucket, we will add some anonymous values to our standardization of the Driver License. Under the project navigator, right click on the **MapleParty/strings/anon** folder and select **New > Other...**



13. Select **General > File** and click the **OK** button.



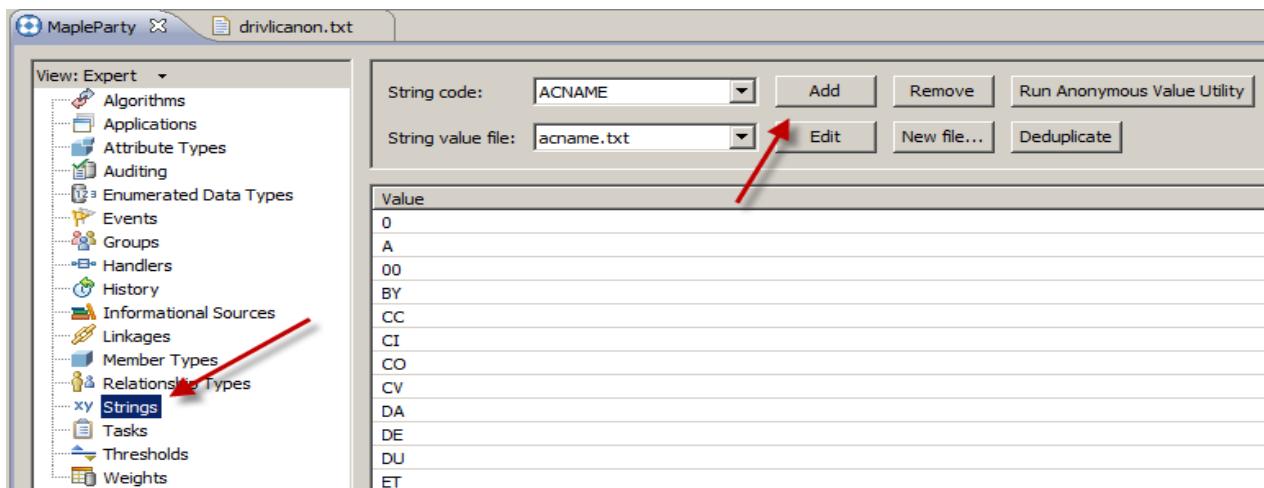
14. Enter **drivlicanon.txt** for the File name and click the **Finish** button.



- 15. After analysis the values of that are in the Sample set, you would have seen that many driver's licenses have the value xx999999999 (where x is a capital letter in the alphabet). Add the anonymous values to the file (999999999, AA999999999, BB999999999, etc).
NOTE: you'll find the same values in the **LabFiles > BucketAnalysis** folder if you just want to copy and paste.

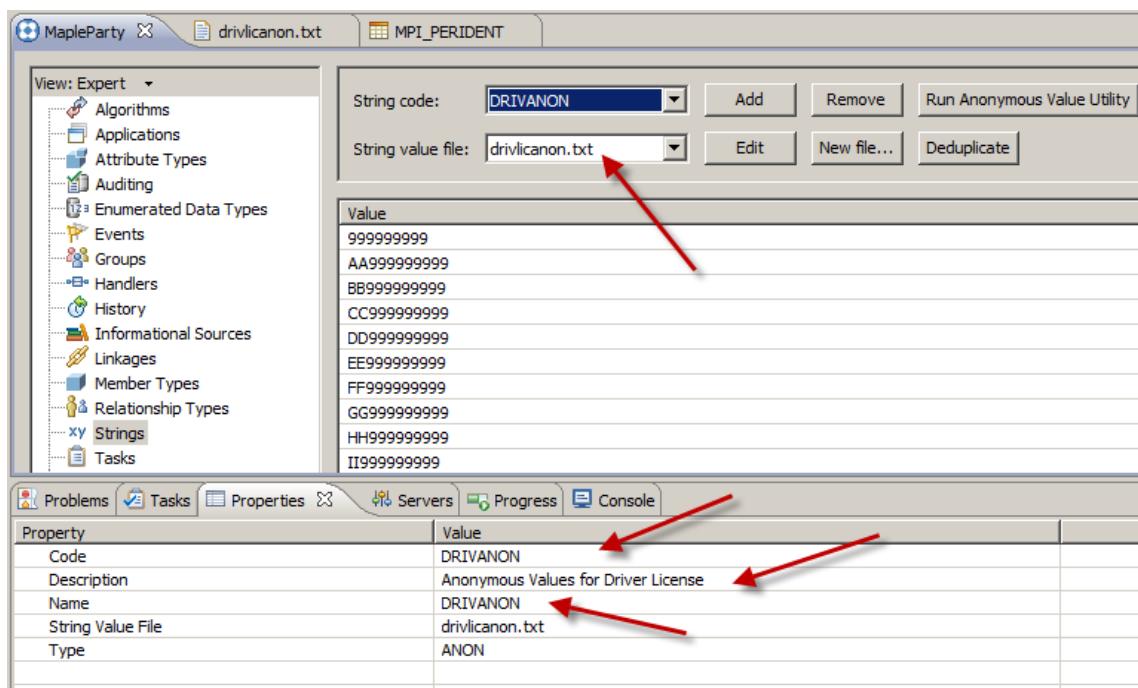
```
999999999
AA999999999
BB999999999
CC999999999
DD999999999
EE999999999
FF999999999
GG999999999
HH999999999
II999999999
JJ999999999
KK999999999
LL999999999
MM999999999
NN999999999
```

- 16. Save the file (CTRL+S)
— 17. Back under the MapleParty configuration file, select the **strings** tab and click the **Add** button to add our new anonymous values.



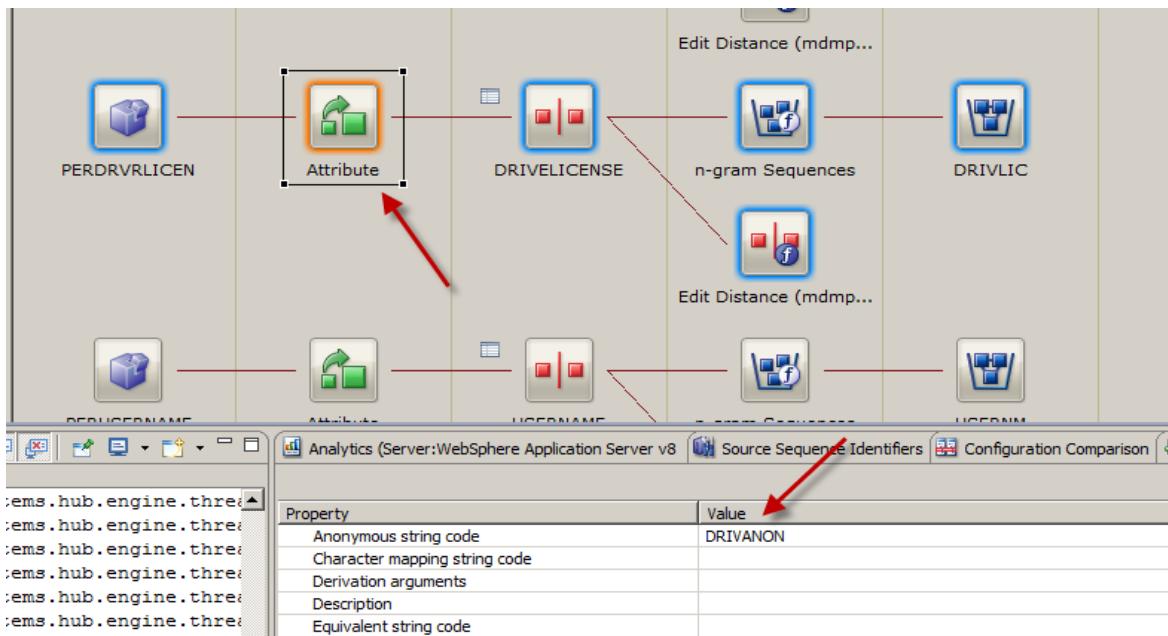
18. Enter the following values under the Properties windows:

- __ a. Code: **DRIVANON**
- __ b. Description: **Anonymous Values for Driver License**
- __ c. Name: **DRIVANON**
- __ d. String Value File: **drvlicanon.txt**
- __ e. Type: **ANON**

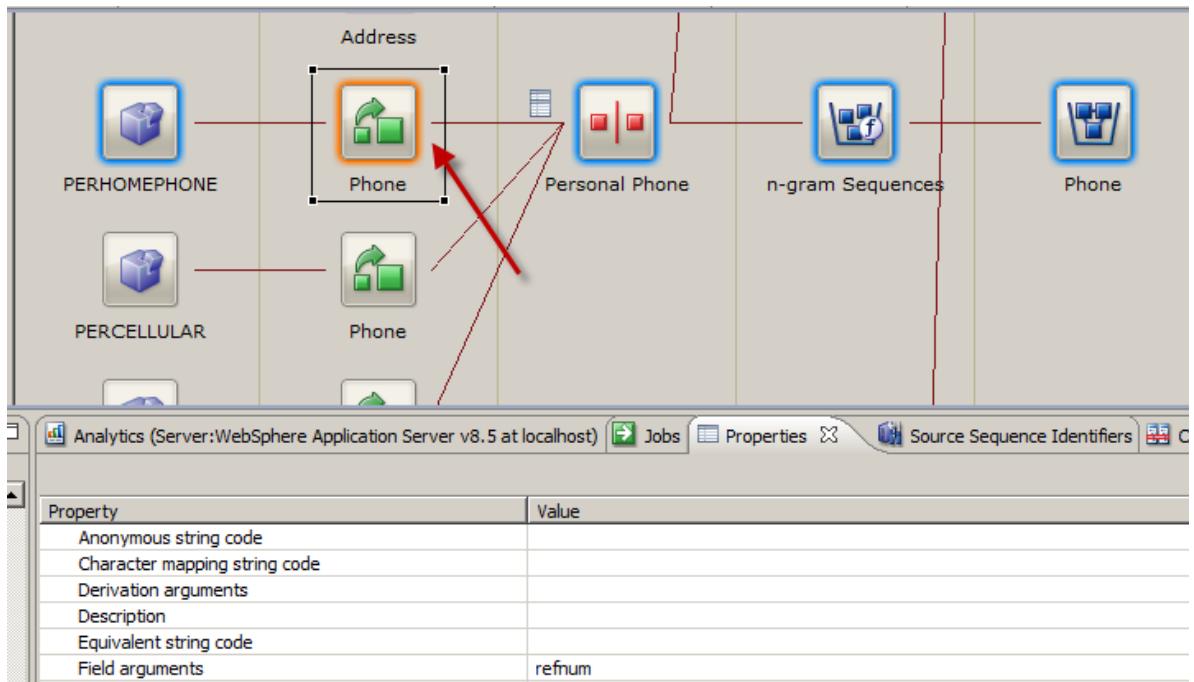


19. Still under the MapleParty configuration file, select the **Algorithms** tab and select the **Attribute** standardization function for the PERDRVVLICEN attribute in our algorithm.

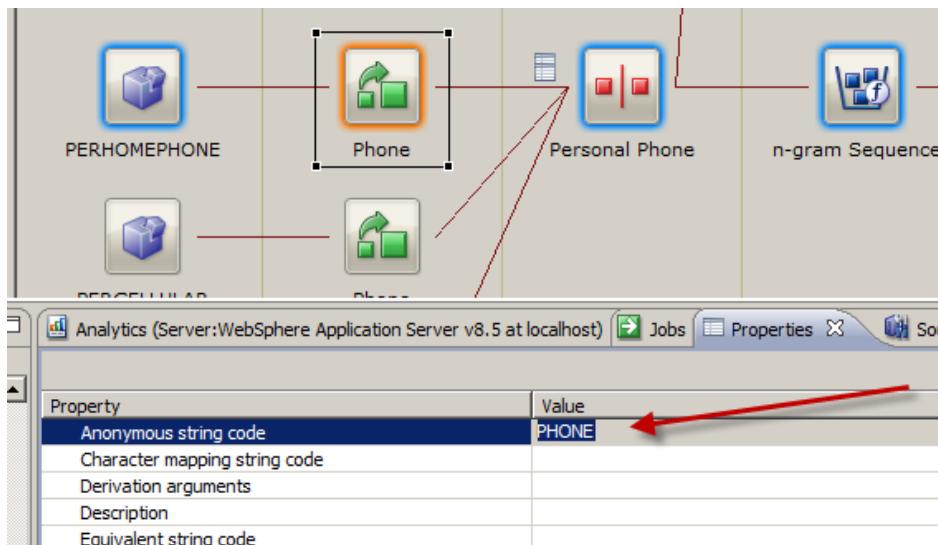
20. Under the **Anonymous string code** change the value to **DRIVANON**.



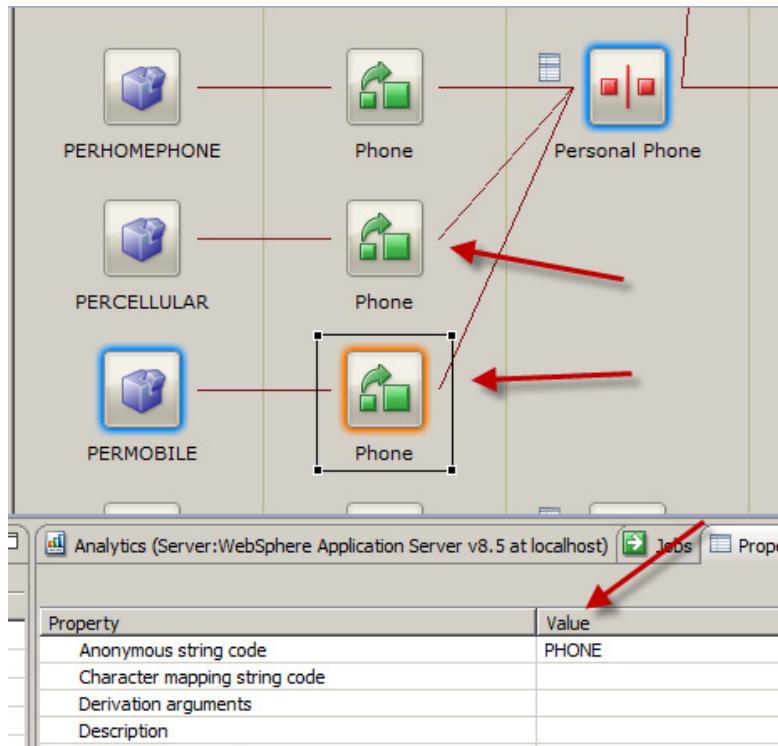
- 21. We also noticed that another bucket role (4) had large buckets of 999999. (as shown in our bucket analysis). This bucket role is the Home and Business Phone Number. Luckily we already have an anonymous file that will capture these values. Select PERHOMEPHONE Attribute standardization icon in the algorithm.



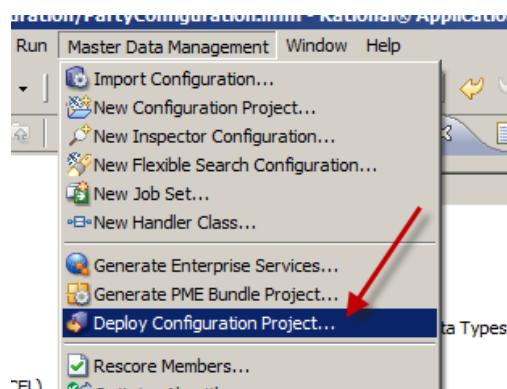
- 22. Change the Anonymous string code to **PHONE**.



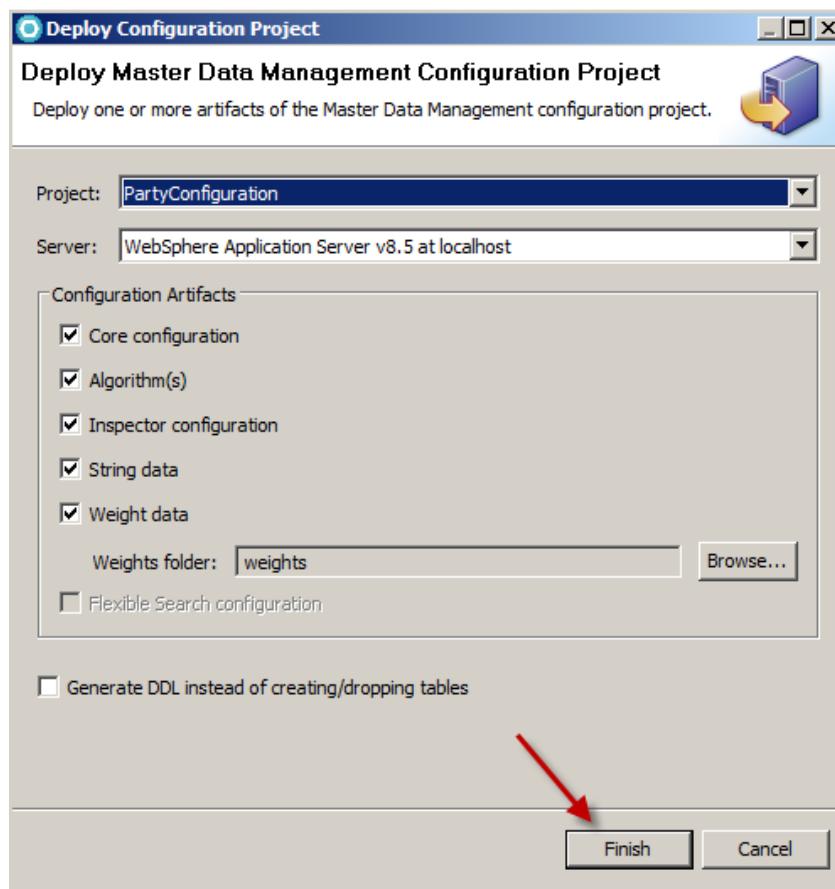
- ___ 23. Repeat the previous steps to add **PHONE** as the Anonymous string code to **PERCELLULAR** and **PERMOBILE**,



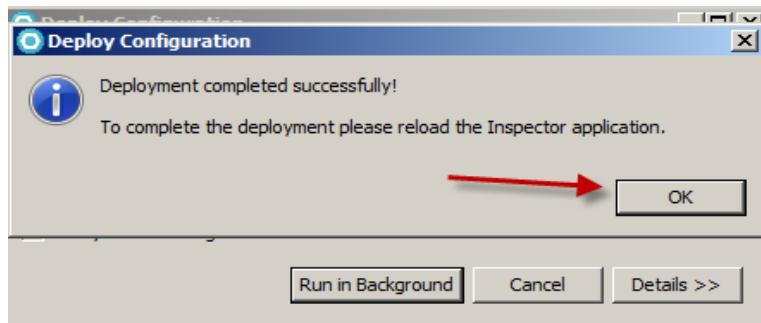
- ___ 24. Save the configuration change (CTRL+S)
- ___ 25. To see our changes we need to redeploy our configuration and rerun our job that loaded the database (so the records are bucketed again). From the RAD file menu, select **Master Data Management > Deploy Configuration**.



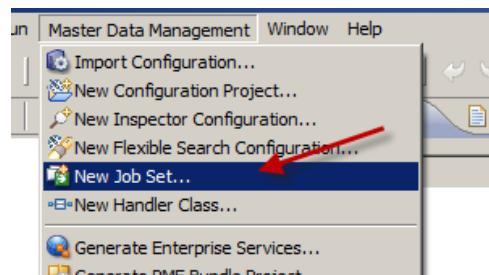
— 26. Accept the defaults and click the **Finish** button.



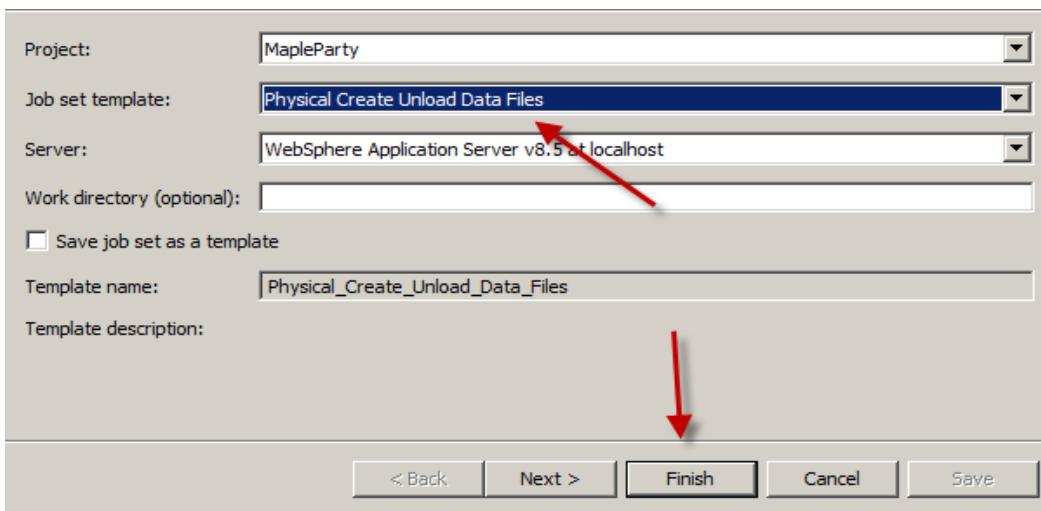
— 27. Once the deployment is complete, click the **OK** button.



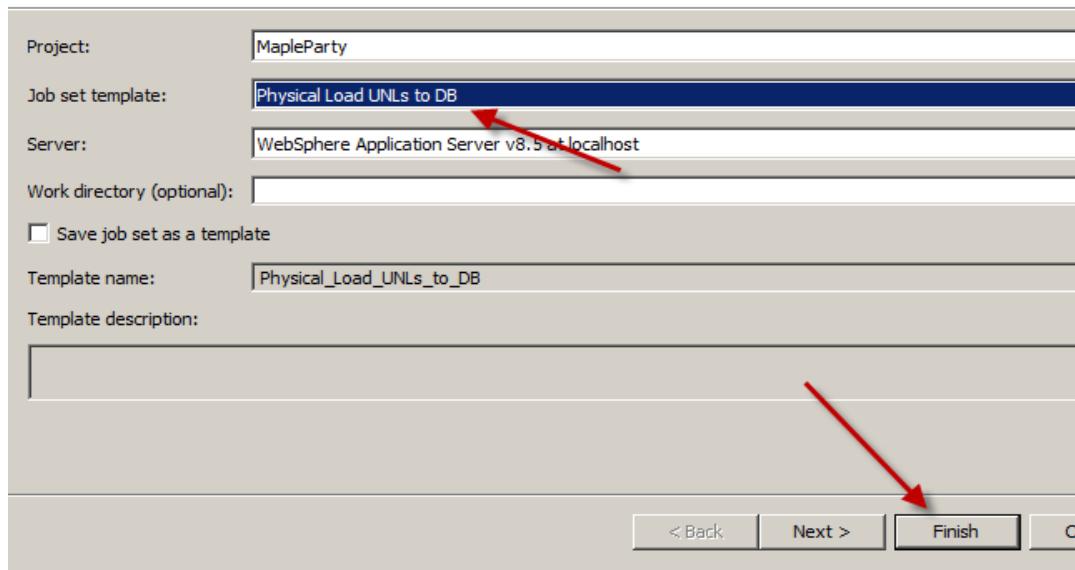
- 28. Next we need to redeploy our members. From the RAD file menu select **Master Data Management > Run Job ...**



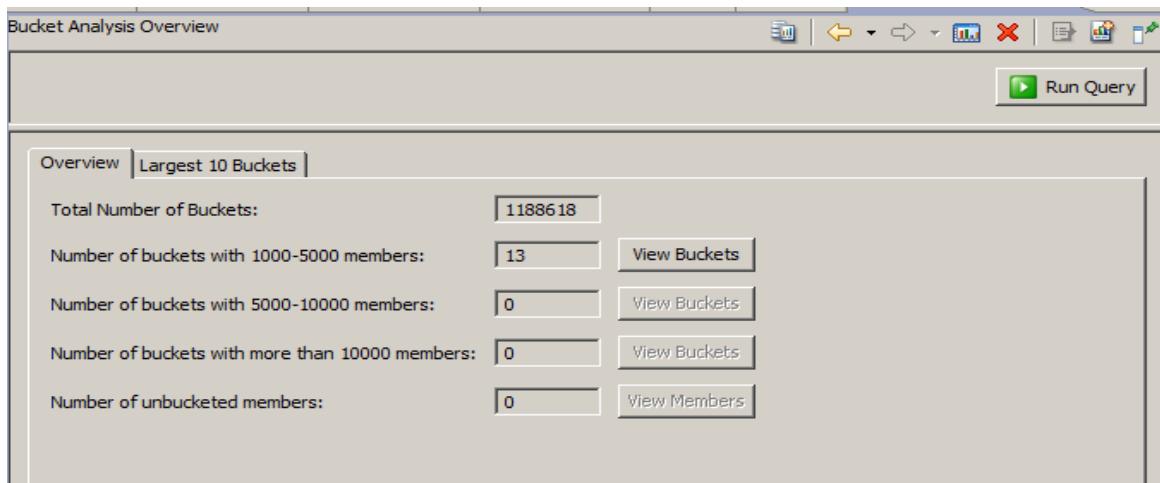
- 29. Select **Physical Create Unload Data Files** for the Job set template and click the **Finish** button.



- 30. Once the job is complete (you can check the jobs window). Run a Job set again but this time select **Physical Load UNLs to DB** for the Job set template and click the **Finish** button.



31. Once the 2nd Job is complete (check the Jobs window), open the **Analytics** window again and re-run the **Bucket Analysis Overview** query.



Congratulation, you have used the bucket analysis to evaluate and improve our algorithm.

End of exercise

Exercise 9c. Generating Weights

What this exercise is about

This exercise covers examining the out of the box weights and generating new weights for our new algorithm.

What you should be able to do

At the end of this exercise, you should be able to:

- Examining the current weight files for the out of the box algorithms
- Generating new weights for our new customized algorithm
- Deploying a new PME configuration for the Physical Module

Introduction

In this exercise we will use some sample data to run a bucket analysis on our algorithm. It is important to run this step during any customization of the algorithm as buckets perform an important role in creating a subset of records we will be comparing.

A bucket that contains too many records will impact performance and algorithm buckets that do not contain two records that could potentially be matches could miss duplicates.

Requirements

Each of these exercise depend on the previous exercise being complete. For this exercise you must have the data model and algorithms defined in the configuration file. The configuration must be deployed with sample data loading into the database.

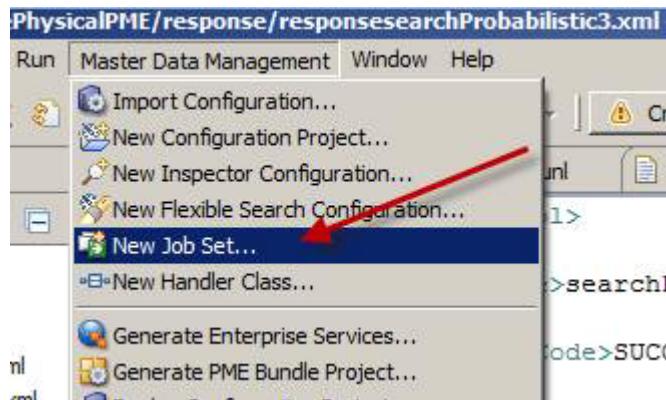
Exercise instructions

Part 1: Generating weights

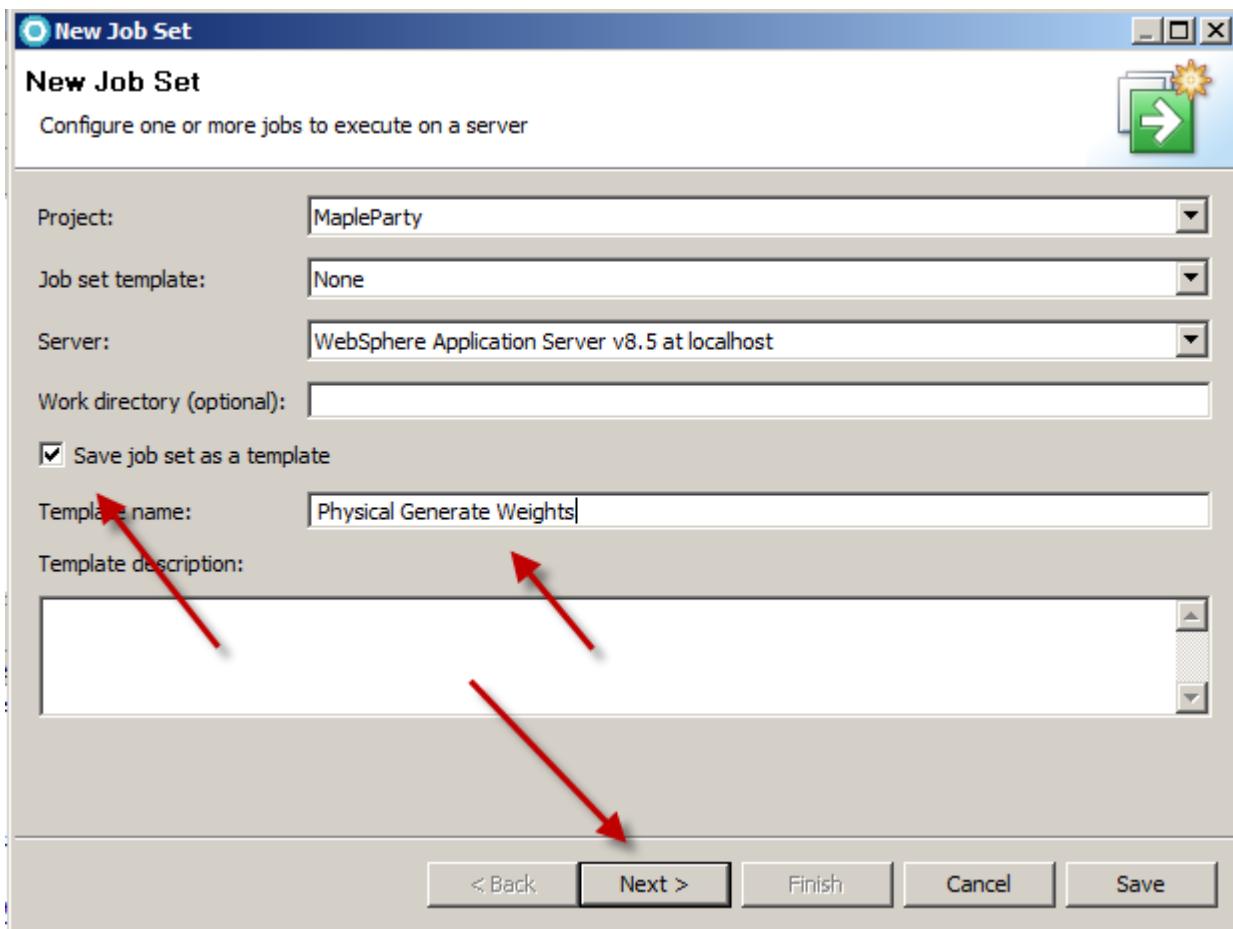
Even though our members are now being bucketed, we do not have weights for our comparison functions.

To generate weights there is a service that we can use.

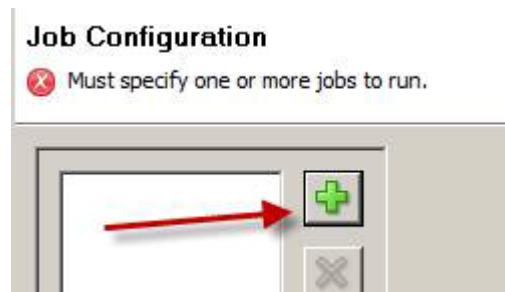
- 1. From the RAD file menu, select **Master Data Management > New Job Set...**



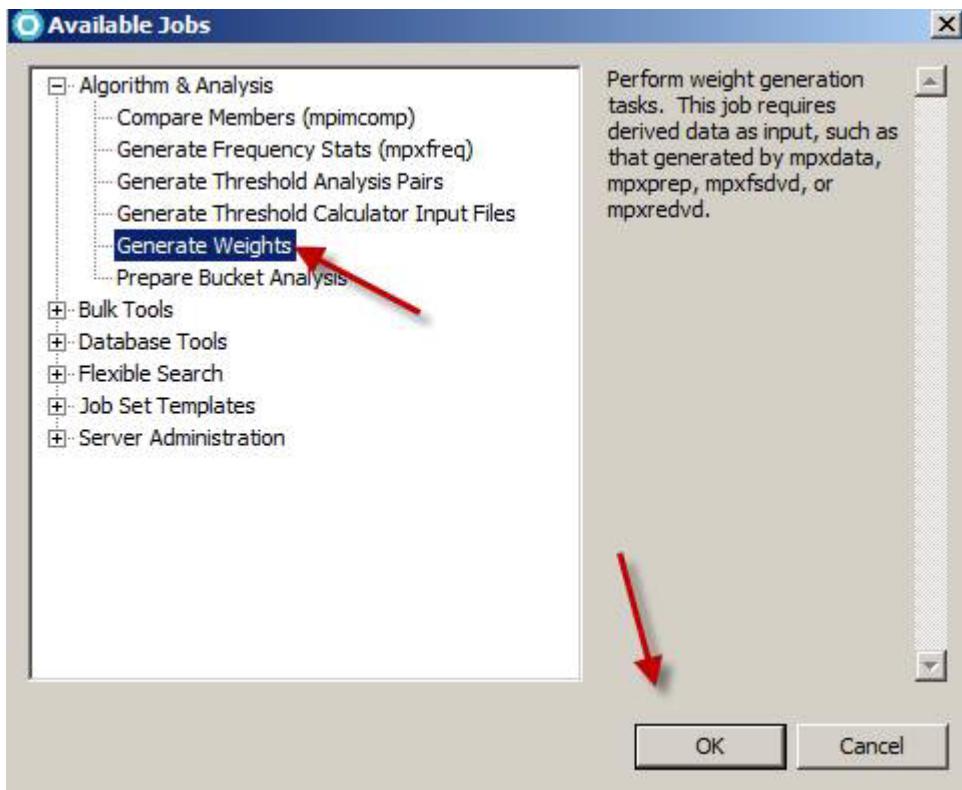
- 2. Select the **Save job set as a template** checkbox and enter **Physical Generate Weights** as the Template name. Click the **Next >** button when complete.



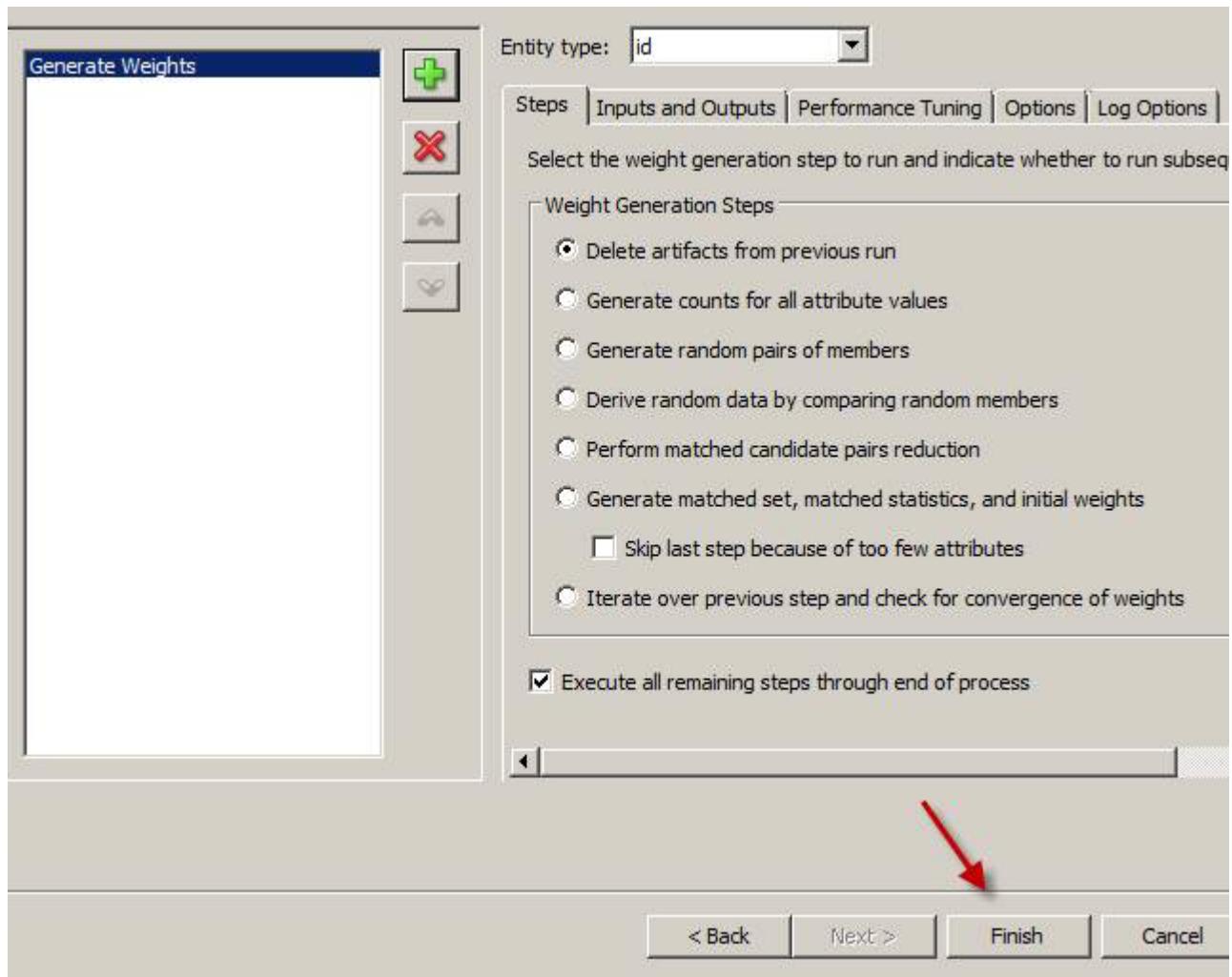
3. Click the green plus (+) button to add a new job.



4. Select **Algorithms & Analysis > Generate Weights** and click the **OK** button.



- ___ 5. Click the **Finish** button.

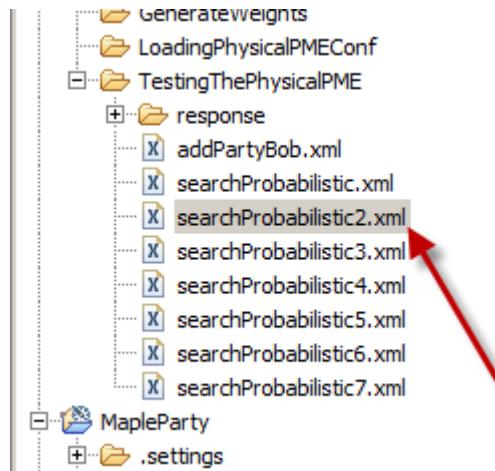


This job can take up to 20 min to complete (time to take a break), you can check the status under the Jobs window.

Part 2: Testing the previous weights

Before we replace the default PME weights (and to give the weight generation some time to finish), we can run some XMLs and see how the weight score is calculated for us.

- 1. Switch back to the **MDM Development** perspective (Window > Open Perspective > Other ... > MDM Development)
- 2. Under the Package explorer, let's take a look at the results of one of our searches and compare the weights to the default weights we imported. Under the **LabFiles > TestingThePhysicalPME** folder, open the **searchProbabilistic2.xml** file.



- 3. Look through the details of the search. In this search, we are sending in the BirthDate and SSN (Identification type 1). Remember that the Party we added had a BirthDate of 1979-03-10 and an Identification of 4363438 (this is found under the addPartyBob.xml). In this search the Month, Day and Identification matches Bob's record, but the year does not match.

```

</RequestControl>
<TCRMTx>
    <TCRMTxType>searchPersonProbabilistic</TCRMTxType>
    <TCRMTxObject>ProbabilisticPersonSearchBObj</TCRMTxObject>
    <TCRMObject>
        <ProbabilisticPersonSearchBObj>
            <MinScore>0</MinScore>
            <TCRMPersonBObj>
                <BirthDate>1990-03-10</BirthDate>
            </TCRMPersonBObj>
            <TCRMPartyIdentificationBObj>
                <IdentificationType>1</IdentificationType>
                <IdentificationNumber>436363438</IdentificationNumber>
            </TCRMPartyIdentificationBObj>
        </ProbabilisticPersonSearchBObj>
    </TCRMObject>
</TCRMTx>
</TCRMService>

```

- 4. Open the MapleParty configuration file and under the **Algorithms** tab locate the comparison function for the **PERBIRTHDATE**. Under the properties tab, you'll see that the comparison functions uses the Date2 function, which will compare the Year, Month and Day separately for weights (NOTE: if Month and Day match, it will use the MonthDay score instead of Month and Day separately)

The screenshot shows a data flow in IBM SPSS Modeler. At the top, there are four source nodes: PERBIRTHDATE (cube icon), Date (green arrow icon), DOB (red square icon), and YYYYMMDD (blue folder icon). Arrows connect PERBIRTHDATE to Date, Date to DOB, and DOB to YYYYMMDD. Below these are four target nodes: PERBIRTHDATE (cube icon), Date (green arrow icon), DOB (red square icon), and YYYYMMDD (blue folder icon). A red arrow points from the DOB target node to a connector in a box labeled "Date (mdmper)" (red square with a blue 'f'). The "Date (mdmper)" box also has a connector pointing to the YYYYMMDD target node. At the bottom, a properties table is displayed:

Property	Value
Comparison arguments	
Comparison function code	DATE2 ←
Comparison group	0
Comparison mode	Use for match, link and search
Comparison specification code	DOB
Description	
Enable Review Identifier Task	No
Entity type	mdmper
Equivalent string code	
Type	Date or Age
Weight table percentage cut-off	80

5. Back under the Project Explorer, open the **mpi_wgtnval.unl** file found under **MapleParty > weights > mdmper** folder.

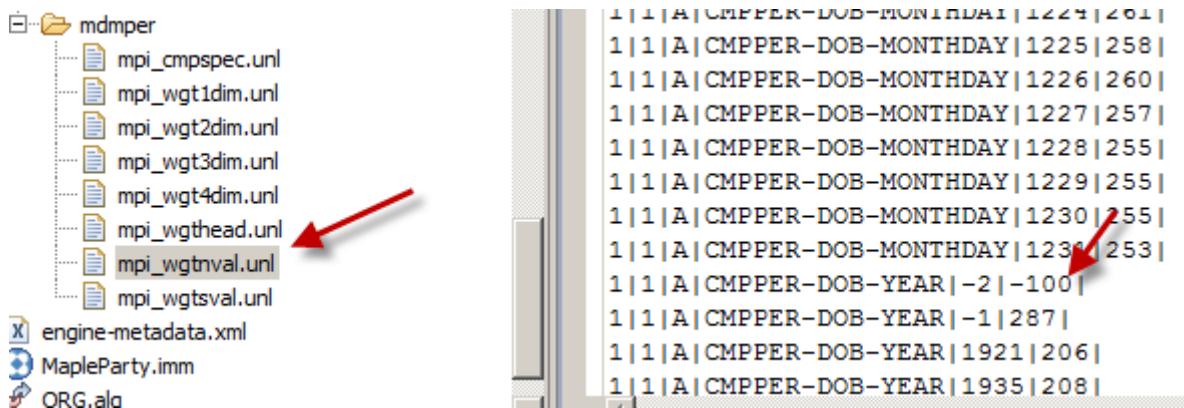
The screenshot shows the Project Explorer on the left and a text editor on the right. The Project Explorer tree shows a folder structure: temp, weights, mdmorg, and mdmper. Inside mdmper, several files are listed: mpi_cmpspec.unl, mpi_wgt1dim.unl, mpi_wgt2dim.unl, mpi_wgt3dim.unl, mpi_wgt4dim.unl, mpi_wgthead.unl, mpi_wgtnval.unl, and mpi_wtsval.unl. A red arrow points from the Project Explorer to the mpi_wgtnval.unl file in the text editor. The text editor displays the following content:

```

1|1|A|CMPPER-DOB-MONTHDAY|302|258|
1|1|A|CMPPER-DOB-MONTHDAY|303|253|
1|1|A|CMPPER-DOB-MONTHDAY|304|256|
1|1|A|CMPPER-DOB-MONTHDAY|305|255|
1|1|A|CMPPER-DOB-MONTHDAY|306|256|
1|1|A|CMPPER-DOB-MONTHDAY|307|256|
1|1|A|CMPPER-DOB-MONTHDAY|308|256|
1|1|A|CMPPER-DOB-MONTHDAY|309|257|
1|1|A|CMPPER-DOB-MONTHDAY|310|253| ←
1|1|A|CMPPER-DOB-MONTHDAY|311|258|
1|1|A|CMPPER-DOB-MONTHDAY|312|256|
1|1|A|CMPPER-DOB-MONTHDAY|313|259|

```

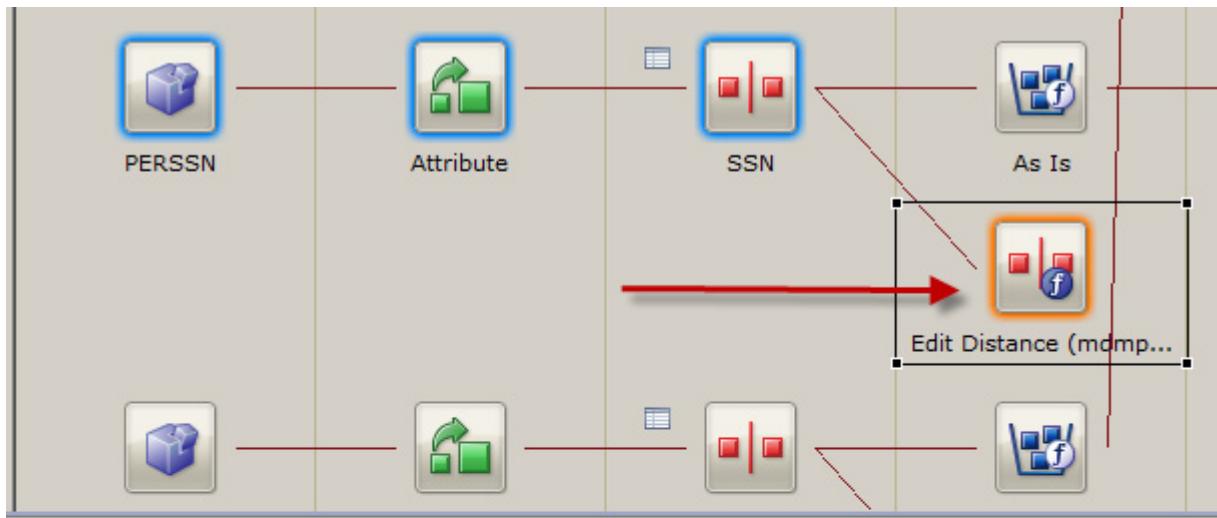
6. We know the Year does not match (1990 vs 1979). Inside the file you'll find the line **CMPPER_DOB_YEAR|-2|-100**. The -2 is a code to indicate a non-match so at this point our score is **-100**.



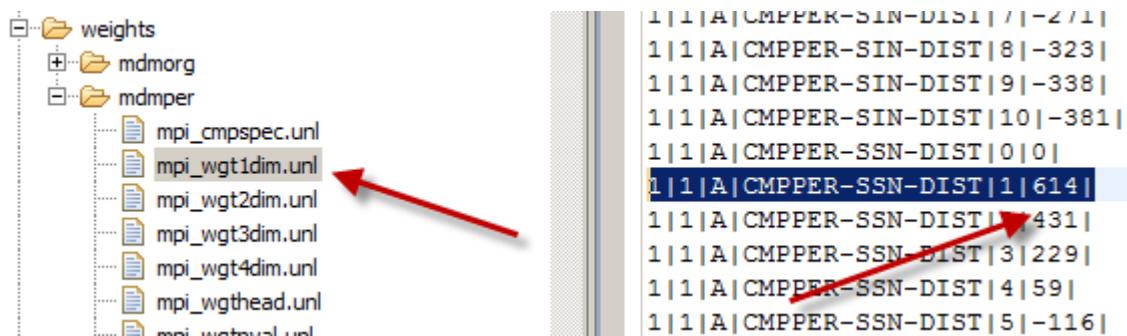
7. In the same file, you'll find the line **CMPPER-DOB-MONTHDAY|310|253**. Since our Month and Day match we receive an 253 value added to our score. Therefor our score is now **153**.

1 1 A CMPPER-DOB-MONTHDAY 305 200
1 1 A CMPPER-DOB-MONTHDAY 306 256
1 1 A CMPPER-DOB-MONTHDAY 307 256
1 1 A CMPPER-DOB-MONTHDAY 308 256
1 1 A CMPPER-DOB-MONTHDAY 309 257
1 1 A CMPPER-DOB-MONTHDAY 310 253
1 1 A CMPPER-DOB-MONTHDAY 311 258
1 1 A CMPPER-DOB-MONTHDAY 312 256
1 1 A CMPPER-DOB-MONTHDAY 313 259

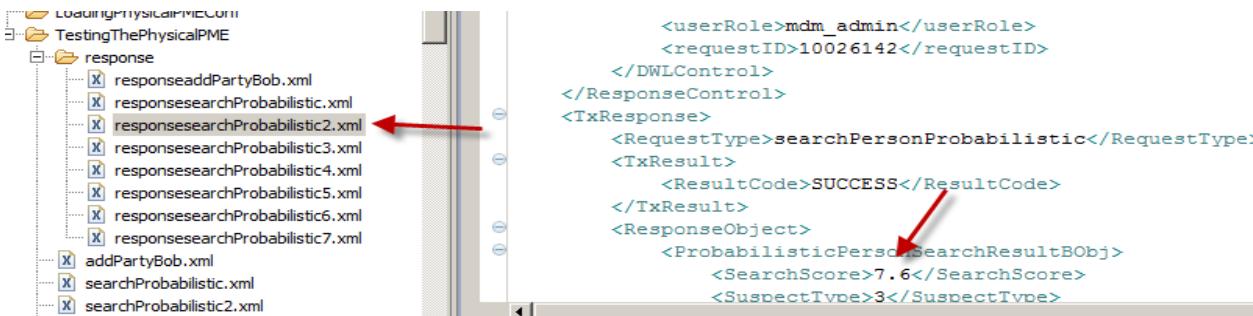
8. The other attribute we provided in the search was the Identification. If you go back to the **Algorithm** and find the **PERSSN** comparison function, you'll find that it uses an 1 dimensional Edit Distance function



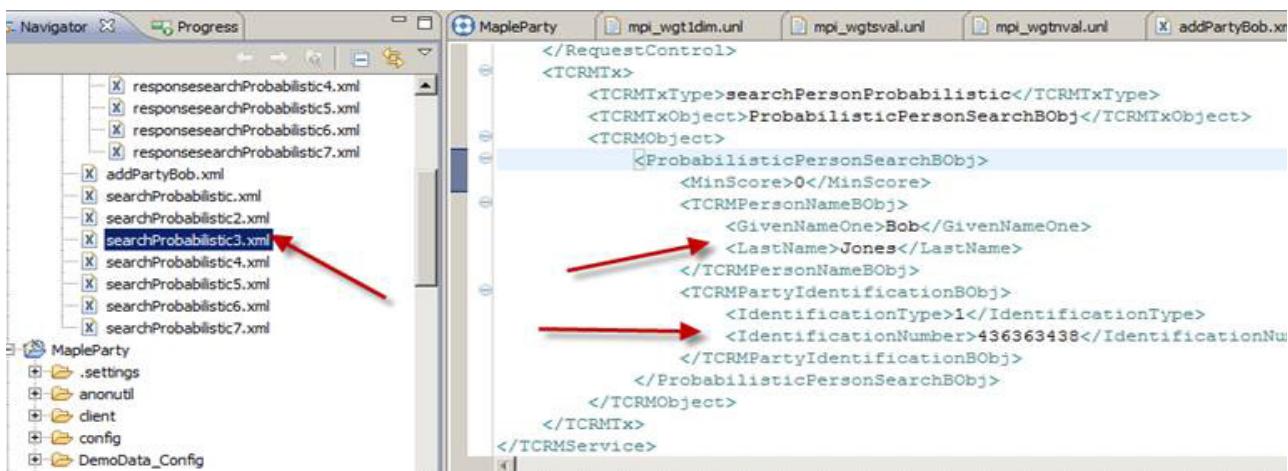
9. Under the **weights > mdmper** folder, open the **mpi_wgt1dim.unl** file. Inside this file you'll find the line **CMPPER-SSN-DIST|1|614**. Since the edit distance between the SSN is 0, the score is increased by 614 to have a value of (153+614) 767



10. If you open the **responseProbabilistic2.xml** under the **LabFiles > TestingThePhysicalPME > response** folder you'll find the match score matches our calculation of 7.6.



11. Let's take a look at another example. Open the **searchProbabilistic3.xml** file. Inside the search you find Bob Jones for the name and 436363483 for the SSN. The search matches our record in the database for the last name, however the first name is Robert from the addPartyBob.xml. Similar to the previous example, the match on the SSN will give us a starting weight of 614.

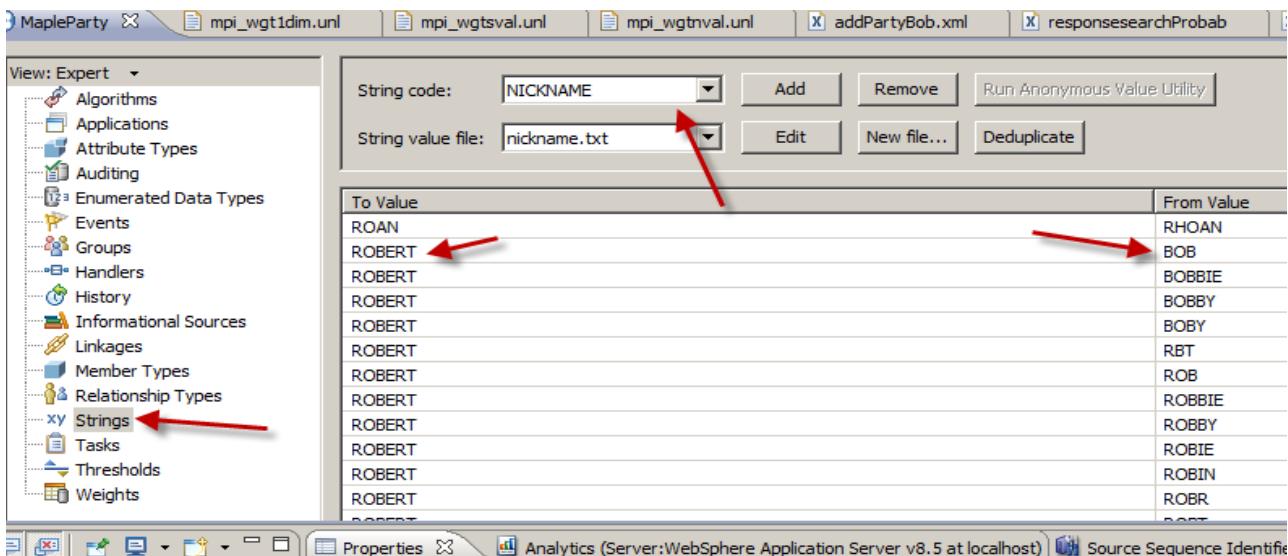


12. If you look at the Algorithm in the MapleParty configuration, for the Name Comparison Function we will find that it uses the **QXNM** function.

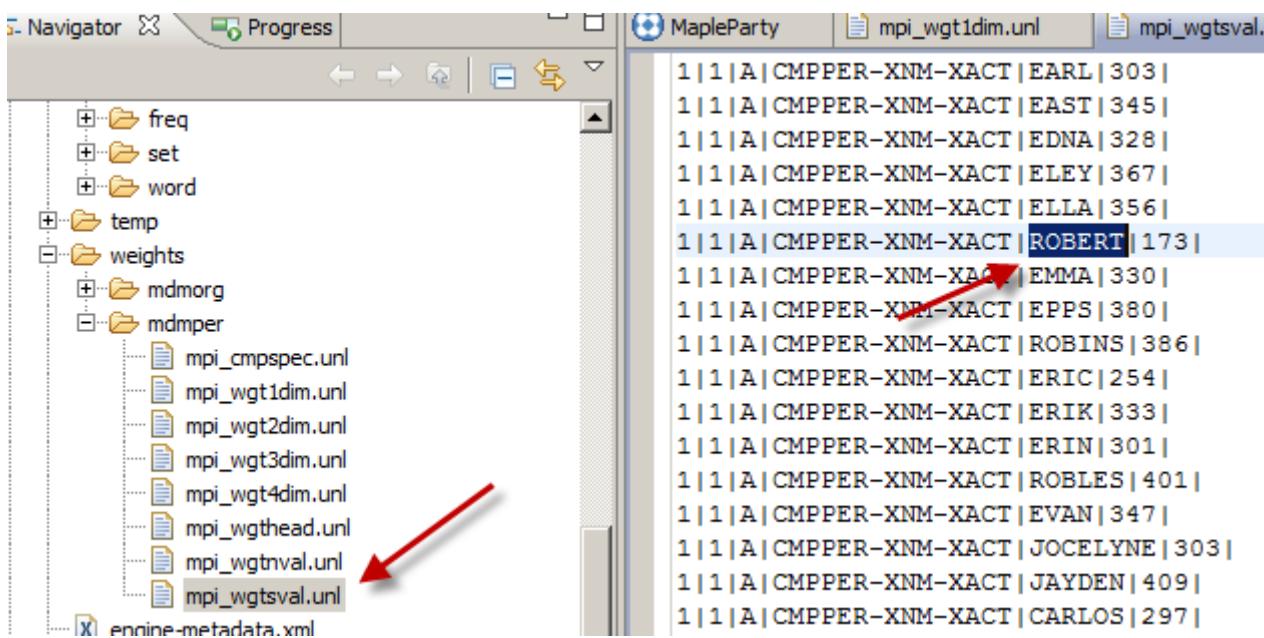
The screenshot shows a configuration interface for MDM Analytics. At the top, there is a diagram with several nodes connected by lines. The nodes include 'PERLEGALNAME' (cube icon), 'Name' (green document icon), 'Name' (red document icon), 'Equivalence' (blue folder icon), 'Equivalence & Phonetic' (blue folder icon), 'Name (mdmper)' (red document icon with a blue 'f'), and another 'Equivalence & Phonetic' node. Red arrows point from the 'Name' (red document) and 'Name (mdmper)' nodes to the 'Properties' table below.

Property	Value
Comparison arguments	METAPHONE
Comparison function code	QXNM
Comparison group	0
Comparison mode	Use for match, link and search
Comparison specification code	XNM
Description	
Enable Review Identifier Task	No
Entity type	mdmper
Equivalent string code	NICKNAME
Type	Person (comprehensive)
Weight table percentage cut-off	80

13. The QXNM function matches on each token to provide higher scores for rare values. The function also uses an equivalency file to provide a score if the token is the equivalent to another token (e.g. Bob and Robert). Inside the **MapleParty** configuration, select the **Strings** tab. Select the **NICKNAME** String code from the dropdown. In this file you will find that Robert is equivalent to Bob.



14. Open the **mpi_wgtsval.unl** file under the **MapleParty > weight > mdmper** folder. In the file you will find the row **CMPPER-XNM-XACT|ROBERT|173**. Since Bob is equivalent to Robert, we add this value to our score 173.



15. Under the same file, since this was a nickname, you'll find the line **CMPPER-XNM-PARM|__NICKNAME_ADJWGT|210**. Since this was a nickname, we subtract the nickname weight (173 – 210) -37

```
1|1|A|CMPPER-XNM-PARM|__ADDR_STREET_NORM|30|
1|1|A|CMPPER-XNM-PARM|__NORM_MCCIDX_PARTIAL|16|
1|1|A|CMPPER-XNM-PARM|__NICKNAME_ADJWGT|210|
1|1|A|CMPPER-XNM-PARM|__EDITDIST_MINWGT|50|
1|1|A|CMPPER-XNM-PARM|__NORM_ANTWGT|0|0|
```

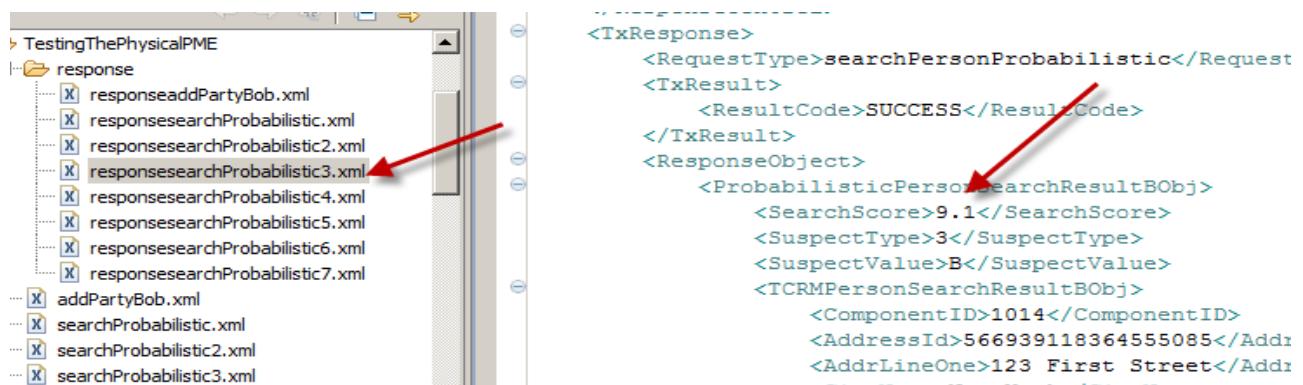
- 16. Although we now have a negative score for the first name, another parameter in the file **_NICKNAME_MINWGT**, give a minimum value of 50 for the nickname, so instead of -37 we have a score of 50 for the first name

```
1|1|A|CMPPER-XNM-PARM|__EDITDIST_FACTOR|5|
1|1|A|CMPPER-XNM-PARM|__NICKNAME_MINWGT|50|
1|1|A|CMPPER-XNM-PARM|__COMPOUND_ADJWGT|50|
1|1|A|CMPPER-XNM-PARM| EDITDIST_MAXWGT|400|
```

- 17. For the last name we match Jones exactly and therefore get the following score of 233. Therefore for the name we receive 50 for the first name, 233 for the last name and since last name matches and is in the same position we get a bonus of 20. $(233 + 50 + 20 + 614) = 917$

```
1|1|A|CMPPER-XNM-XACT|JOANN|50|
1|1|A|CMPPER-XNM-XACT|JOHNS|413|
1|1|A|CMPPER-XNM-XACT|JONES|233|
1|1|A|CMPPER-XNM-XACT|JORGE|322|
1|1|A|CMPPER-XNM-XACT|JOSEPH|268|
```

- 18. If you open the **responsesearchProbabilistic3.xml** file under the **LabFiles > TestingThePhysicalPME > response** folder, you will find that our calculation matches the score from the search.

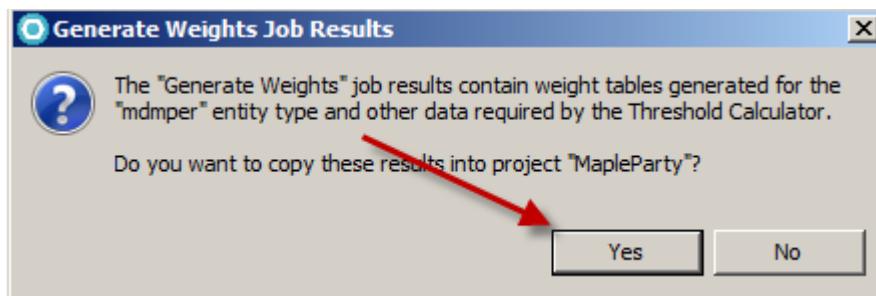


Part 3: Loading the new weights

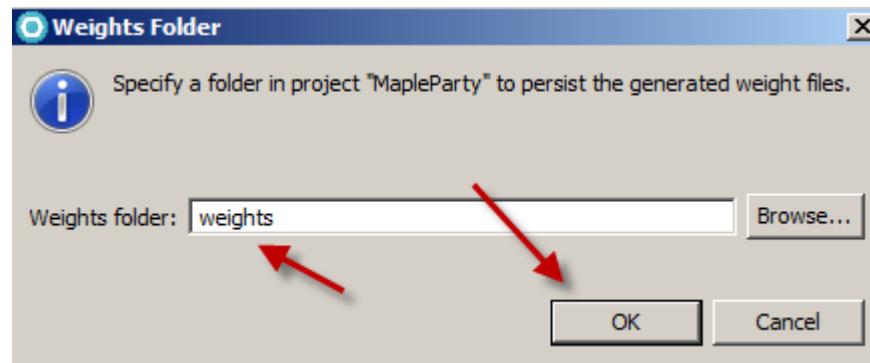
- 1. Once the Generate Weights is complete (you can check under the **Jobs** tab), right click on the Generate Weights job and select **Get job results**.

Description	Status	Project	Work Directory	Submitted By	Create Time
[+] madunupload (2)	SUCCESS	MapleParty		ibmclass	02/10/14 11:18:44 AM
[+] BucketAnalysisOverview	SUCCESS	MapleParty		ibmclass	02/10/14 12:32:57 PM
[+] MembersByBucketCount	SUCCESS	MapleParty		ibmclass	02/10/14 01:27:52 PM
[+] MemberBucketValues	SUCCESS	MapleParty		ibmclass	02/10/14 01:28:13 PM
[+] Generate Weights (2)	SUCCESS	MapleParty		ibmclass	02/10/14 02:08:02 PM
Generate Weights	02/10/14 02:08:02 PM				
Generate Weights Log	02/10/14 02:21:28 PM				

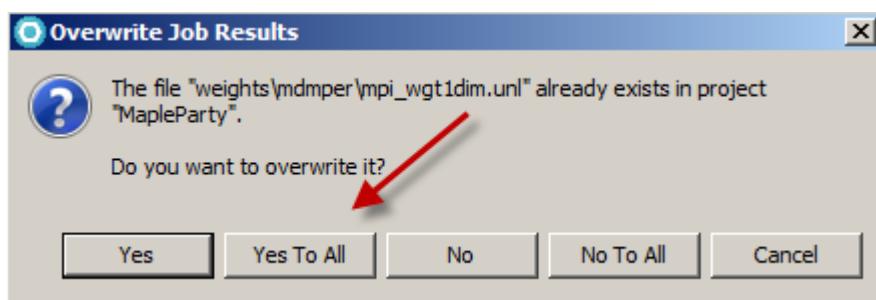
- ___ 2. Click the **Yes** button to copy the results into the MapleParty project.



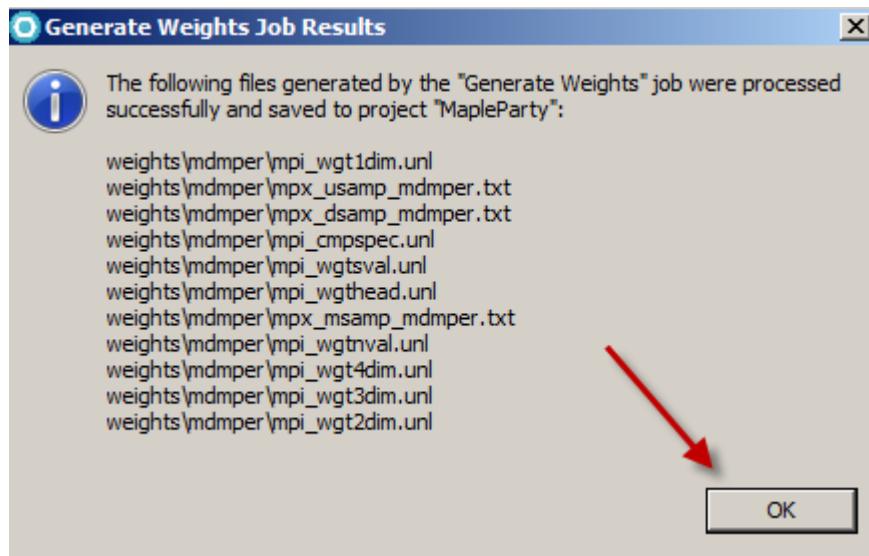
- ___ 3. Accept the default for the Weights folder (weights) and click the **OK** button.



- ___ 4. Click the **Yes To All** button to overwrite the previous weights



- ___ 5. Click the **OK** button once the weights have been copied.

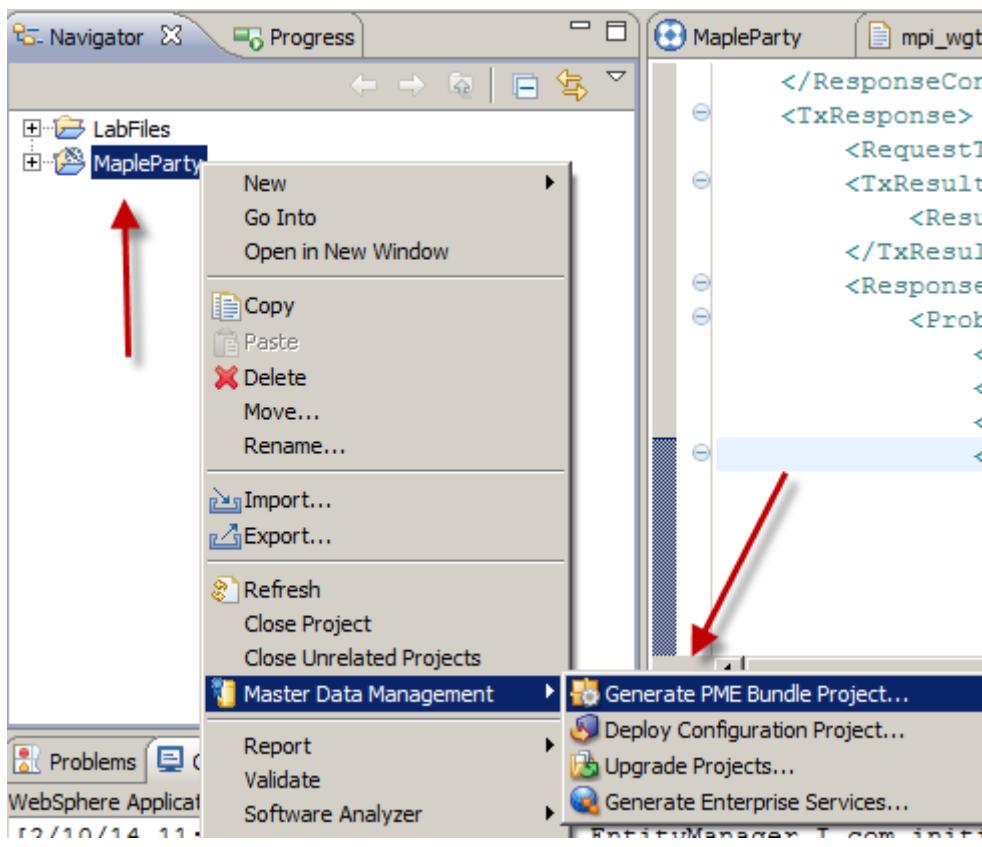


Part 4: Deploying the Physical PME Algorithm

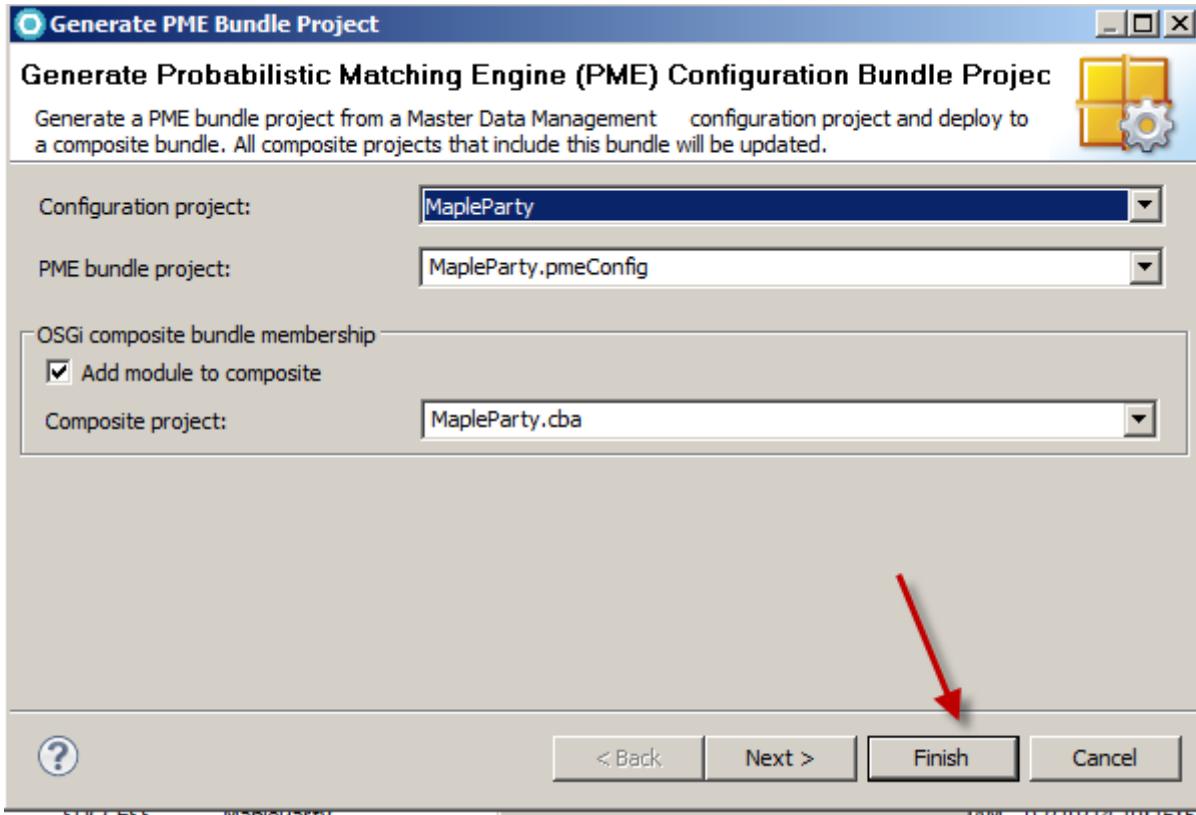
At this point during the PME configuration, we would run the Bulk Cross Load, Pair Manager (for thresholds) and Entity Analytics similar to the Virtual MDM. We will skip this process in these exercises, however the exercise exist in Appendix C, D, and E.

In this part of the exercise we will deploy the configuration project to the Physical MDM module.

- 1. Under the package explorer, Right click on the **MapleParty** project and select **Master Data Management > Generate PME Bundle Project**

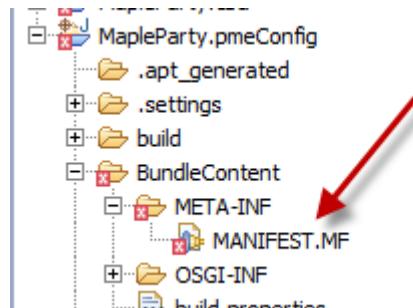


2. Keep the default values and click the **Finish** button.

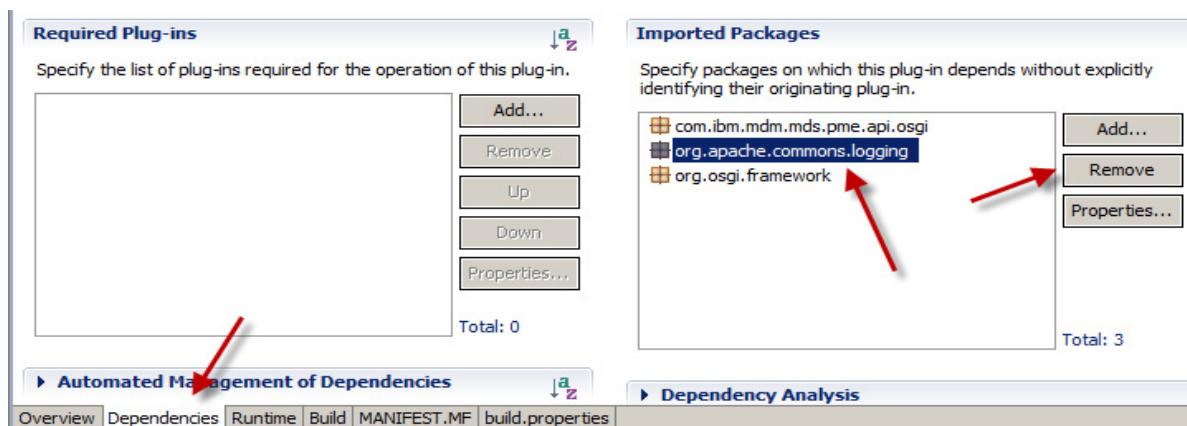


After the generation of the PME projects, you will have a few errors. In the next steps we will clean up the errors.

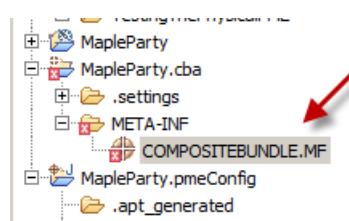
- 3. Open the **MANIFEST.MF** file found under the **MapleParty.pmeConfig** project under **BundleContent > META-INF** folder



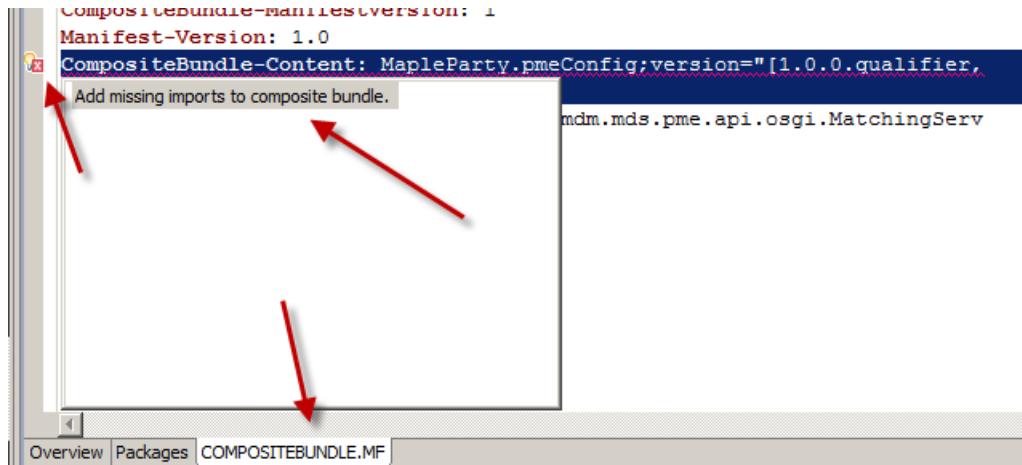
- 4. Select the **Dependencies** tab, select the **org.apache.commons.logging** under the Imported Packages and click the **Remove** button.



- 5. Next open the **COMPOSITEBUNDLE.MF** file found under the **MapleParty.cba** project.



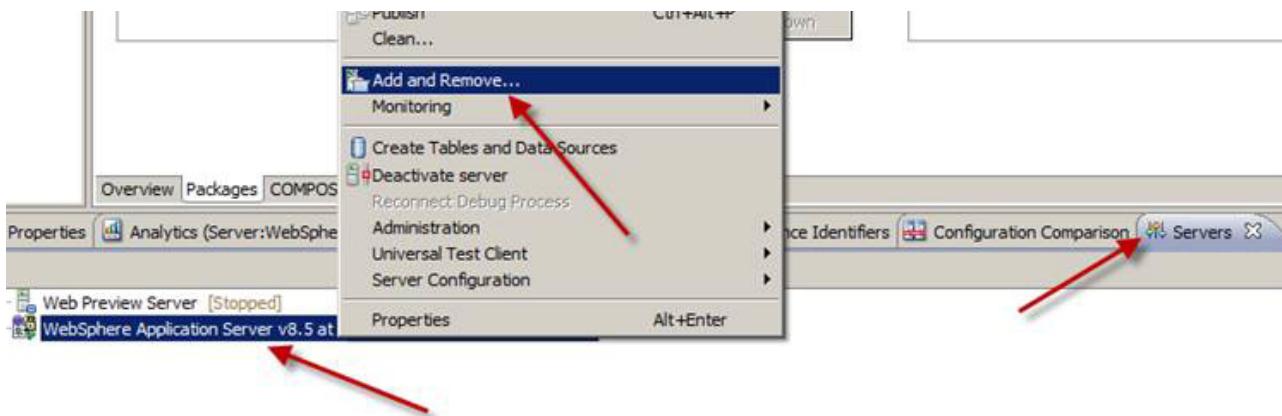
- 6. Under the **COMPOSITEBUNDLE.MF** tab, click on the red x and select **Add missing imports to composite bundle**.



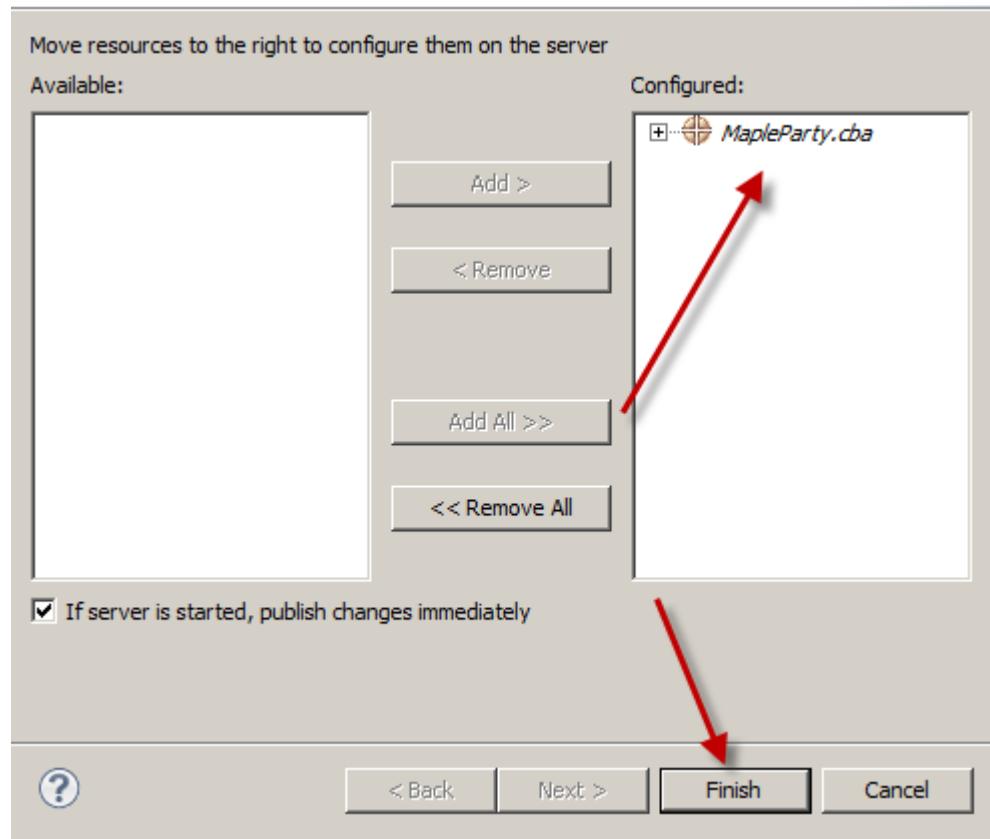
___ 7. Save the file

Now we are ready to add the new package to the Server.

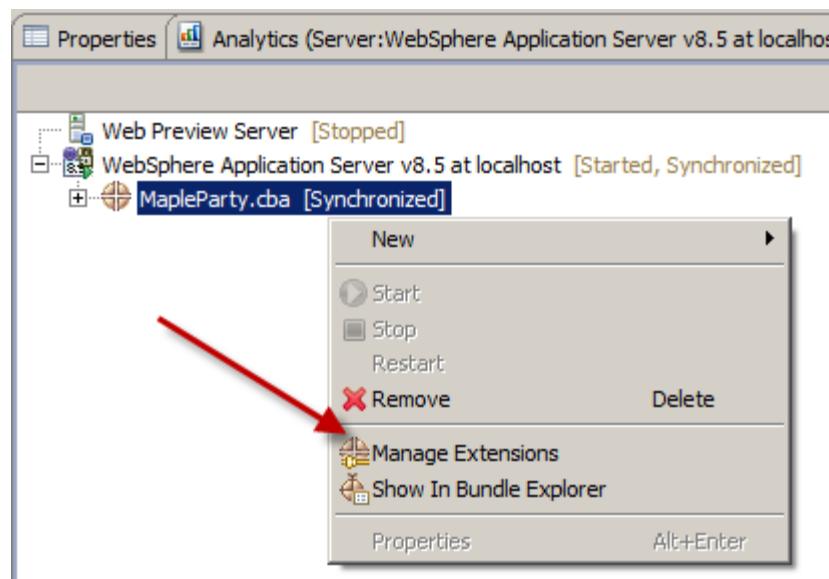
___ 8. Under the **Servers** tab, right click on the **WebSphere Application Server** and select **Add and Remove...**



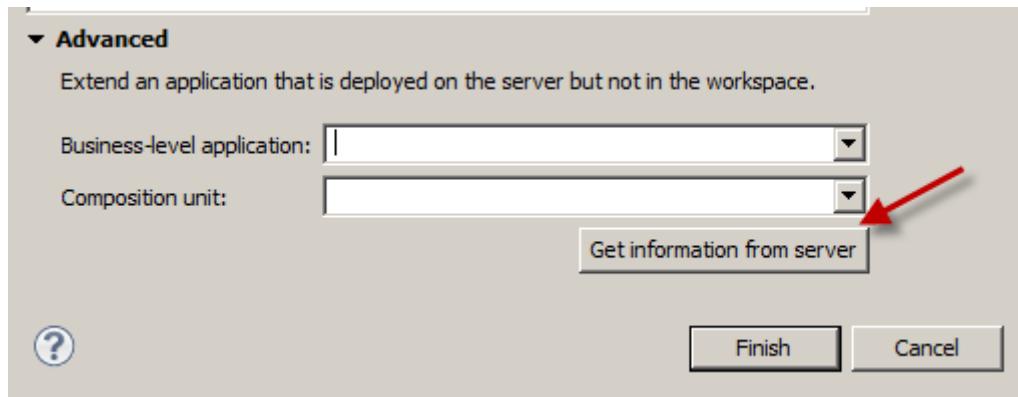
___ 9. Move the MapleParty.cba to the Configured text box, by selecting the project and clicking the **Add >** button. Once the project is moved, click the Finish button.



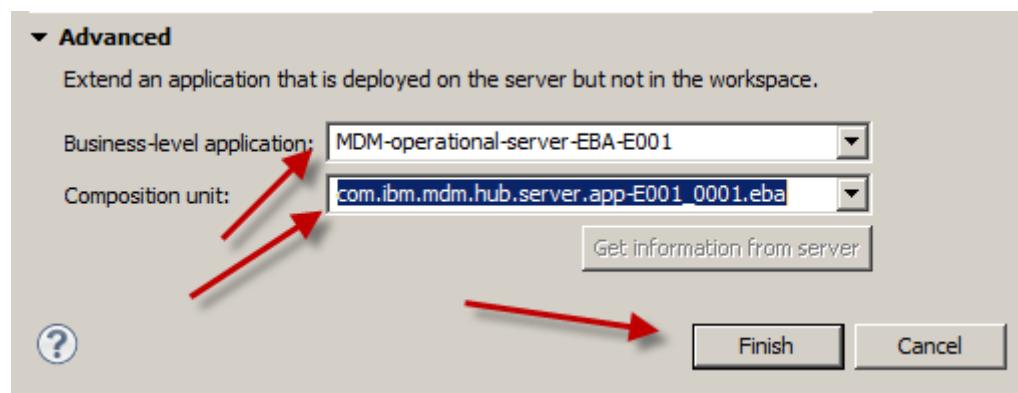
- 10. The previous step will deploy the package to the Application Server, however at this point the bundle is not yet attached to the InfoSphere MDM. To add it to the InfoSphere MDM, we must add it as an extension to the MDM. Right click on the **MapleParty.cba** found under the Servers tab and select **Manage Extensions**.



- 11. Under the **Advanced** tab, click the **Get information from server** button.

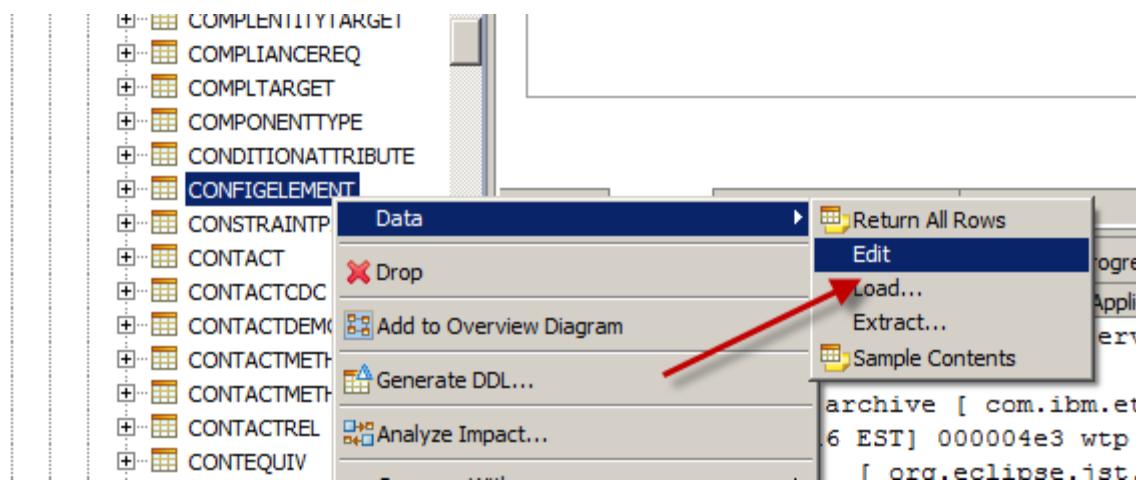


12. Select the following values from the dropdowns and click the **Finish** button:
- Business-level application: **MDM-operational-server-EBA-E001**
 - Composition unit: **com.ibm.mdm.hub.server.app-E001_0001.eba**



The deployment can take up to 5 mins, so now would be a good time to take a quick break.

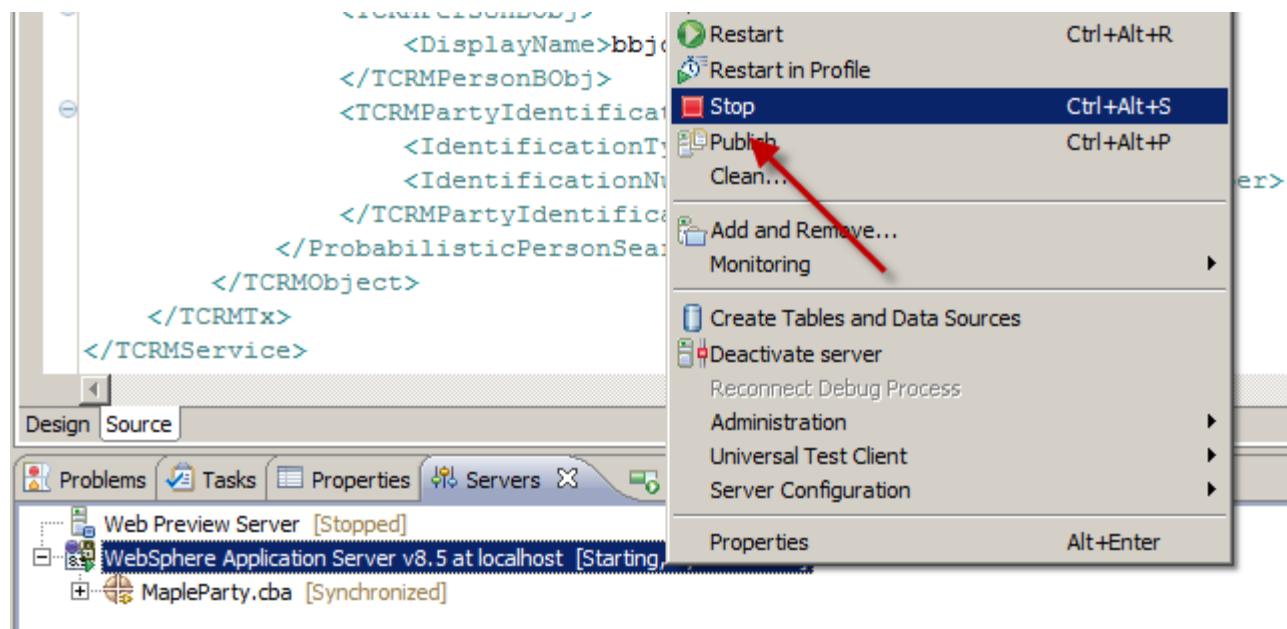
13. Once the deployment is complete, we still need to tell the InfoSphere MDM to use our new configuration. Under the Data Source Explorer, navigate to the **CONFIGELEMENT** table, right click on the table and select **Data > Edit**.



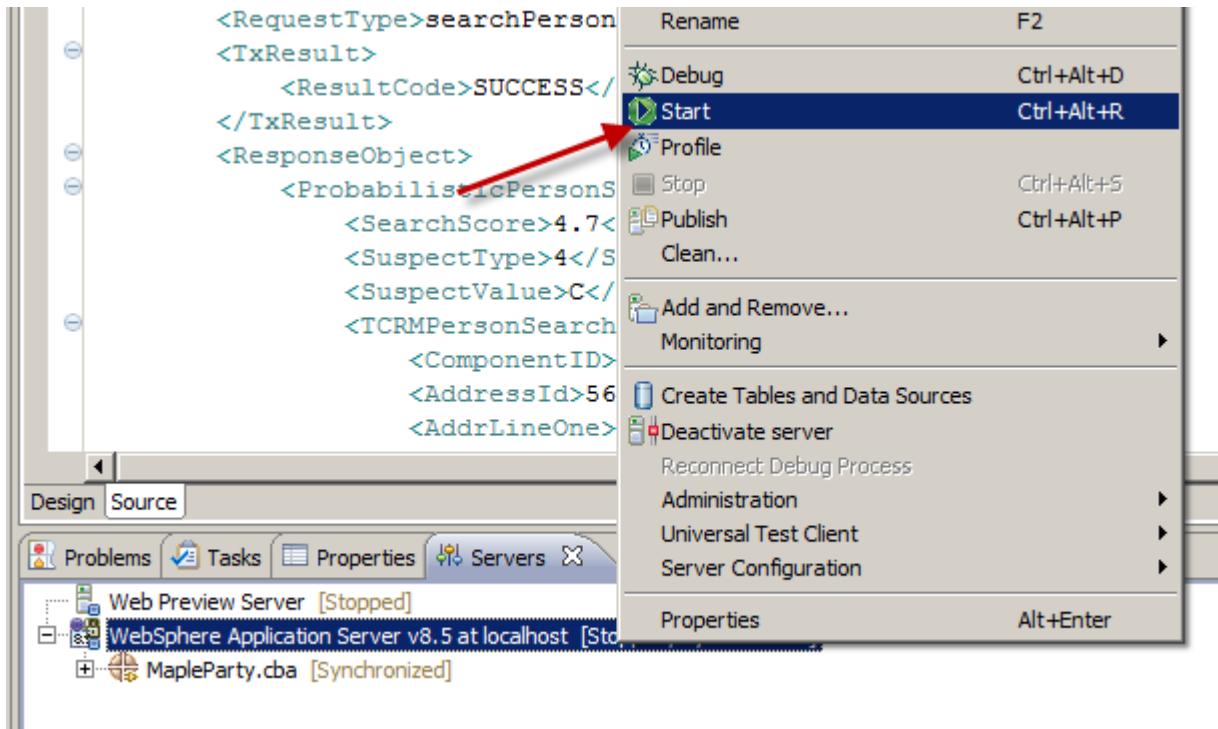
14. Find the PME configuration entry (id 1022, /IBM/BusinessServices/PME/configuration). Change the value to **mapleparty** and save the changes.

1000	/IBM/Hybrid/PersistEntity/mdmorg/memberStatu...	A,O,M
1000	/IBM/DWLCommonServices/KeyGeneration/defa...	S2R2T12I2
1000	/IBM/Hybrid/PersistEntity/contextPoolSize	5
1000	/IBM/BusinessServices/PME/configuration	mapleparty
1000	/IBM/BusinessServices/PME/lookupTimeout	120000
1000	/IBM/EventManager/MessagesInQueue/max	500
1000	/IBM/EventManager/Integration/WebSphereMQ/...	yourMQHostName

15. In order for our new changes to take effect, we will need to restart the server. Under the **Servers** tab, right click on the **WebSphere Application Server** and select **Stop**.



16. Once the server has stopped, right click again on the **WebSphere Application Server** and select **Start**.



Part 5: Testing our new Configuration

While the server is starting, you can perform this first step. We will delete the party that we created in the first exercise so that we can add him again to derive the data once again for the user.

- 1. Open the **CONTACT** table for editing. (Right click and select Data > Edit). Change the **DO_NOT_DELETE_IND** to **1** for our party (bbjones79)

STAMP	LAST_UPDATE_USER [VARCHAR(20)]	LAST_UPDATE_TX_ID [BIGINT]	DO_NOT_DELETE_IND [CHAR(1)]	LAST_UPDATE_TS [TIMESTAMP]
3	mdmadmin	586738791726842252		
2	mdmadmin	627438791728237688		
7	mdmadmin	897438791728551618		
4	mdmadmin	528038911036212434	1	
		567439213128107118		

- 2. Also inside the CONTACT table, under the **CONT_ID** column, right click on the bbjones79 cont_id and select **Copy**. (we will use to this in another service)

CONT_ID [BIGINT]	ACCE_COMP_TP_CD [BIGINT]	PREF_LANG_TP
111111111	1	100
311111111	1	100
911111111	1	100
840001	1	100
568239213128116		
<new row>		

3. Under the Package Explorer, open the **DeleteParty.xml** found under the **LabFiles > GenerateWeights** folder. Change the PartyId in the file to match the cont_id value from the table.

The screenshot shows the Eclipse IDE's Package Explorer on the left and the XML editor on the right. The XML editor displays the following code:

```

<?xml version="1.0" encoding="UTF-8"?>
<TCRMService xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  xmlns="http://www.ibm.com/mdm/schema" xsi:schemaLocation=">
    <RequestControl>
      <requestID>100012</requestID>
      <DWLControl>
        <requesterName>CoreParty</requesterName>
        <requesterLanguage>100</requesterLanguage>
      </DWLControl>
    </RequestControl>
    <TCRMTx>
      <TCRMTxType>deleteParty</TCRMTxType>
      <TCRMObject>TCRMPartyBObj</TCRMObject>
      <TCRMObject>
        <TCRMPartyBObj>
          <PartyId>568239213128116515</PartyId>
        </TCRMPartyBObj>
      </TCRMObject>
    </TCRMTx>
  </TCRMService>

```

4. Run the XML (right click and select Run As > MDM Transaction). You may need to select the server that this service need to run. Check the response folder for the response and look for **SUCCESS** for the ResultCode to ensure that the XML executed correctly. (NOTE: the server will need to be started at this point)

```

<TCRMService xmlns="http://www.ibm.com/mdm/schem
<ResponseControl>
    <resultCode>SUCCESS</resultCode>
    <ServiceTime>3562</ServiceTime>
    <DWLControl>
        <requesterName>mdmadmin</requesterName>
        <requesterLanguage>100</requesterLanguage>
        <requesterLocale>en</requesterLocale>
        <userRole>mdm_admin</userRole>
        <requestID>100012</requestID>
    </DWLControl>
</ResponseControl>
<TxResponse>
    <RequestType>deleteParty</RequestType>
    <TxResult>
        <resultCode>SUCCESS</resultCode>
    </TxResult>
</TxResponse>

```

- ___ 5. Run the **addPartyBob.xml** found under TestingThePhysicalPME to readd the record and derive bob's data using our new PME algorithm.
- ___ 6. Run the following XML again (take a look at what you are searching and the results):
 - ___ a. searchProbabilistic.xml
 - Search Criteria = Name, Username, DOB, Gender, Address, Driver's License, SSN, Phone Number
 - Score = _____
 - Previous Score = 21.9
 - ___ b. searchProbabilistic2.xml
 - Search Criteria = DOB, SSN
 - Score = _____
 - Previous Score = 7.6
 - ___ c. searchProbabilistic3.xml
 - Search Criteria = Name, SSN
 - Score = _____
 - Previous Score = 9.1
 - ___ d. searchProbabilistic4.xml
 - Search Criteria = Name, Address
 - Score = _____
 - Previous Score = 7.5
 - ___ e. searchProbabilistic5.xml
 - Search Criteria = DOB, Contact Method
 - Score = _____
 - Previous Score = 4.7
 - ___ f. searchProbabilistic6.xml

- Search Criteria = DOB, Driver's License
 - Score = _____
 - Previous Score = No Record Found
- ___ g. searchProbabilistic7.xml
- Search Criteria = Display Name, Driver's License
 - Score = _____
 - Previous Score = Insufficient Search Criteria
- ___ h. searchProbabilistic8.xml
- Search Criteria = Display Name
 - Score = _____
 - Previous Score = Insufficient Search Criteria

End of exercise

Exercise 10. Customizing MDM Converters

What this exercise is about

This exercise covers customizing the MDM converters that map the physical module business objects to the PME configuration project.

What you should be able to do

At the end of this exercise, you should be able to:

- Create a new project for an external rule
- Locate and load the OOTB converters into the RAD environment
- Modify a converter used for Duplicate Suspect Processing
- Deploy a new converter to the Physical module

Introduction

In certain use cases you may need to customize the Converter to pass information to the PME record model, this could include an extension or data addition to the Physical MDM that you want to use in the Search or Matching process. In our exercise we will use the DisplayName however this process would be the same if you were passing any custom attribute over to the algorithm.

This exercise includes some custom code and deployment of custom code. We've provided the code for you, however if you are interested in more information around development for InfoSphere MDM, you can also take another MDM course ZZ840: InfoSphere MDM Workbench.

Requirements

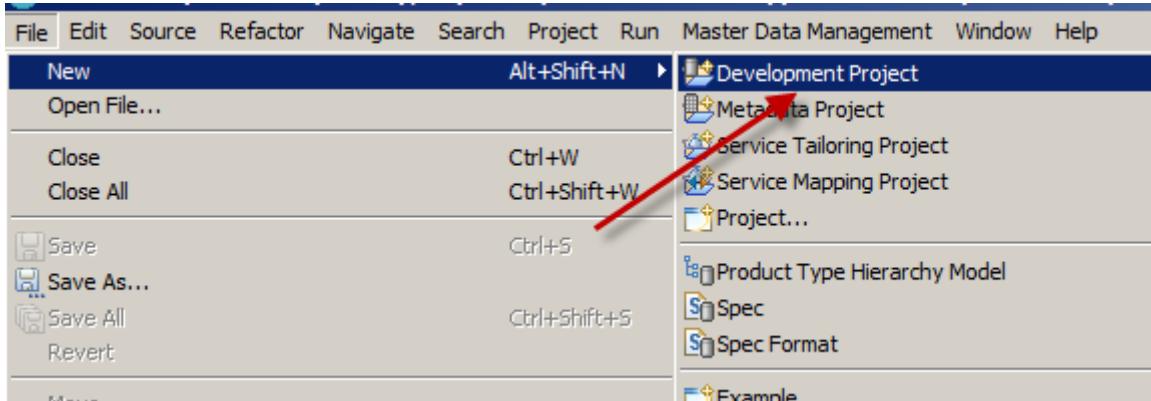
Each of these exercises depend on the previous exercise being complete. For this exercise you must have the Physical PME algorithm configured and deployed that contains the PERUSERNAME field.

Exercise instructions

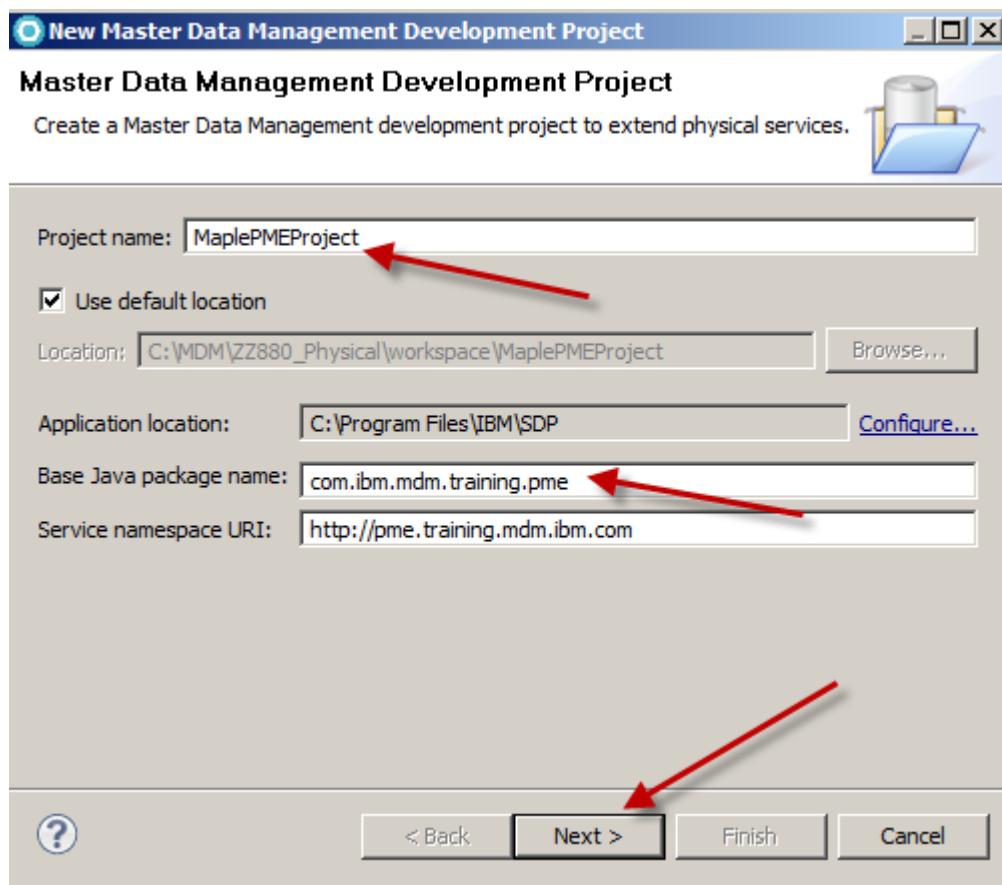
Part 1: Creating our new Development Project

For this exercise we will be working in the MDM Development perspective (Window > Open Perspective > Other ... > MDM Development)

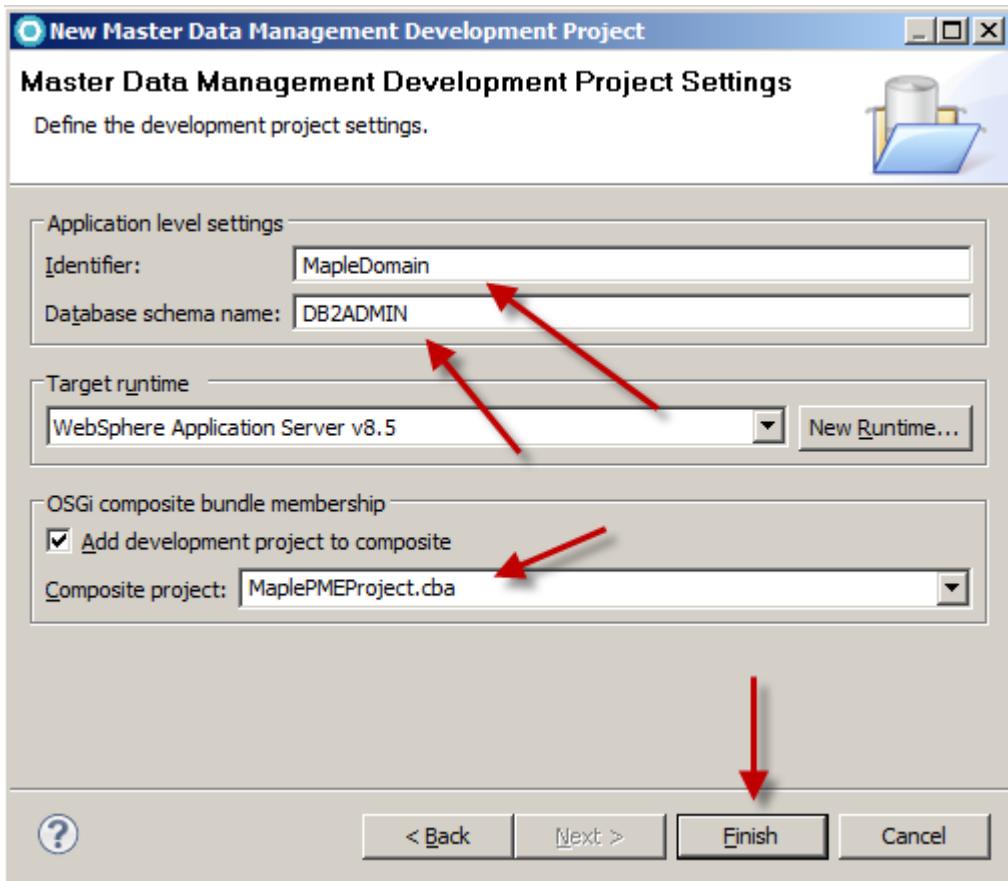
- ___ 1. To begin creating our custom converter, we will first need to create a new Project in our environment. In RAD, select **File > New > Development Project**



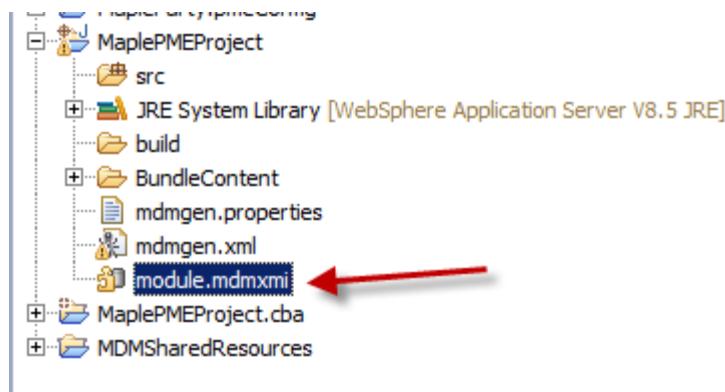
- ___ 2. Enter **MaplePMEProject** for the project name, **com.ibm.mdm.training.pme** for the Base Java package name and click the **Next >** button.



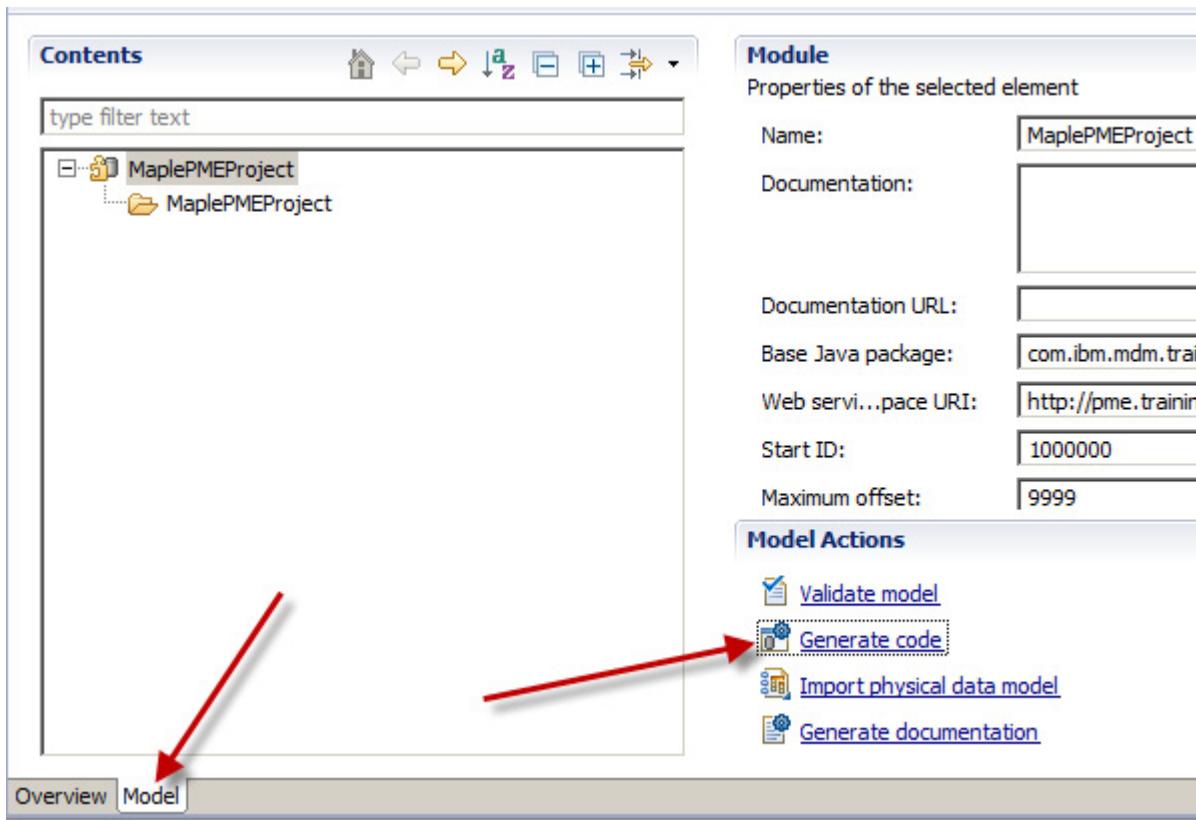
- ___ 3. Enter the following details and click the Finish button:
- ___ a. Identifier: **MapleDomain**
 - ___ b. Database schema name: **DB2ADMIN**
 - ___ c. Composite project: **MaplePMEProject.cba**



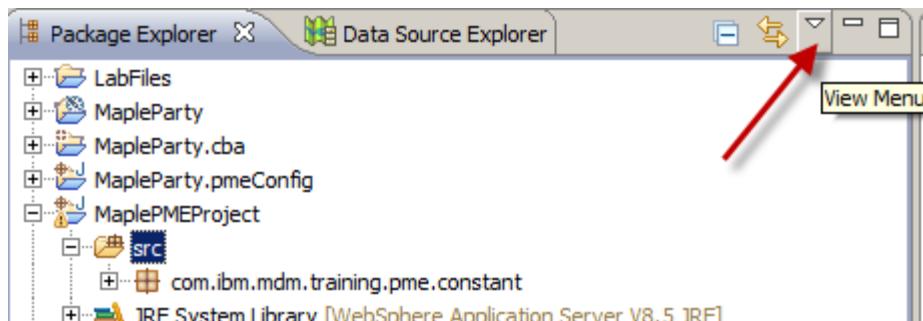
4. Inside the **MaplePMEProject**, open the **module.mdmxmi** file.



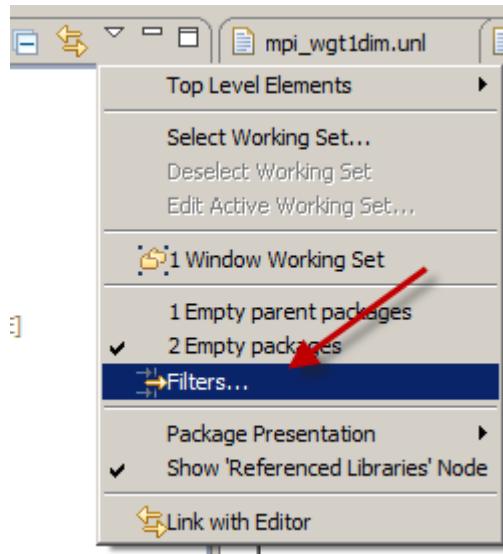
5. Select the **Model** tab at the bottom of the file and click the **Generate Code** link (this will create all of our necessary file structure)



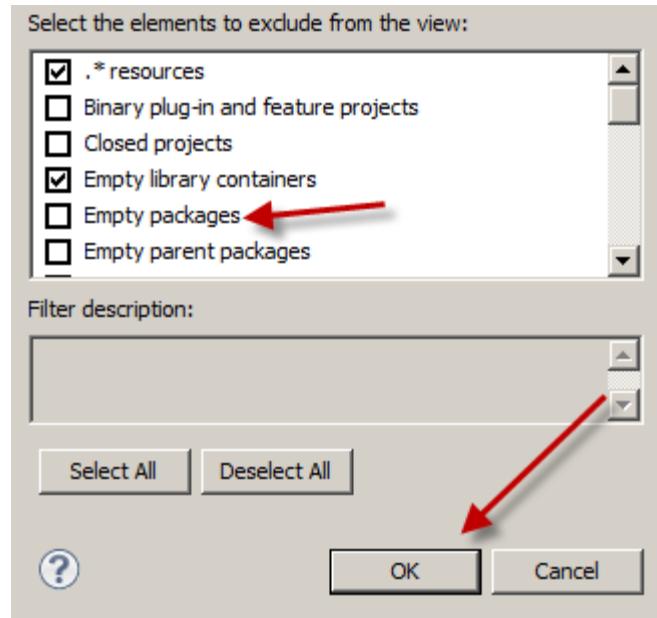
- ___ 6. At the top of the Package Explorer, select the **View Menu** button



- ___ 7. Select the **Filters...** option

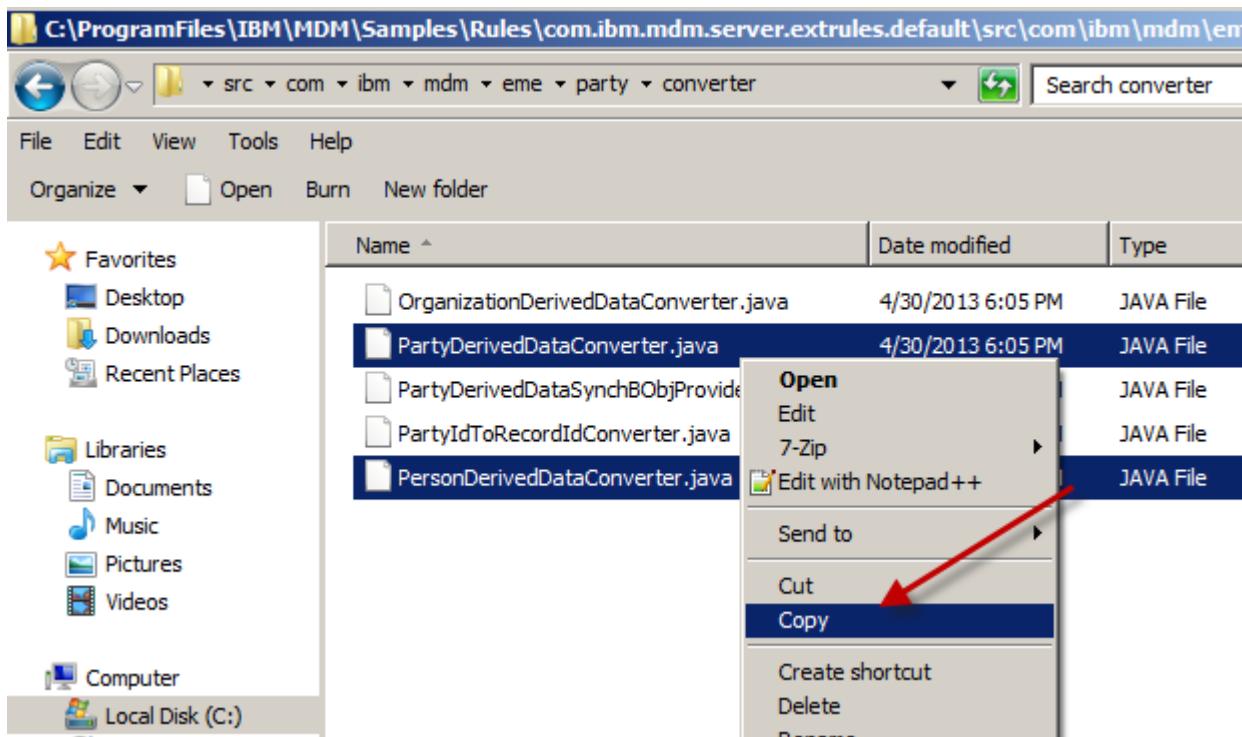


- ___ 8. De-select the **Empty parent packages** option and click the **OK** button

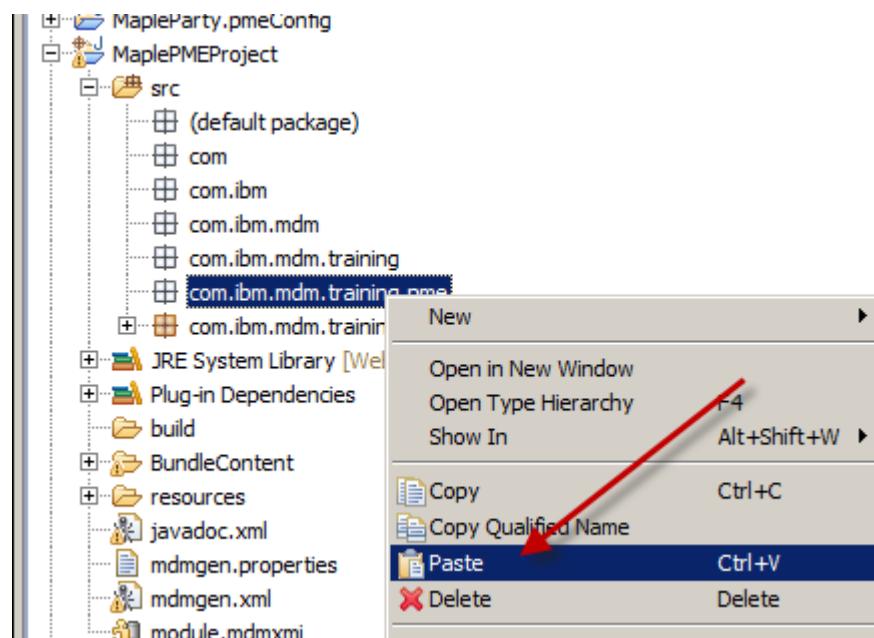


Part 2: Creating Custom Converters

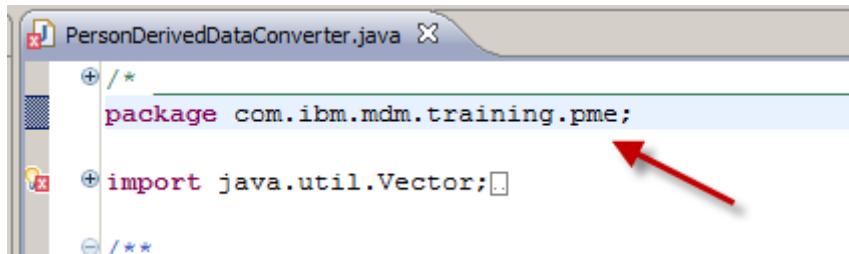
- ___ 1. When we install the InfoSphere MDM on this image, we added the **Samples** as an installation. In the samples you'll find the default converter source code that we will use as a base to create our new converter. Open a Windows explorer and navigate to **C:\ProgramFiles\IBM\MDM\Samples\Rules\com.ibm.mdm.server.exrules.default\src\com\ibm\mdm\eme\party\converter**.
- ___ 2. Right click on the **PartyDerivedDataConverter.java** and **PersonDerivedDataConverter.java** files (used CTRL to select both) and select Copy.



- ___ 3. Back under the RAD environment, right click on our new package and select **Paste**.



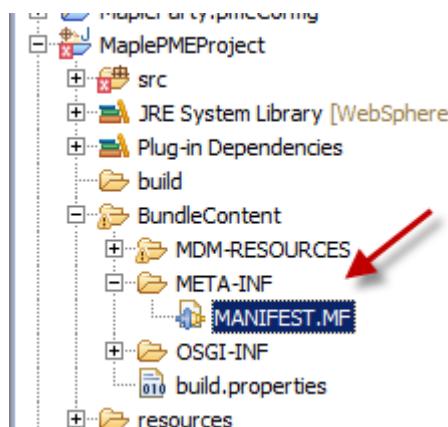
- ___ 4. Once you've pasted the files, you will find an error in each of the files. Open both files and modify the package name to **com.ibm.mdm.training.pme**.



```
PersonDerivedDataConverter.java
/*
package com.ibm.mdm.training.pme;
import java.util.Vector;
/**
```

Part 3: Resolving Errors and configuration

- 1. You will still find a number of errors in the workspace, to fix these, we will need to import a number of packages in the Manifest file. Open **MANIFEST.MF** file found under the **MaplePMEProject > BundleContent > META-INF** folder



- 2. Select the **Dependencies** tab and click the **Add...** button under the **Imported Packages** window.

Dependencies

Required Plug-ins

Specify the list of plug-ins required for the operation of this plug-in.

Total: 0

Imported Packages

Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.

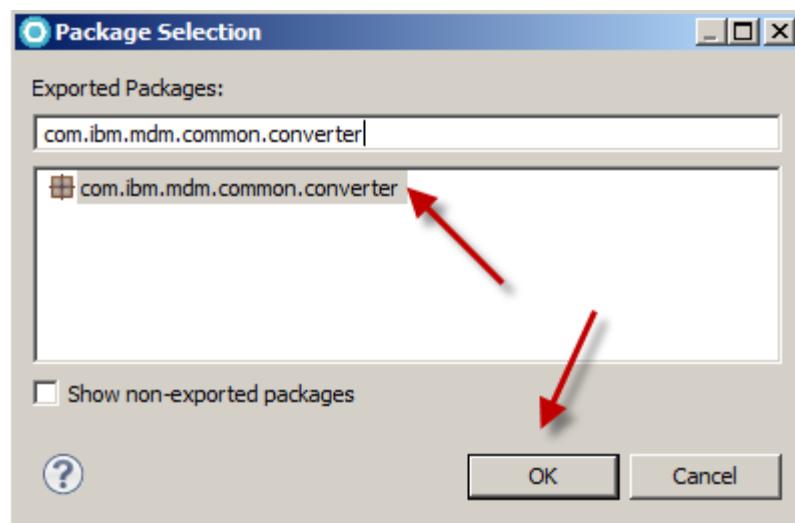
Total: 50

Automated Management of Dependencies

Dependency Analysis

Overview Dependencies Runtime Build MANIFEST.MF build.properties

3. Enter **com.ibm.mdm.common.converter** and click the **OK** button



4. Save the file (CTRL+S)

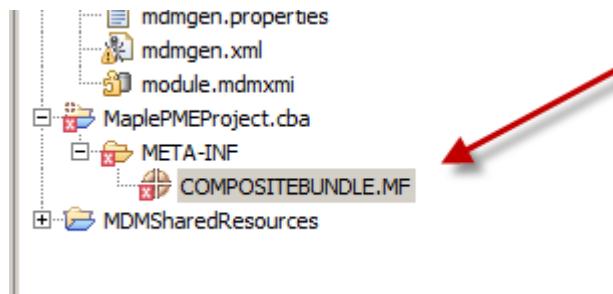
- 5. Repeat the previous steps for all the packages that are highlighted as errors in both the PartyDerivedDataConverter.java and PersonDerivedDataConverter.java that we pasted into our project. You'll find the classpath errors under the imports at the top of the file. The list includes:
- a. com.dwl.tcrm.coreParty.component
 - b. com.dwl.tcrm.coreParty.constant
 - c. com.ibm.mdm.mds.pme.api.bean
 - d. com.ibm.mdm.eme.metadata
 - e. com.dwl.tcrm.coreParty.entityObject

```

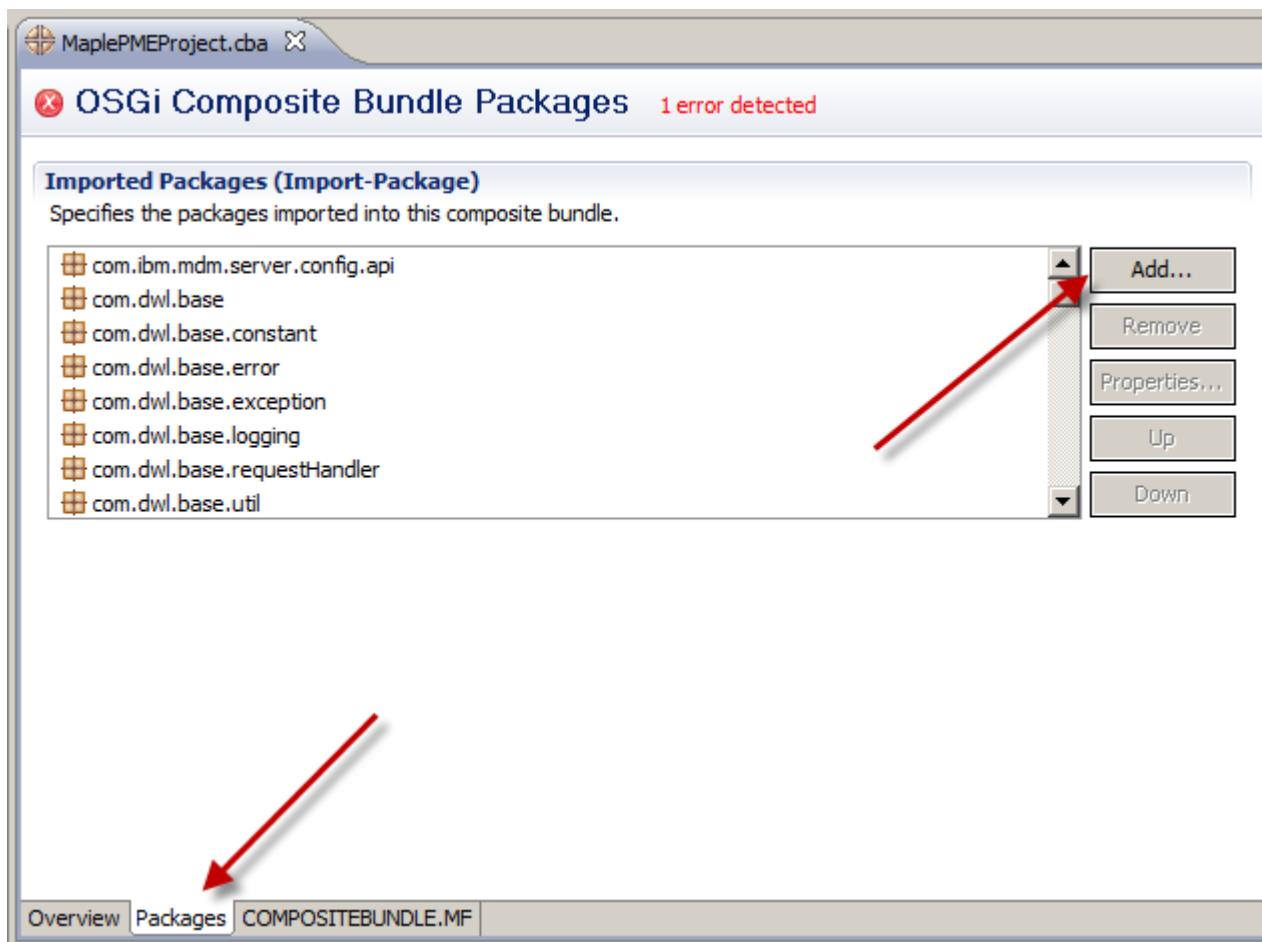
import com.dwl.base.DWLCommon;
import com.dwl.base.IDWLErrorMessage;
import com.dwl.base.error.DWLStatus;
import com.dwl.base.logging.DWLLoggerManager;
import com.dwl.base.logging.IDWLLogger;
import com.dwl.base.util.StringUtils;
import com.dwl.tcrm.common.TCRMErrorCode;
import com.dwl.tcrm.coreParty.component.ProbabilisticPersonSe
import com.dwl.tcrm.coreParty.component.TCRMAddressBObj;
import com.dwl.tcrm.coreParty.component.TCRMContactMethodBObj
import com.dwl.tcrm.coreParty.component.TCRMPartyAddressBObj;
import com.dwl.tcrm.coreParty.component.TCRMPartyBankAccountB
import com.dwl.tcrm.coreParty.component.TCRMPartyChargeCardBO
import com.dwl.tcrm.coreParty.component.TCRMPartyContactMetho
import com.dwl.tcrm.coreParty.component.TCRMPartyIdentificati
import com.dwl.tcrm.coreParty.component.TCRMPersonBObj;
import com.dwl.tcrm.coreParty.component.TCRMPersonNameBObj;
import com.dwl.tcrm.coreParty.constant.TCRMCoreComponentTD:

```

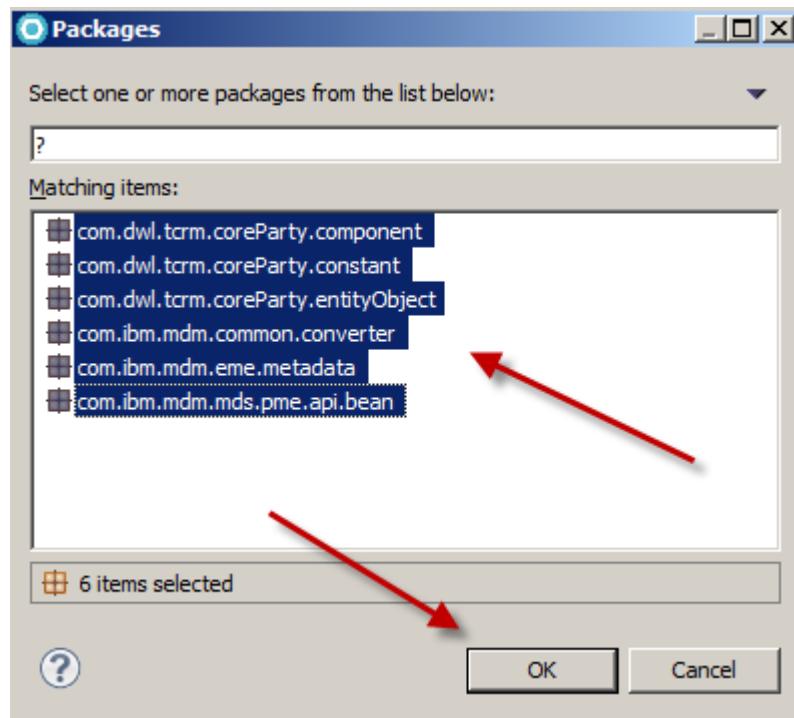
- 6. You should at this point have no errors in the MaplePMEProject, however you will see errors in the MaplePMEProject.cba. This is because the composite bundle is not importing the same packages. To fix this error, open the **COMPOSITEBUNDLE.MF** under the **MaplePMEProject.cba > META-INF** directory



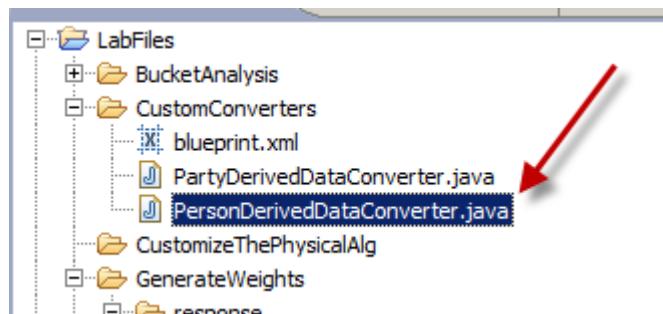
- 7. Under the **Packages** tab, click the **Add...** button.



8. Select all the packages that are listed and click the **OK** button.



9. We have now successfully imported the default converters and are now ready to make our modifications. For our converters, we will add a couple of lines of code that will convert the DisplayName off the Person Business Object to the PERUSERNAME on the configuration. We've already coded the solution for you. Under the **LabFiles > CustomConverters** folder, you'll find the solution **PersonDerivedDataConverter.java**. Open the java file.



10. Inside the file you'll find some lines of code that deal with the Display Name (and create an attribute called PERUSERNAME for the configuration). **Copy** the contents of the solution file and replace the contents of the **PersonDerivedDataConverter.java** in your MaplePMEProject.

```

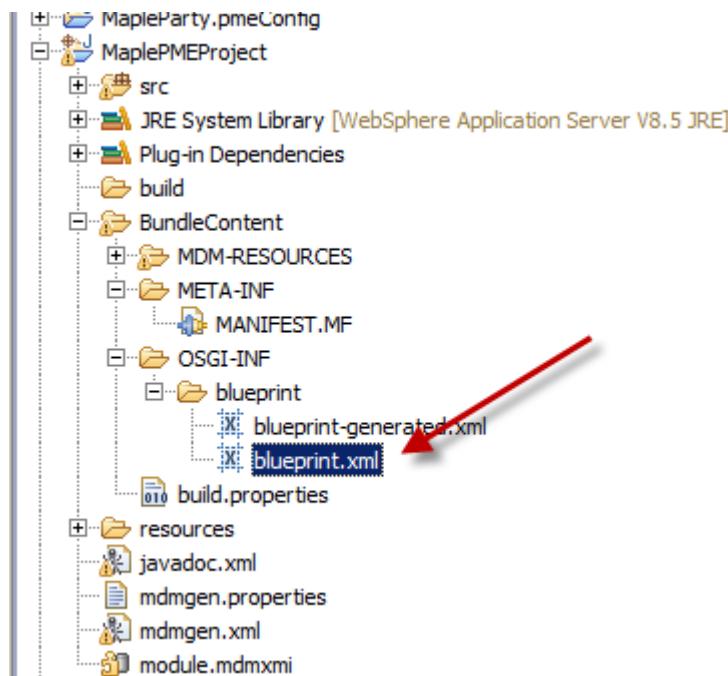
// synchronized to EME.
for (CriticalData criticalData : vecCritVectorForPerson) {

    String fieldName = criticalData.getElementName();
    if (StringUtils.isNotBlank(personObj.getBirthDate()))
        && fieldName
        .equalsIgnoreCase(BIRTH_DATE_MDM_ATTRIBUTE_NAME)) {
        // convert to Date.
        personRecord.addAttribute(buildAttribute("PERBIRTHDATE",
            "val",
            DateFormatter.getDate(personObj.getBirthDate())));
    }
    if (StringUtils.isNotBlank(personObj.getDisplayName()))
        && fieldName
        .equalsIgnoreCase(USERNAME_MDM_ATTRIBUTE_NAME)) {
        // convert to Date.
        personRecord.addAttribute(buildAttribute("PERUSERNAME",
            "idnum",
            personObj.getDisplayName()));
    }
}

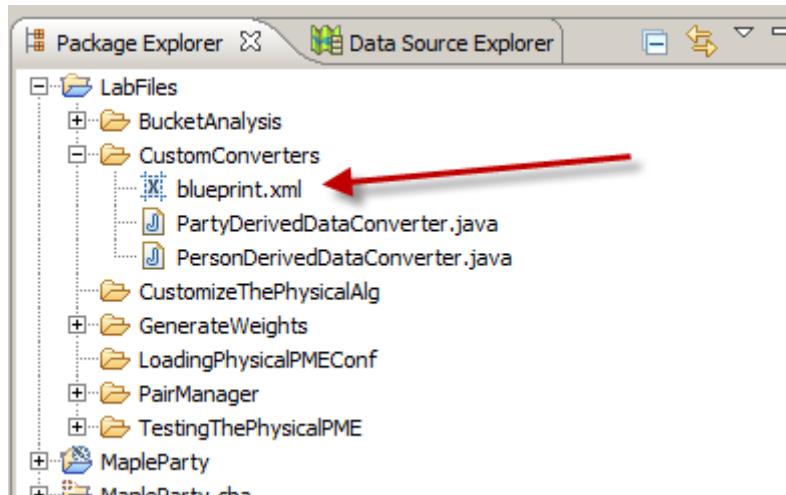
```



11. We have the code complete, but we still haven't registered our new Converter service for MDM to use. Open the blueprint.xml file found under the **MaplePMEProject > BundleContent > OSGI-INF > blueprint** folder.



12. This file will be empty, but we have also created the solution for you in the **blueprints.xml** file found under the **LabFiles > CustomConverters** folder.



- ___ 13. Copy the contents of the solution blueprints.xml file found under the LabFile in the blueprint.xml file found under the MaplePMEProject. You'll notice 3 services we define for our bundle:
 - ___ a. Match – used to suspect duplicate processing when a business object is added or updated
 - ___ b. Search – used for the searchPersonProbabilistic service
 - ___ c. Sync – used to synchronize data when critical fields are added or updated.

```

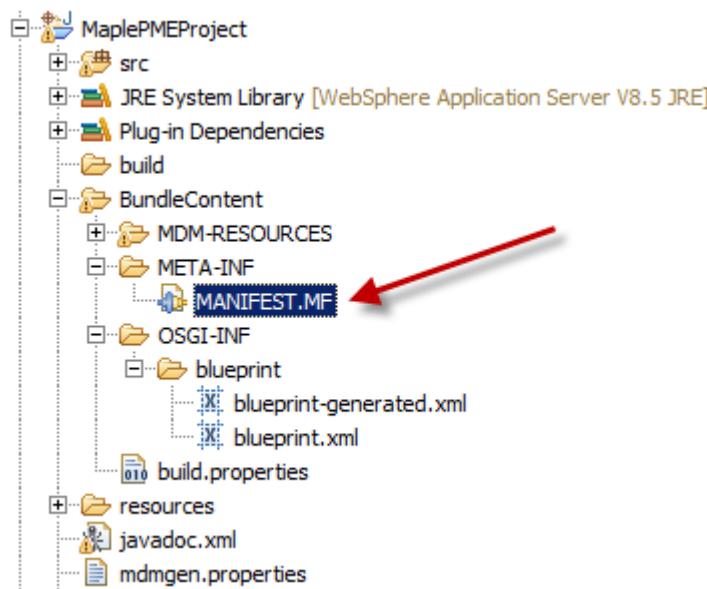
<service id="ConverterFactoryService.PersonDerivedDataConverter.Match" interface="com.ibm.mdm.common.converter.ConverterFactoryService">
  <service-properties>
    <entry key="object.to.convert" value="com.dwl.tcrm.coreParty.component.TCRMPersonBObj"/>
    <entry key="converter.type" value="eme.matching.converter"/>
  </service-properties>
  <bean class="com.ibm.mdm.common.converter.ConverterFactoryServiceImpl">
    <argument ref="blueprintBundle"/>
    <argument value="com.ibm.mdm.training.pme.PersonDerivedDataConverter"/>
  </bean>
</service>

<service id="ConverterFactoryService.PersonDerivedDataConverter.Search" interface="com.ibm.mdm.common.converter.ConverterFactoryService">
  <service-properties>
    <entry key="object.to.convert">
      <list>
        <value>com.dwl.tcrm.coreParty.component.TCRMPersonBObj</value>
        <value>com.dwl.tcrm.coreParty.component.ProbabilisticPersonSearchBObj</value>
      </list>
    </entry>
    <entry key="converter.type" value="eme.searchSuspects.converter"/>
  </service-properties>
  <bean class="com.ibm.mdm.common.converter.ConverterFactoryServiceImpl">
    <argument ref="blueprintBundle"/>
    <argument value="com.ibm.mdm.training.pme.PersonDerivedDataConverter"/>
  </bean>
</service>

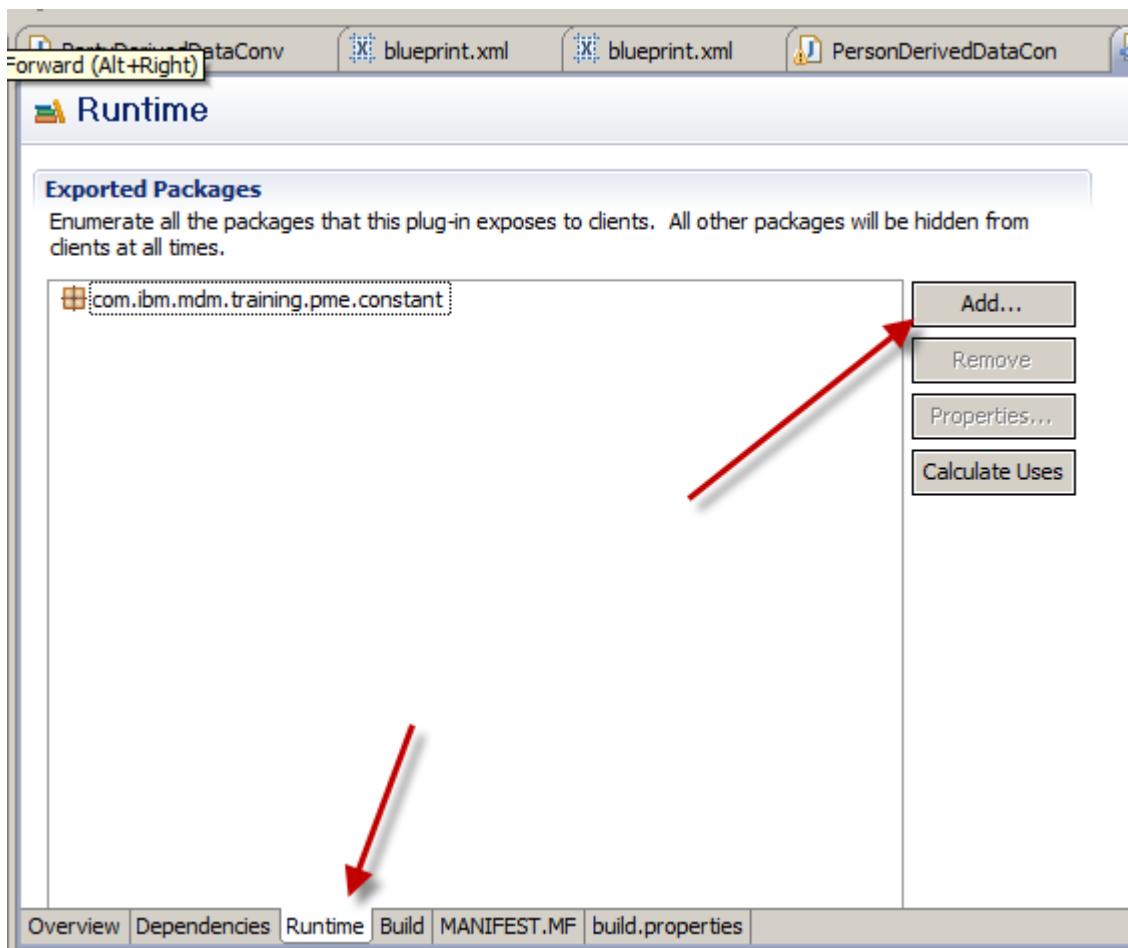
<service id="ConverterFactoryService.PersonDerivedDataConverter.Sync" interface="com.ibm.mdm.common.converter.ConverterFactoryService">
  <service-properties>
  </service-properties>

```

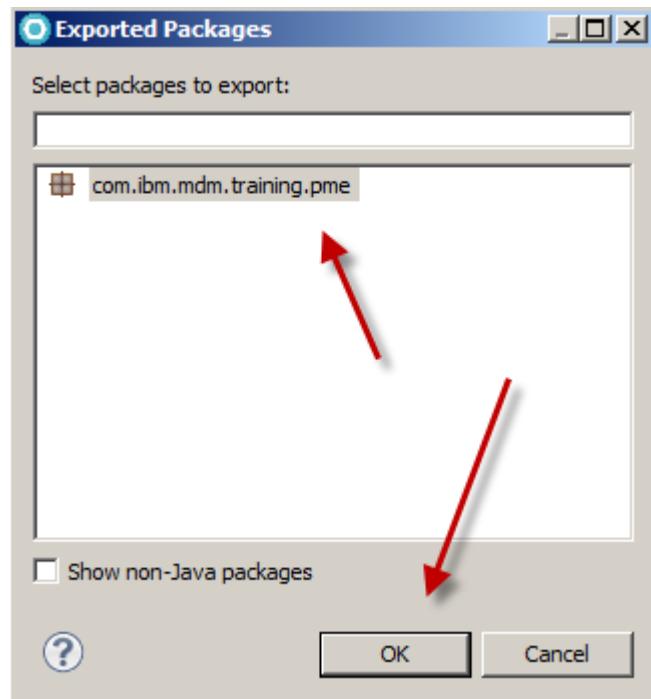
- ___ 14. Now that we have the code and have registered the server, the last step is to export our packages so that the MDM can access our services and converter. First we will update our MaplePMEProject. Open the **MANIFEST.MF** file found under the **MaplePMEProject > BundleContent > META-INF** folder.



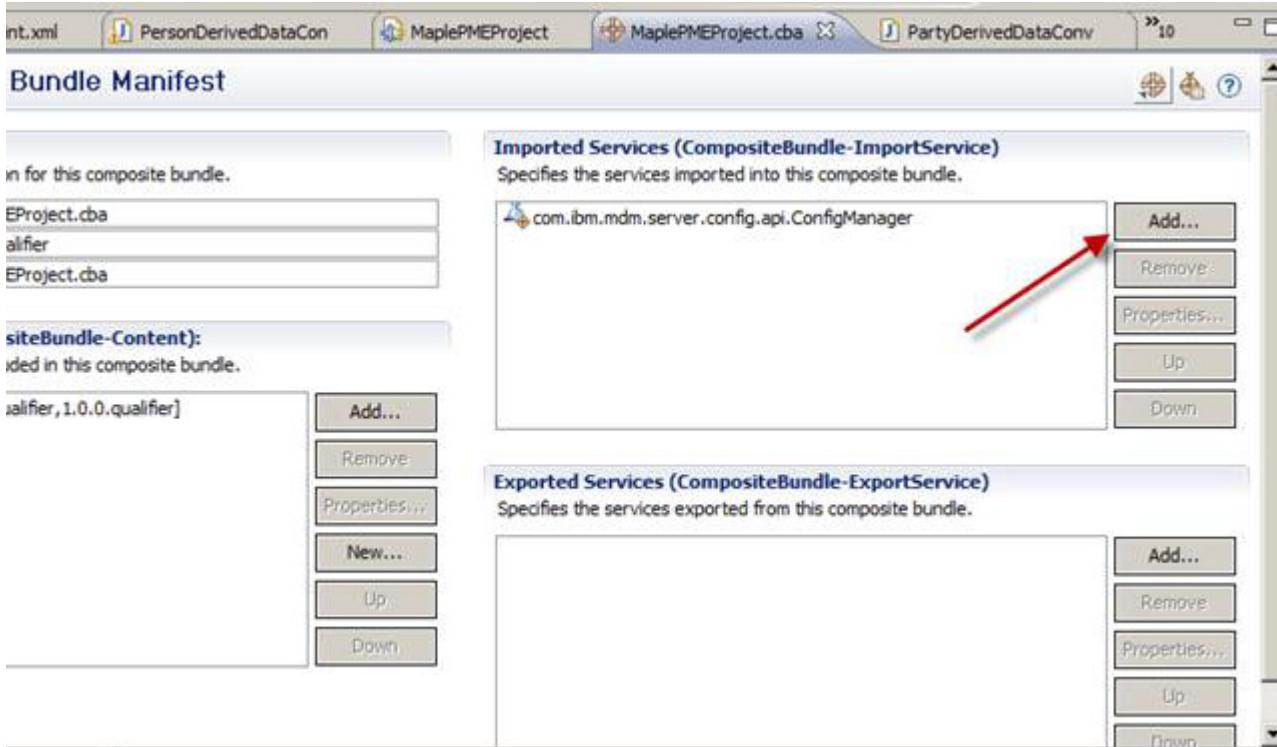
- 15. Under the **Runtime** tab, click the **Add...** button.



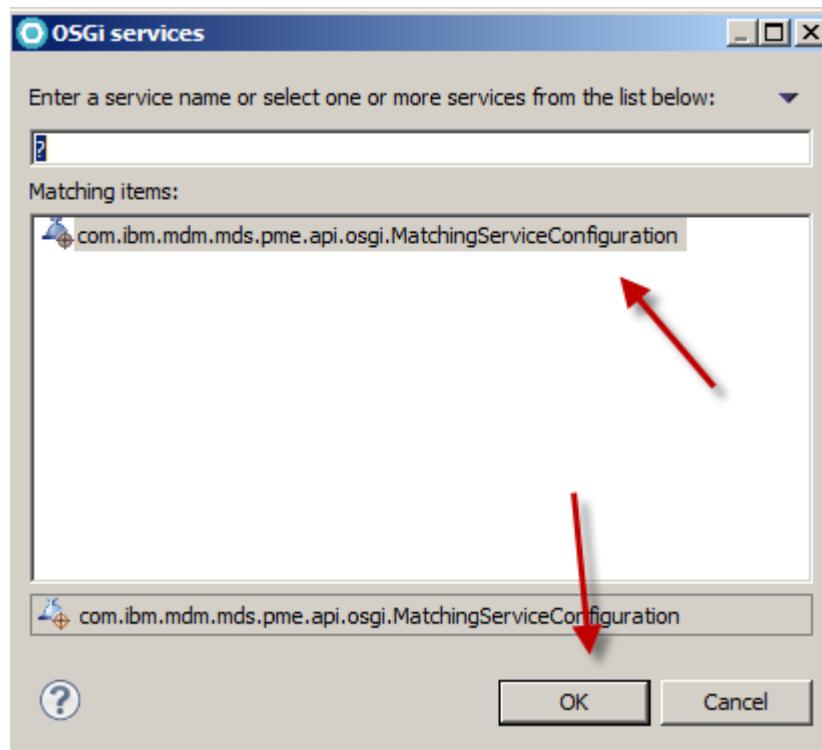
- 16. Select the **com.ibm.mdm.training.pme** package and click the **OK** button.



- ___ 17. Save the file (CTRL+S)
- ___ 18. Next we will update the composite bundle. Open the **COMPOSITEBUNDLE.MF** file found under the **MaplePMEProject.cba > META-INF** folder. Under the **Overview** tab, click the **Add...** button for the Imported Servers.



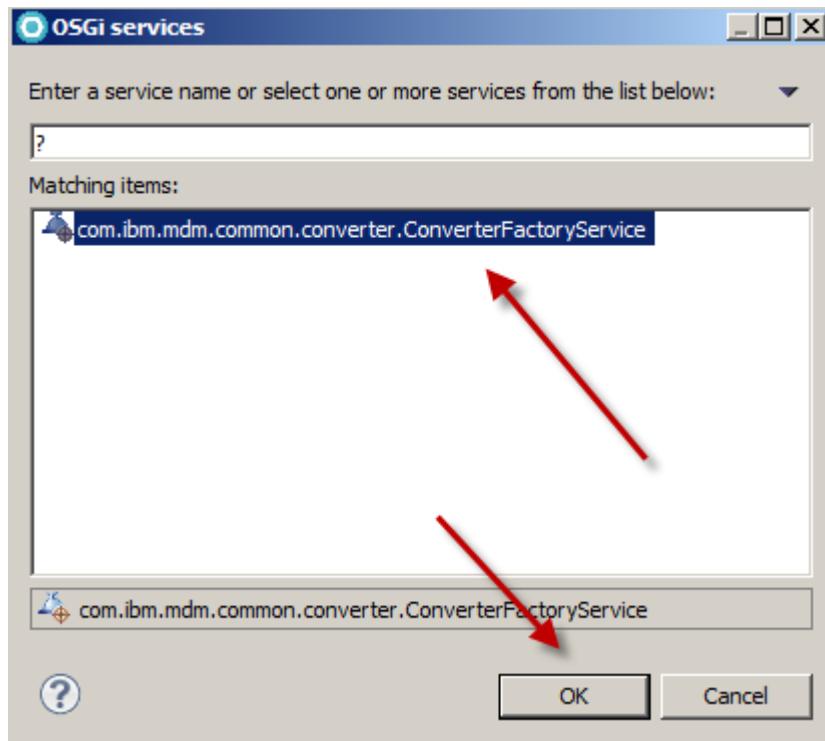
- 19. Select the **com.ibm.mdm.mds.pme.api.osgi.MatchingServiceConfiguration** and click the **OK** button



- 20. Next, click the **Add...** button under the **Exported Services**.

The screenshot shows the 'Composite Bundle' configuration interface. It has two main sections: 'Imported Services (CompositeBundle-ImportService)' and 'Exported Services (CompositeBundle-ExportService)'.
The 'Imported Services' section lists 'com.ibm.mdm.server.config.api.ConfigManager' and 'com.ibm.mdm.mds.pme.api.osgi.MatchingServiceConfiguration'. To its right are buttons for 'Add...', 'Remove', 'Properties...', 'Up', and 'Down'.
The 'Exported Services' section is currently empty. To its right are buttons for 'Add...', 'Remove', 'Properties...', 'Up', and 'Down'. A red arrow points to the 'Add...' button in the 'Exported Services' section.

- ___ 21. Select the **com.ibm.mdm.common.converter.ConverterFactoryService** and click the **OK** button.

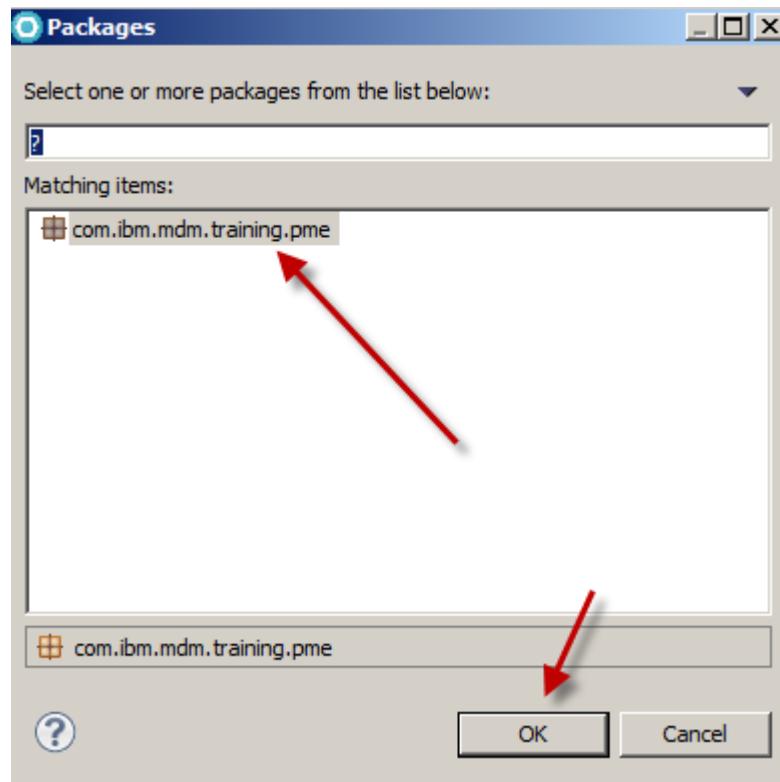


- ___ 22. Under the **Packages** tab, click the **Add...** button under the **Exported Packages**.

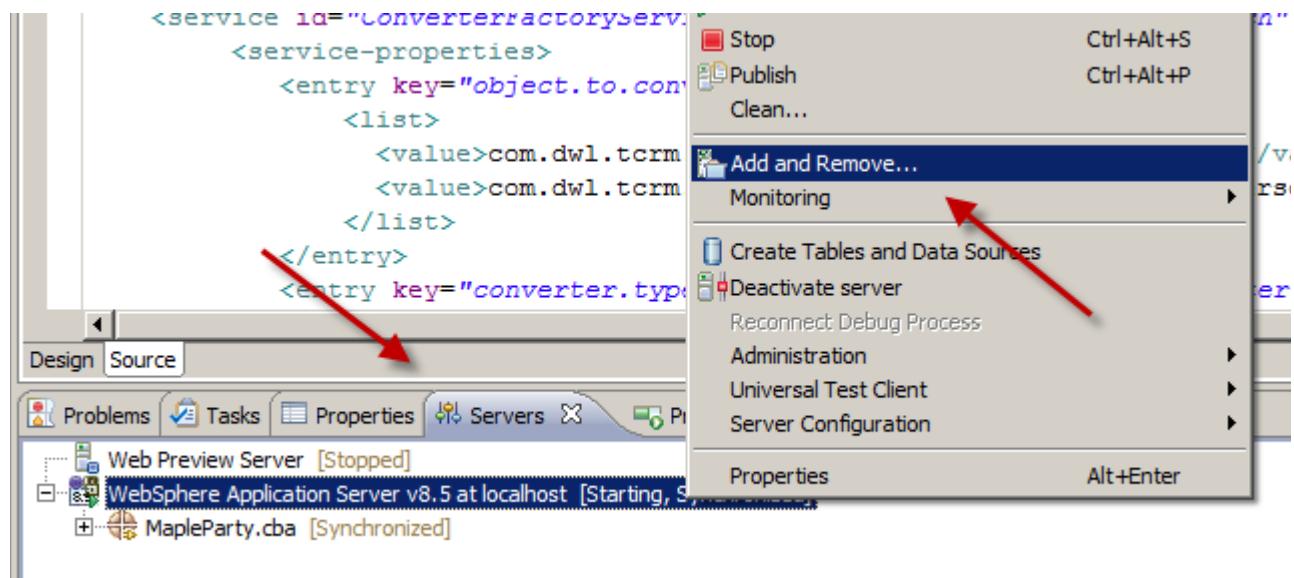
Imported Packages (Import-Package)		Exported Packages (Export-Package)	
Specifies the packages imported into this composite bundle.		Specifies the packages exported from this composite bundle.	
om.ibm.mdm.suspect.interfaces	Add...	com.ibm.mdm.training.pme.constant	Add...
om.ibm.mdm.suspect.component	Remove		Remove
om.ibm.mdm.common.brokers	Properties...		Properties...
om.ibm.mdm.common.codetype.obj	Up		Up
om.ibm.mdm.common.codetype.interfaces	Down		Down
om.ibm.mdm.common.codetype.componer			
om.dwl.tcrm.coreParty.component			

Packages COMPOSITEBUNDLE.MF

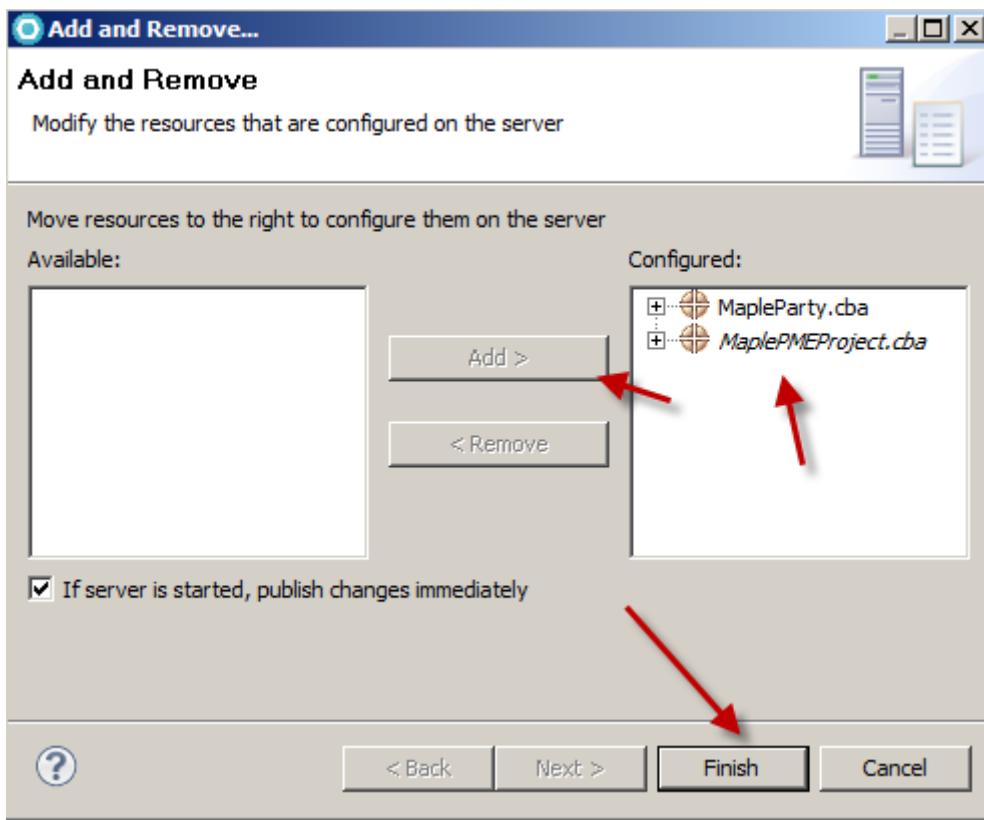
- ___ 23. Select the **com.ibm.mdm.training.pme** and click the **OK** button.



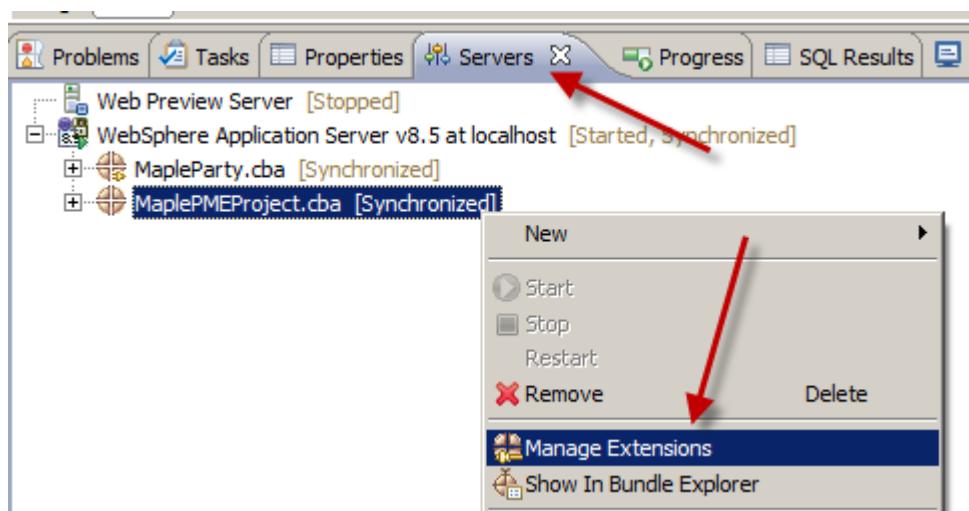
- ___ 24. Save the file (CTRL+S)
- ___ 25. Now we are ready to deploy our new package. Under the **Servers** tab, right click on the WebSphere Application Server and select **Add and Remove...**



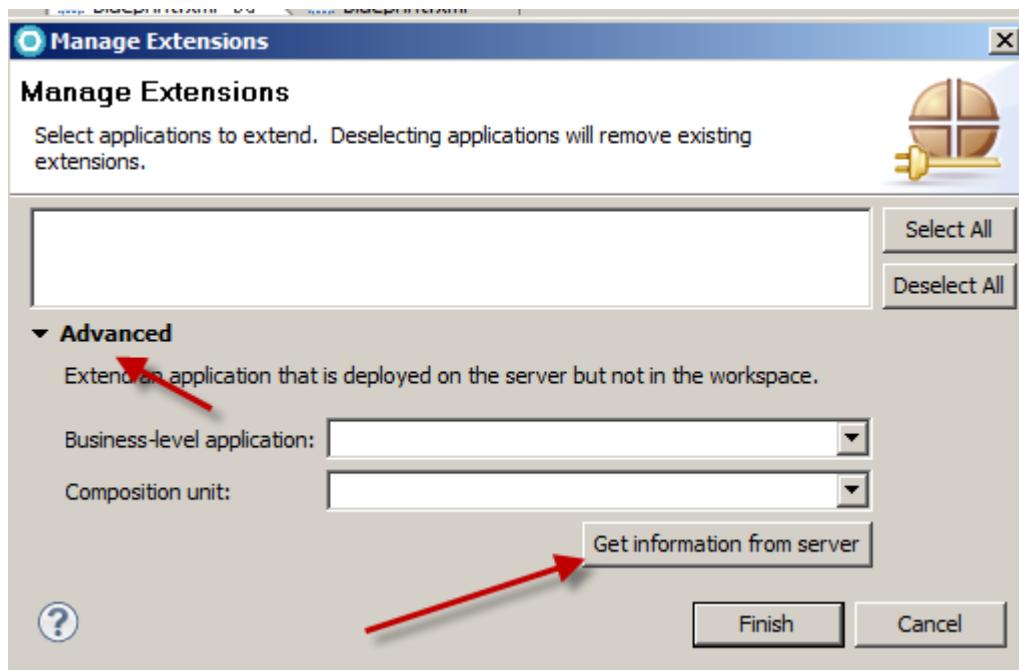
- ___ 26. Move the **MaplePMEProject.cba** under the **Configured** projects using the **Add >** button and click the **Finish** button.



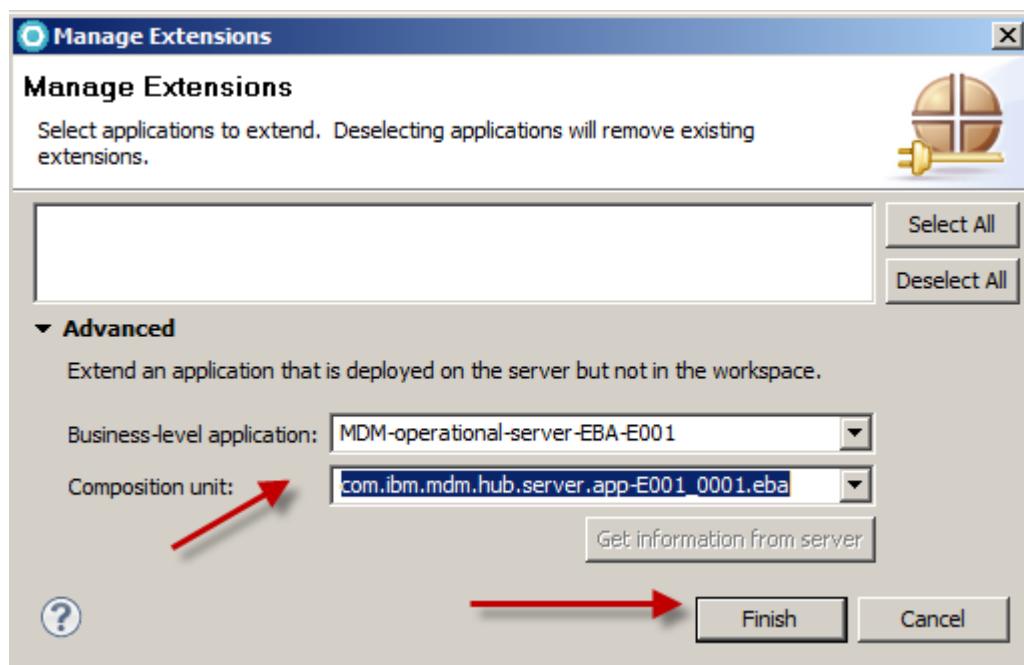
- 27. We need to attach the bundle to the MDM, under the **Servers** window, right click on the **MaplePMEProject.cba** and select **Manage Extensions**.



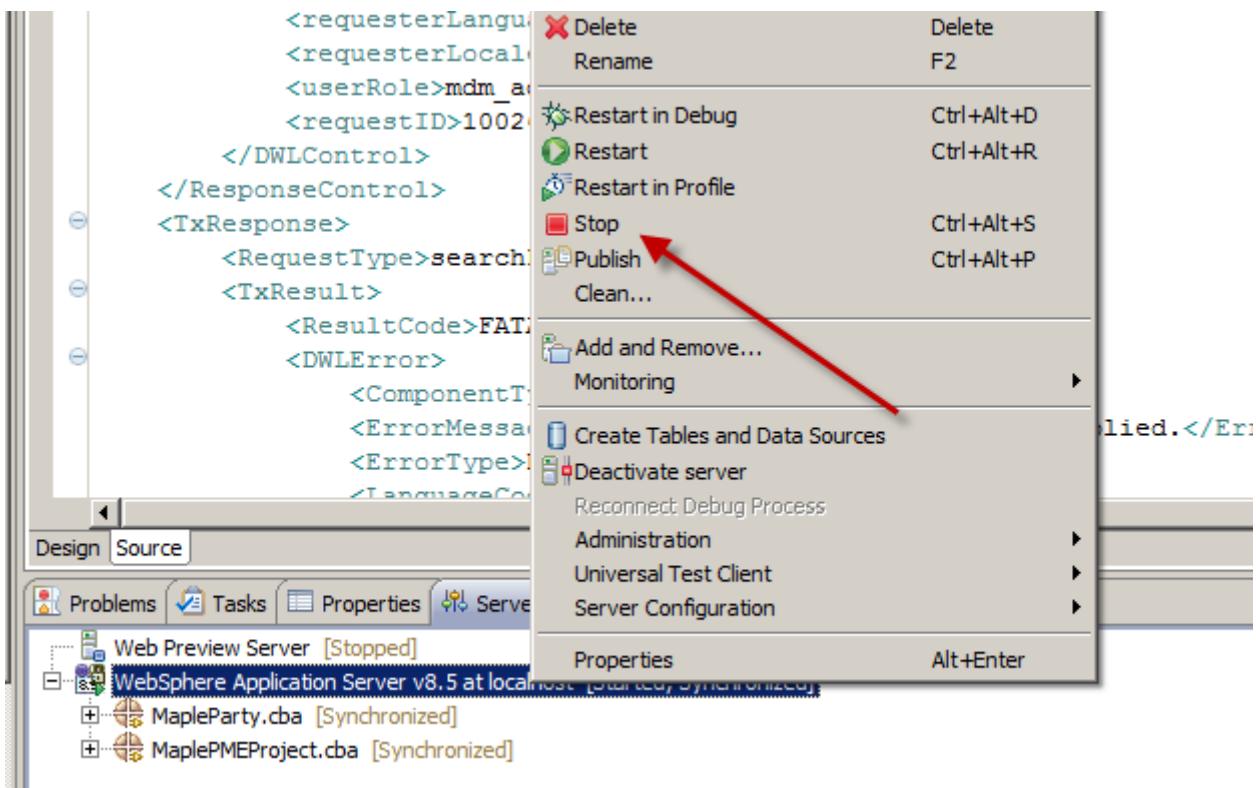
- 28. Select the **Advanced** tab and click the **Get information from server** button.



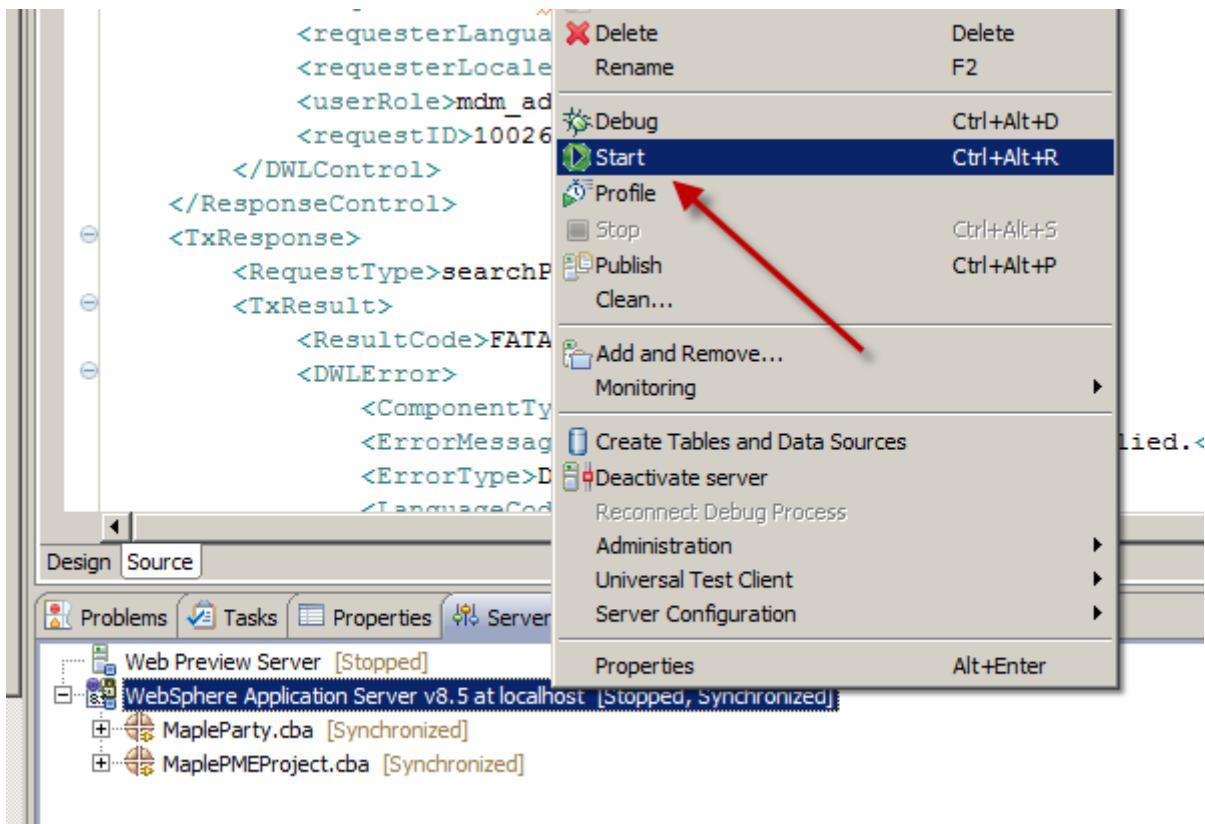
- ___ 29. Select the following values and click the **Finish** button:
- Business-level application: **MDM-operational-server-EBA-E001**
 - Composite unit: **com.ibm.mdm.hub.server.app-E001_0001.eba**



- ___ 30. The deployment will take up to 5 minutes. When the deployment is complete, we'll still need to restart the server. Right click on the **WebSphere Application Server** and select **Stop**.



31. Once the server has stopped, right click again on the WebSphere Application Server and select **Start**.



Part 4: Testing our new Configuration

While the server is starting, you can perform this first step. We will delete the party that we created in the first exercise so that we can add him again to derive the data once again for the user.

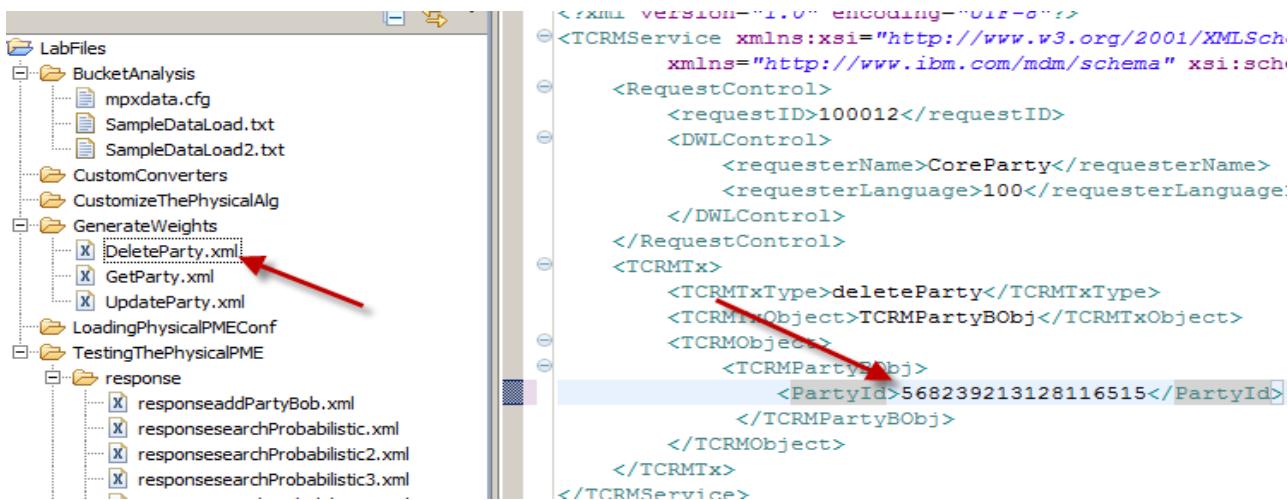
- 1. Open the **CONTACT** table for editing. (Right click and select Data > Edit). Change the **DO_NOT_DELETE_IND** to 1 for our party (bbjones79)

STAMP	LAST_UPDATE_USER [VARCHAR(20)]	LAST_UPDATE_TX_ID [BIGINT]	DO_NOT_DELETE_IND [CHAR(1)]	LAST...
3	mdmadmin	586738791726842252		
2	mdmadmin	627438791728237688		
7	mdmadmin	897438791728551618	528038911036212434	
4	mdmadmin	567439213128107118	1	

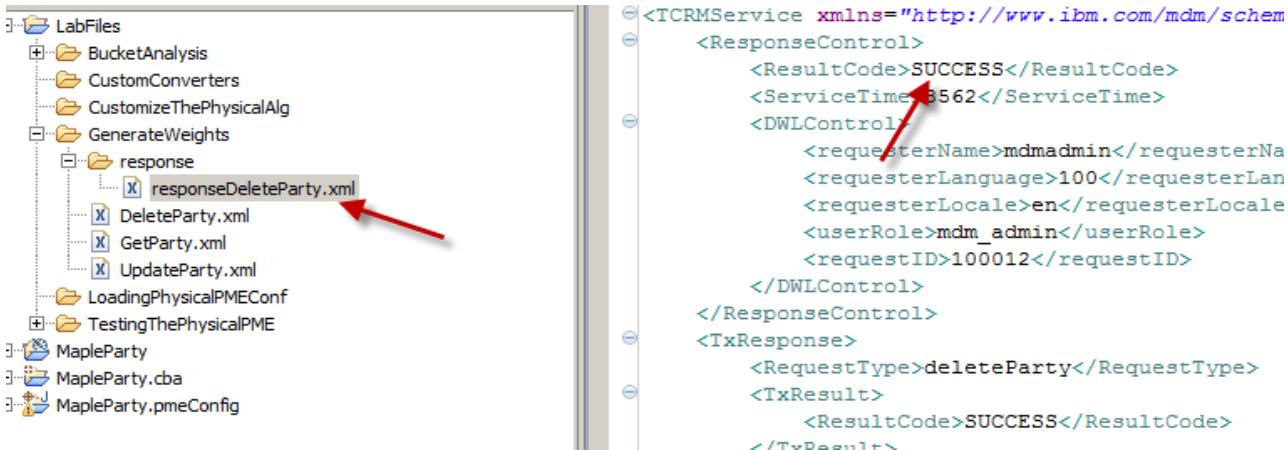
- 2. Also inside the CONTACT table, under the **CONT_ID** column, right click on the bbjones79 cont_id and select **Copy**. (we will use to this in another service)

CONT_ID [BIGINT]	ACCE_COMP_TP_CD [BIGINT]	PREF_LANG_CD [CHAR(1)]
111111111	1	100
311111111	1	100
911111111	1	100
840001	1	100
568239213128116		
<new row>		

- 3. Under the Package Explorer, open the **DeleteParty.xml** found under the **LabFiles > GenerateWeights** folder. Change the PartyId in the file to match the cont_id value from the table.



4. Run the XML (right click and select Run As > MDM Transaction). Check the response folder for the response and look for **SUCCESS** for the ResultCode to ensure that the XML executed correctly. (NOTE: the server will need to be started at this point)



5. Run the **addPartyBob.xml** found under TestingThePhysicalPME to readd the record and derive bob's data using our new PME algorithm.
6. Run the following XML again (take a look at what you are searching and the results):
- searchProbabilistic.xml
 - Search Criteria = Name, Username, DOB, Gender, Address, Driver's License, SSN, Phone Number
 - Score = _____
 - Previous Score = 28.1
 - searchProbabilistic2.xml
 - Search Criteria = DOB, SSN
 - Score = _____
 - Previous Score = 6.3

-
- ___ c. searchProbabilistic3.xml
 - Search Criteria = Name, SSN
 - Score = _____
 - Previous Score = 12.4
 - searchProbabilistic4.xml
 - Search Criteria = Name, Address
 - Score = _____
 - Previous Score = 9.8
 - ___ d. searchProbabilistic5.xml
 - Search Criteria = DOB, Contact Method
 - Score = _____
 - Previous Score = 3.2
 - ___ e. searchProbabilistic6.xml
 - Search Criteria = DOB, Driver's License
 - Score = _____
 - Previous Score = 9.6
 - ___ f. searchProbabilistic7.xml
 - Search Criteria = Display Name, Driver's License
 - Score = _____
 - Previous Score = 5.7
 - ___ g. searchProbabilistic8.xml
 - Search Criteria = Display Name
 - Score = _____
 - Previous Score = Insufficient Search Criteria

End of exercise

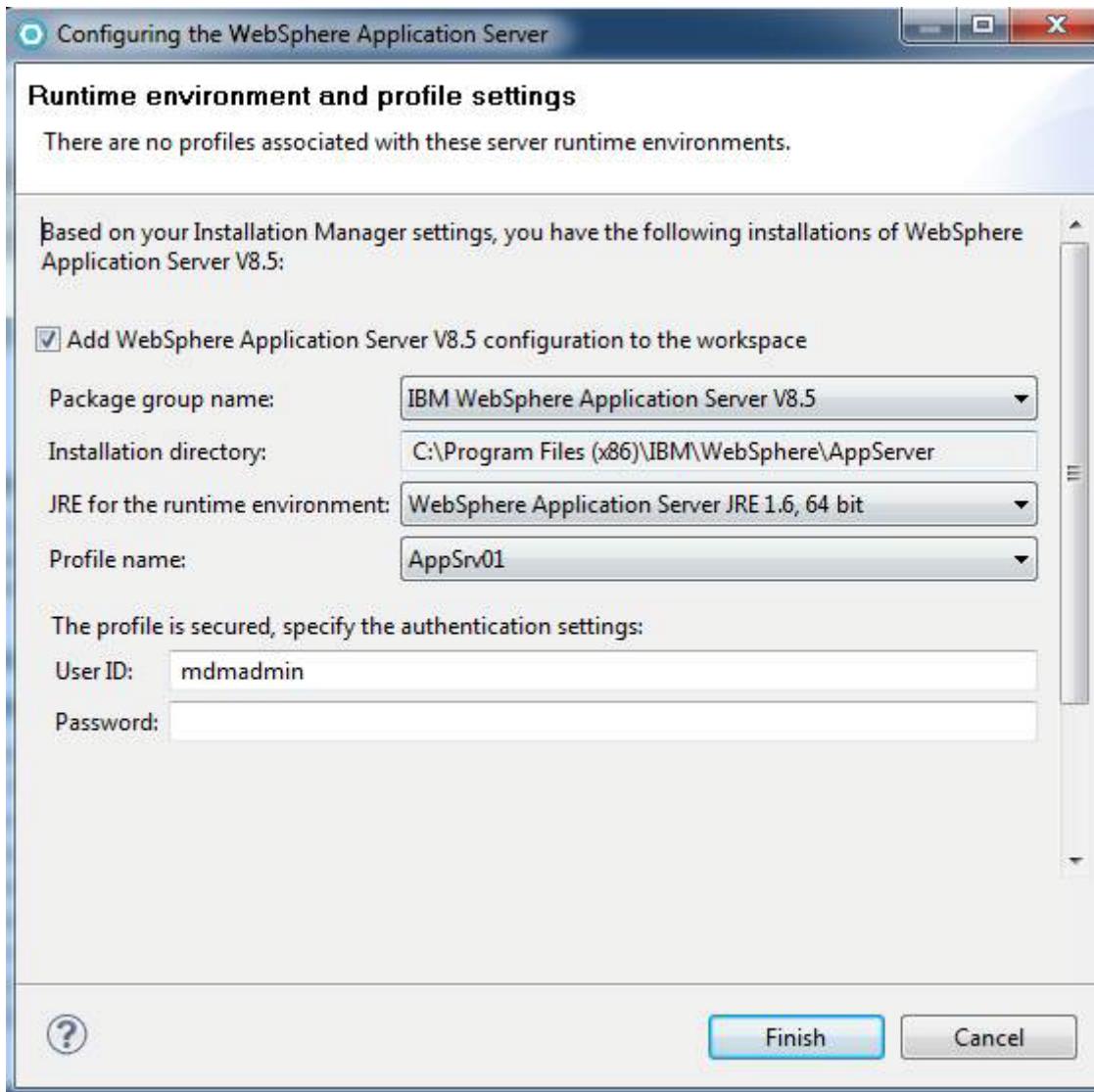
Appendix A. MDM Virtual Data Model

Part 1: Creating a new Virtual MDM Configuration project

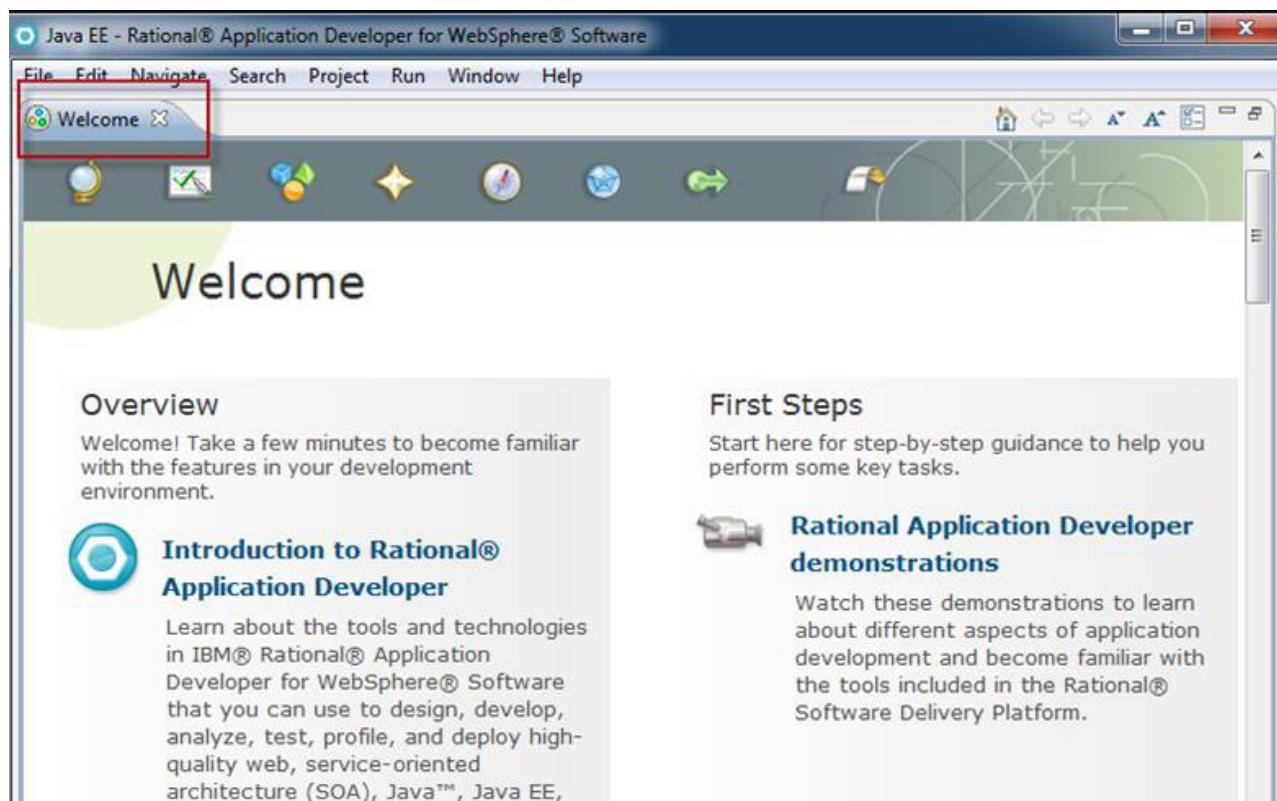
- 1. Make sure you are logged in as Administrator (password: passw0rd).
- 2. Start **Rational Application Developer** by clicking on the **ZZ880 Virtual Workspace** link on the desktop



- 3. When a brand new workspace is opened, the **Configuration the WebSphere Application Server** dialog will appear. Completing this dialog will create a connection to the WebSphere profile that has been installed on the virtual machine. Enter **mdmadmin** for the password and click the **Finish** button.



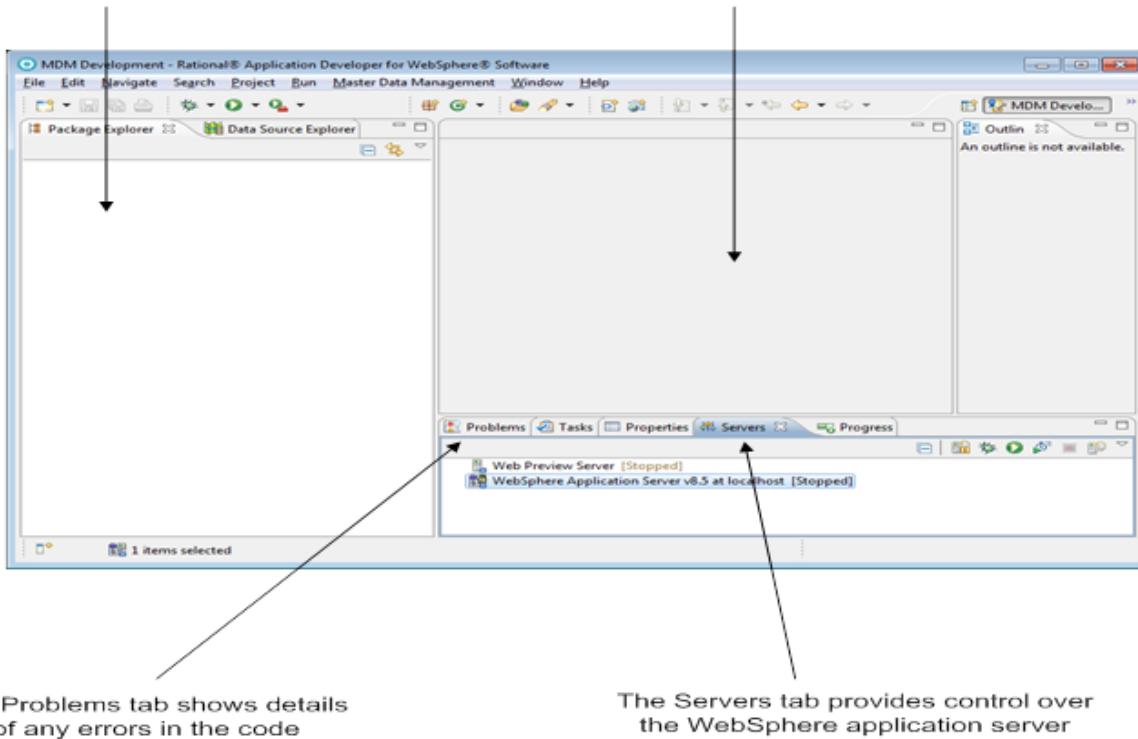
- ___ 4. Once the workspace opens, close the Welcome screen by clicking the X on the Welcome tab.



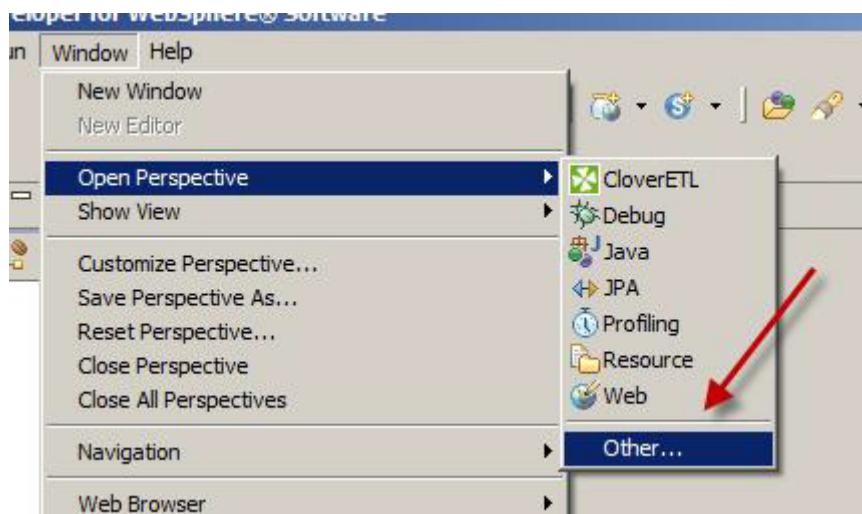
5. You are now ready to start developing a solution in the workspace. The diagram below shows the main components of the RAD or RSA workspace.

New "Development Projects" are created in here and contain models and code developed in the workspace

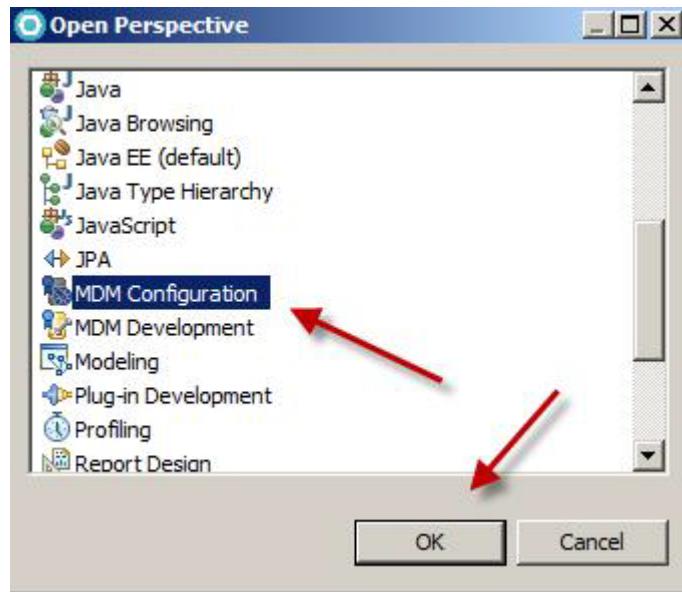
Resources being edited appear in the right handle panel



6. Next we will create an MDM configuration project. Switch to the **MDM Configuration** perspective by selecting **Window >Open Perspective >Other ...** from the RAD file menu



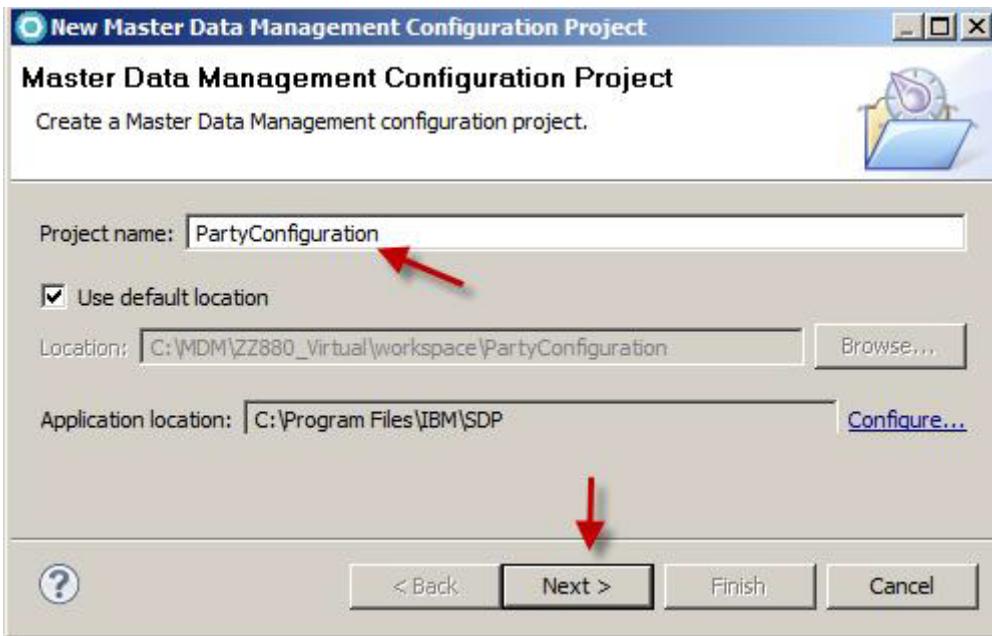
7. Select the **MDM Configuration** perspective from the list of available perspective and click the **OK** button.



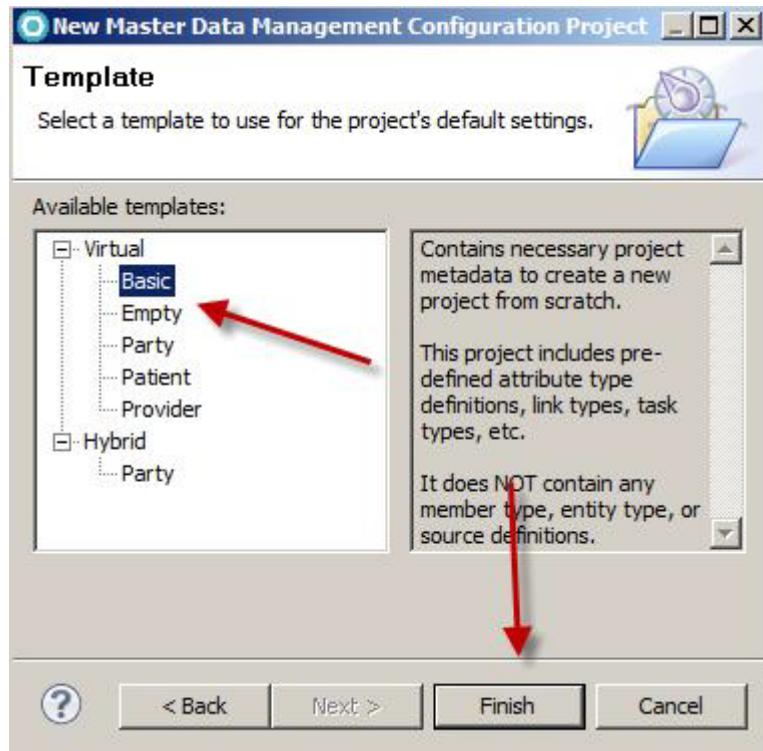
- 8. Now we can create our Configuration Project, from the RAD file menu, select **File > New > Configuration Project**



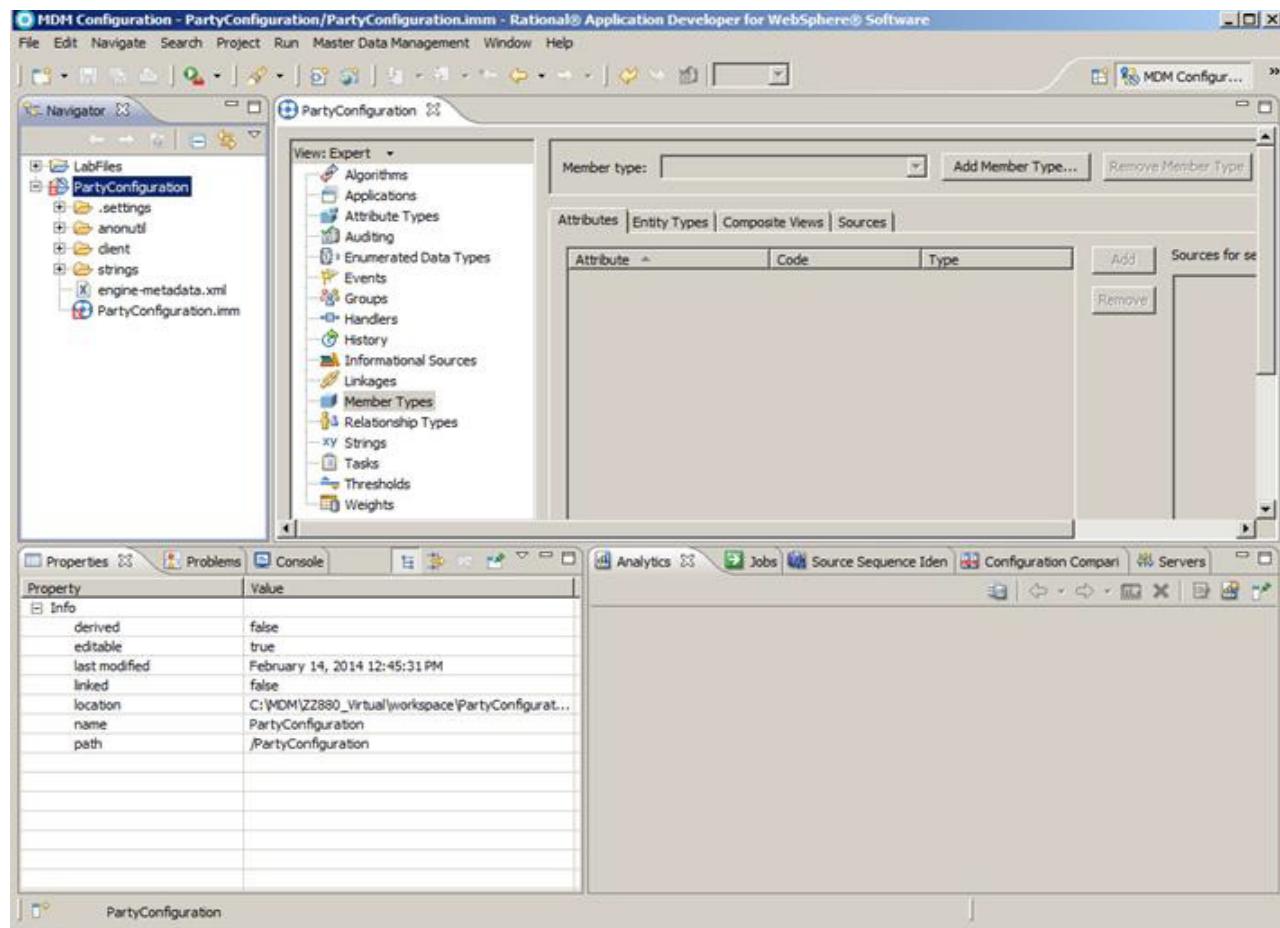
- 9. Enter **PartyConfiguration** as the Project Name, leave **Use default location** checked and click the **Next >** button.



- ___ 10. Select the **Basic** template and click the **Finish** button.

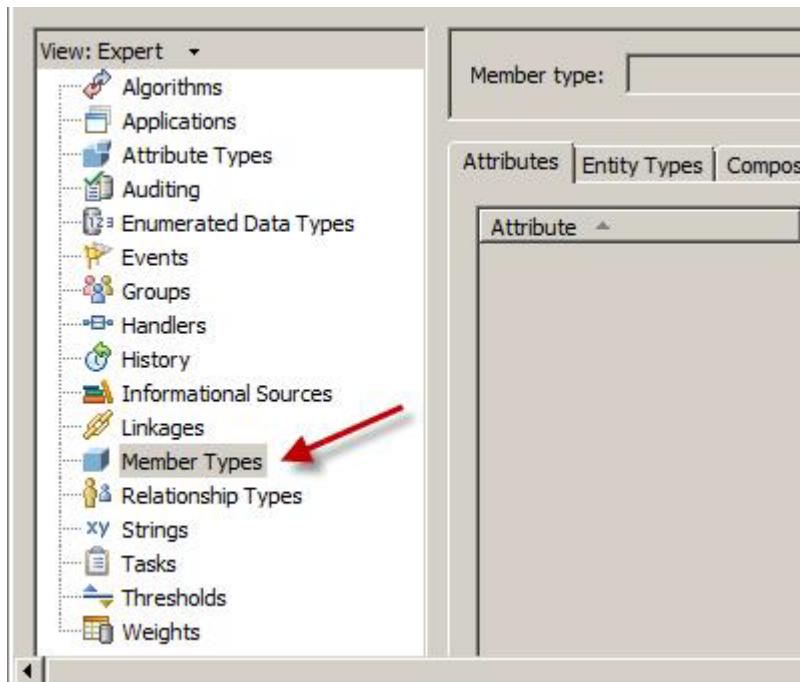


- ___ 11. Expand the **PartyConfiguration** project in the Navigator pane.

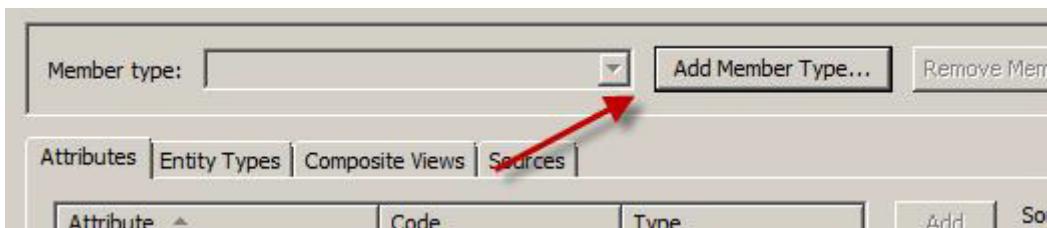


Part 2: Adding a new member type

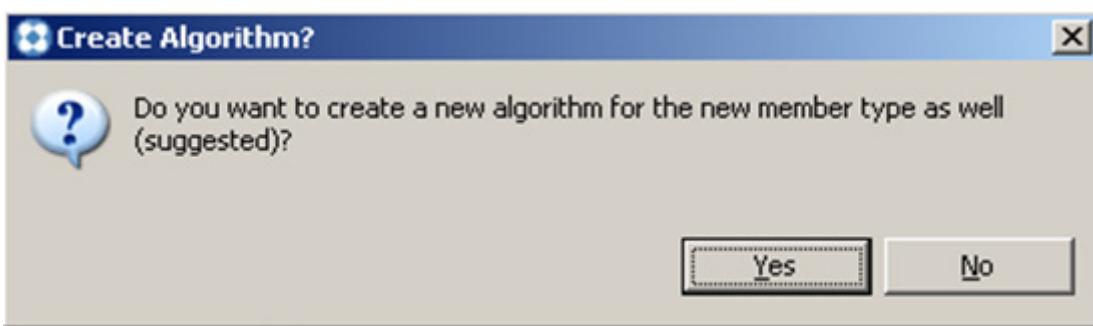
12. Select the **Member Types** tab in the Configuration pane – if not already highlighted.



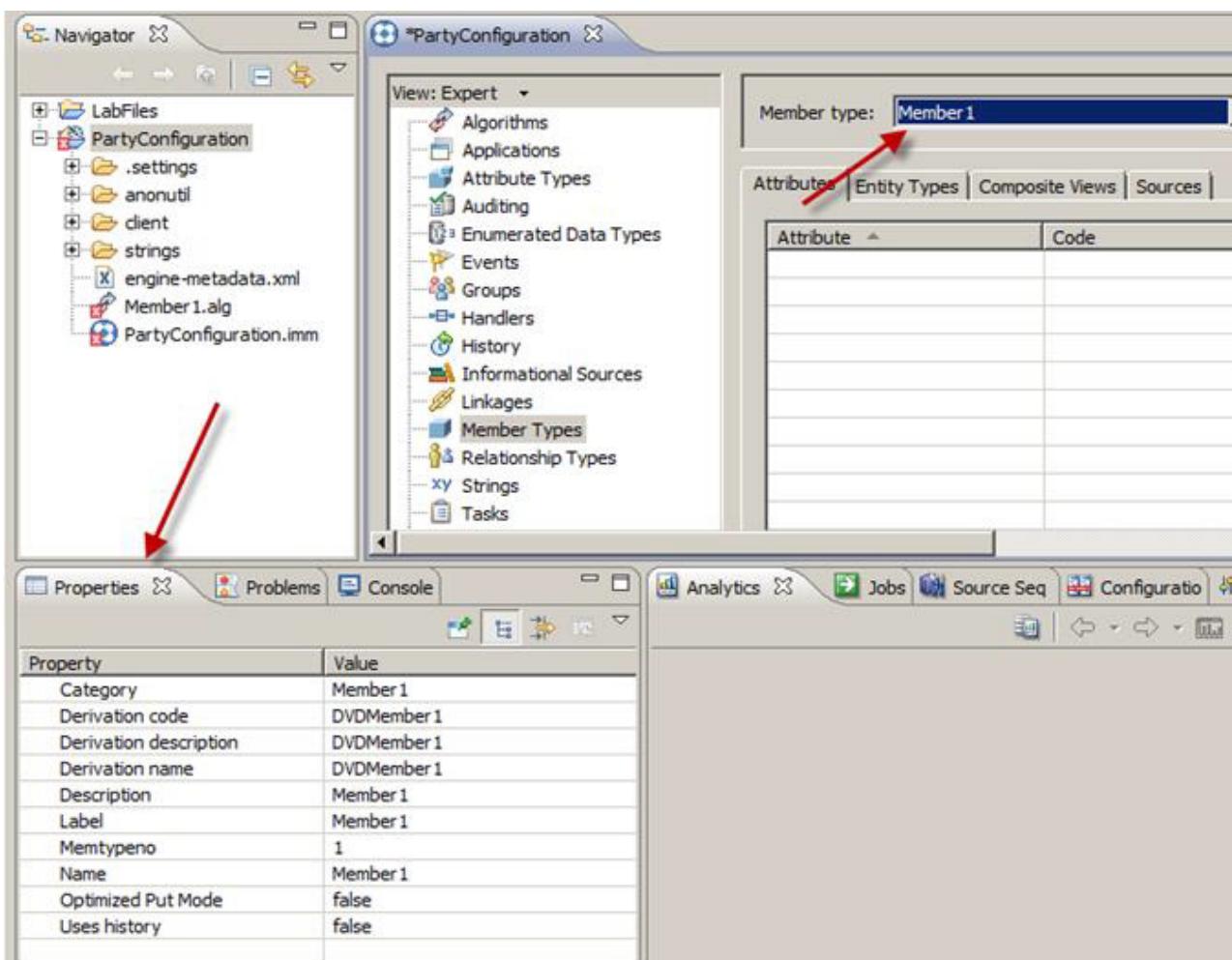
- ___ 13. Click the **Add Member Type...** button (to the right of the Member Type drop-down list) in the Editor pane to create a new member type.



- ___ 14. Click **Yes** when prompted to create new algorithm.



- ___ 15. Select **Member1** as the **Member Type** and open the RAD Properties window tab in the Information pane.

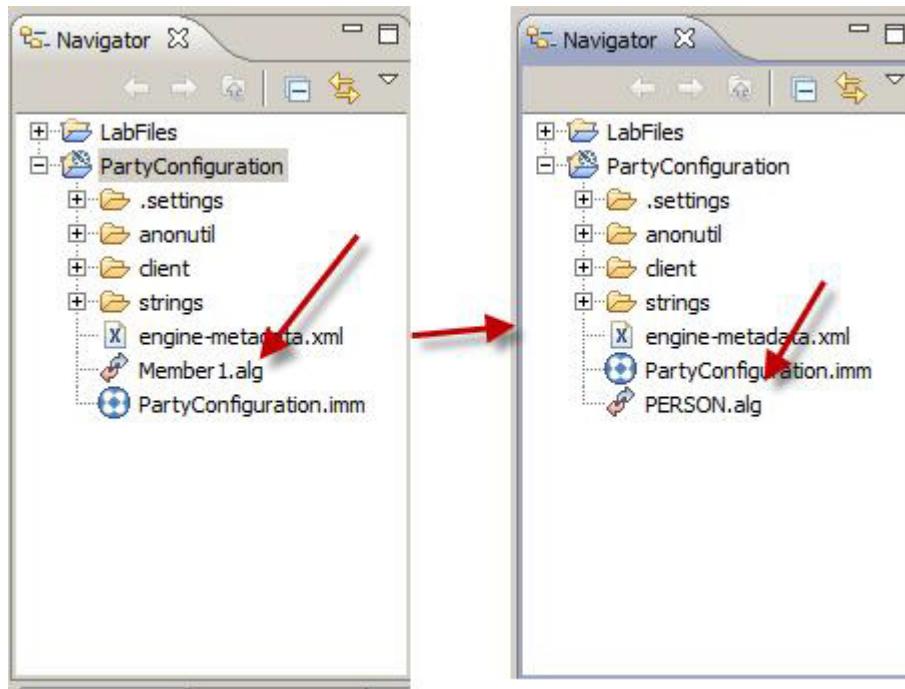


16. Change the following properties in the Properties tab:
- Category / Description / Label / Name values: **PERSON**.
 - Derivation code / Derivation description / Derivation name: **DVDPERSON**.

The screenshot shows the Eclipse IDE's Properties view. The 'Memtypeno' property is selected and has a value of '1'. Other properties listed include Category (PERSON), Derivation code (DVDPERSON), Derivation description (DVDPERSON), Derivation name (DVDPERSON), Description (PERSON), Label (PERSON), Name (PERSON), Optimized Put Mode (false), and Uses history (false).

Property	Value
Category	PERSON
Derivation code	DVDPERSON
Derivation description	DVDPERSON
Derivation name	DVDPERSON
Description	PERSON
Label	PERSON
Memtypeno	1
Name	PERSON
Optimized Put Mode	false
Uses history	false

17. To keep naming in sync, right-click **Member1.alg** in the navigator pane and rename to **PERSON.alg**.

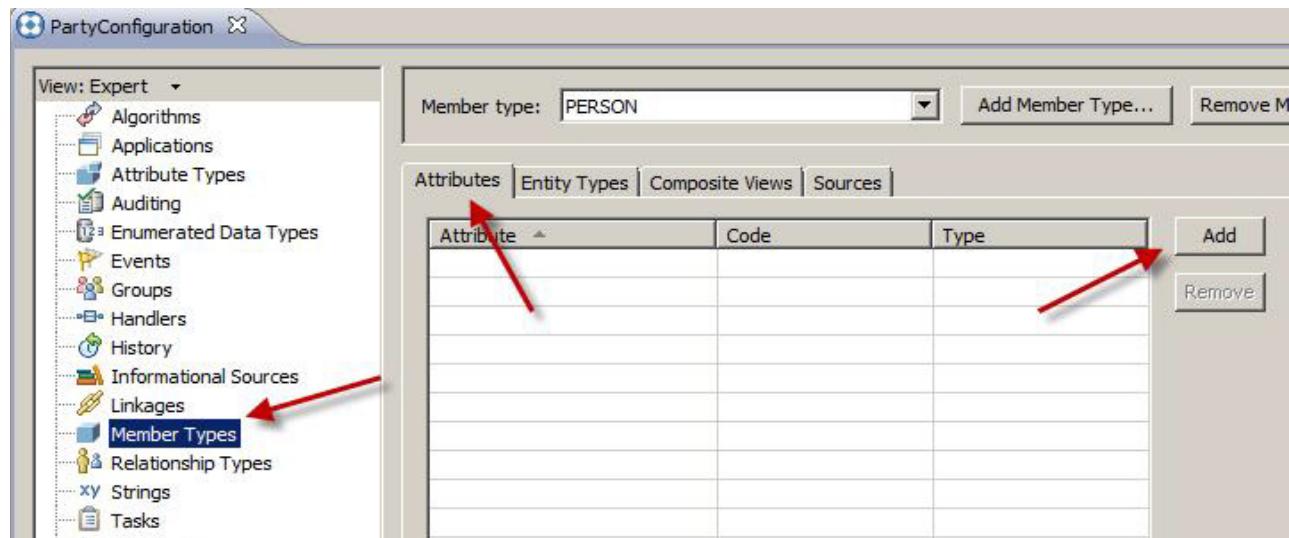


18. Click the Save button to save your project.

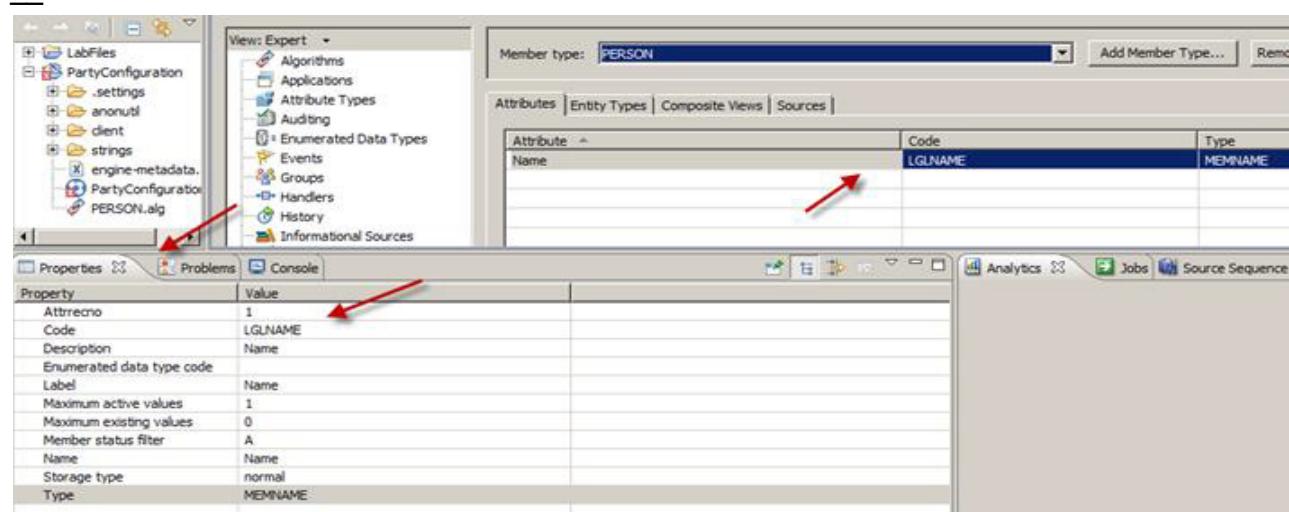
Part 3: Adding attributes

In this section we will add the attribute that belong to a Person member type.

- ___ 19. From the configuration file, under the Member Types section, select the **Attributes** tab in the Editor pane. Click the **Add** button (to the right of the fields) to add a new attribute.
- ___ 20.



- ___ 21. Under the RAD Properties window, change the following values for our new attribute:
- Code: **LGLNAME**.
 - Description / Label / Name: **Name**.
 - Type: **MEMNAME**.
- ___ 22.



- ___ 23. Repeat the previous steps to add the following attributes.
- Home Address**
 - Code: **HOMEADDR**
 - Description / Label / Name: **Home Address**.
 - Maximum active values: **1**

- Maximum existing values: **10**
- Type: **MEMADDR**.

The screenshot shows the 'Properties' tab of the IBM InfoSphere MDM interface. The table lists various properties and their values:

Property	Value
Attrrecno	2
Code	HOMEADDR
Description	Home Address
Enumerated data type code	
Label	Home Address
Maximum active values	1
Maximum existing values	10
Member status filter	A
Name	Home Address
Storage type	normal
Type	MEMADDR

b. **Gender**

- Code: **SEX**
- Description / Label / Name: **Gender**.
- Maximum active values: **1**
- Maximum existing values: **1**
- Type: **MEMATTR**.

The screenshot shows a software window with a toolbar at the top containing icons for Properties, Problems, Console, and other functions. The main area is a table titled 'Properties' with two columns: 'Property' and 'Value'. The properties listed are Attrrecno (3), Code (SEX), Description (Gender), Enumerated data type code, Label (Gender), Maximum active values (1), Maximum existing values (1), Member status filter (A), Name (Gender), Storage type (normal), and Type (MEMATTR). The 'Attrrecno' row is highlighted in blue.

Property	Value
Attrrecno	3
Code	SEX
Description	Gender
Enumerated data type code	
Label	Gender
Maximum active values	1
Maximum existing values	1
Member status filter	A
Name	Gender
Storage type	normal
Type	MEMATTR

___ c. National ID Number

- Code: **NATID**
- Description / Label / Name: **National ID Number**.
- Maximum active values: **1**
- Maximum existing values: **1**
- Type: **MEMIDENT**.

The screenshot shows a software window with a toolbar at the top containing icons for Properties, Problems, Console, and other functions. The main area is a table titled 'Properties' with two columns: 'Property' and 'Value'. The properties listed are Attrrecno (4), Code (NATLID), Description (National ID Number), Enumerated data type code, Label (National ID Number), Maximum active values (1), Maximum existing values (1), Member status filter (A), Name (National ID Number), Storage type (normal), and Type (MEMIDENT). The 'Attrrecno' row is highlighted in blue.

Property	Value
Attrrecno	4
Code	NATLID
Description	National ID Number
Enumerated data type code	
Label	National ID Number
Maximum active values	1
Maximum existing values	1
Member status filter	A
Name	National ID Number
Storage type	normal
Type	MEMIDENT

___ d. Birth Date

- Code: **BIRTHDT**
- Description / Label / Name: **Birth Date**.

- Maximum active values: **1**
- Maximum existing values: **1**
- Type: **MEMDATE**.

The screenshot shows the 'Properties' tab of the IBM InfoSphere MDM interface. The table lists the following attributes:

Property	Value
Attrrecno	5
Code	BIRTHDT
Description	Birth Date
Enumerated data type code	
Label	Birth Date
Maximum active values	1
Maximum existing values	1
Member status filter	A
Name	Birth Date
Storage type	normal
Type	MEMDATE

__ e. **Home Phone**

- Code: **HOMEPHON**
- Description / Label / Name: **Home Phone**
- Maximum active values: **1**
- Maximum existing values: **10**
- Type: **MEMPHONE**.

The screenshot shows the SAP Data Services interface with the 'Properties' tab selected. The table lists the following properties for an attribute:

Property	Value
Attrrecno	6
Code	HOMEPHON
Description	Home Phone
Enumerated data type code	
Label	Home Phone
Maximum active values	1
Maximum existing values	10
Member status filter	A
Name	Home Phone
Storage type	normal
Type	MEMPHONE

___ f. Mobile Phone

- Code: **MOBILPHON**
- Description / Label / Name: **Mobile Phone**
- Maximum active values: **1**
- Maximum existing values: **10**
- Type: **MEMPHONE**.

The screenshot shows the SAP Data Services interface with the 'Properties' tab selected. The table lists the following properties for an attribute:

Property	Value
Attrrecno	7
Code	MOBILEPHON
Description	Mobile Phone
Enumerated data type code	
Label	Mobile Phone
Maximum active values	1
Maximum existing values	10
Member status filter	A
Name	Mobile Phone
Storage type	normal
Type	MEMPHONE

___ 24. Click the **Save** button to save your project.

Attribute	Code	Type
National ID Number	NATID	MEMIDENT
Name	LGLNAME	MEMNAME
Home Phone	HOMEPHON	MEMATTR
Home Address	HOMEADDR	MEMADDR
Gender	SEX	MEMATTR
Birth Date	BIRTHDT	MEMDATE
Mobile Phone	MOBILEPHON	MEMPHONE

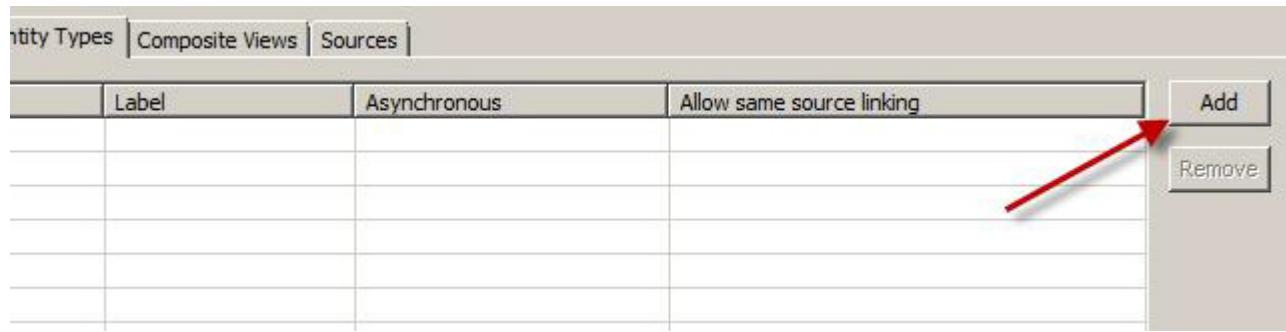
Part 4: Adding the Entity Types

In this part we will define our Entity Types for our Members.

- ___ 25. Click the **Entity Types** tab in the Editor pane and select **PERSON** in the Member Type drop-down list if it is not already selected.

Member type: PERSON

- ___ 26. Click the **Add** button (to the right of the fields) to add a new entity type.



- ___ 27. Under the **Properties** window, enter the following values for our new Entity Type:
- Entity Type: **id**.
 - Allow same source linking / Asynchronous: **true**.
 - Category / Description / Label: **Identity**.
 - Comparison Code: **CMPID**.
 - Comparison Description / Comparison Name: **Identity Comparison**.

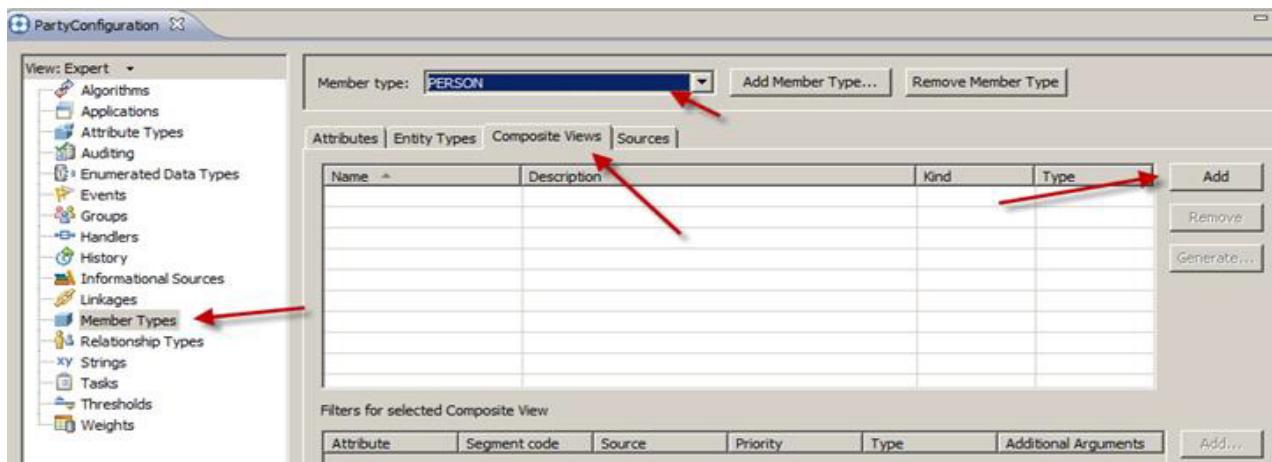
Property	Value
Allow same source linking	true
Asynchronous	true
Category	Identity
Comparison code	CMPID
Comparison description	Identity Comparison
Comparison name	Identity Comparison
Description	Identity
Entity Type	id
Enttypeno	1
Label	Identity
Maximum bucket role	0
Maximum candidate count for dynam	0
Member status filter	AO
Minimum bucket role	0
Transitive	true
Uses an input queue	true
Uses linkage history	true
Uses linkages	true
Uses notes	true
Uses rules	true
Uses tasks	true

- ___ 28. Click the **Save** button to save your project.

Part 5: Adding composite views

In this part, we will add 2 new composite view to our member and entities.

- ___ 29. Under the Party Configuration, under the **Member Types**, select the **Composite Views** tab in the Editor pane. Select **PERSON** in the Member Type drop-down list if it is not already selected. Click the **Add** button (to the right of the fields) to add a new composite view.



- 30. Select the **Properties** window and change the enter the following values for our new Composite View:

- __ a. Name: **EMCA**.
- __ b. Description: **Entity Most Current Attribute**.
- __ c. Kind: **Entity**.
- __ d. Number of current attributes to return: **1**.
- __ e. Number of results to return: **20**.

Name	Description	Kind	Type
EMCA	Entity Most Current Attribute	Entity	Standard

Property	Value
Additional arguments	
Cvwirecno	1
Description	Entity Most Current Attribute
Kind	Entity
Most current attribute date type	Operational Server
Name	EMCA
Number of current attributes to return	1
Number of results to return	20
Record status filter	A
Restricted	false

- 31. Click **Add** button again to add a second composite view with the following values

- __ a. Name: **MMCA**.
- __ b. Description: **Member Most Current Attribute**.
- __ c. Kind: **Member**.
- __ d. Number of current attributes to return: **1**.
- __ e. Number of results to return: **20**.

Name	Description	Kind	Type
MMCA	Member Most Current Attribute	Member	Standard
EMCA	Entity Most Current Attribute	Entity	Standard

Property Grid:

- Additional arguments: Cvirecno = 2
- Description: Member Most Current Attribute
- Kind: Member
- Most current attribute date type: Operational Server
- Name: MMCA
- Number of current attributes to return: 1
- Number of results to return:** 20
- Record status filter: A
- Restricted: false

- 32. Click the **Save** button to save your project.

Part 6: Adding definitional sources

The sources are the systems where the member or stored and maintained. We will add 2 sources to our project, our member will come from one of these 2 sources.

- 33. Under the **Member Types**, select the Sources tab. Click the **Add** button (to the right of the fields) to add a new source.

Source Name	Source Code

- 34. Inside the RAD **Properties** windows, add the following values:

- Source code / Physical code: **A**.
- Enable Review Identifier Task: **Yes**.
- Source name: **Archway**.

The screenshot shows the IBM InfoSphere MDM interface. On the left, there's a navigation tree with items like Auditing, Events, Groups, Handlers, and History. The main area has tabs for Markers, Properties, Servers, Data Source Explorer, Snippets, Annotations, and Console. Below the tabs is a table with two columns: Source Name and Source Code. The first row has 'Source Name' as 'Archway' and 'Source Code' as 'A'. Below this table is a 'Property' table with rows for various properties and their values. Red arrows point from the following property values to the corresponding cells in the table above:

Property	Value
Default Entity Priority	100
Enable Review Identifier Task	Yes
Is virtual	No
Physical code	A
Potential overlay comparison code	
Potential overlay score	0
Record status	Active
Source code	A
Source name	Archway
Srcrecno	1

35. Click the **Add** button again to add a second source with the following values:

- __ a. Source code / Physical code: **B**.
- __ b. Enable Review Identifier Task: **Yes**.
- __ c. Source name: **Bellwood**.

The screenshot shows the IBM InfoSphere MDM interface. The left navigation tree includes Auditing, Events, Groups, Handlers, and History. The main area displays a table with two rows: 'Archway' and 'Bellwood'. Below this is a 'Property' table with various properties and their values. Red arrows point from the following property values to the corresponding cells in the table above:

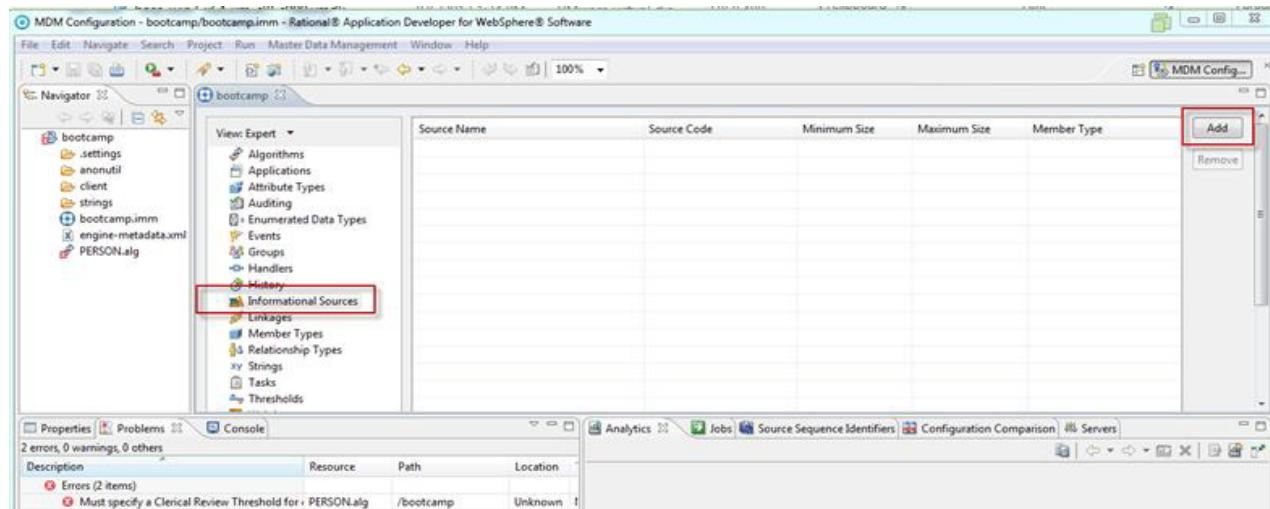
Property	Value
Default Entity Priority	100
Enable Review Identifier Task	Yes
Is virtual	No
Physical code	B
Potential overlay comparison code	
Potential overlay score	0
Record status	Active
Source code	B
Source name	Bellwood
Srcrecno	2

36. Click the **Save** button to save your project.

Part 7: Adding informational sources

Informational sources are thought of as “look up” data sources, typically providing “legal” values used to identify certain member attributes (for example, state codes, credit card number and issuer or frequent customer number and issuer).

- 37. Under the PartyConfiguration, select the **Informational Sources** tab in the Configuration pane. Click the Add button to the right.



- 38. Click the **Properties** tab in the Information pane and enter the following information:
- Source code / Physical code: **NIA**.
 - Max length / Min length: **9**.
 - Source name: **National ID Administration**.

Property	Value
Default Entity Priority	100
Enable Review Identifier Task	Yes
Max length	9
Member type	All
Min length	9
Physical code	NIA
Record status	Active
Source code	NIA
Source name	National ID Administration
Srcrecno	3

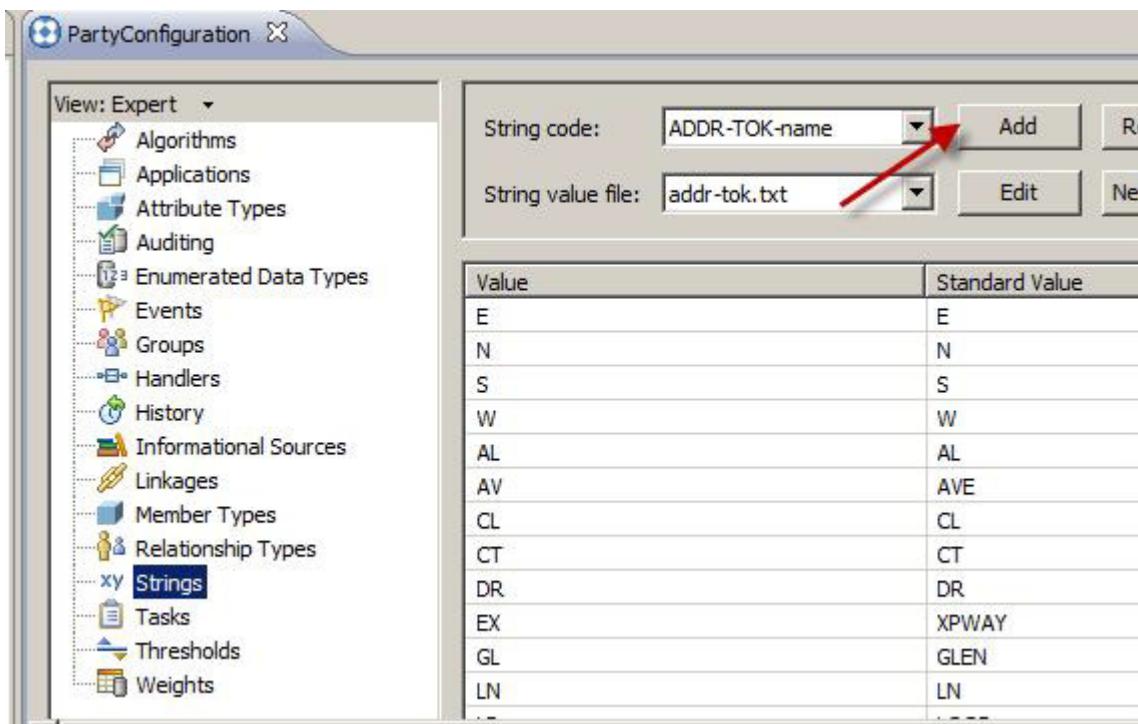
- 39. Click the **Save** button to save your project.

Part 8: Incorporating strings

Strings enable you to create “rules” or “guidelines” that instruct the algorithm on how to handle certain incoming data values. A common use of Strings is to provide a list of values that are Equivalent (e.g. Robert and Bob) for comparison or a list of Anonymous values to not be evaluated during standardization (e.g. 111111111). We will add some Strings to our project that have already been created for us.

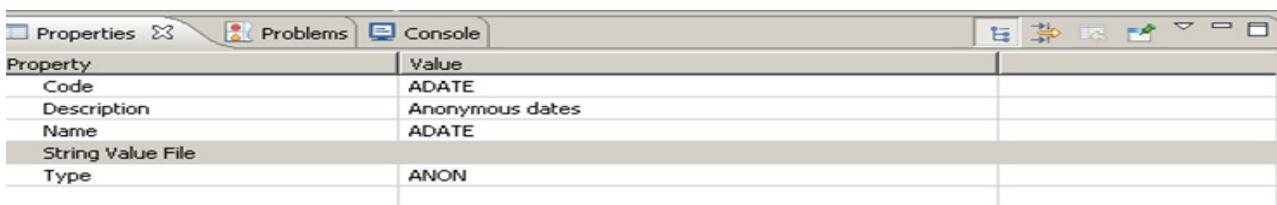
First we'll start by adding some anonymous dates to our configuration

- ___ 40. Under the Party Configuration, Select the **Strings** tab in the Configuration pane. Click the Add button to the right of the String code field to add a new String code.

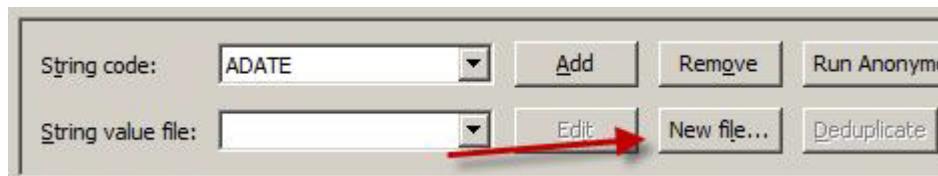


- ___ 41. Under the **Properties** window, add the following values:

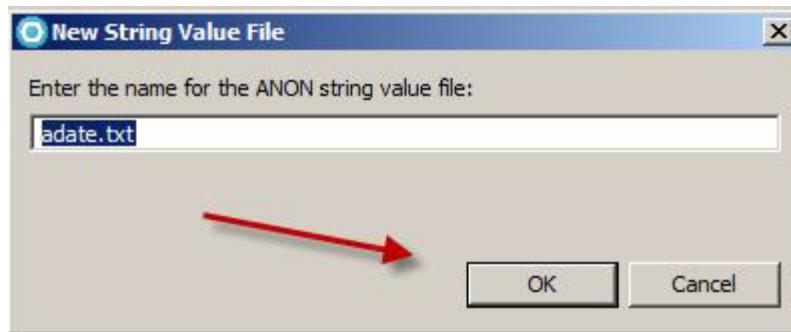
- ___ a. Code / Name: **ADATE**.
- ___ b. Description: **Anonymous dates**.
- ___ c. Type: **ANON**.



- ___ 42. In the Editor pane, click the **New file** button to the right of the String value file field



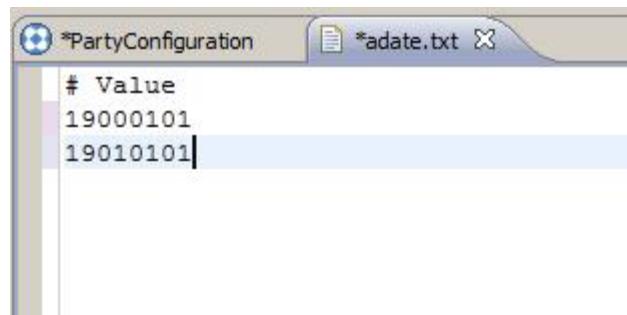
- ___ 43. Accept the default file name and click the **OK** button.



- ___ 44. Click the **Edit** button to the right of the String value file field to open a text editor.

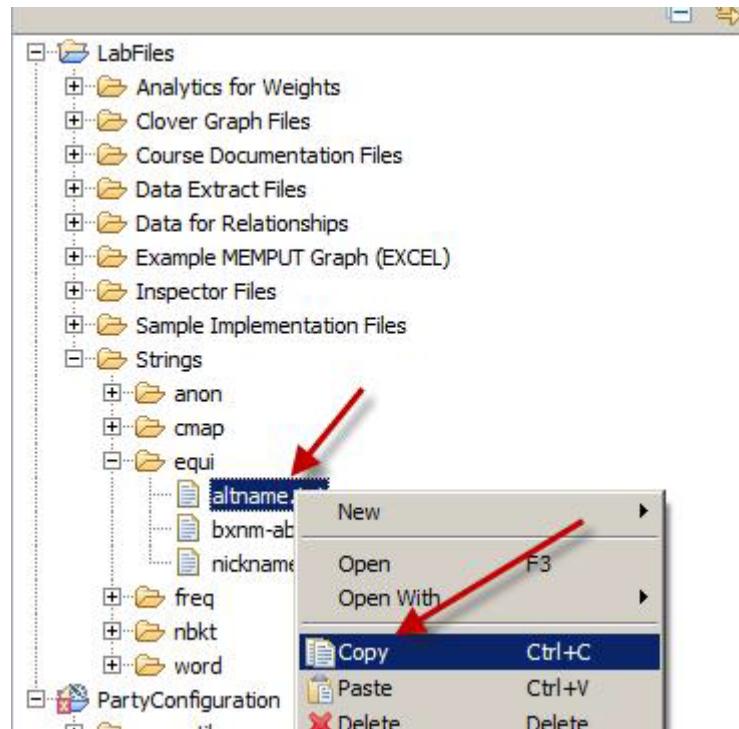


- ___ 45. Add to values to the file (19000101 and 19010101) and save the file. Close the editor tab by clicking the "x" button.

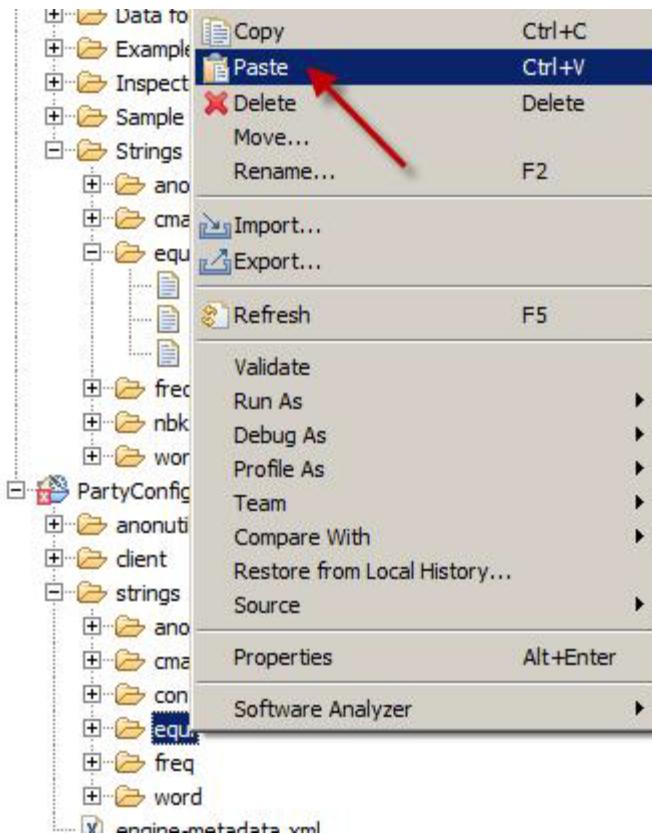


To save some time, we have provided the String files that we will be using in our exercise.

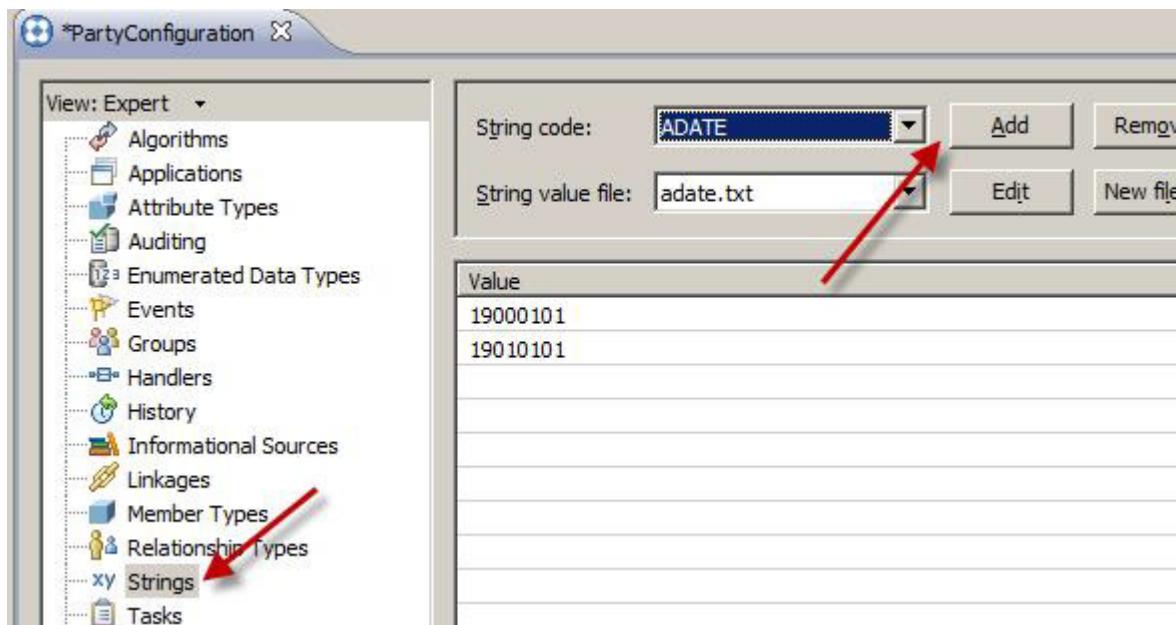
- ___ 46. Inside the RAD environment you will find a LabFiles project under the Enterprise Explorer. Right click on the **altname.txt** (found under Strings > equi) and select **Copy**.



47. Under the PartyConfiguration, right click on the **strings > equi** folder and select Paste.

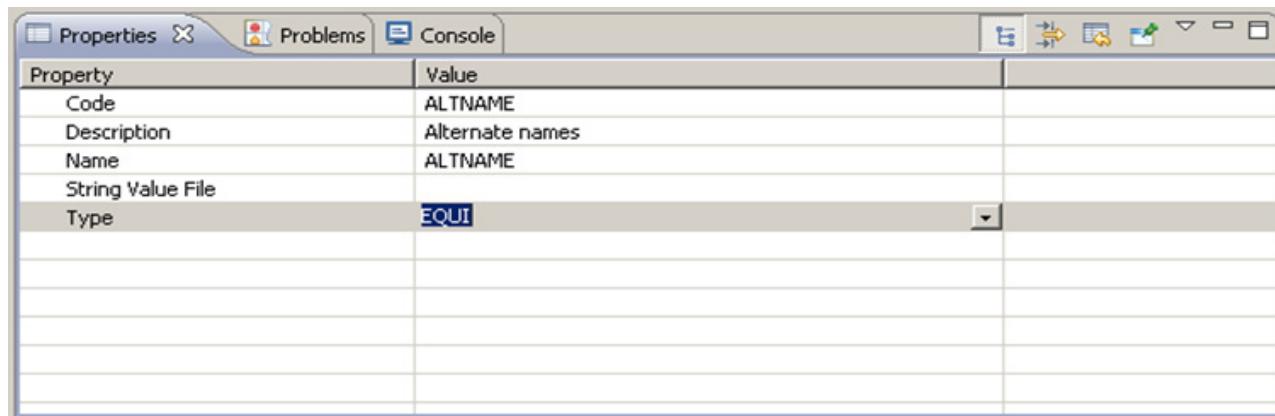


48. From the Party Configuration file, select the **Strings** tab. Click the **Add** button to the right of the String code field.

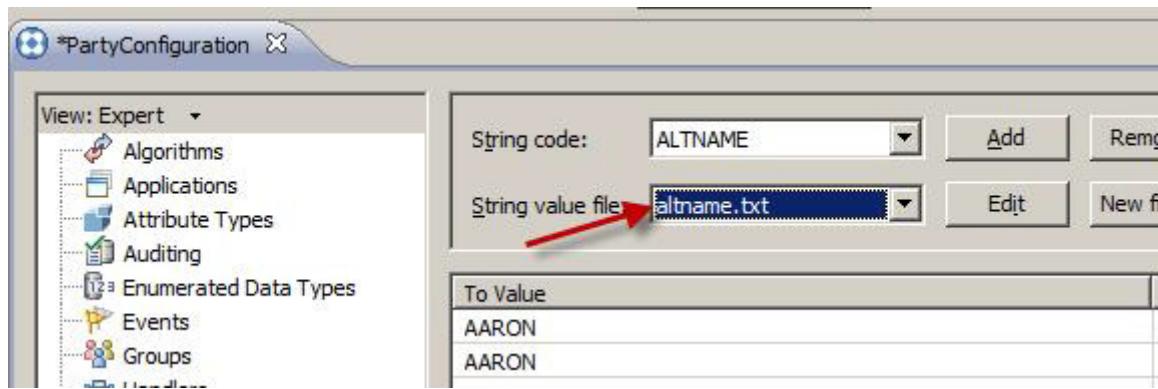


___ 49. Under the RAD Properties window, enter the following values:

- ___ a. Code / Name: **ALTNAMES**.
- ___ b. Description: **Alternate names**.
- ___ c. Type: **EQUI**.



___ 50. In the Editor pane, expand the **String value file** list and select **altnames.txt**.



- ___ 51. Repeat the previous steps to define an anonymous values file for the PartyConfiguration (file can be copied from LabFiles > Strings > anon > **anonname.txt**). Use the following values when defining the String file in the PartyConfiguration:
- Code / Name: **ANONNAME**.
 - Description: **Anonymous names**.
 - Type: **ANON**.

The screenshot shows the 'PartyConfiguration' window. The 'String code' is now 'ANONNAME'. The 'String value file' dropdown is set to 'anonname.txt', indicated by a red arrow. Below, the 'Value' table lists 'XXX', 'ZZZ', 'GIRL1', and 'GIRL2'. At the bottom, the properties table shows:

Property	Value
Code	ANONNAME
Description	Anonymous Names
Name	ANONNAME
String Value File	anonname.txt
Type	ANON

A red arrow also points to the 'Description' row in the properties table.

- ___ 52. Click the **Save** button to save your project.

At the end of this lab, you will see two errors. Both errors are related to the PERSON algorithms. Since we haven't configured the any algorithms for the person member type, it is OK to proceed to the next lab.

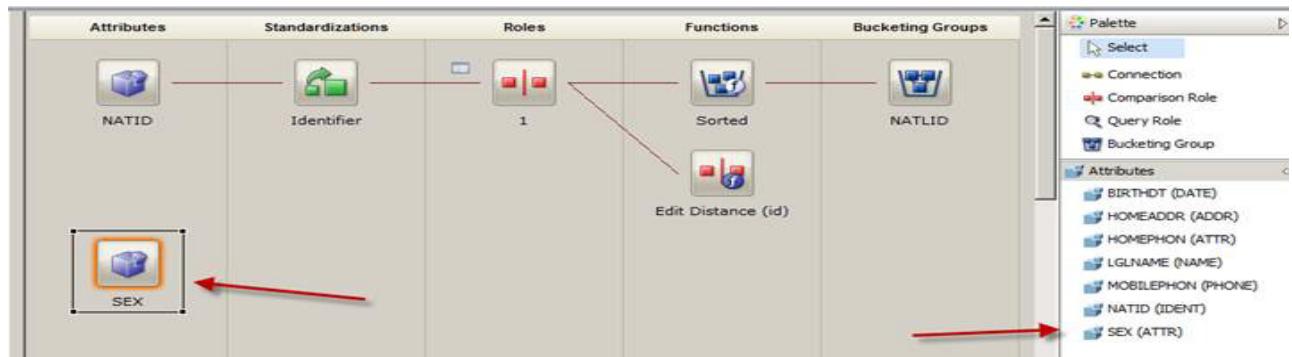
2 errors, 0 warnings, 0 others					
Description	Resource	Path	Location	Type	
Errors (2 items)					
Must specify a Clerical Review Threshold for	PERSON.alg	/PartyConfiguration	Unknown	Master Data...	
Must specify an Autolink Threshold for	PERSON.alg	/PartyConfiguration	Unknown	Master Data...	

End of exercise

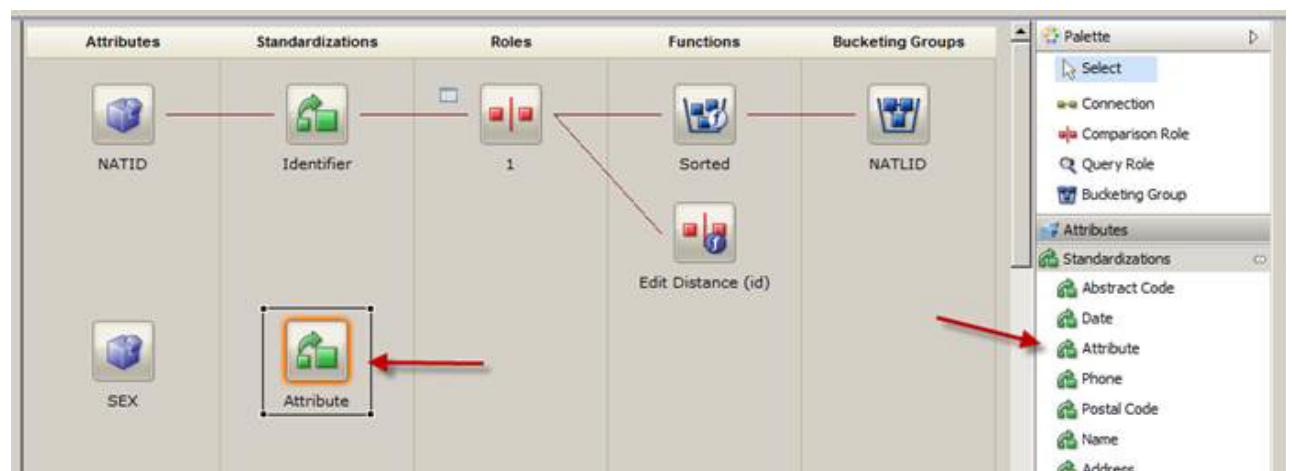
Appendix B. Algorithm Configuration

Part 1: Configuring the SEX Algorithm

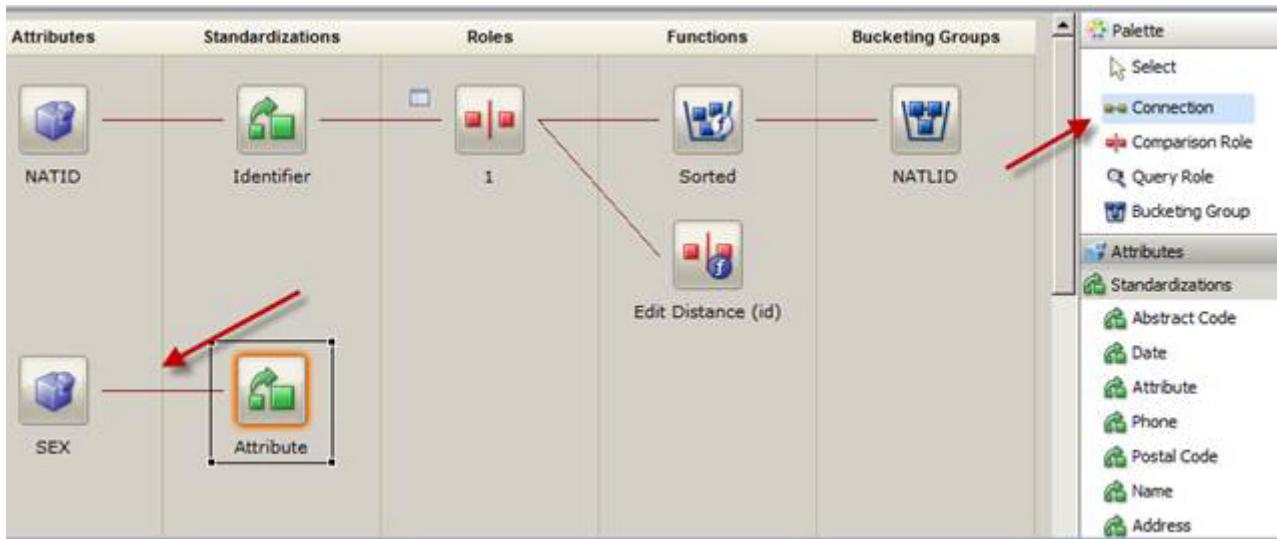
- 1. Expand the **Attributes** view in the palette. Select the SEX (**ATTR**) attribute and place the attribute under the first column.



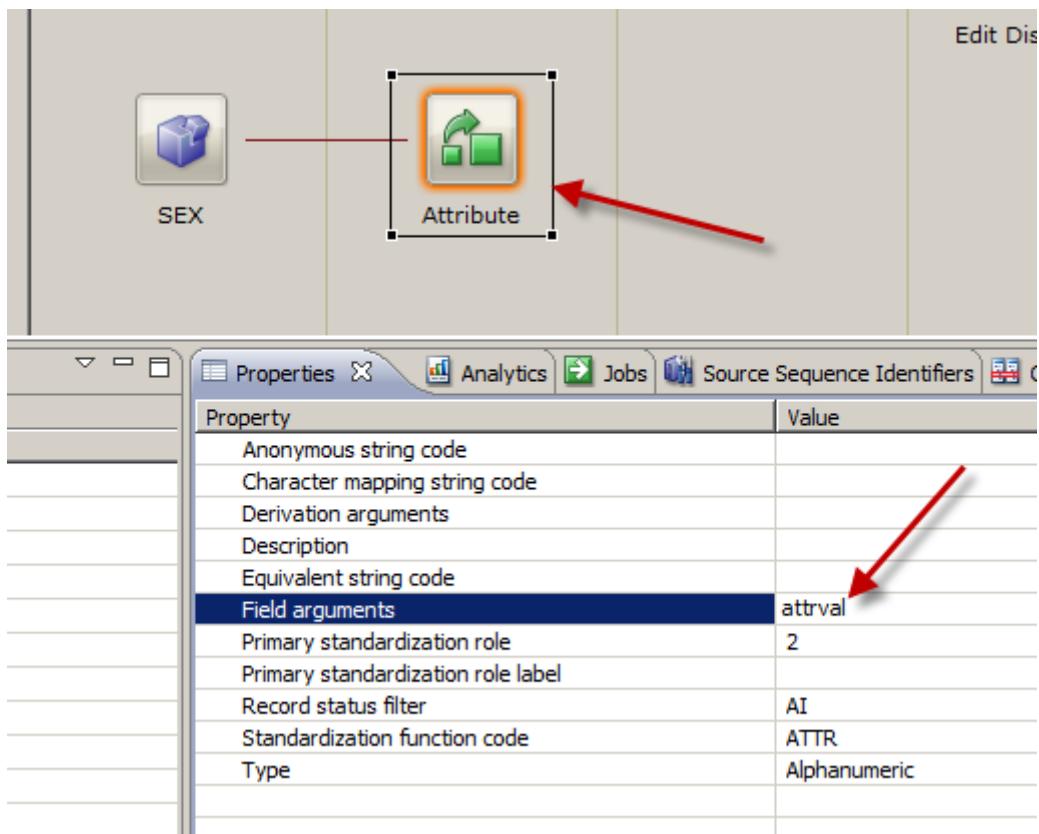
- 2. Expand the **Standardizations** view in the palette and select the Attribute icon. Place the Attribute standardization function in the 2nd column beside the SEX attribute.



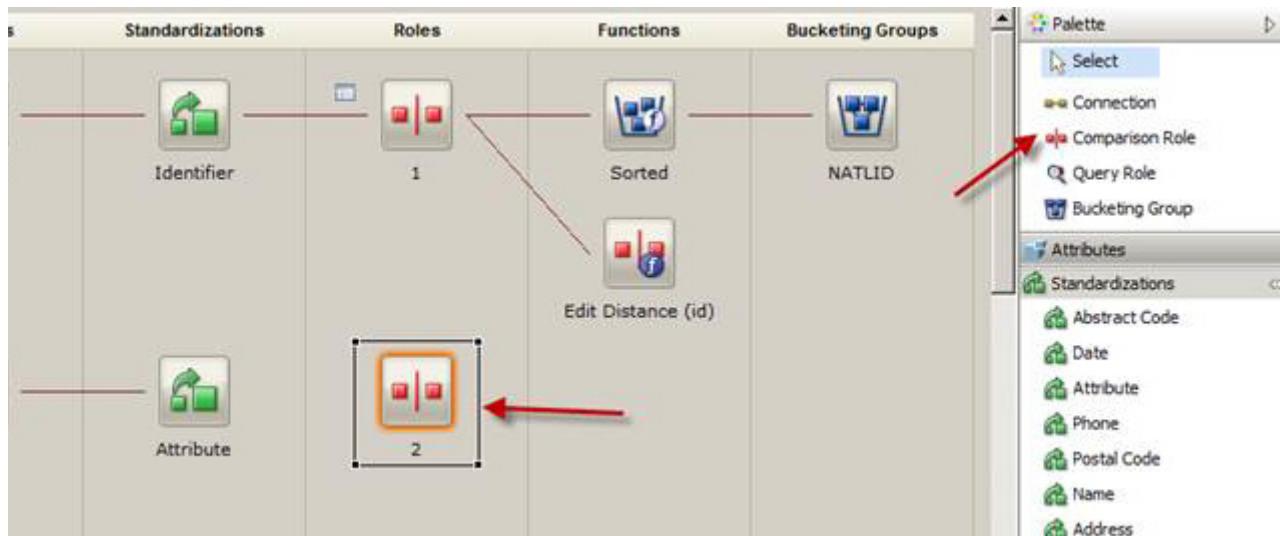
- 3. Connect the SEX attribute to the Attribute standardization function using the palette **Connection** tool.



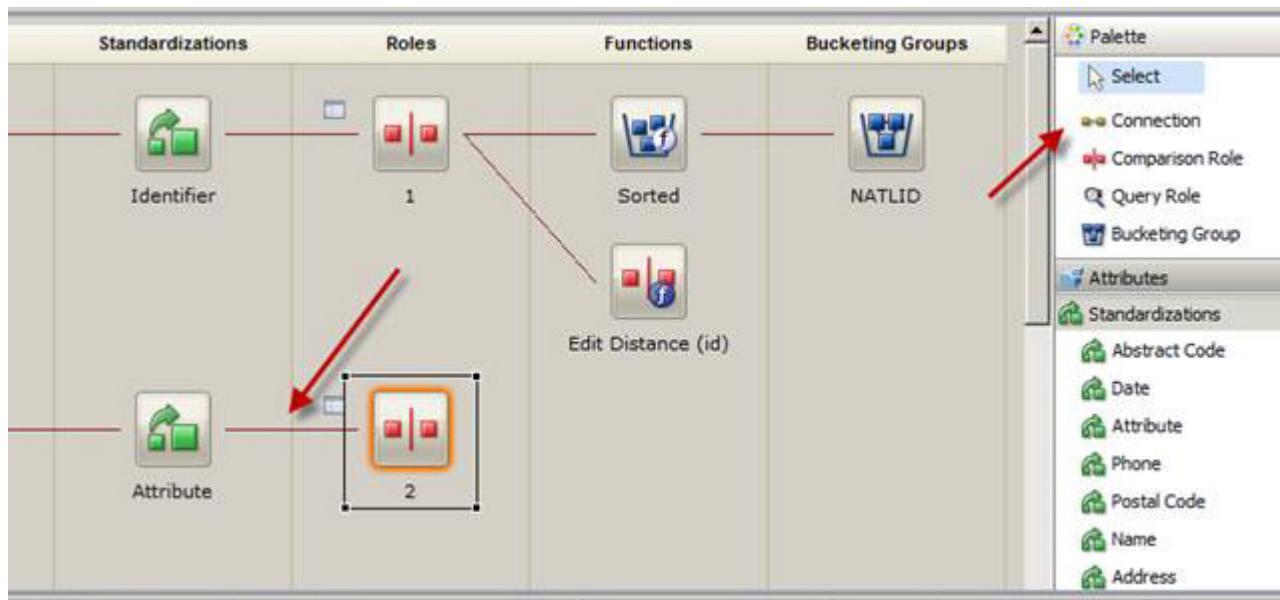
- ___ 4. Press the **ESC** key to exit the Connection tool.
- ___ 5. Select the **Attribute** icon and open the RAD Properties window. Change the following values:
 - ___ a. Field arguments: **1 - attrval**
 - ___ b. Record Status Filter: **AI**.
 - ___ c. Type: **Alphanumeric**.



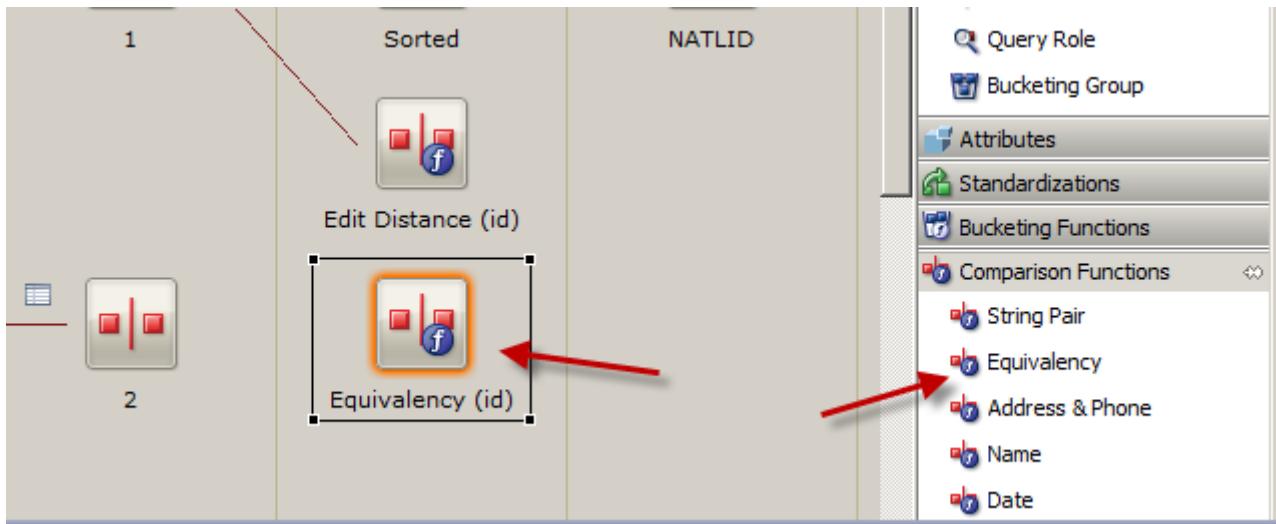
6. Select the **Comparison Role** icon in the palette and place it in the 3rd column.



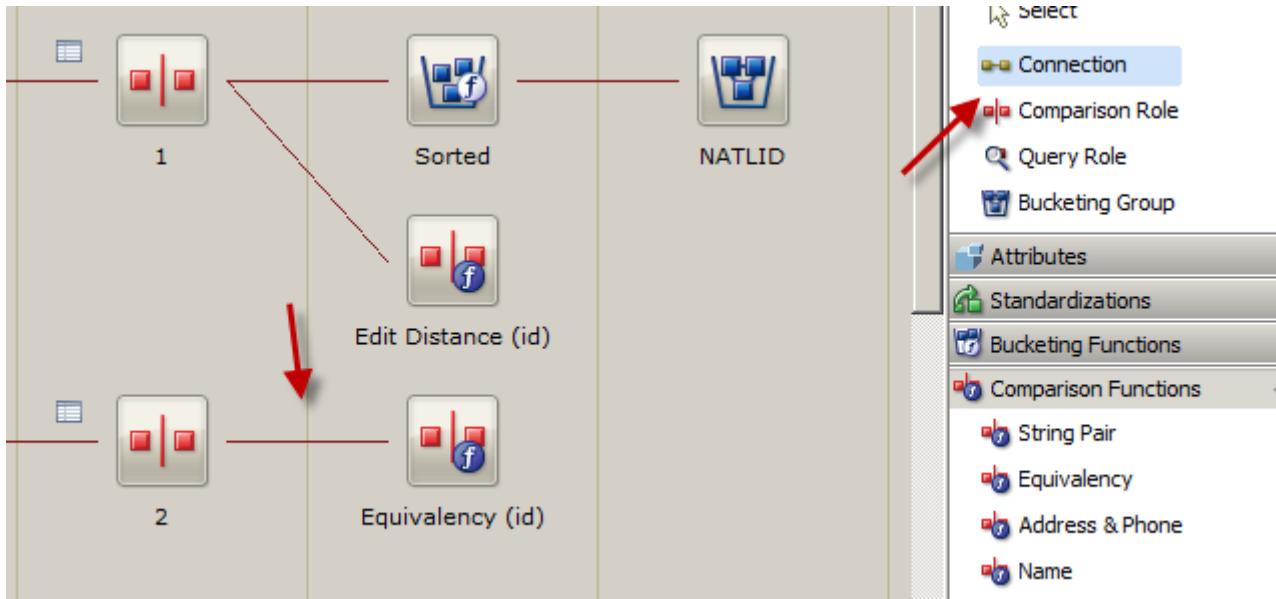
7. Connect the new Comparison Role to the Attribute standardization function using the palette's **Connection** tool.



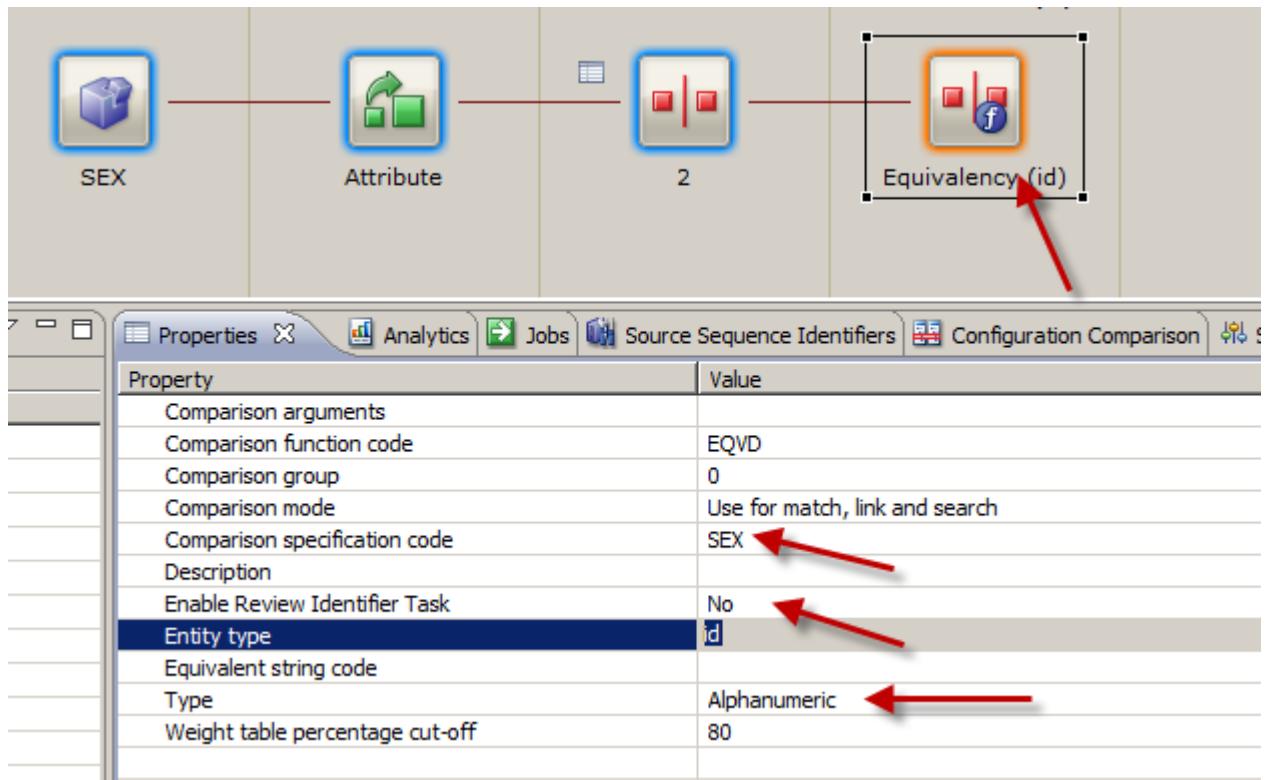
8. Press the **ESC** key to exit the connection tool.
 9. Expand the **Comparison Functions** view in the palette. Select the **Equivalency** comparison function and place it in the 4th row.



- ___ 10. Connect the **Comparison Role 2** to the **Equivalency (id)** icon using the palette's **Connection** tool.

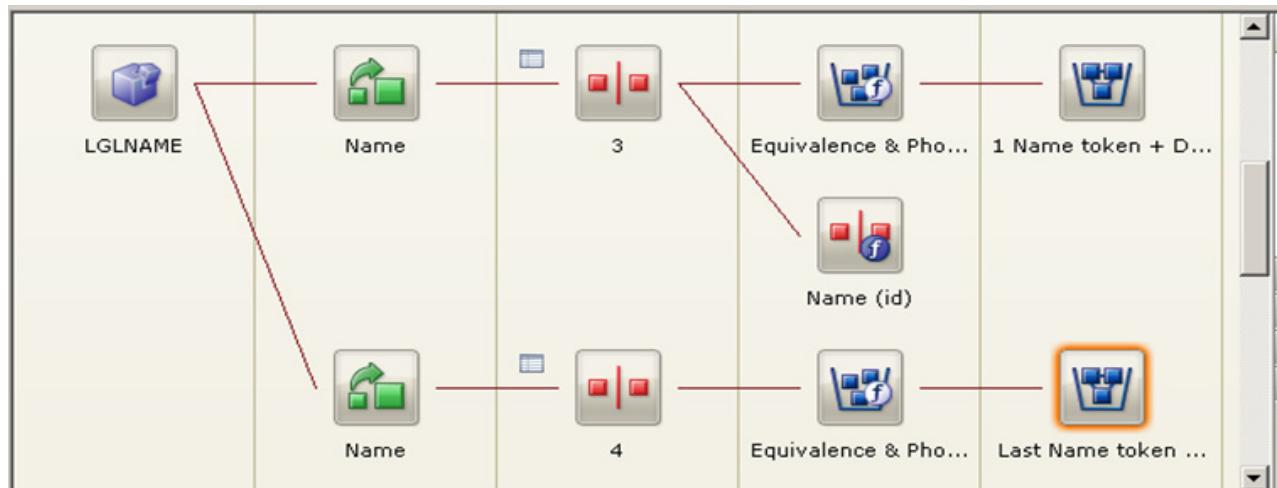


- ___ 11. Press the **ESC** key to exit the connection tool.
- ___ 12. Select the **Equivalency (id)** icon in the 4th column and open the RAD Properties window. Enter the following values:
- ___ a. Comparison Spec Code: **SEX**.
 - ___ b. Enable Review ID Task: **No**.
 - ___ c. Type: **Alphanumeric**.



- 13. Click the **Save** button to save your project.

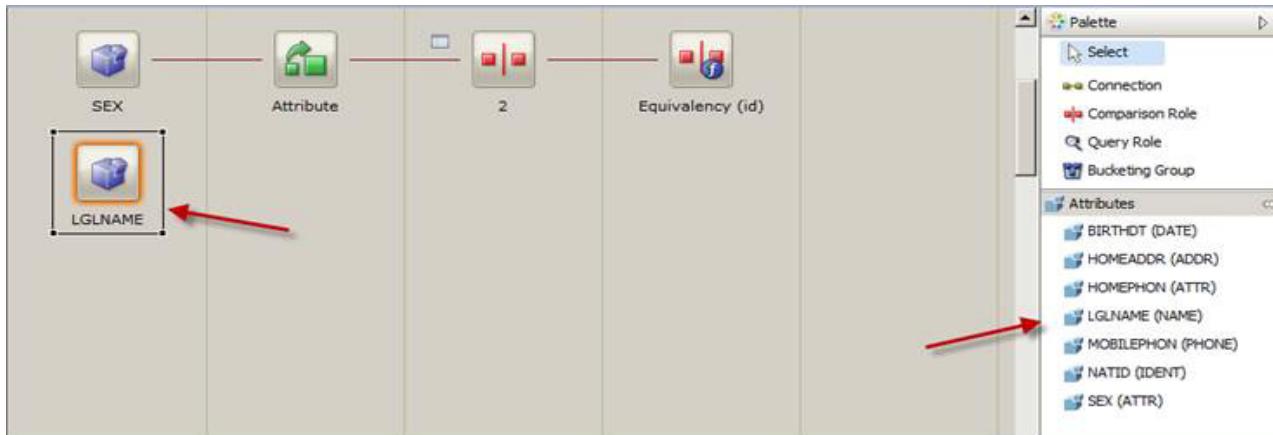
Part 2: Configure the legal name attribute



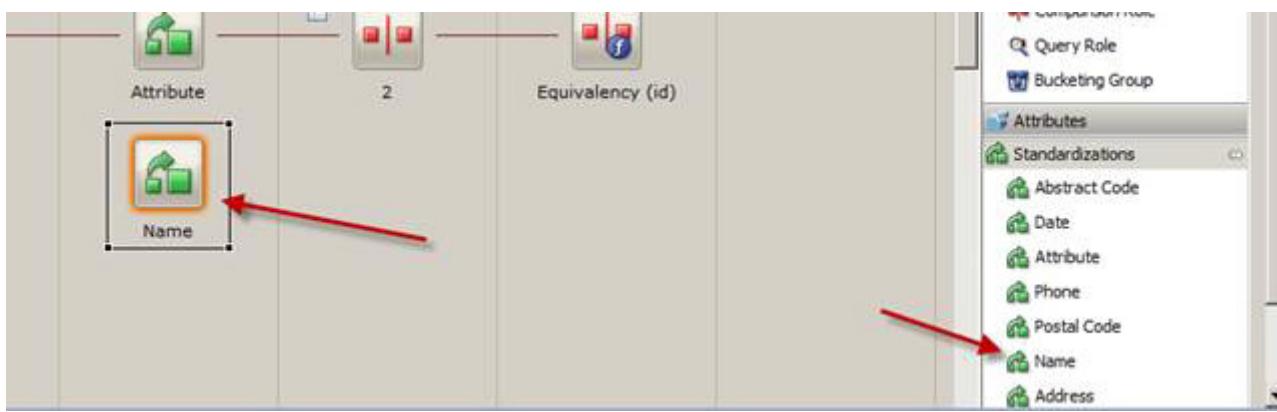
Name plus Birth Date

In this section we will create a bucket for the Name plus Birth Date.

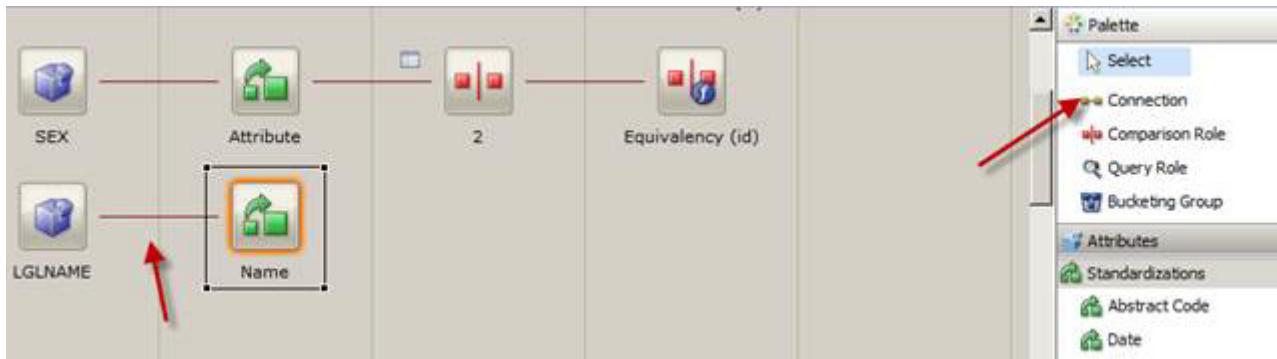
- 1. Inside the Algorithm, expand the **Attributes** view in the palette and select the LGLNAME (NAME) attribute. Place the attribute in the 1st column.



- 2. Expand the **Standardizations** view in the palette and select the Name function. Place the Name standardization function in the 2nd column.



- 3. Connect the **LGLNAME** attribute to the **Name** standardization function using the palette's **Connection** tool.

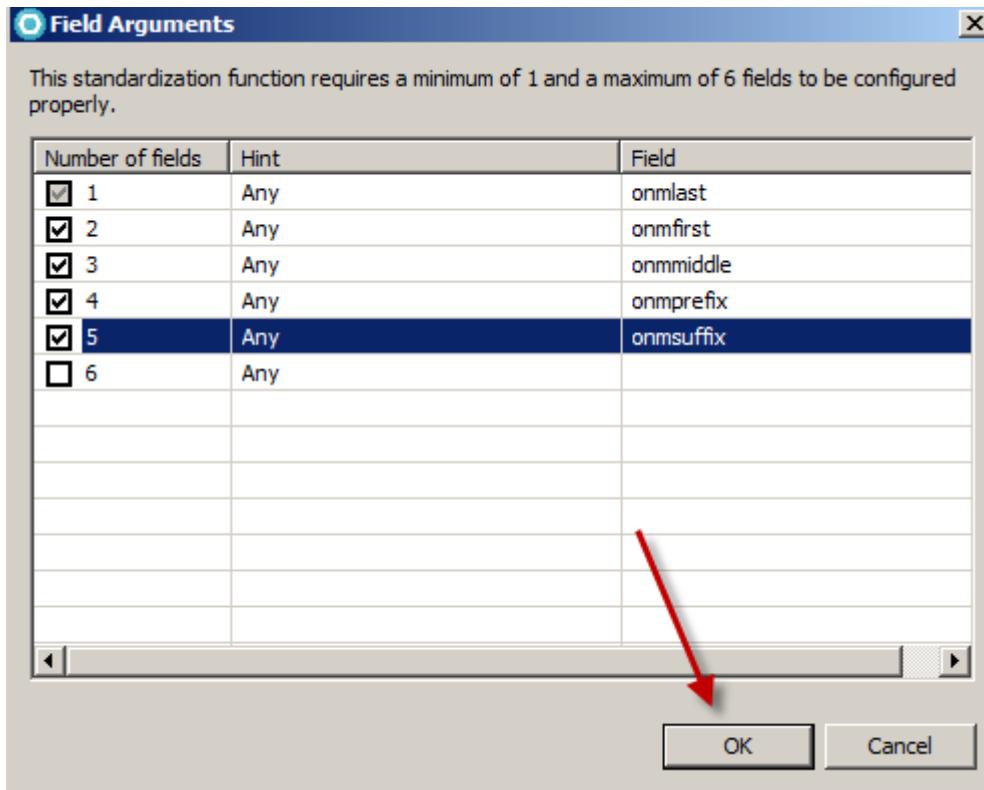


- 4. Press the **ESC** key to exit the connection tool.
 — 5. Select the Name standardization icon and open the RAD Properties window. Click on the button inside the **Field arguments** Value column.



Property	Value
Anonymous string code	
Character mapping string code	
Derivation arguments	
Description	
Equivalent string code	
Field arguments	
Primary standardization role	3
Primary standardization role label	AI
Record status filter	PXNM
Standardization function code	
Type	Person

- ___ 6. Check the first 5 checkboxes and set the Fields from 1 to 5 with the following values. Click the **OK** button when complete.
- ___ a. 1 : **onmlast**.
 - ___ b. 2 : **onmfirst**.
 - ___ c. 3 : **onmmiddle**.
 - ___ d. 4 : **onmprefix**.
 - ___ e. 5 : **onsuffix**.



- ___ 7. Still in the Properties window, set the following values:
- ___ a. Primary standardization role label: **Name Tokens**.

- __ b. Record Status Filter: **AI**.
- __ c. Type: **Person**.
- __ d. Anonymous string code: **ANONNAME**.

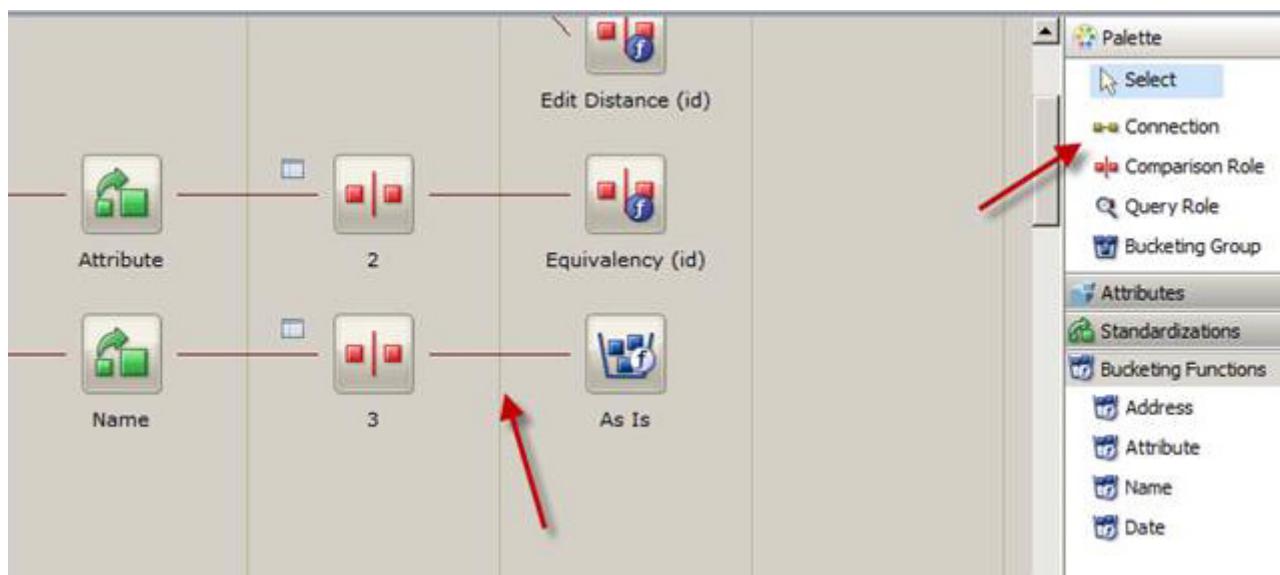
Property	Value
Anonymous string code	ANONNAME
Character mapping string code	
Derivation arguments	
Description	
Equivalent string code	
Field arguments	onmlast,onmfirst,ommmiddle,ompprefix,ommsuffix
Primary standardization role	3
Primary standardization role label	Name Tokens
Record status filter	AI
Standardization function code	PXNM
Type	Person

- __ 8. Connect the **Name** standardization function to the new **Comparison Role** using the palette's **Connection** tool.

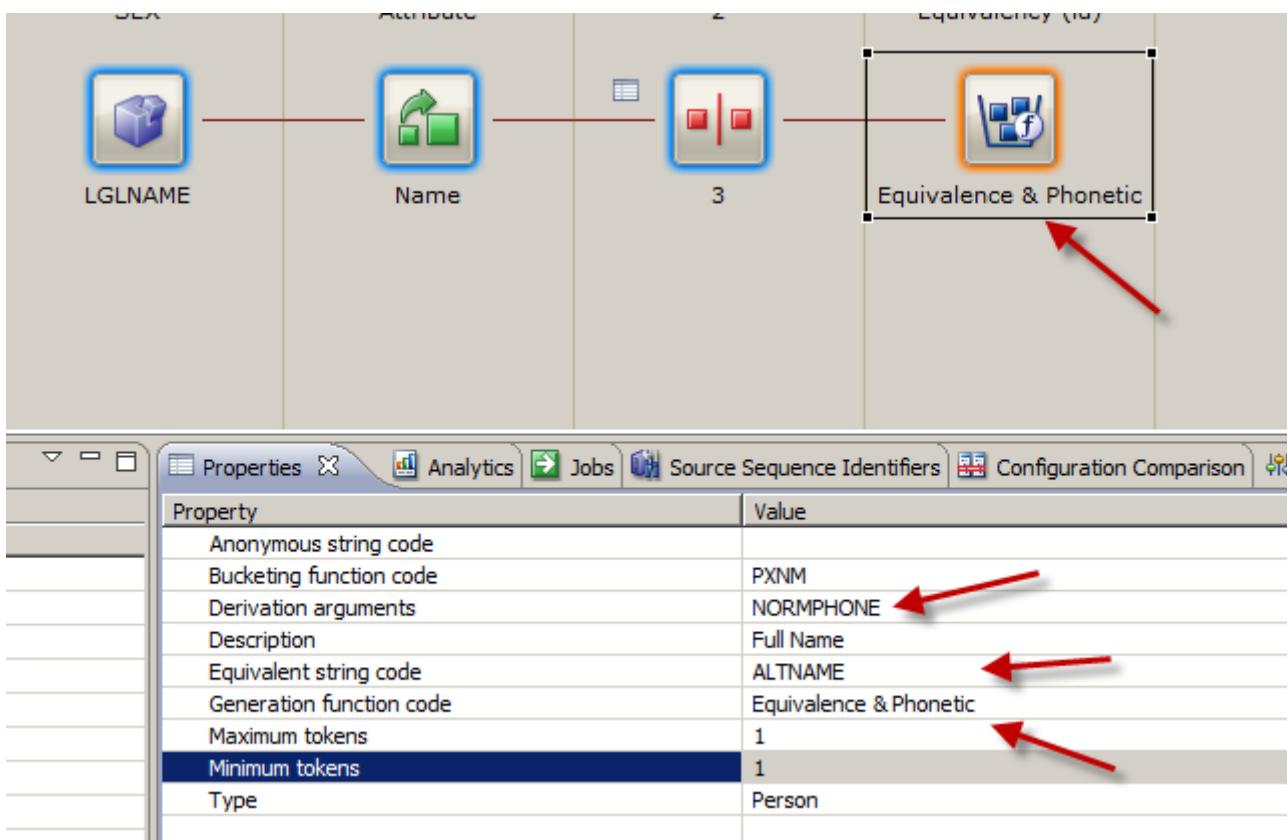
- __ 9. Press the **ESC** key to exit the connection tool.
- __ 10. Expand the **Bucketing Functions** view in the palette and select the Name function icon. Place the bucketing function in the 4th column.



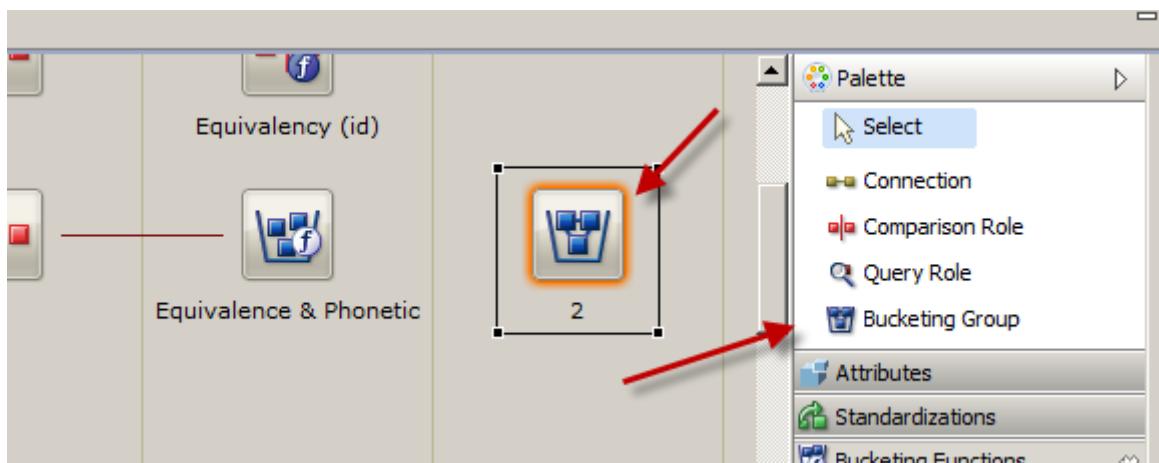
- ___ 11. Connect the **Comparison Role 3** to the new bucketing function (**As Is**) using the palette's **Connection** tool.



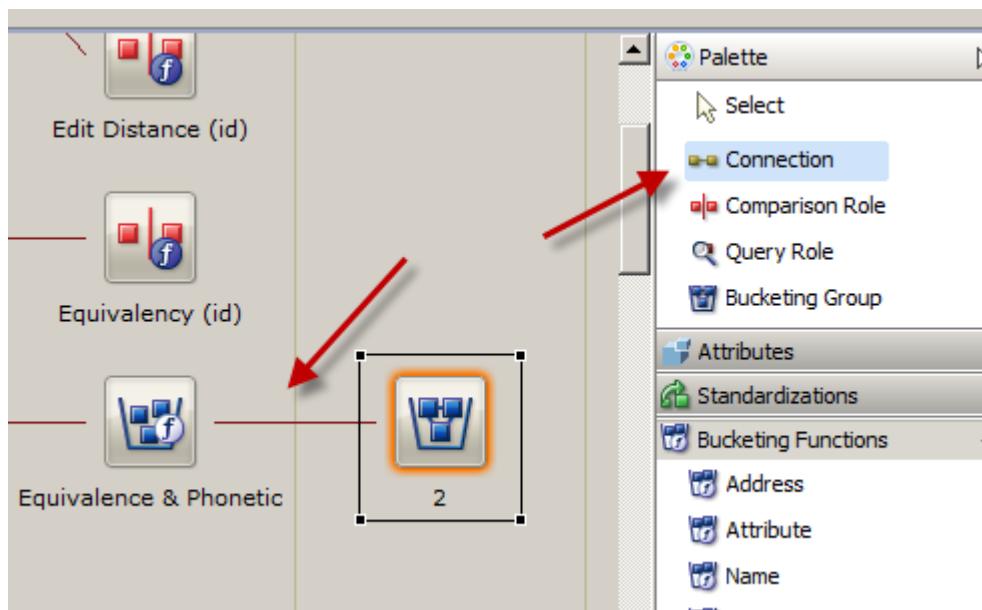
- ___ 12. Press the **ESC** key to exit the connection tool.
- ___ 13. Select the **As Is** bucketing icon in the Functions column and open the RAD Properties window. Enter the following values:
- Description: **Full Name**.
 - Generation function code: **Equivalence & Phonetic**.
 - Derivation arguments: **NORMPHONE**
 - Equivalent string code: **ALTNNAME**.
 - Maximum tokens: **1**.
 - Minimum tokens: **1**.
 - Type: **Person**.



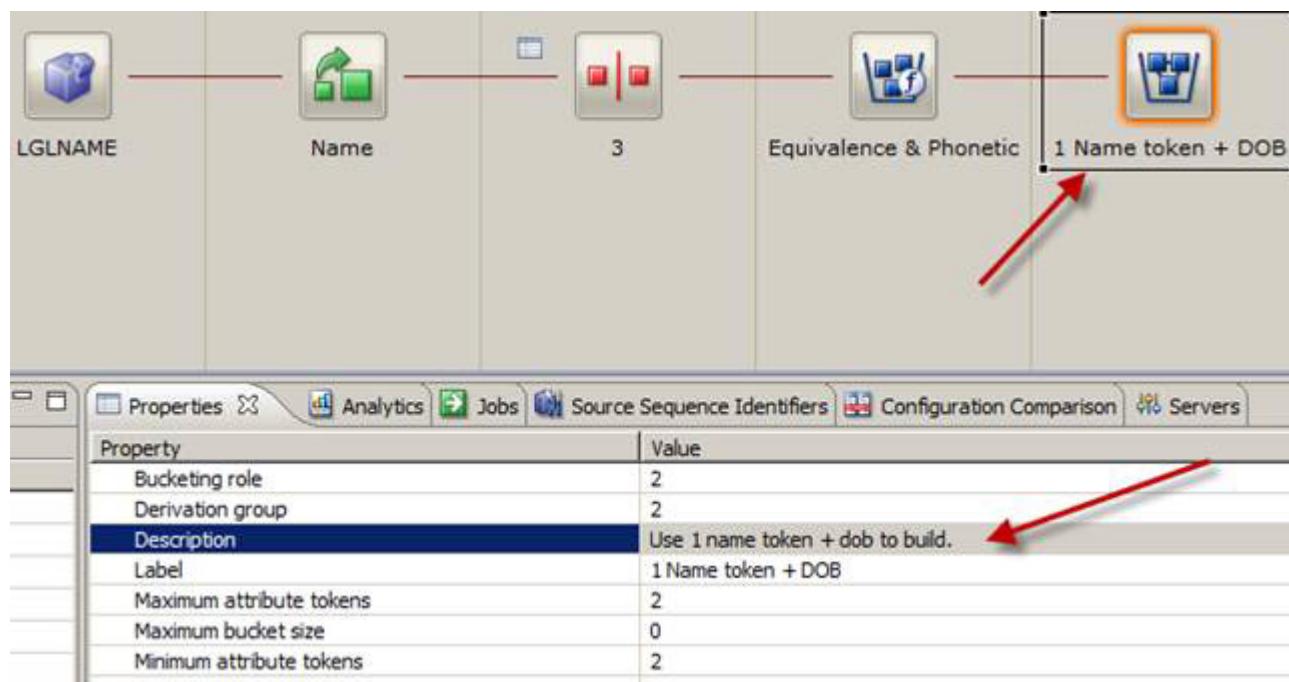
14. Select the **Bucketing Group** icon in the palette and place the bucketing group in the 5th column.



15. Connect the **Equivalence & Phonetic** bucketing function to the new **Bucketing Group 2** using the palette's **Connection** tool.

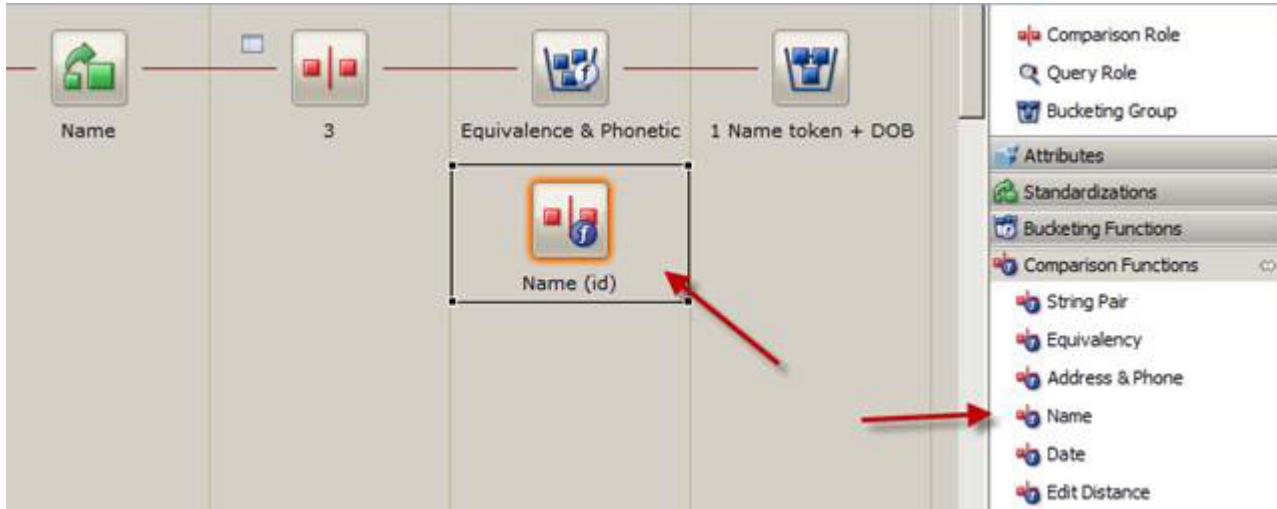


- ___ 16. Press the **ESC** key to exit the connection tool.
- ___ 17. Click on the new Bucketing Group 2 icon and open the RAD Properties window. Enter the following information:
 - ___ a. Description: **Use 1 name token + dob to build.**
 - ___ b. Label: **1 Name token + DOB.**
 - ___ c. Maximum attribute tokens: **2.**
 - ___ d. Maximum bucket size: **0.**
 - ___ e. Minimum attribute tokens: **2.**

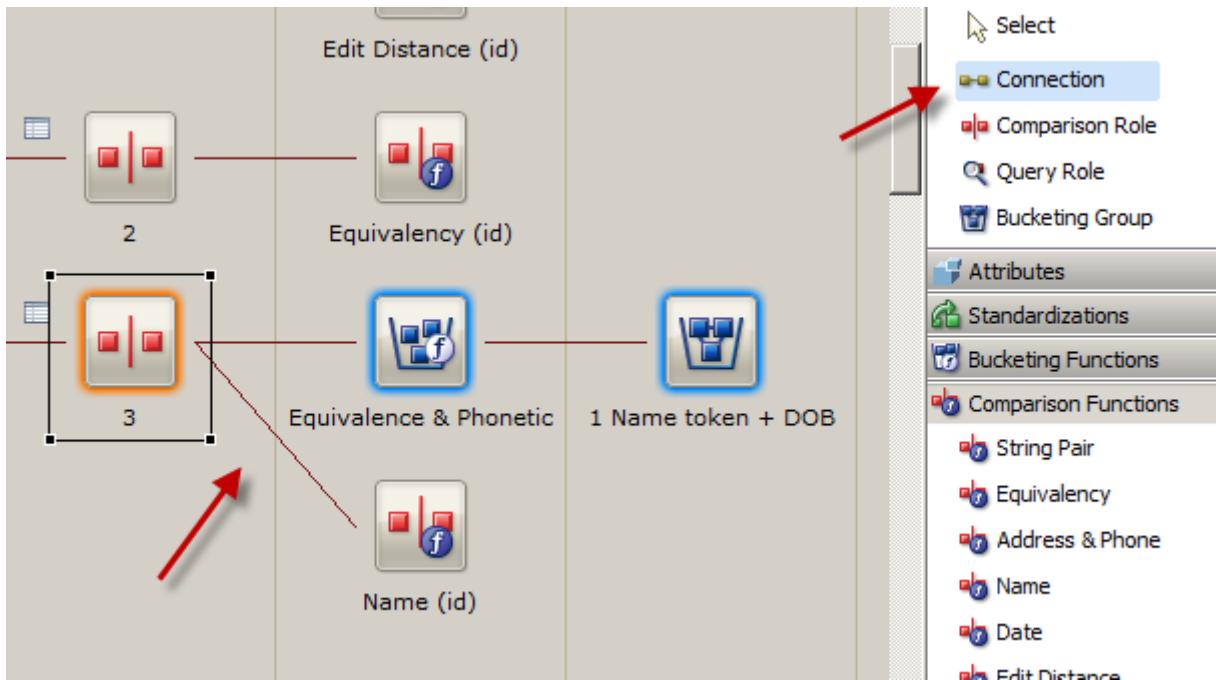


Next we will add a comparison function to our Legal Name to contribute to the match score.

- 18. Expand the **Comparison Functions** view in the palette, select the **Name** function and place the function in the 4th column under the Equivalence & Phonetic bucketing function.

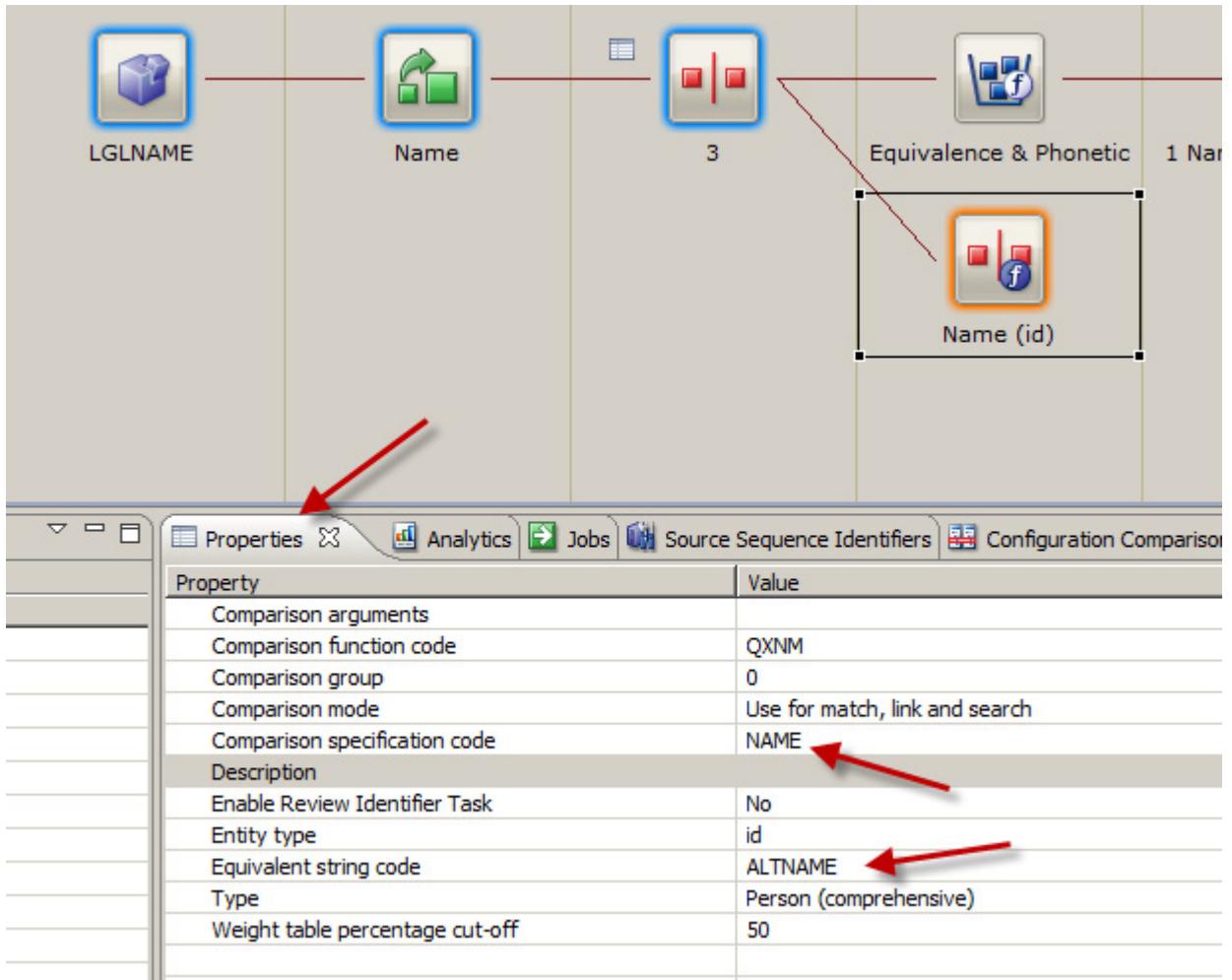


- 19. Connect the **Comparison Role 3** to the **Name (id)** comparison function using the palette's **Connection** tool.



- 20. Press the **ESC** key to exit the connection tool.
- 21. Select the **Name (id)** comparison function and open the RAD Properties window. Enter the following values:
 - a. Comparison specification code: **NAME**.
 - b. Equivalent string code: **ALTNAMES**.

- ___ c. Type: **Person (comprehensive)**.



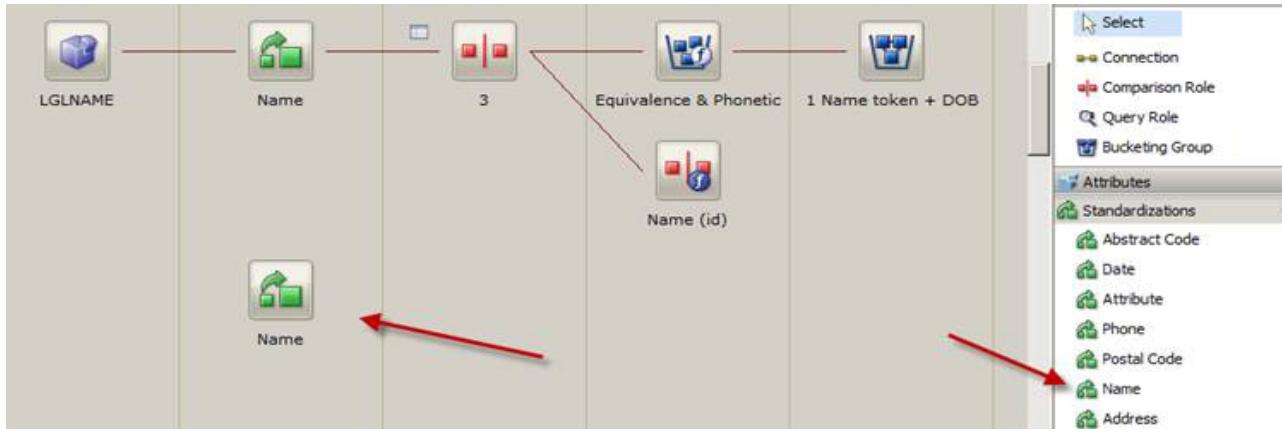
- ___ 22. Click the **Save** button to save your project.

Currently, our bucketing only includes one token from the Name. Later in the exercise, we will be adding the Birth Data attribute and connecting it to the bucket.

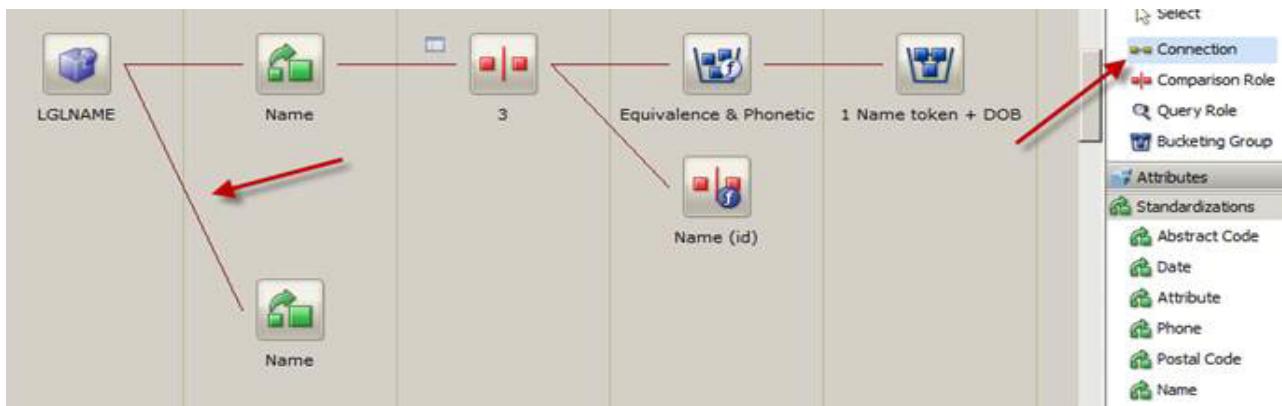
Last Name and Zip Code Bucketing

In this section we will create a bucket that will include the Last Name and Zip Codes. Because the first standardization function we created for the Name included all 5 name tokens (last, first, middle, prefix, suffix), we will first create a new Standardization function that will only include the last name.

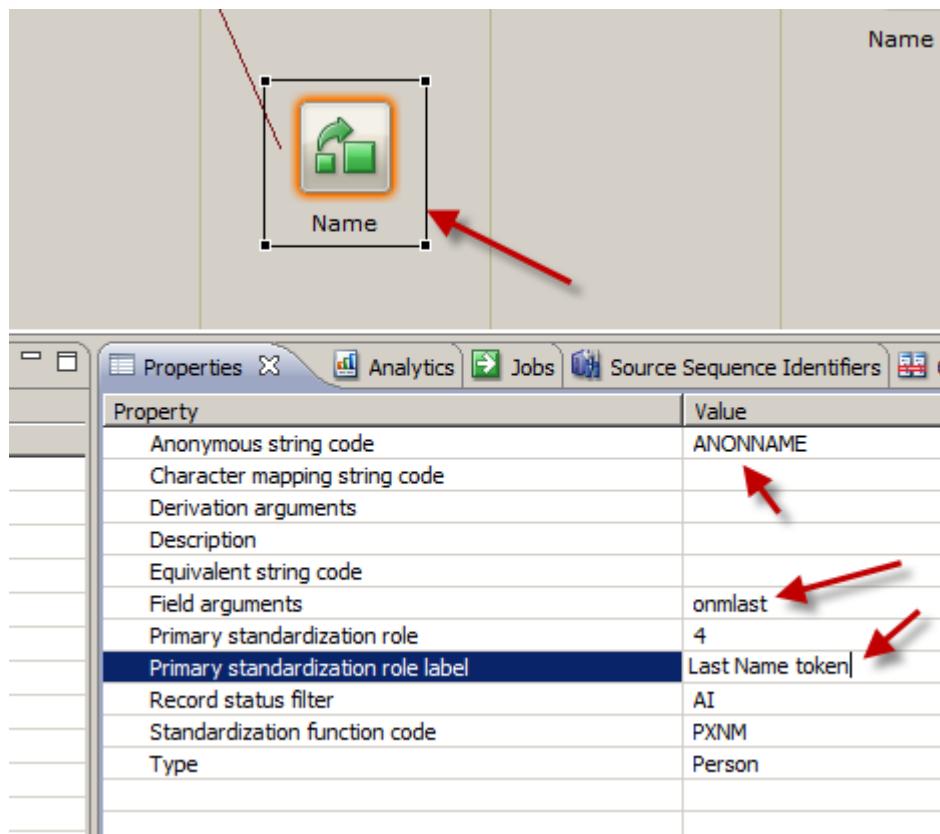
- ___ 23. Expand the **Standardizations** view in the palette and select the Name function. Place the **Name** function under the 2nd column.



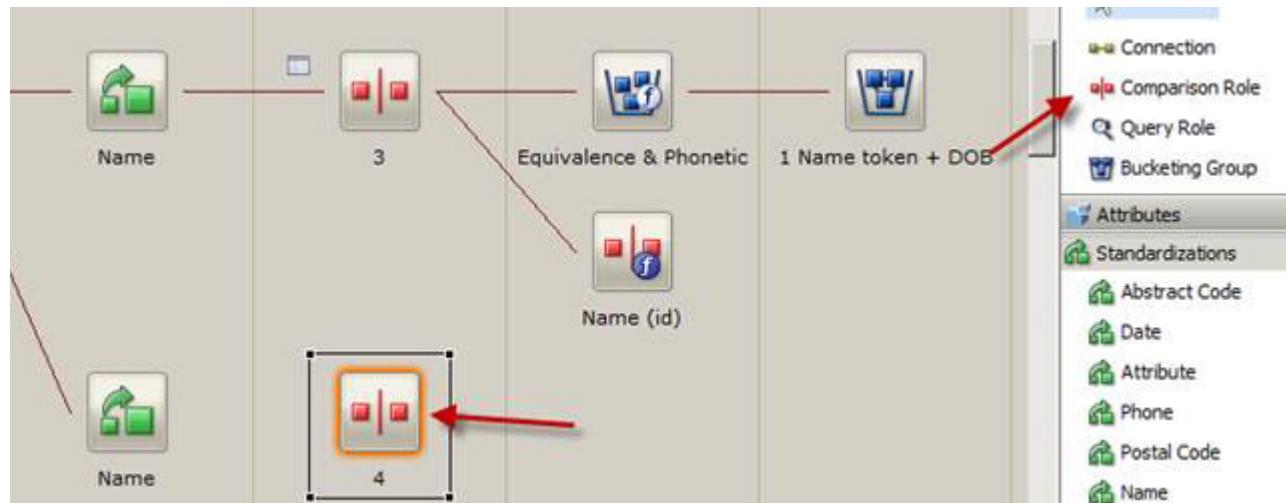
- ___ 24. Connect the **LGLNAME** attribute to the **Name** standardization function using the palette's **Connection** tool.



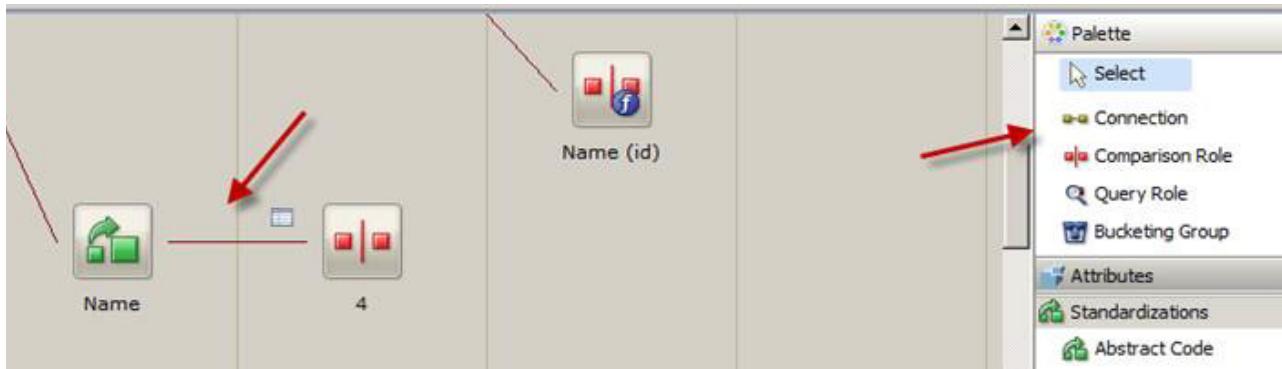
- ___ 25. Press the **ESC** key to exit the connection tool.
- ___ 26. Select the new Standardization function and open the RAD Properties window. Enter the following values:
- Field arguments: **omnlst**
 - Primary standardization role label: **Last Name token**.
 - Record status filter: **All**.
 - Type: **Person**.
 - Anonymous string code: **ANONNAME**.



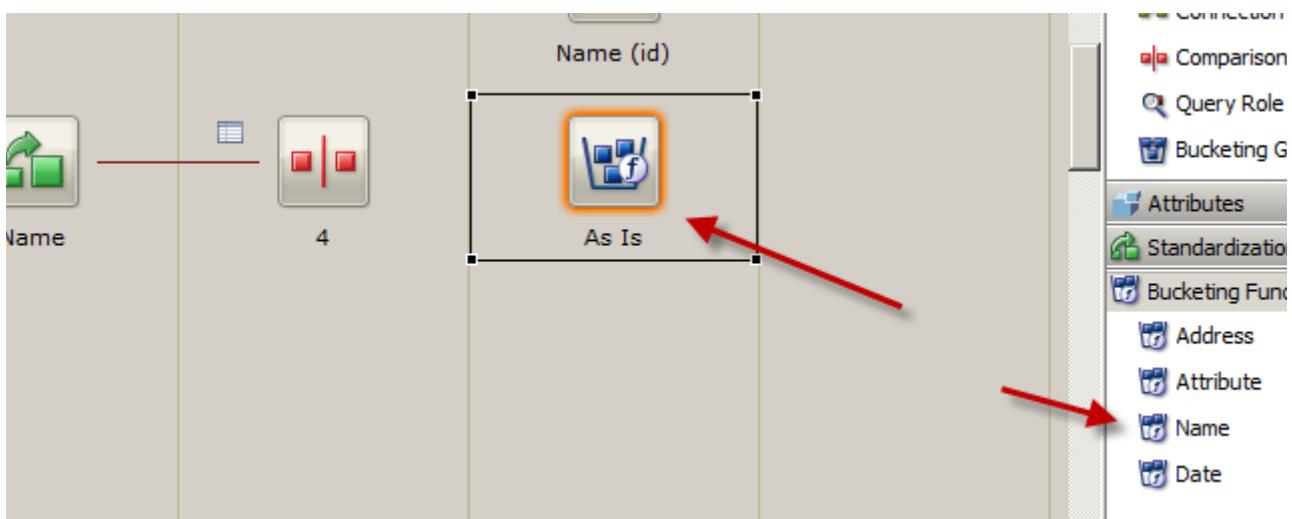
- 27. Select the **Comparison Role** icon in the palette and add it to the 3rd column.



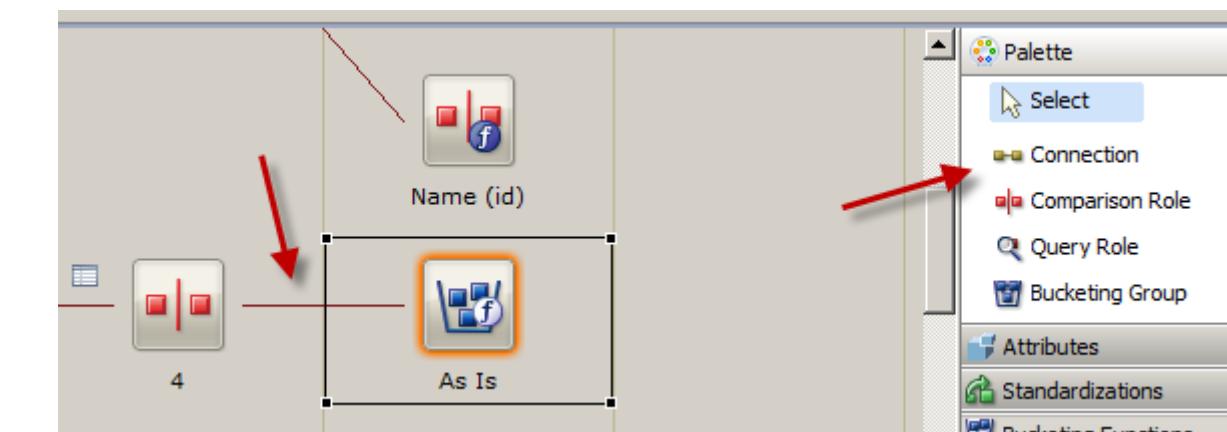
- 28. Connect the **Name** standardization function to the **Comparison Role 4** using the palette's **Connection** tool.



- ___ 29. Press the **ESC** key to exit the connection tool.
- ___ 30. Expand the **Bucketing Functions** view in the palette and select the Name function icon. Place the Name bucketing function in the 4th column.

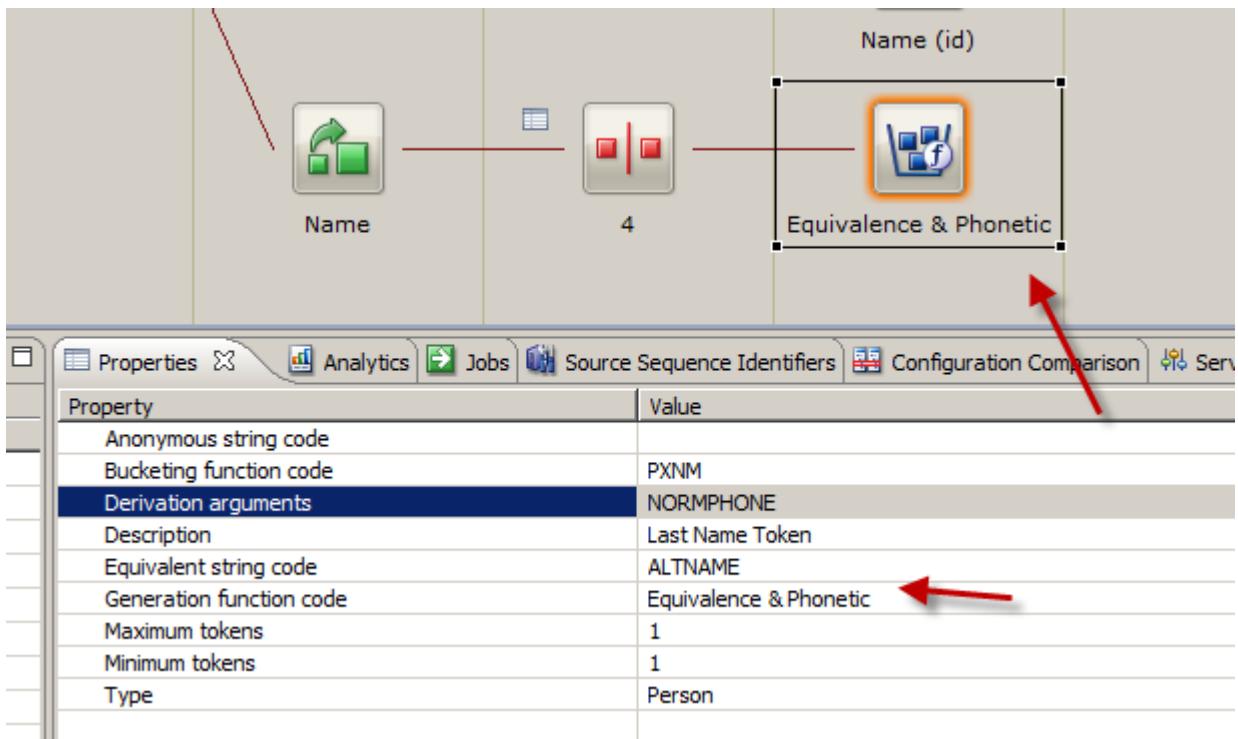


- ___ 31. Connect the **Comparison Role 4** to the new **Name** bucketing function (**As Is**) using the palette's **Connection** tool.

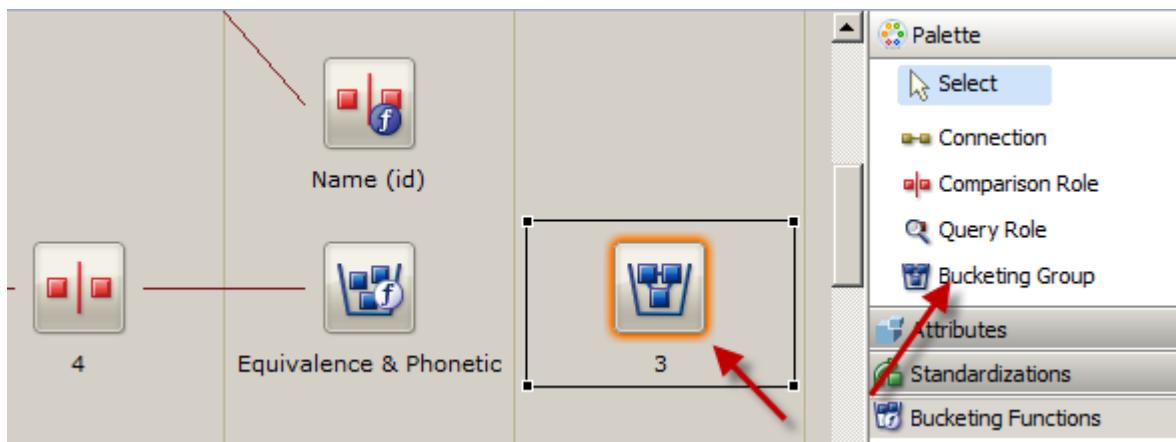


- ___ 32. Press the **ESC** key to exit the connection tool.

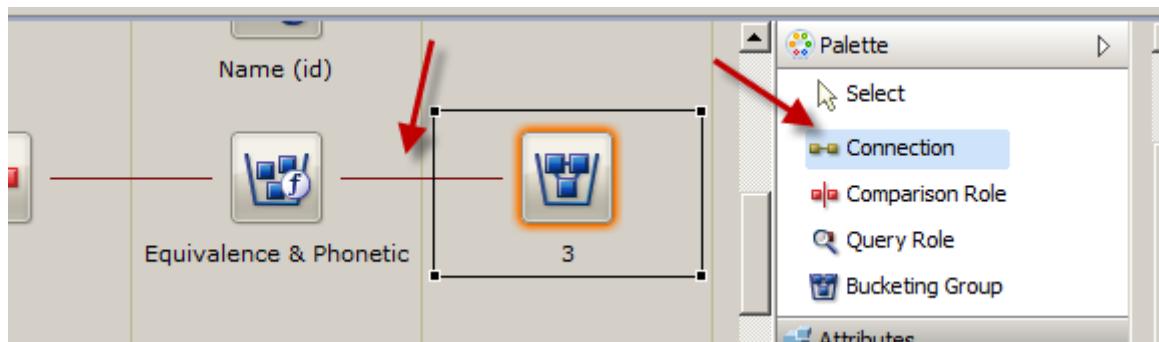
- ___ 33. Select the new **As Is** bucketing icon and open the RAD Properties window. Enter the following values:
- Description: **Last Name token**.
 - Generation function code: **Equivalence & Phonetic**.
 - Derivation arguments: **NORMPHONE**
 - Equivalent string code: **ALTNAMES**.
 - Maximum tokens: **1**.
 - Minimum tokens: **1**.
 - Type: **Person**.



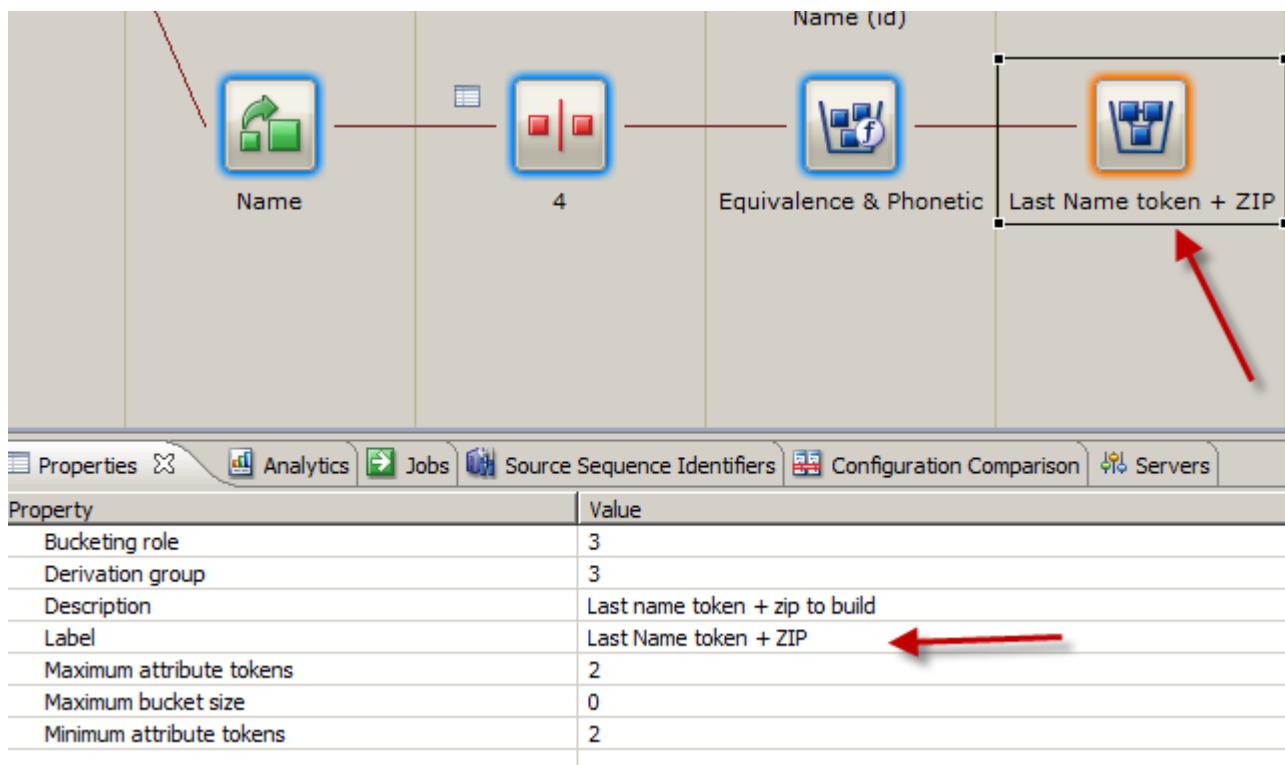
- ___ 34. Select the **Bucketing Group** icon in the palette and place the bucketing group under the 5th column.



- ___ 35. Connect the **Equivalence & Phonetic** bucketing function to the new **Bucketing Group 3** using the palette's **Connection** tool.



- ___ 36. Press the **ESC** key to exit the connection tool.
 ___ 37. Select the **Bucketing Group 3** and open the RAD **Properties** window. Enter the following values:
 ___ a. Description: **Last name token + zip to build**.
 ___ b. Label: **Last Name token + ZIP**.
 ___ c. Maximum attribute tokens: **2**.
 ___ d. Maximum bucket size: **0**.
 ___ e. Minimum attribute tokens: **2**.



- 38. Click the **Save** button to save your project.

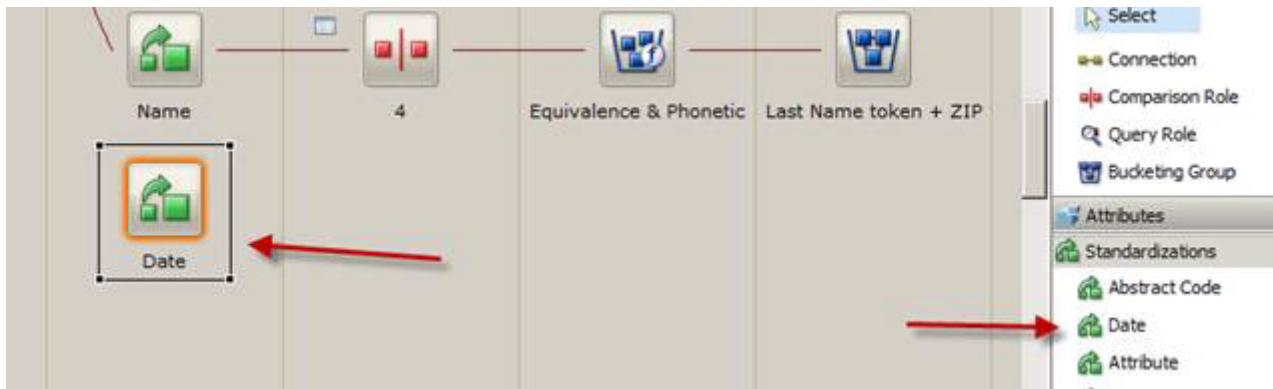
Part 3: Configure the birth date attribute

For the Birth Date, we will attach the attribute to our Name plus DOB bucket and also create a comparison function to add a score to the match algorithm.

- 39. Expand the **Attributes** view in the palette and select the **BIRTHDT (DATE)** attribute. Place the attribute in the 1st column.



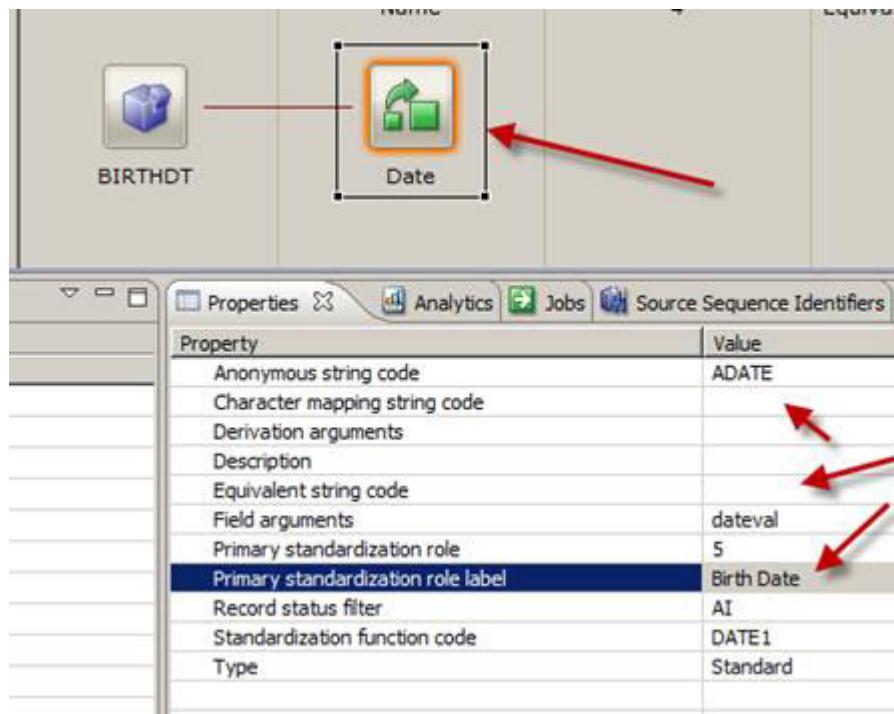
- 40. Expand the **Standardizations** view in the palette and select the **Date** function. Place the Date standardization function under the 2nd column.



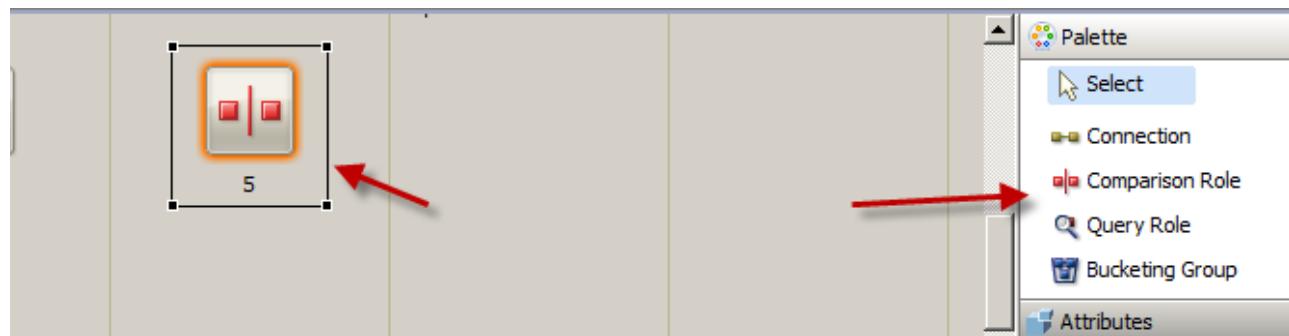
- 41. Connect the **BIRTHDT** attribute to the **Date** standardization function using the palette's **Connection** tool.



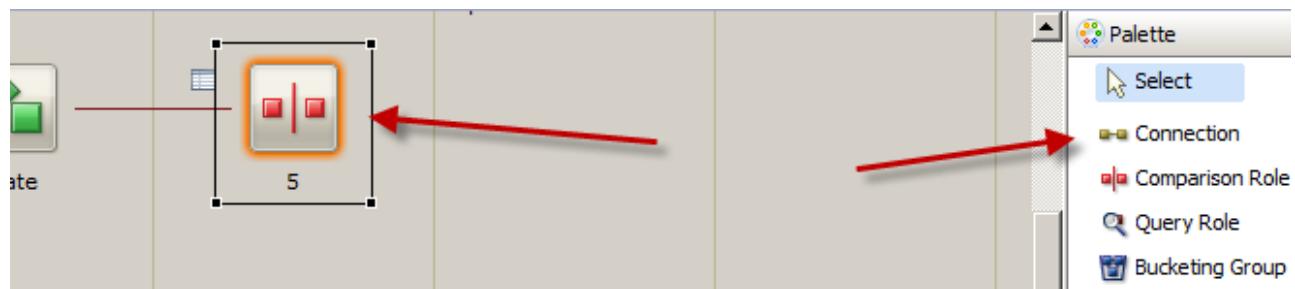
- 42. Press the **ESC** key to exit the connection tool.
- 43. Select the **Date** standardization icon and open the RAD Properties window. Enter the following values:
- Field Arguments: **dateval**
 - Anonymous string code: **ADATE**.
 - Primary Standardization role label: **Birth Date**.
 - Record status filter: **All**.
 - Type: **Standard**.



- 44. Select the **Comparison Role** icon in the palette and place it in the 3rd row.

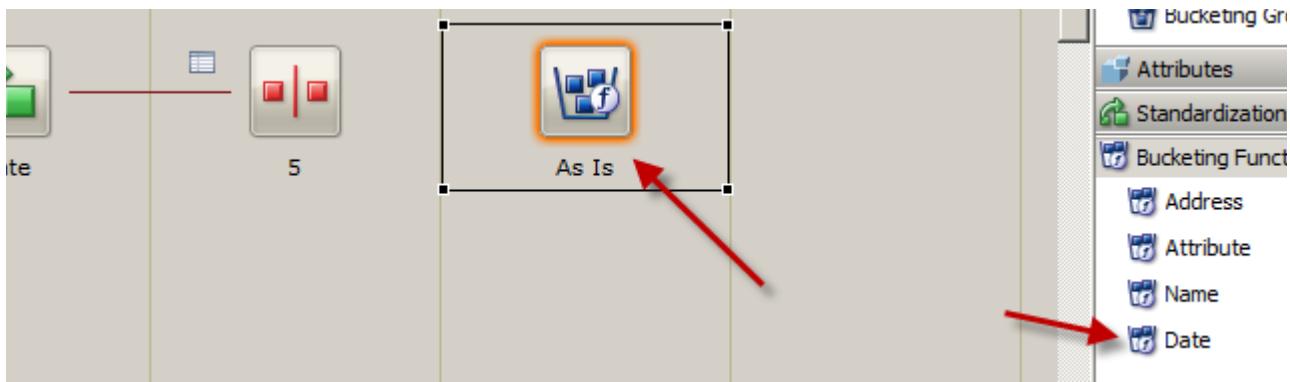


- 45. Connect the **Date** standardization function to the **Comparison Role 5** using the palette's **Connection** tool.

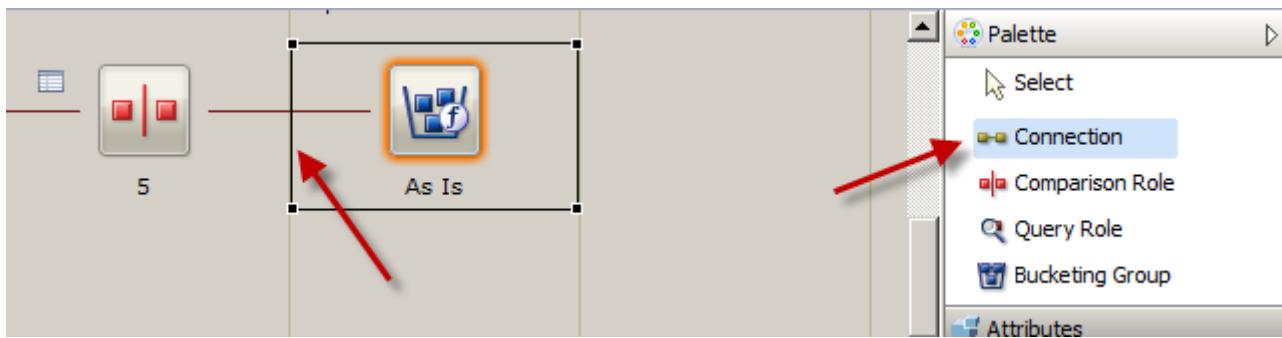


- 46. Press the **ESC** key to exit the connection tool.

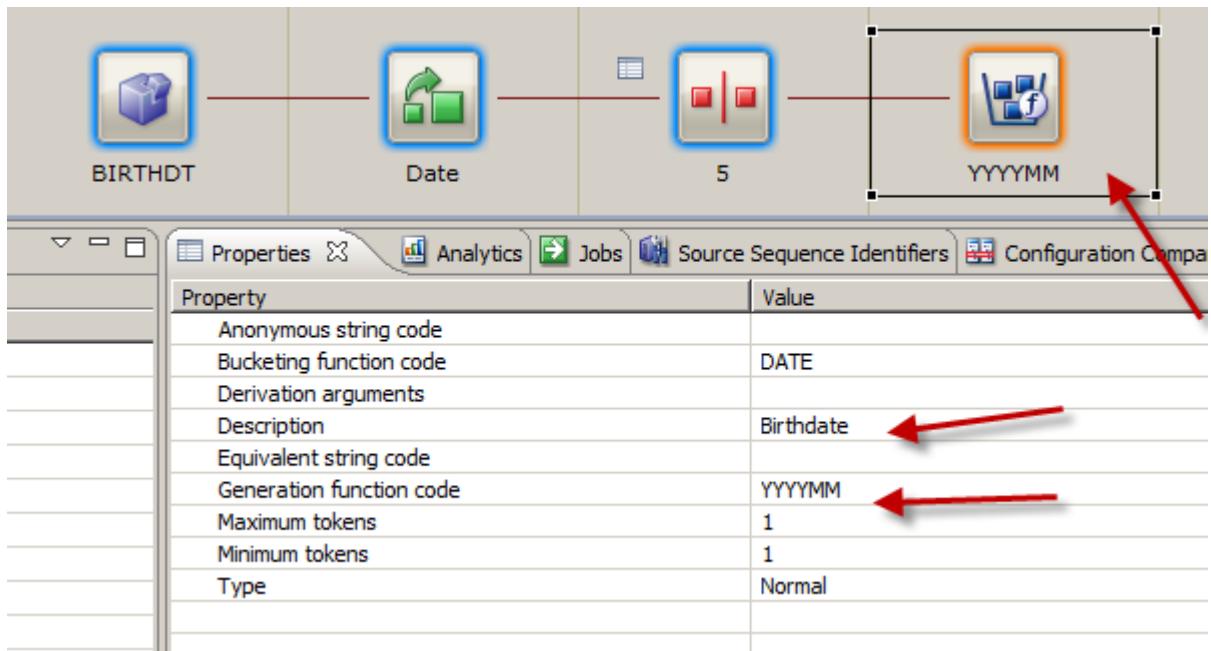
- 47. Expand the **Bucketing Functions** view in the palette and select the **Date** icon. Place the bucketing function in the 4th column.



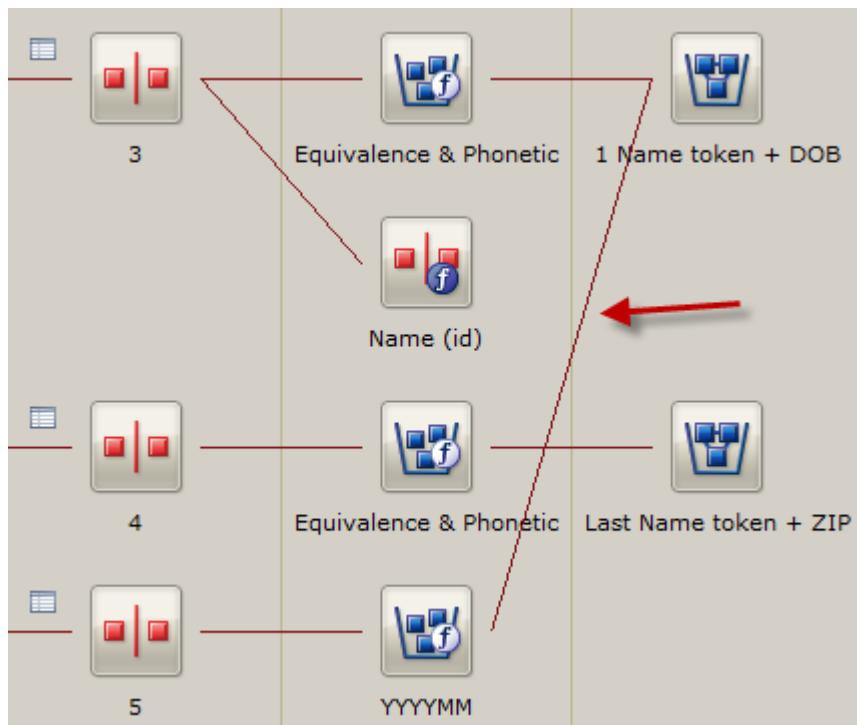
- 48. Connect the **Comparison Role 5** to the new Data bucketing function (**As Is**) using the palette's **Connection** tool.



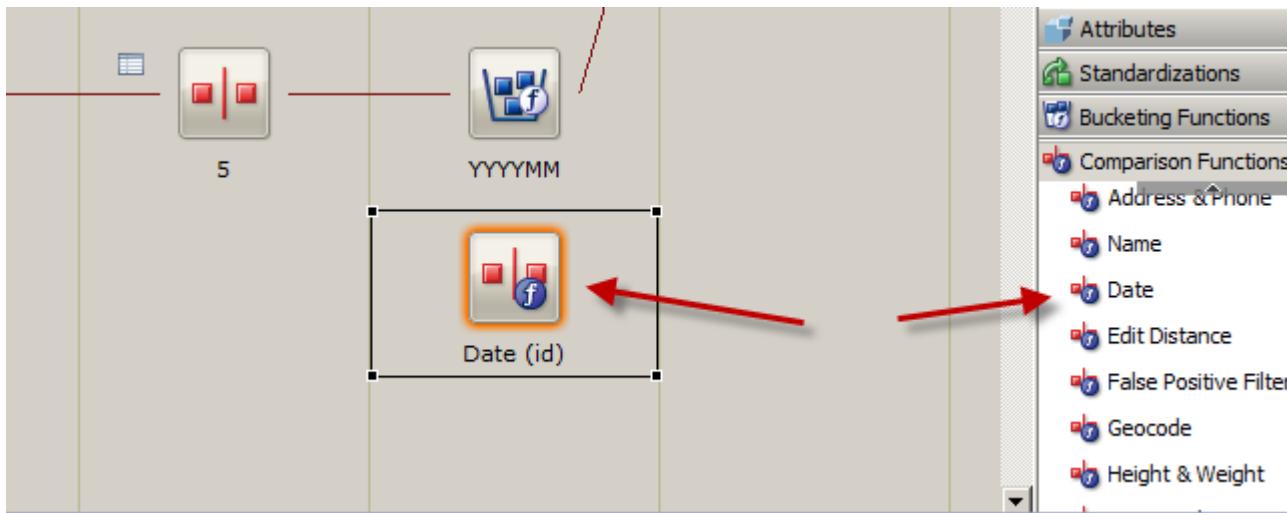
- 49. Press the **ESC** key to exit the connection tool.
 — 50. Select the **As Is** bucketing icon and open the RAD Properties window. Enter the following values:
 — a. Description: **Birthdate**.
 — b. Generation function code: **YYYYMM**.
 — c. Maximum tokens: **1**.
 — d. Minimum tokens: **1**.
 — e. Type: **Normal**.



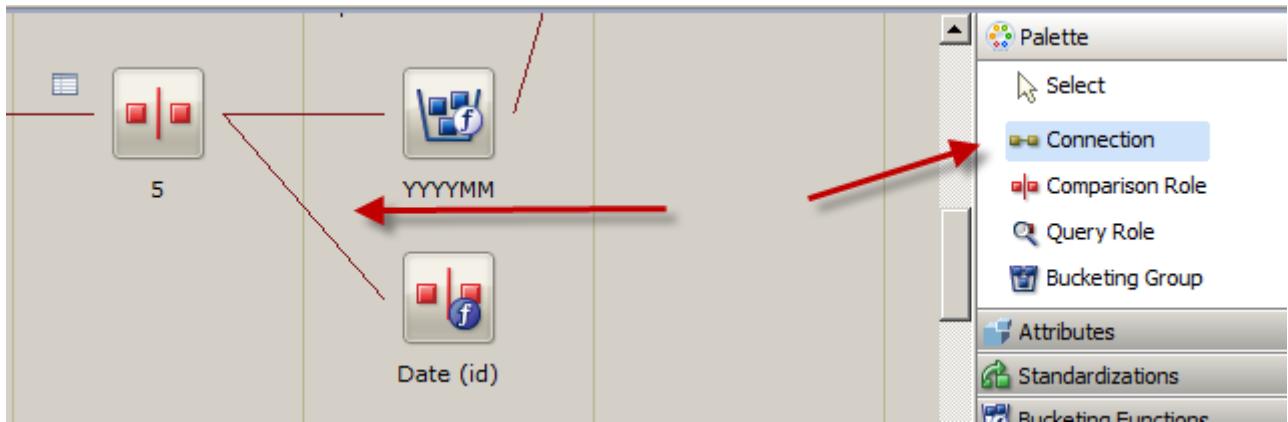
51. Connect the **YYYYMM** bucketing function to the **1 Name token + DOB** bucketing group that created previously with the name using the palette's **Connection tool**.



52. Press the **ESC** key to exit the connection tool.
53. Expand the **Comparison Functions** view in the palette and select the Date function. Place the Date comparison function under the 4th column.



- ___ 54. Connect the **LGLNAME** attribute to the **Name** standardization function using the palette's **Connection** tool.

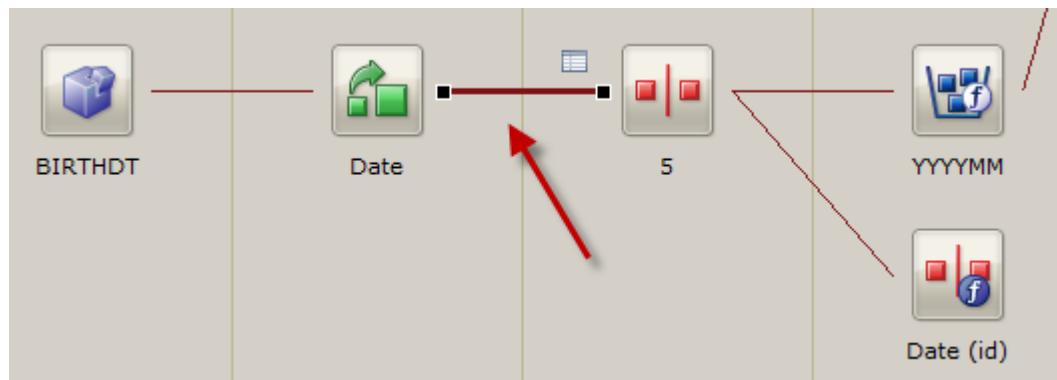


- ___ 55. Press the **ESC** key to exit the connection tool.
- ___ 56. Select the **Date (id)** comparison function and open the RAD Properties window. Enter the following information:
- Comparison specification code: **DOB**.
 - Type: **Date**.

The screenshot shows a data flow configuration. It starts with a source node 'BIRTHDT' (represented by a blue cube icon) connected to a 'Date' standardization node (green document icon). This leads to a 'Comparison Role 5' node (red square icon with two red squares). From there, two paths emerge: one leading to a target node 'YYYYMM' (blue folder icon), and another leading to a 'Date (id)' node (red square icon with a blue 'f'). A red arrow highlights the connection between the 'Date' standardization node and the 'Comparison Role 5' node.

Property	Value
Comparison arguments	
Comparison function code	DATE
Comparison group	0
Comparison mode	Use for match, link and search
Comparison specification code	DOB
Description	
Enable Review Identifier Task	No
Entity type	id
Equivalent string code	
Type	Date
Weight table percentage cut-off	80

57. Click the connection between the **Date** Standardization icon and the **Comparison Role 5**.



58. Under the **Properties** window, change the Minimum weight frequency to 1.

The screenshot shows the 'Properties' window with the following configuration:

Property	Value
Minimum weight frequency	1
Record status filter	AI

A red arrow points to the 'Value' column for 'Minimum weight frequency', highlighting the change from 'AI' to '1'.

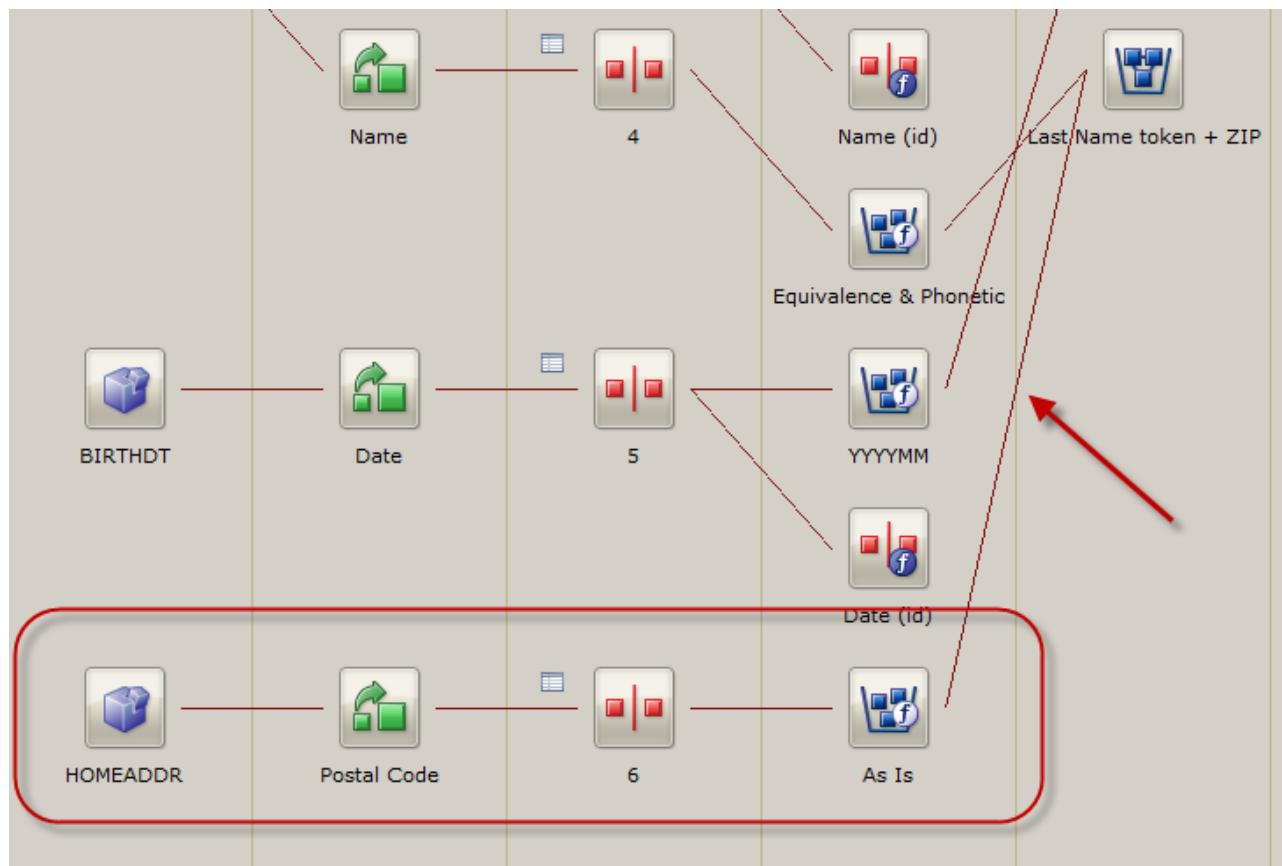
59. Click the **Save** button to save your project.

Part 4: Configure the home address attribute

To configure the home address, we will break it into 2 steps. The first step is to separate the Zip Code (or Postal Code) from the address so that we can use it in the Last Name token + Zip code bucket. The second step will be to create a comparison algorithm that will use the entire address (and phone) for the comparison. We will not create a bucket for the address alone, because even with multiple tokens in the buckets (e.g. New York, St, NY), you could have too many members. If we were storing business addresses however, it might work to create a bucket for a complete match of the address. By using the a 2 dimensional comparison with Address and Phone, we can give someone bonus points for have both matches (Phone and Address), and adjust the weights as they get further away on both attributes.

Postal Code (Zip Code for Last Name bucket)

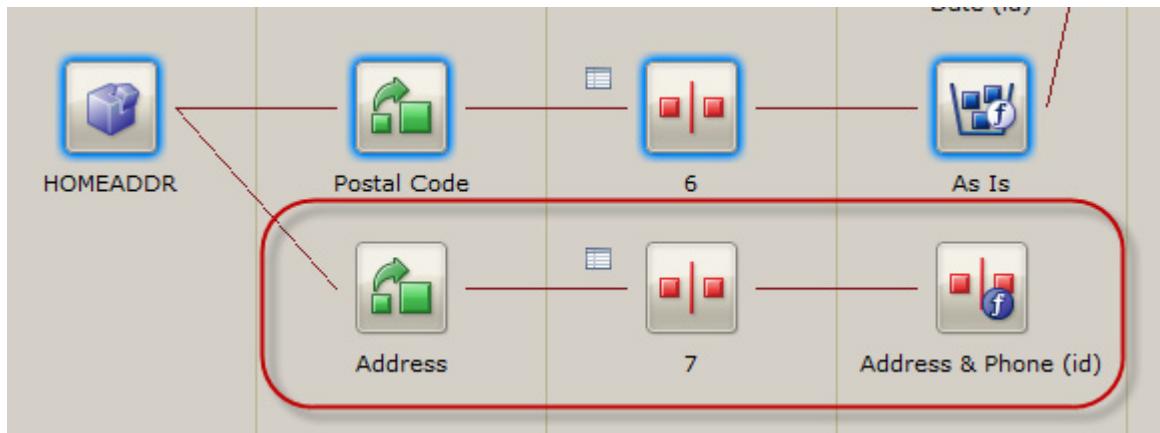
- ___ 60. Create the following algorithm pieces (and connections) for the **HOMEADDR(ADDR)** attribute with the following values:
- ___ a. Attribute: **HOMEADDR(ADDR)**
 - ___ b. Standardization: **Postal Code**
 - Field arguments: **zipcode**
 - Locale: **United States**.
 - Record status filter: **All**.
 - ___ c. Comparison Role
 - ___ d. Bucketing Function: **Attribute**
 - Description: **Zip code**.
 - Generation function code: **As Is**.
 - Maximum tokens: **1**.
 - Minimum tokens: **1**.
 - ___ e. Bucketing Group: Connect the Bucketing Function to the Last Name token + Zip Bucketing Group



Address and Phone Comparison

For our address comparison, we will need to use all the tokens, not just the Zip Code (or Postal Code). To pull all tokens we can reuse the HOMEADDR attribute, but will need to create a new Standardization function that takes all tokens. (NOTE: We will deal with the Phone part of the comparison later in this exercise)

- ___ 61. Create the following algorithm pieces (and connections) for the **HOMEADDR(ADDR)** attribute with the following values:
 - ___ a. Standardization: **Address** (Connect the existing HOMEADDR to this standardization function)
 - Field arguments: **stline1,stline2,stline3,stline4,city,state,zipcode**
 - Record status filter: **AI**.
 - Region: **United States - expanded**.
 - ___ b. Comparison Role
 - ___ c. Comparison Function: **Address & Phone**
 - ___ d. Comparison specification code: **AXP**



- __ 62. Click the **Save** button to save your project.

Part 5: Configure the phone attribute

We have 2 phone types defined in our data model: Home Phone and Mobile Phone. We will combine both types into the same Comparison Roles as the same data type (e.g. someone might enter a mobile phone as their Home Phone).

We will also create a bucket for our phone based on the sorted values of the phone number (e.g 555-4212-9053 would be bucketed as 01223455559).

Creating the Phone Bucket

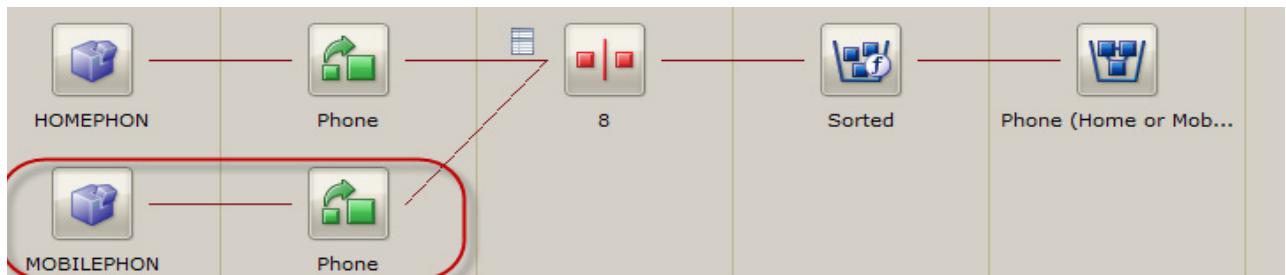
- __ 63. Create the following algorithm pieces (and connections) for the **HOMEPHON (PHONE)** attribute with the following values:
- Attribute: **HOMEPHON (PHONE)**
 - Standardization: **Phone**
 - Field arguments: **phnumber**
 - Primary Standardization role label: **Home Phone**.
 - Record status filter: **All**.
 - Type: **North America**.
 - Comparison Role
 - Bucketing Function: **Attribute**
 - Description: **Home Phone**.
 - Generation function code: **Sorted**.
 - Maximum tokens: **1**.
 - Minimum tokens: **1**
 - Bucketing Group
 - Description: **Phone (Home or Mobile) to build**.

- Label: **Phone (Home or Mobile)**.
- Maximum attribute tokens: **1**.
- Maximum bucket size: **0**.
- Minimum attribute tokens: **1**



We are not complete yet, remember that we also want to include the Mobile Phone in this bucket (and comparison in the next part of the exercise), so we want to add it to the same Comparison Role (since it's the same data, just different type).

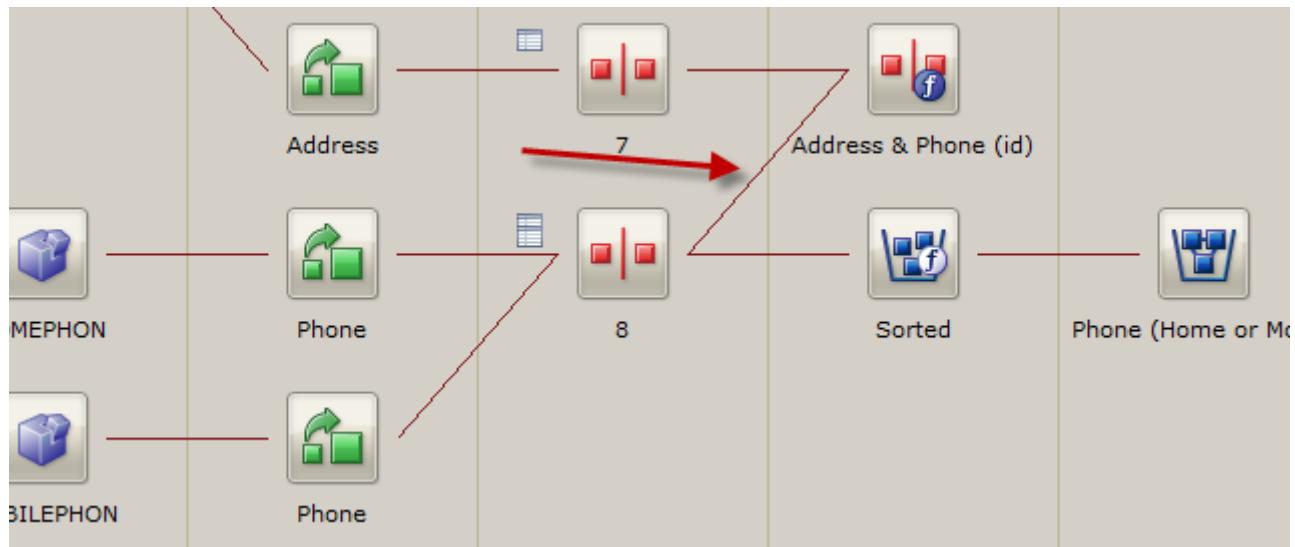
- 64. Create the following algorithm pieces (and connections) for the **MOBILEPHON (PHONE)** attribute with the following values:
- a. Attribute: **MOBILEPHON (PHONE)**
 - b. Standardization: **Phone**
 - Field arguments: phonenumber
 - Primary Standardization role label: Home Phone.
 - Record status filter: AI.
 - Type: North America.
 - c. Comparison Role: connect the Standardization function (Phone) to the same comparison role as the HOMEPHON attribute.



Phone Number comparison

To compare our phone numbers, we have already created a 2 dimensional comparison with the Address, all that is left is to connect our Home Phone and Mobile Phone comparison role to our Address and Phone comparison function.

- 65. Using the palette's connection tool, connect the **Comparison Role 8** (the phone comparison role) to the **Address & Phone (id)** comparison function.



___ 66. Click the **Save** button to save your project.

End of exercise

Appendix C. Bulk Cross Load

What this exercise is about

This exercise covers bulk cross matching.

What you should be able to do

At the end of this exercise, you should be able to:

- Run a bulk cross match and load the entity/linkage data

Introduction

Although we have loaded sample data into MDM and they have been bucketed, not one member has been compare at this point. To compare our members for a bulk load, we must run the bulk cross match (BXM) process.

The bulk cross match (BXM) is a process that allows you to compare and link thousands of records per second. The BXM is used when we are working with the Physical MDM to test our algorithm and set our Thresholds. The BXM process is made up of two primary jobs; Compare Members in Bulk (mpxcomp) and Link Entities (mpxlink). The BXM process creates entities so after running the compare and link, the entity data will need to be loaded into the database.

Requirements

Each of these exercise depend on the previous exercise being complete. For this exercise you must have the data model and algorithms defined in the configuration file. The configuration must be deployed with sample data loading into the database and the weights must be generated and deployed.

Exercise instructions

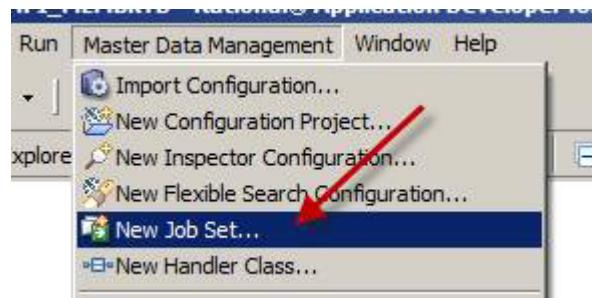
Part 1: Run a bulk cross match and load entity data

For the Bulk Cross Match Load we will create a job that will run 3 tasks:

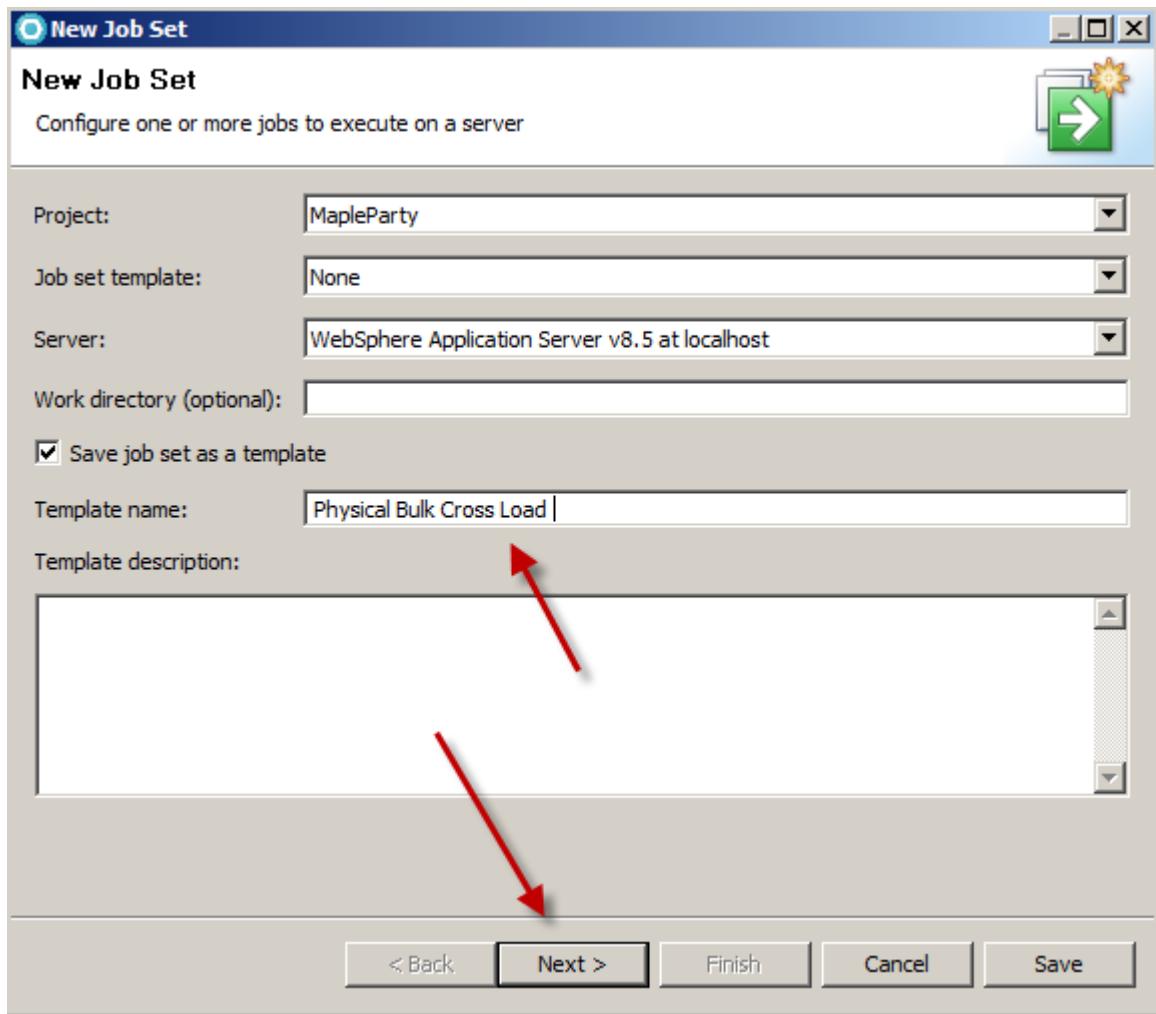
- Compare Members in Bulk (mpxcomp)
- Link Entities (mpxlink)
- Load UNLs to DB (madunload)

For this step you will need to change the RAD perspective to **MDM Configuration**.

- 1. From the RAD file menu, select **Master Data Management > New Job Set...**



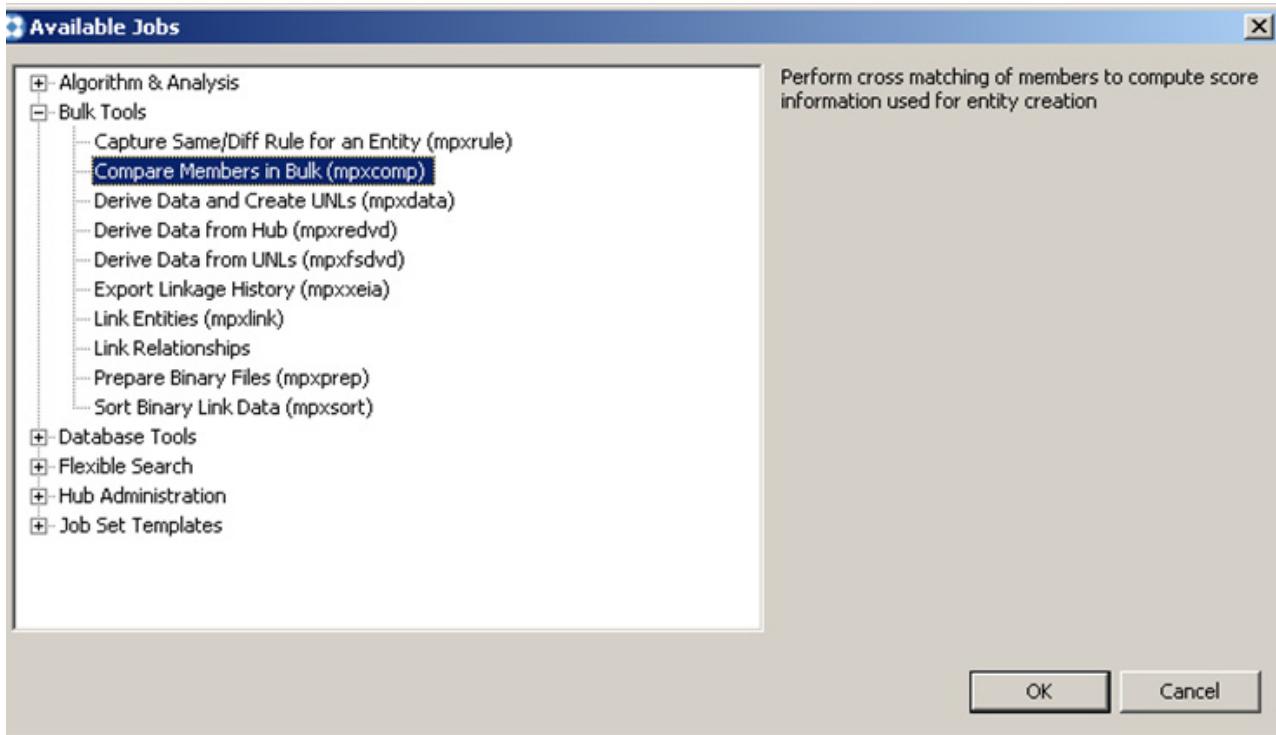
- 2. Enter **Physical Bulk Cross Load** as the job set template name and click the **Next >** button.



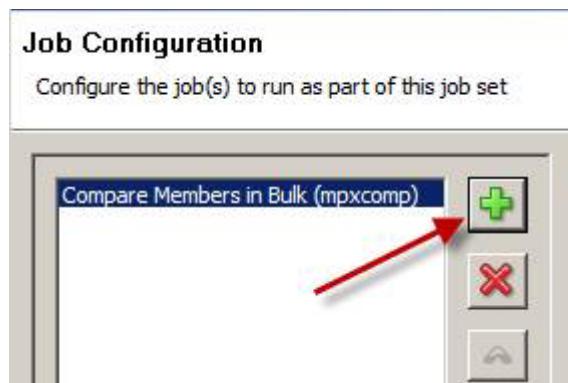
- ___ 3. Click the **Add Job** (the green plus sign) button.



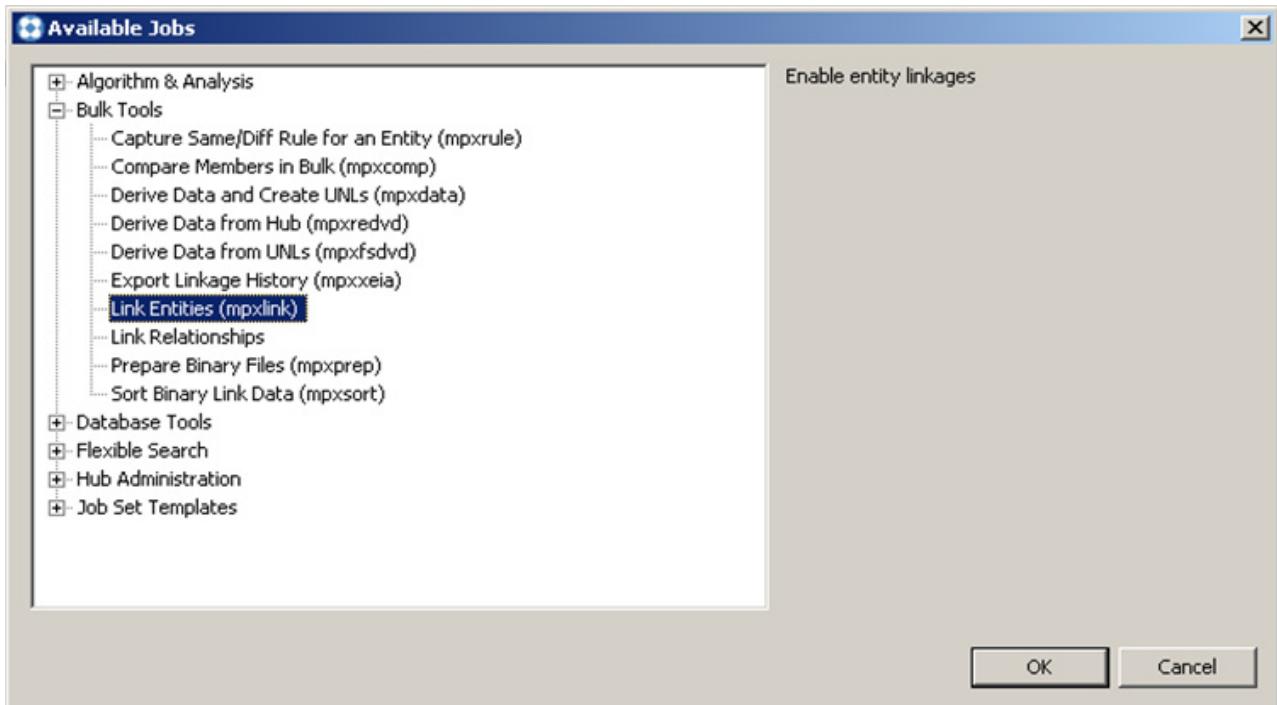
- ___ 4. Expand Bulk tools, select **Compare Members in Bulk (mpxcomp)** and click the **OK**.



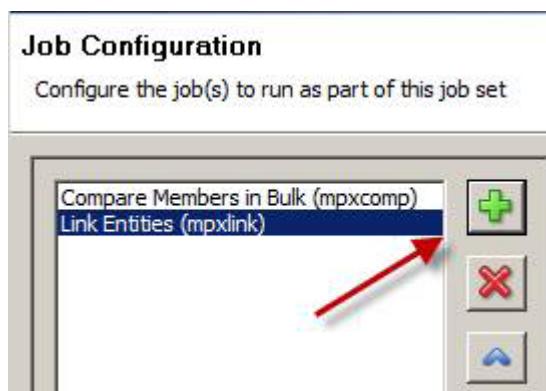
- ___ 5. Click the **Add Job** (the green plus sign) button.



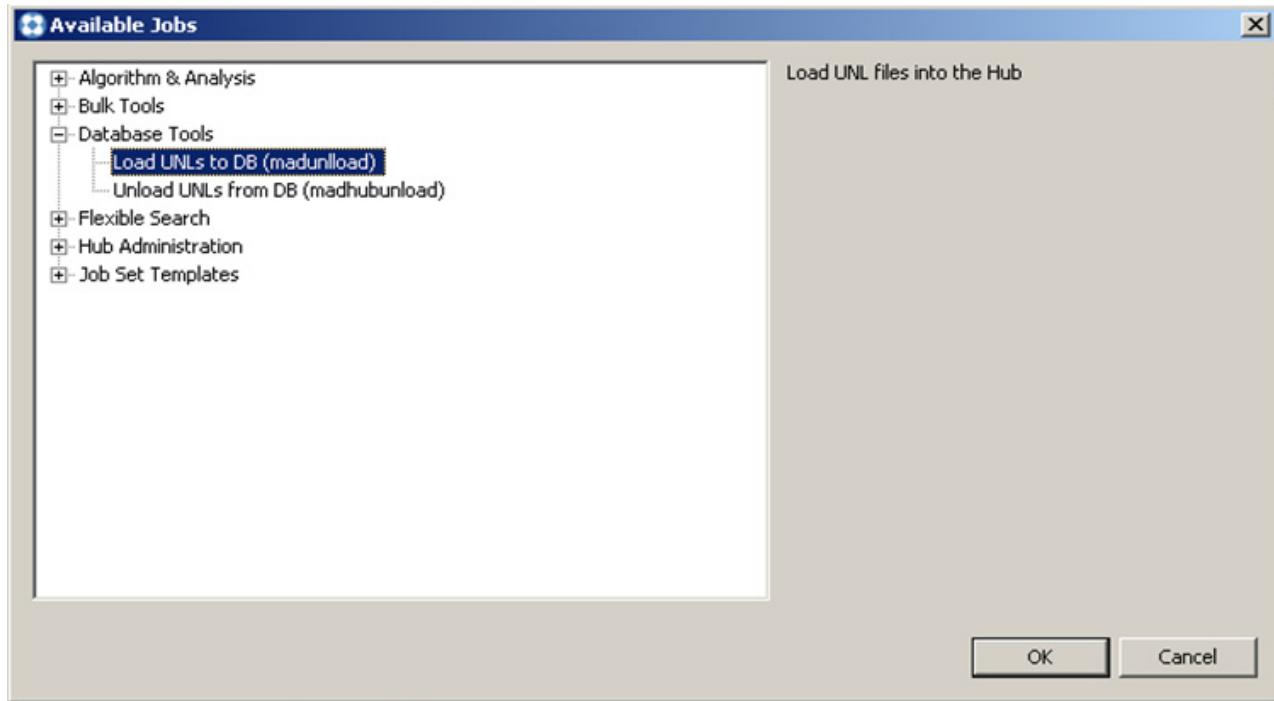
- ___ 6. Expand Bulk Tools, select **Link Entities (mpxlink)** and click the **OK** button.



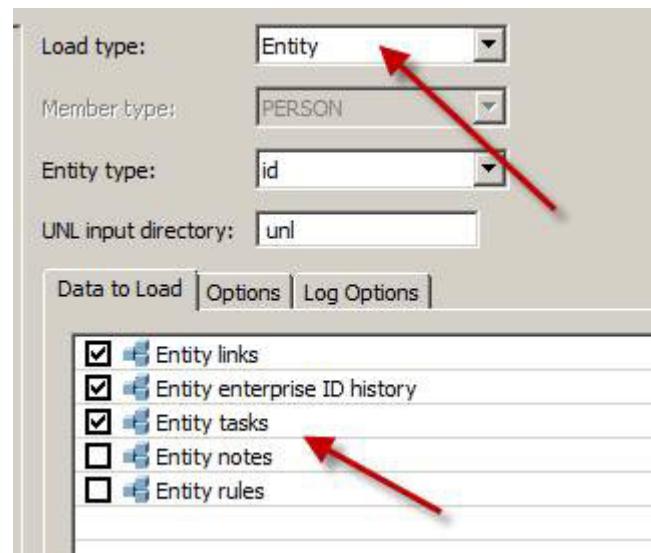
- ___ 7. Click the **Add Job** (the green plus sign) button.



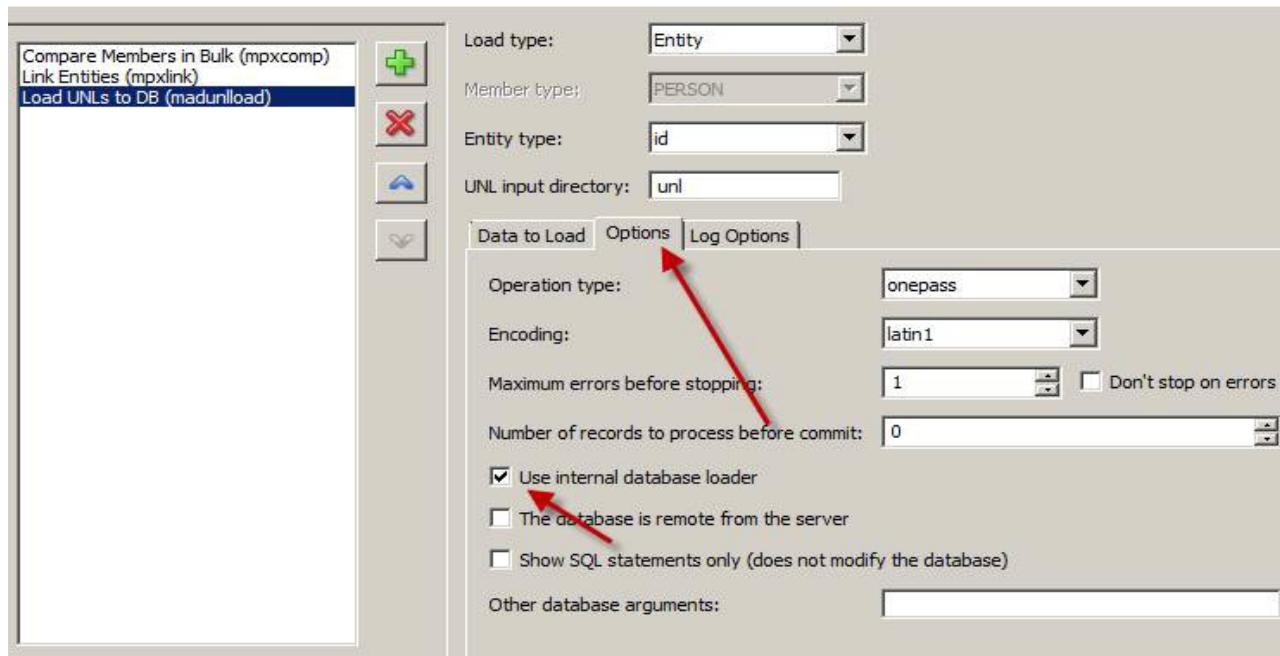
- ___ 8. Expand Database Tools, select **Load UNLs to DB (madunlload)** and click the **OK** button.



- ___ 9. Select **Entity** in the Load type dropdown and ensure that only **Entity links**, **Entity enterprise ID history** and **Entity tasks** are selected for Data to Load.



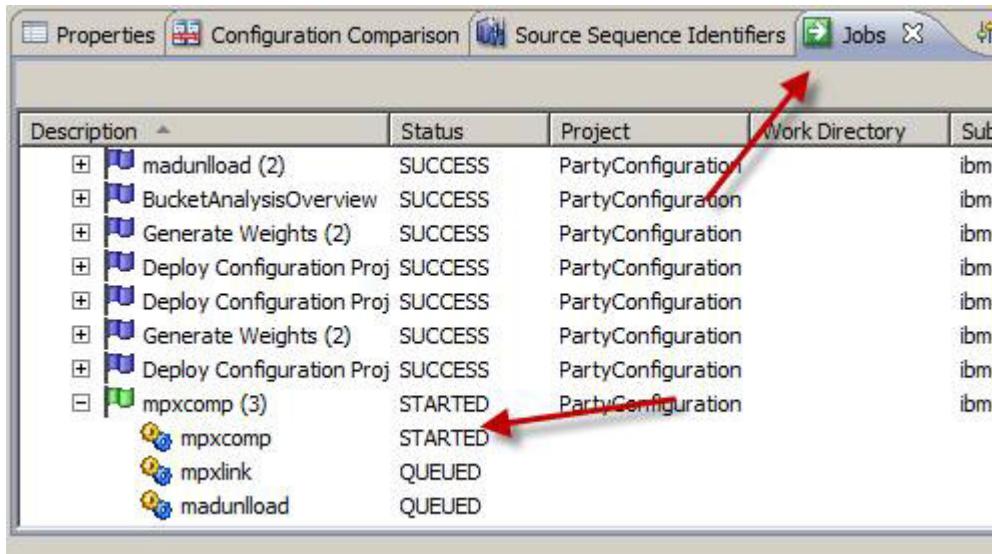
- ___ 10. Select the Options tab and check the **Use internal database loader** checkbox.



- 11. Click the **Finish** button to start running the bulk cross match job set. (This job can take up to 20 min)



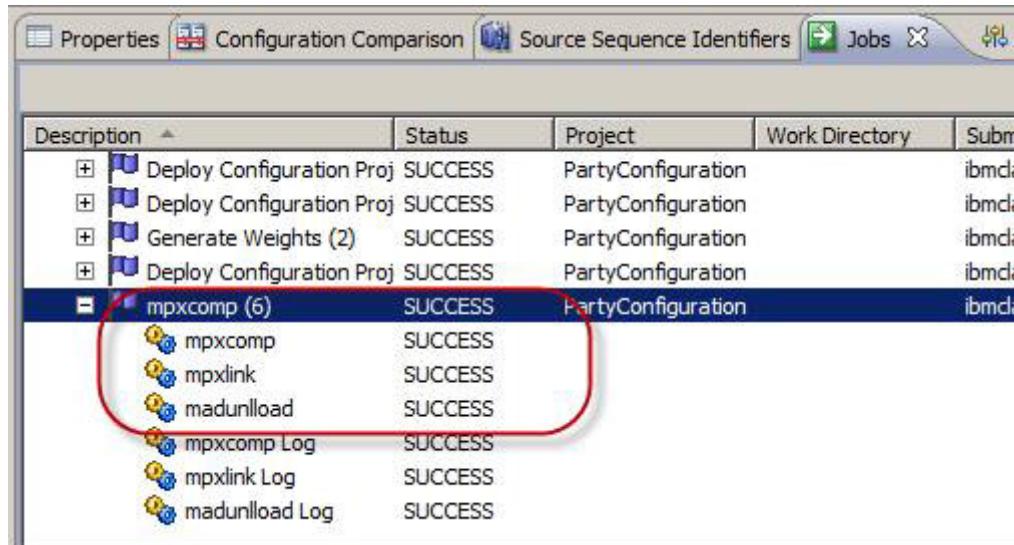
- 12. Open the RAD Jobs window and expand the **mpxcomp** job set that is running to view status.



The screenshot shows a 'Jobs' window with a table of tasks. A red arrow points from the top right towards the 'Work Directory' column. Another red arrow points from the bottom left towards the 'Status' column of the 'mpxcomp (3)' row.

Description	Status	Project	Work Directory	Subm
[+] madunload (2)	SUCCESS	PartyConfiguration	ibmc	
[+] BucketAnalysisOverview	SUCCESS	PartyConfiguration	ibmc	
[+] Generate Weights (2)	SUCCESS	PartyConfiguration	ibmc	
[+] Deploy Configuration Proj	SUCCESS	PartyConfiguration	ibmc	
[+] Deploy Configuration Proj	SUCCESS	PartyConfiguration	ibmc	
[+] Generate Weights (2)	SUCCESS	PartyConfiguration	ibmc	
[+] Deploy Configuration Proj	SUCCESS	PartyConfiguration	ibmc	
[+] mpxcomp (3)	STARTED	PartyConfiguration	ibmc	
[+] mpxcomp	STARTED			
[+] mpxlink	QUEUED			
[+] madunload	QUEUED			

13. Verify that the Bulk Cross Match job set has completed.



The screenshot shows a 'Jobs' window with a table of tasks. A red rounded rectangle highlights the 'mpxcomp (6)' row and its sub-tasks. All tasks are listed as 'SUCCESS'.

Description	Status	Project	Work Directory	Subm
[+] Deploy Configuration Proj	SUCCESS	PartyConfiguration	ibmcla	
[+] Deploy Configuration Proj	SUCCESS	PartyConfiguration	ibmcla	
[+] Generate Weights (2)	SUCCESS	PartyConfiguration	ibmcla	
[+] Deploy Configuration Proj	SUCCESS	PartyConfiguration	ibmcla	
[+] mpxcomp (6)	SUCCESS	PartyConfiguration	ibmcla	
[+] mpxcomp	SUCCESS			
[+] mpxlink	SUCCESS			
[+] madunload	SUCCESS			
[+] mpxcomp Log	SUCCESS			
[+] mpxlink Log	SUCCESS			
[+] madunload Log	SUCCESS			

We should now have Entities in our MDM Virtual Implementation.

End of exercise

Appendix D. Pair Manager and Threshold Calculations

What this exercise is about

This exercise covers how to use the IBM® Master Data Management Pair Manager to review sample pairs to determine system thresholds.

What you should be able to do

At the end of this exercise, you should be able to:

- Generate and review Threshold Analysis sample pairs
- Use Pair Manager to review Threshold Analysis sample pairs
- Set thresholds using Threshold Calculator
- Re-deploy the configuration
- Re-run Bulk Cross Match

Introduction

Pair Manager is a stand-alone tool that reads the sample pair file created from the Threshold Analysis Pair Generation job. The Threshold Analysis Pair Generation job supports dividing the generated sample pairs into multiple.xls files. This tool is primarily used during the implementation phase to speed the process of sample-pair evaluation.

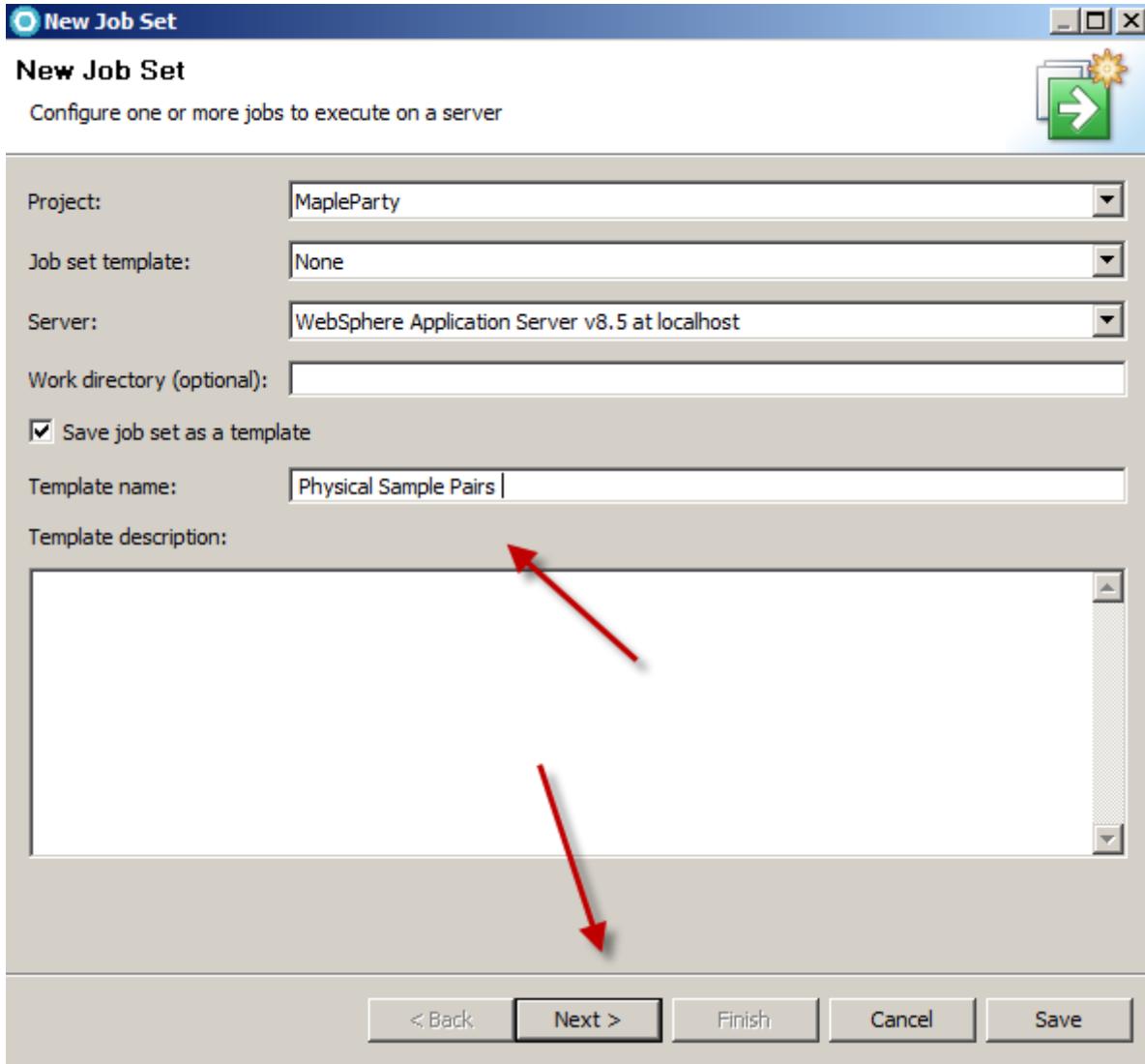
Requirements

Each of these exercise depend on the previous exercise being complete. For this exercise you must have the data model and algorithms defined in the configuration file. The configuration must be deployed with sample data loading into the database. The weights must be generated and a bulk cross match and load process must have been run on the sample data.

Exercise instructions

Part 1: Generate and review Threshold Analysis sample pairs

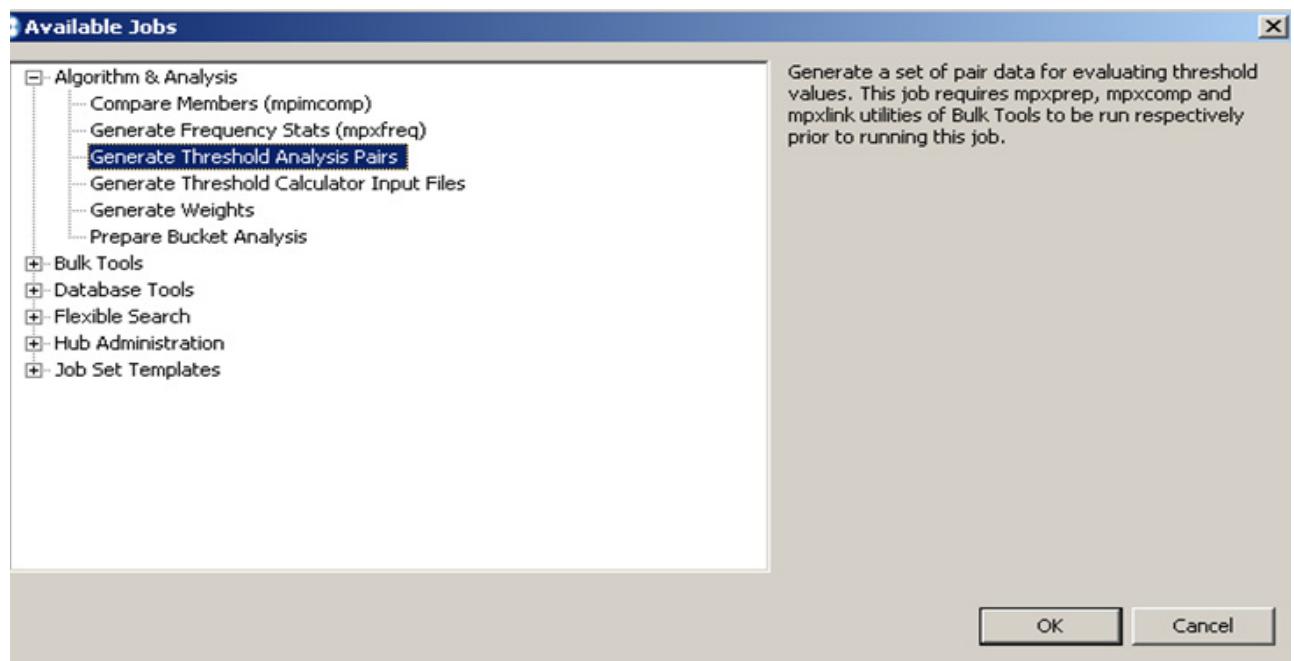
- __ 1. From the RAD file menu, select **Master Data Management > New Job Set...**
- __ 2. Enter **Physical Sample Pairs** as the job set template name and click the **Next >** button.



- __ 3. Click the **Add Job** (the green plus sign) button.

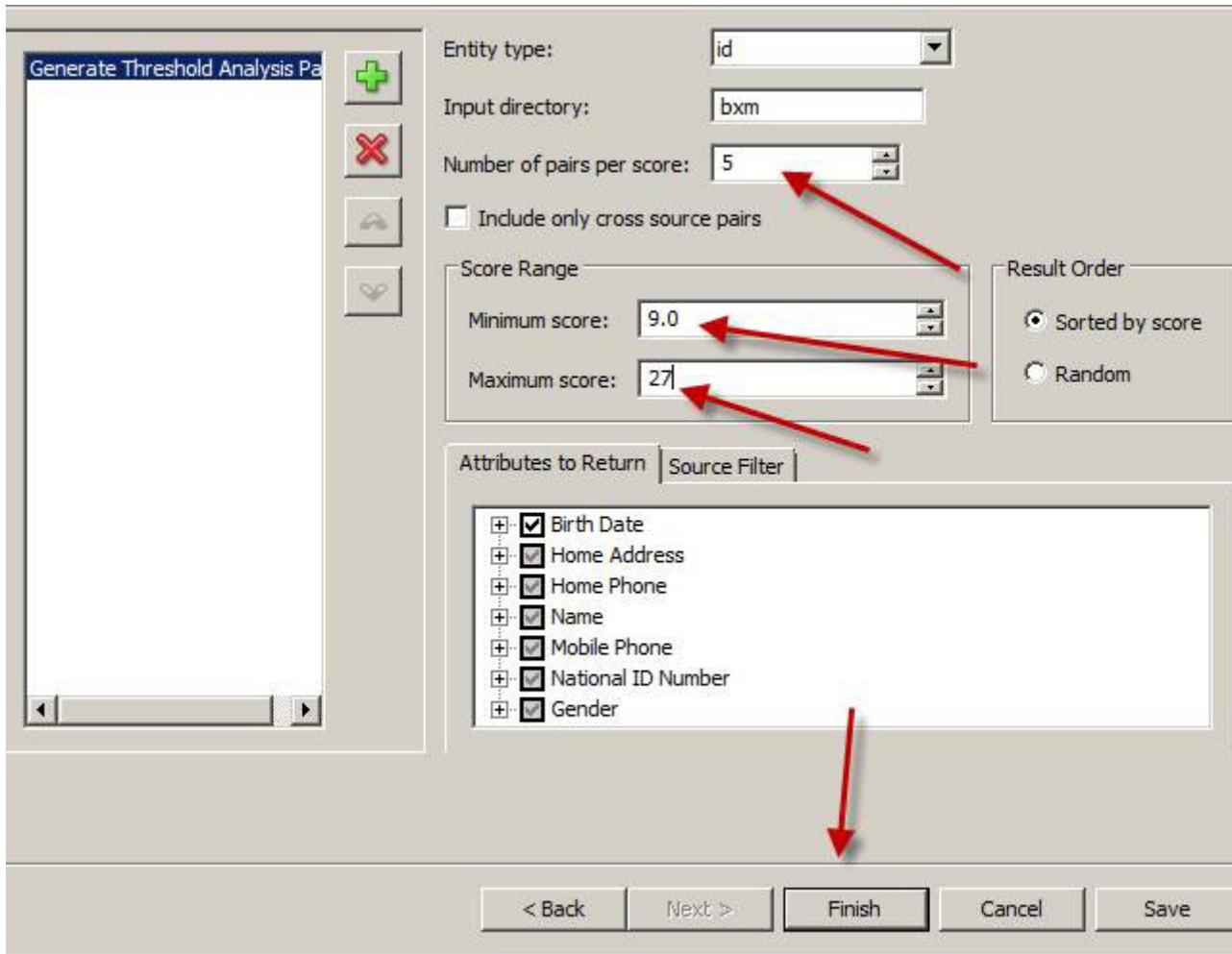


- 4. Expand Algorithm & Analysis, select **Generate Threshold Analysis Pairs** and click the **OK** button.

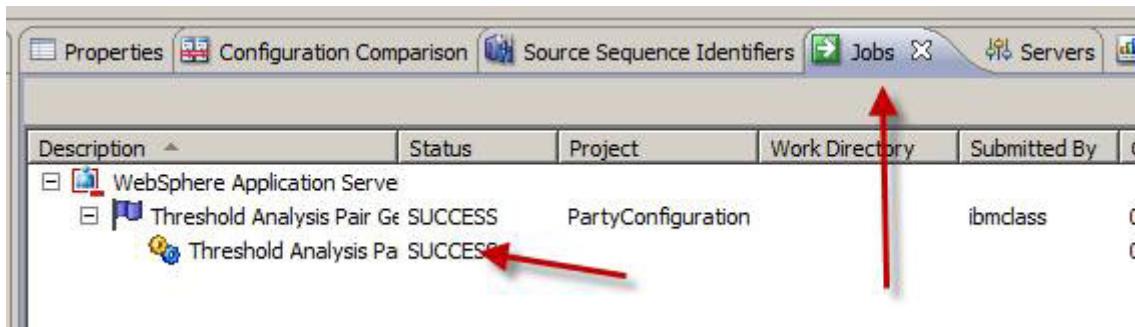


- 5. Change the following values and click on the **Finish** button:

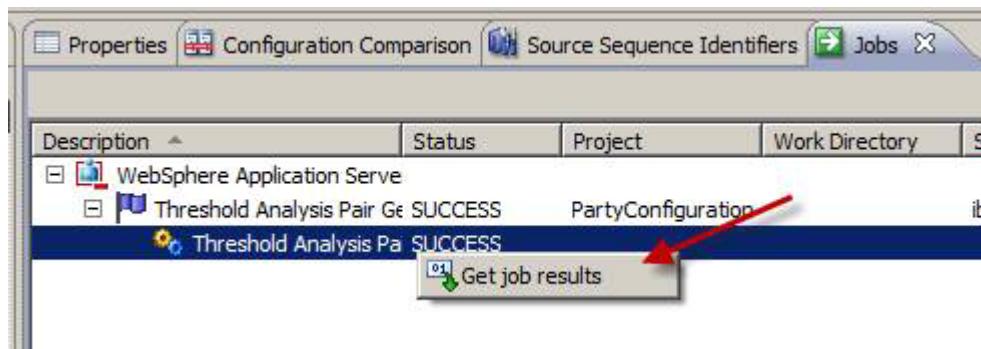
- a. Number of pairs per score: **5**.
- b. Minimum Score: **9.0**.
- c. Maximum Score: **27.0**.



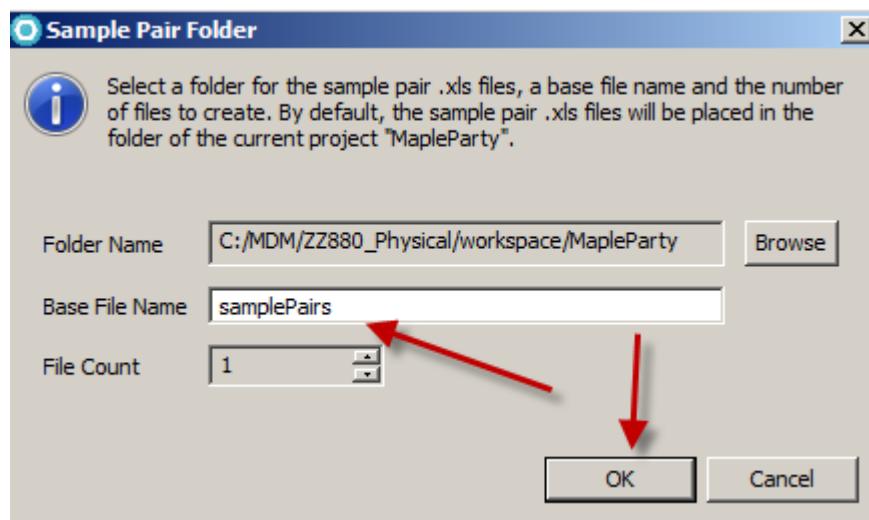
6. Validate that the job has started by verifying in the **Jobs** window.



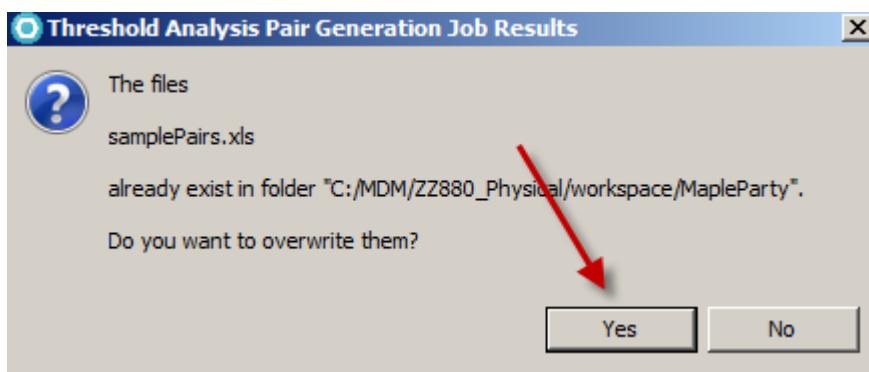
7. When job completes, right click on **Threshold Analysis Pair Generation** and select **Get job results**.



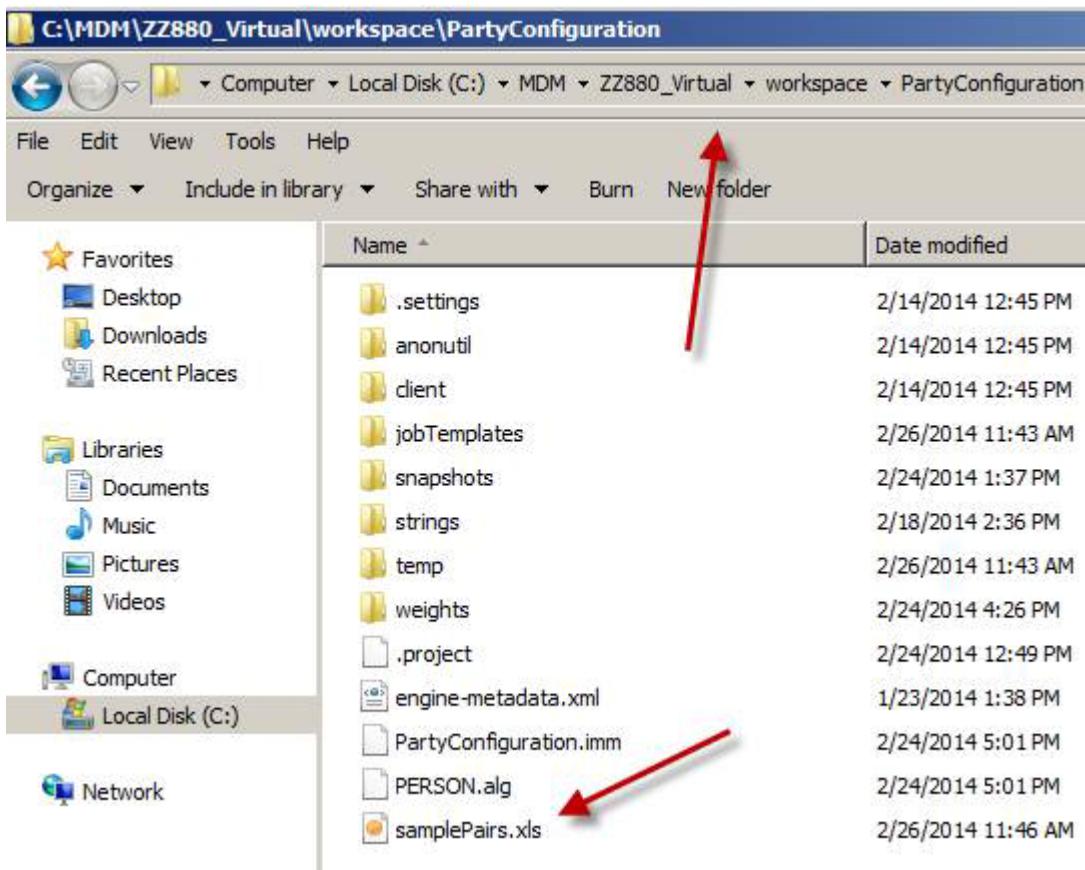
- 8. Click **OK** to use **samplePairs** as the Base File Name and to put the file in the default Folder Name.



- 9. Click **OK** to accept the job and copy the samplePairs.xls file into your project.



- 10. Open a windows explorer and navigate to **C:\MDM\ZZ880_Physical\workspace\MapleParty**. Open the **samplePairs.xls** spreadsheet.



11. Inside this file, you'll find pairs of member that are to be compared. This manual comparison will help the InfoSphere MDM learn which threshold values should be in place. You can go through the spreadsheet and change column A to Yes, No or Maybe, or we can use the Pair Manager.

samplePairs.xls - Spreadsheet - IBM Lotus Symphony

File Edit View Create Layout Data Tools Window Help

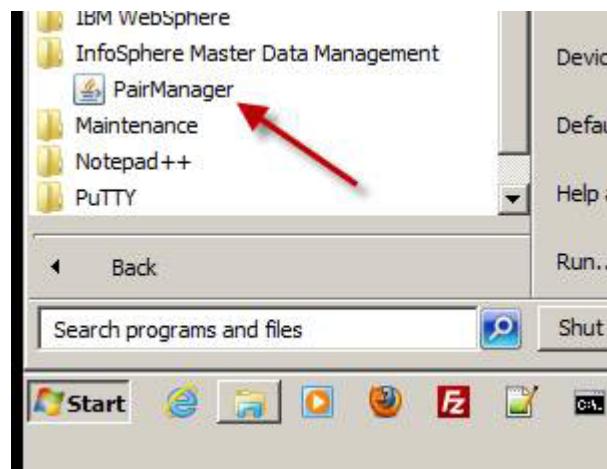
New samplePairs.xls

A2

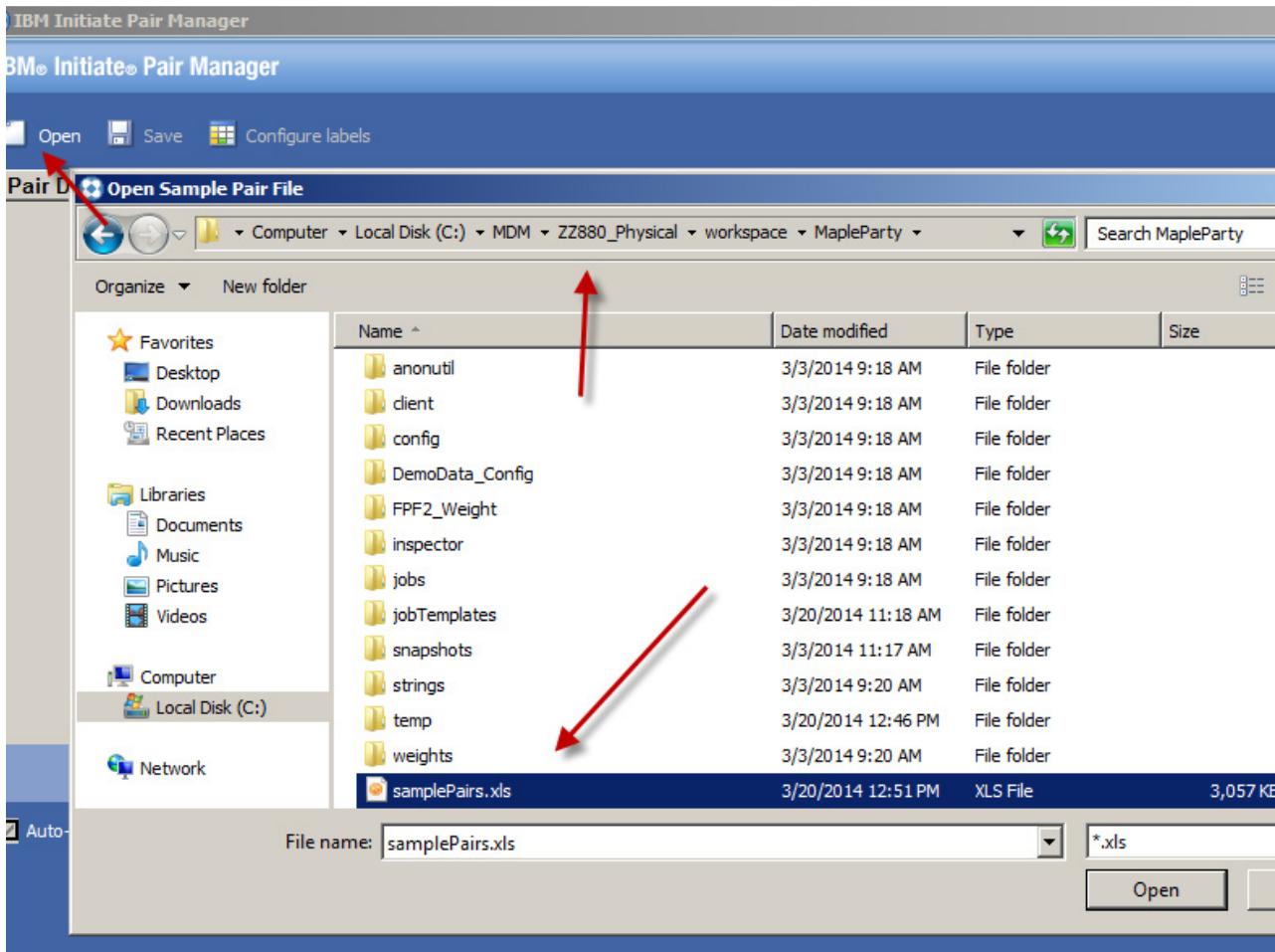
	A	B	C	D	E	F	G	H
1	Same?	Pair number	Score	Original score	memrecno	Source	Has more data?	Per AKA No Pe
2		1	90	90	163464	MDMS Person	FALSE	
3	Yes	1	90	90	410335	MDMS Person	FALSE	
4	No	2	90	90	242490	MDMS Person	FALSE	
5	Maybe	2	90	90	432828	MDMS Person	FALSE	
6		3	90	90	309372	MDMS Person	FALSE	
7		3	90	90	400613	MDMS Person	FALSE	
8		4	90	90	146760	MDMS Person	FALSE	
9		4	90	90	448975	MDMS Person	FALSE	
10		5	90	90	82927	MDMS Person	FALSE	
11		5	90	90	459582	MDMS Person	FALSE	
12		6	91	91	376939	MDMS Person	FALSE	

Part 2: Review Threshold Analysis sample pairs

- 1. Start the Pair Manager from the Windows Start menu. Select **Start > All programs > InfoSphere Master Data Management > Pair Manager**



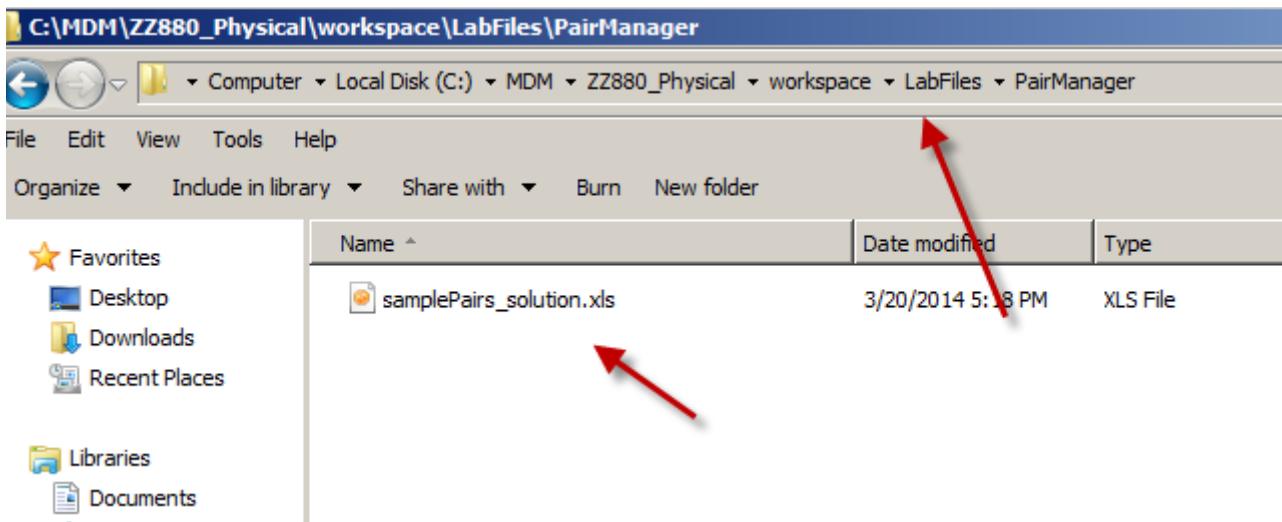
- 2. Click the **Open** icon and navigate to C:\MDM\ZZ880_Physical\workspace\MapleParty



3. Select **samplePairs.xls** and click **Open** to display the first row of sample pairs.

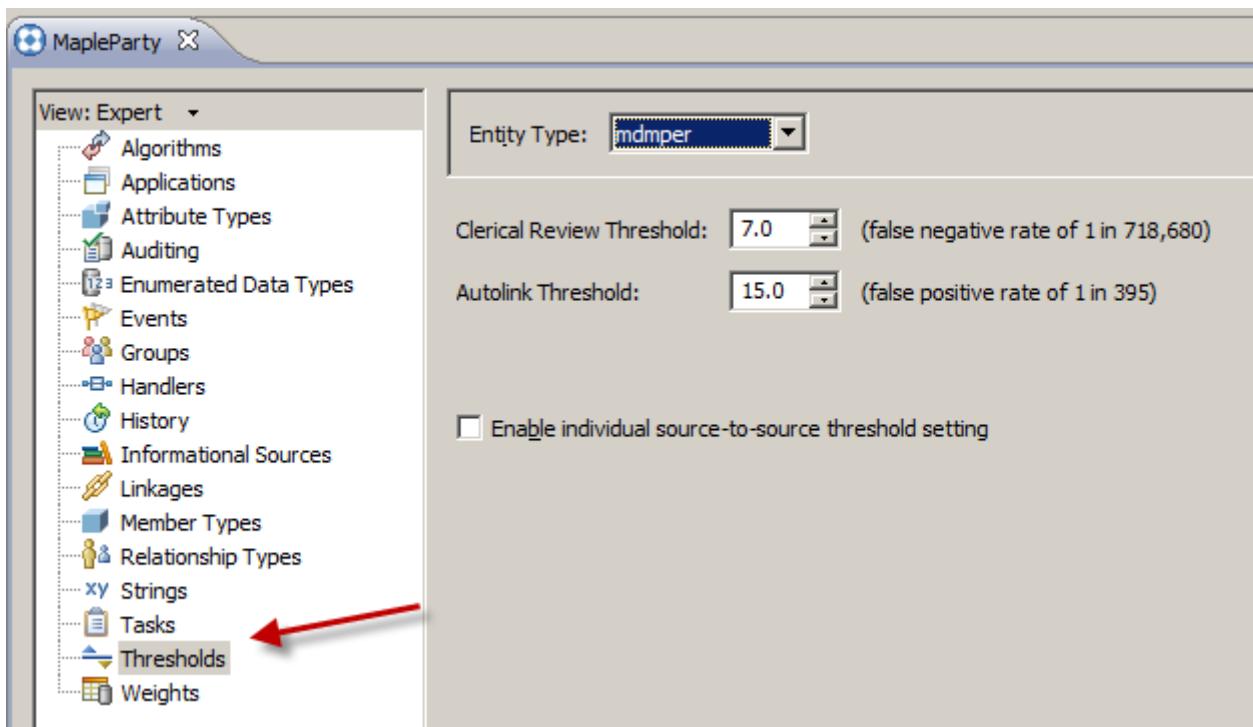
- 4. Review the sample pair set and indicate:
- **Yes** – green checkmark to say that they are a matching pair.
 - **No** – red line through circle to say that they are NOT a matching pair.
 - **Maybe** – yellow question mark to say that there is not enough information for you to decide if they are a matching pair or not.
- (NOTE: you don't have to review all of them, we have a solution file for you)
- 5. Once you feel comfortable with the Pair Manager and how it works, save the file, you will notice that it also produces an xls file. (The xls file is exactly the same as the one we loaded with the first column populated during your reviews)

We have provided you with a solution file that can be found under
C:\MDM\ZZ880_Physical\workspace\LabFiles\PairManager\samplePairs_solution.xls.
We will use this file for our Threshold Calculations,

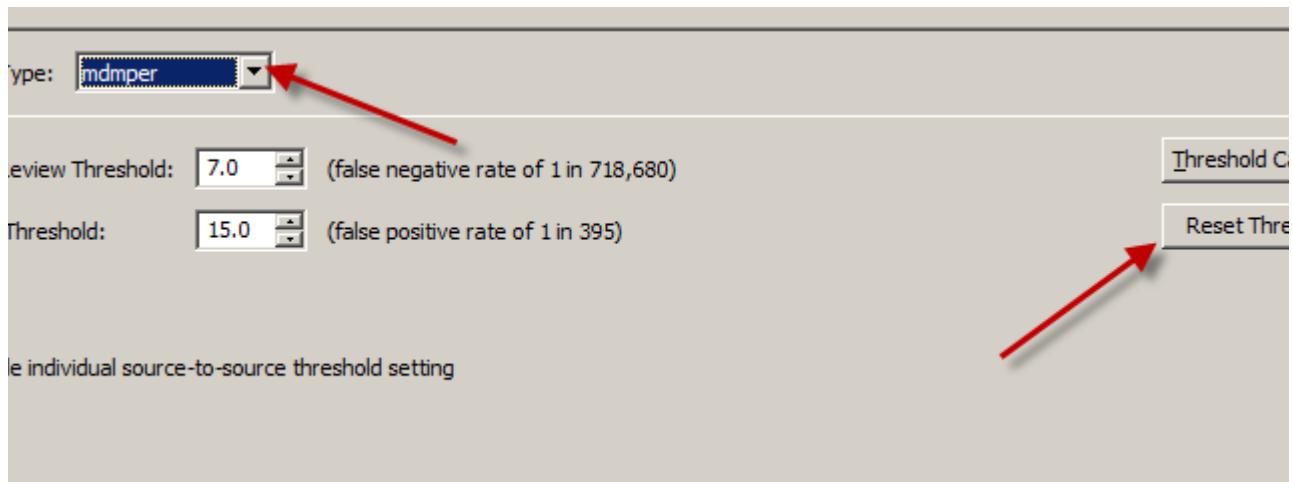


Part 3: Setting new thresholds using Threshold Calculator

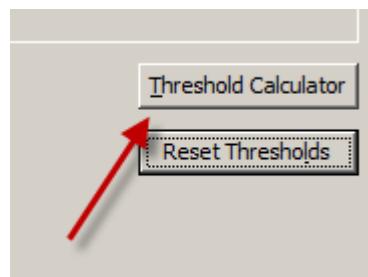
1. Back in the RAD environment, open the MapleParty.imm file and select the **Thresholds** tab.



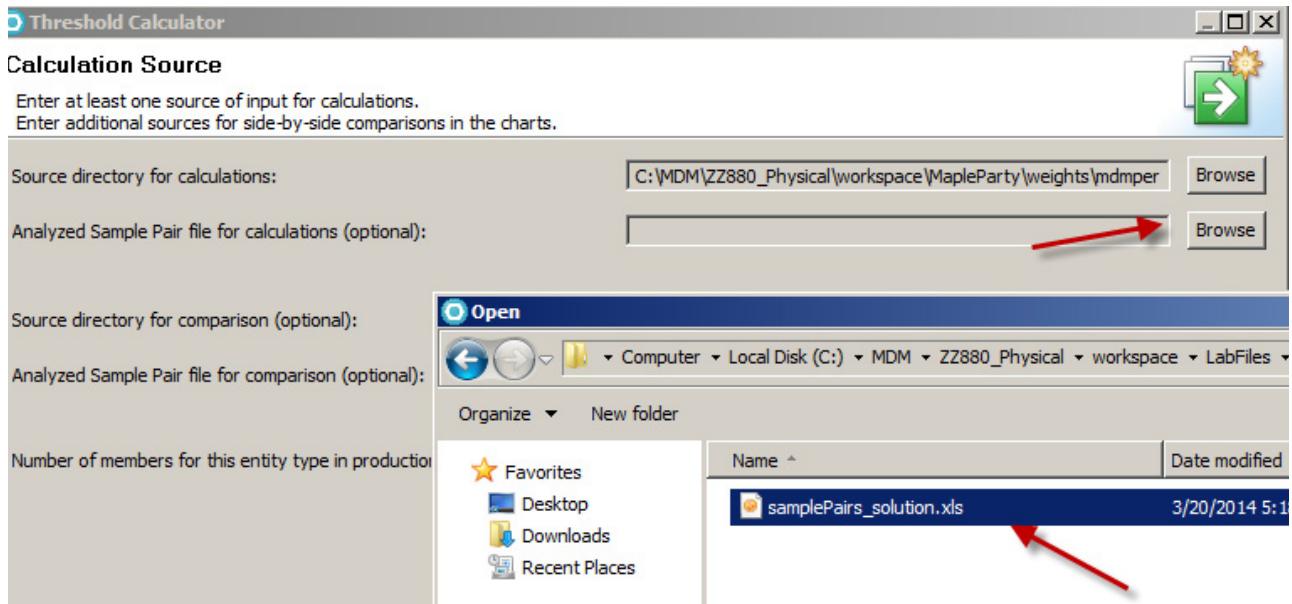
2. Select the **mdmper** Entity Type and click the **Reset Thresholds** button.



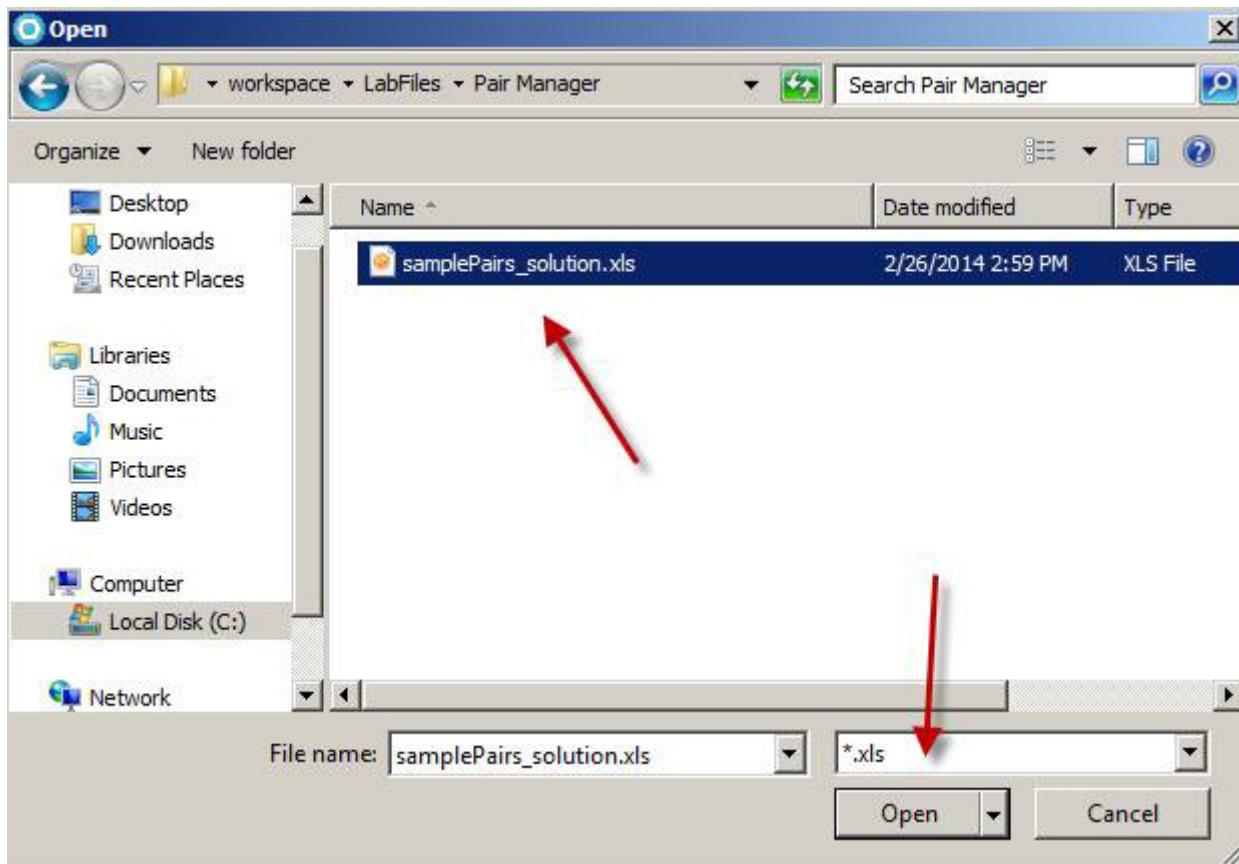
- ___ 3. Click the **Threshold Calculator** button.



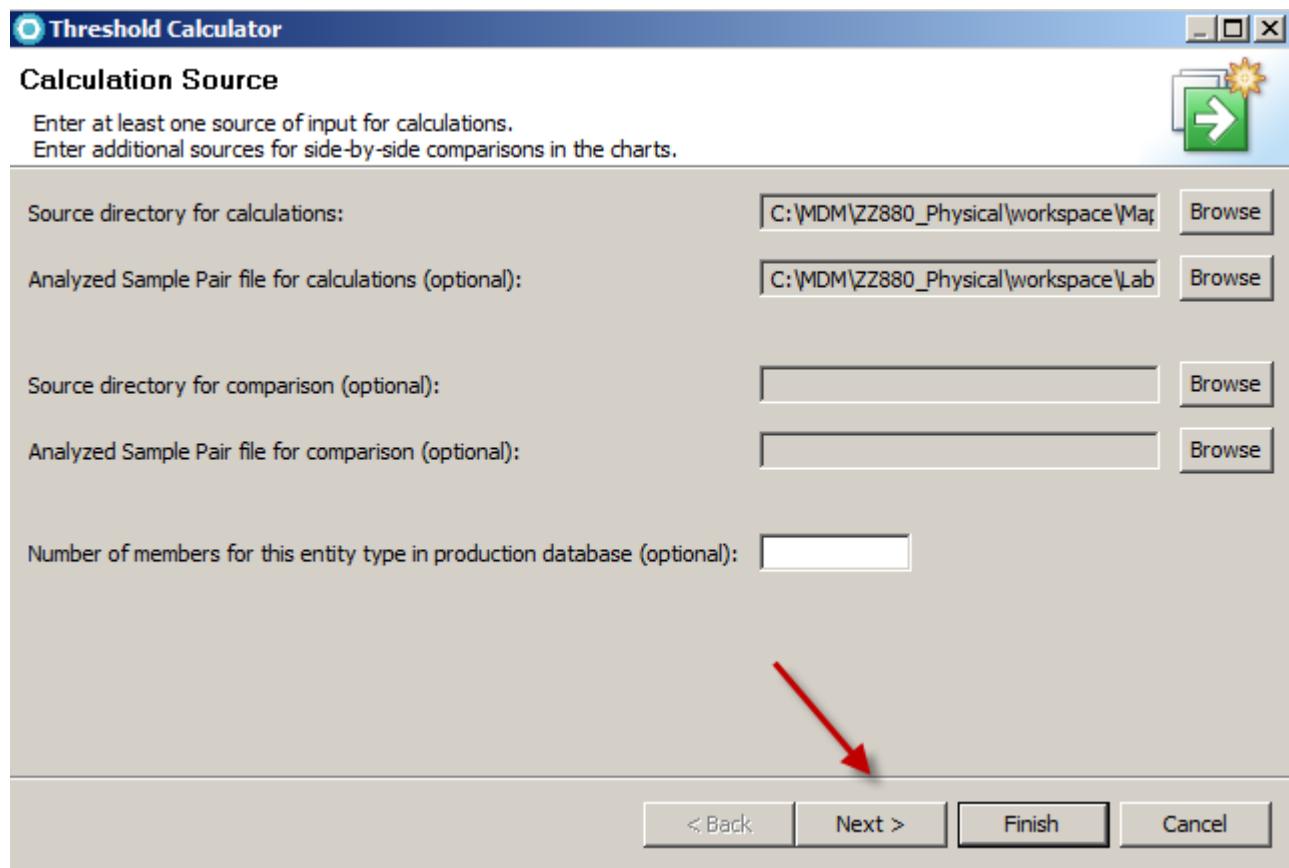
- ___ 4. Click the **Browse** button to change property **Analyzed Sample Pair file for calculations (optional)**: navigate to the directory
C:\MDM\ZZ880_Physical\workspace\LabFiles\PairManager

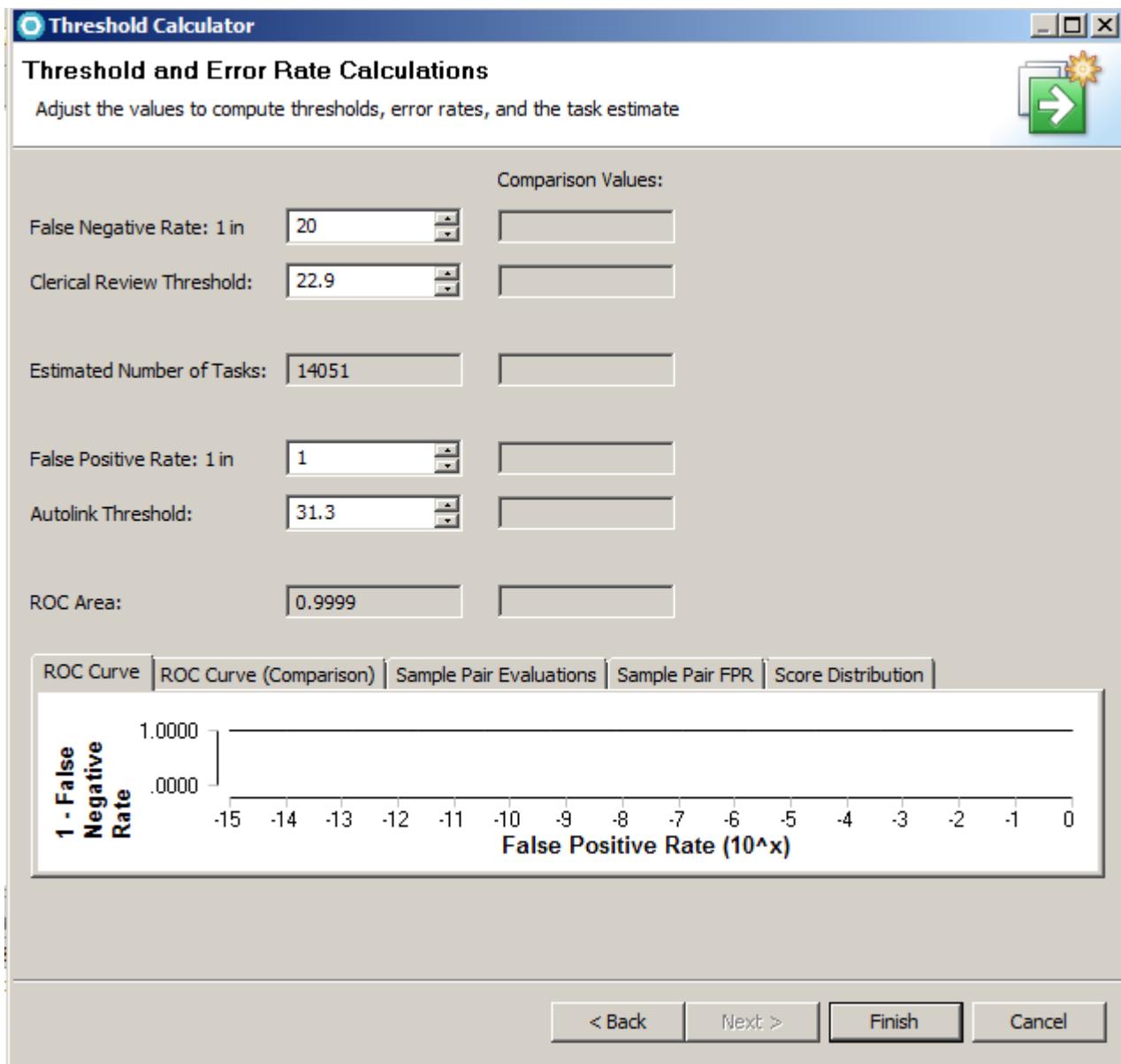


- ___ 5. Select **samplePairs_solution.xls** and click **Open**.

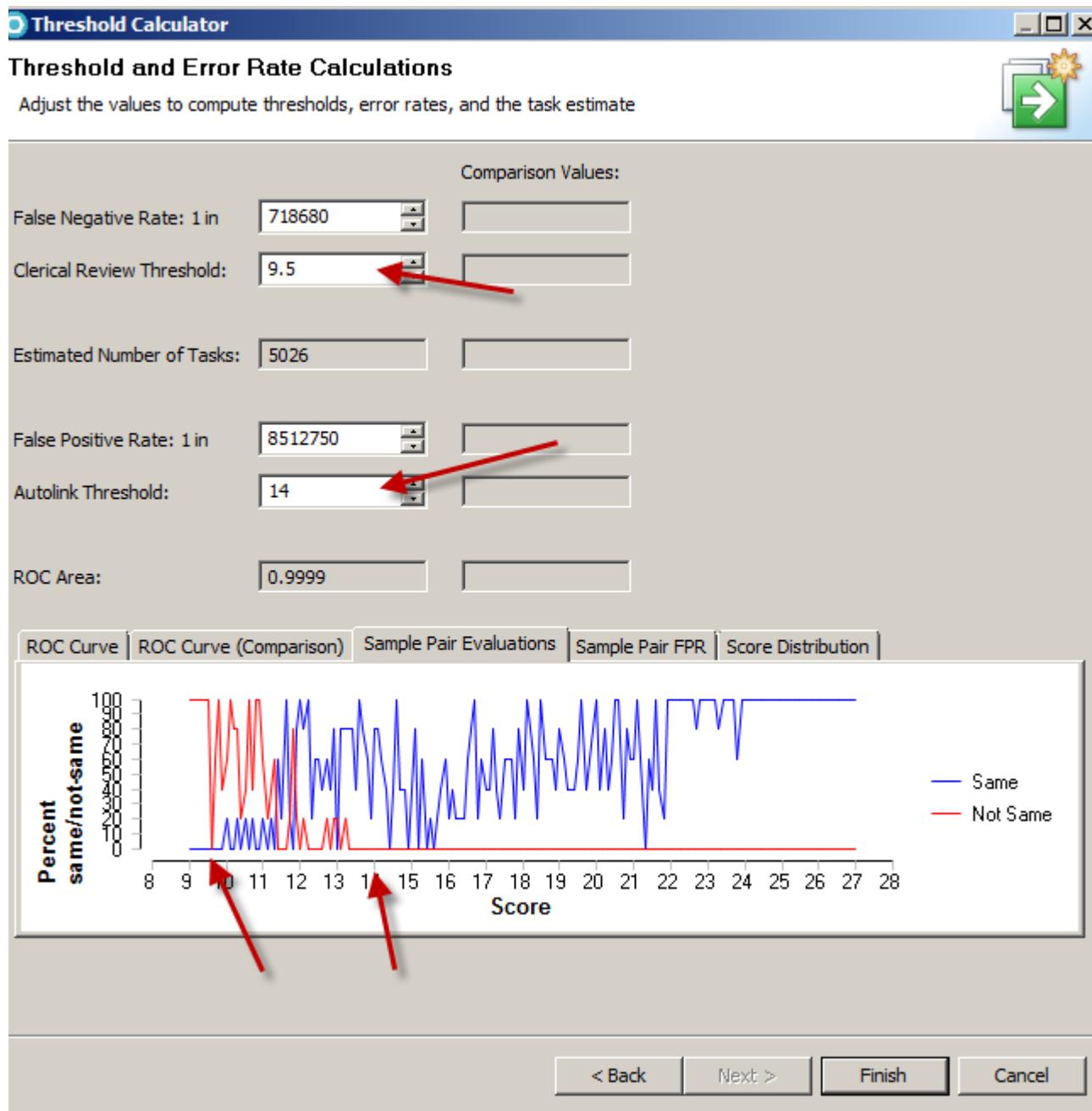


- ___ 6. Click the **Next >** button to display the ROC (Receiver Operating Characteristics) curve.

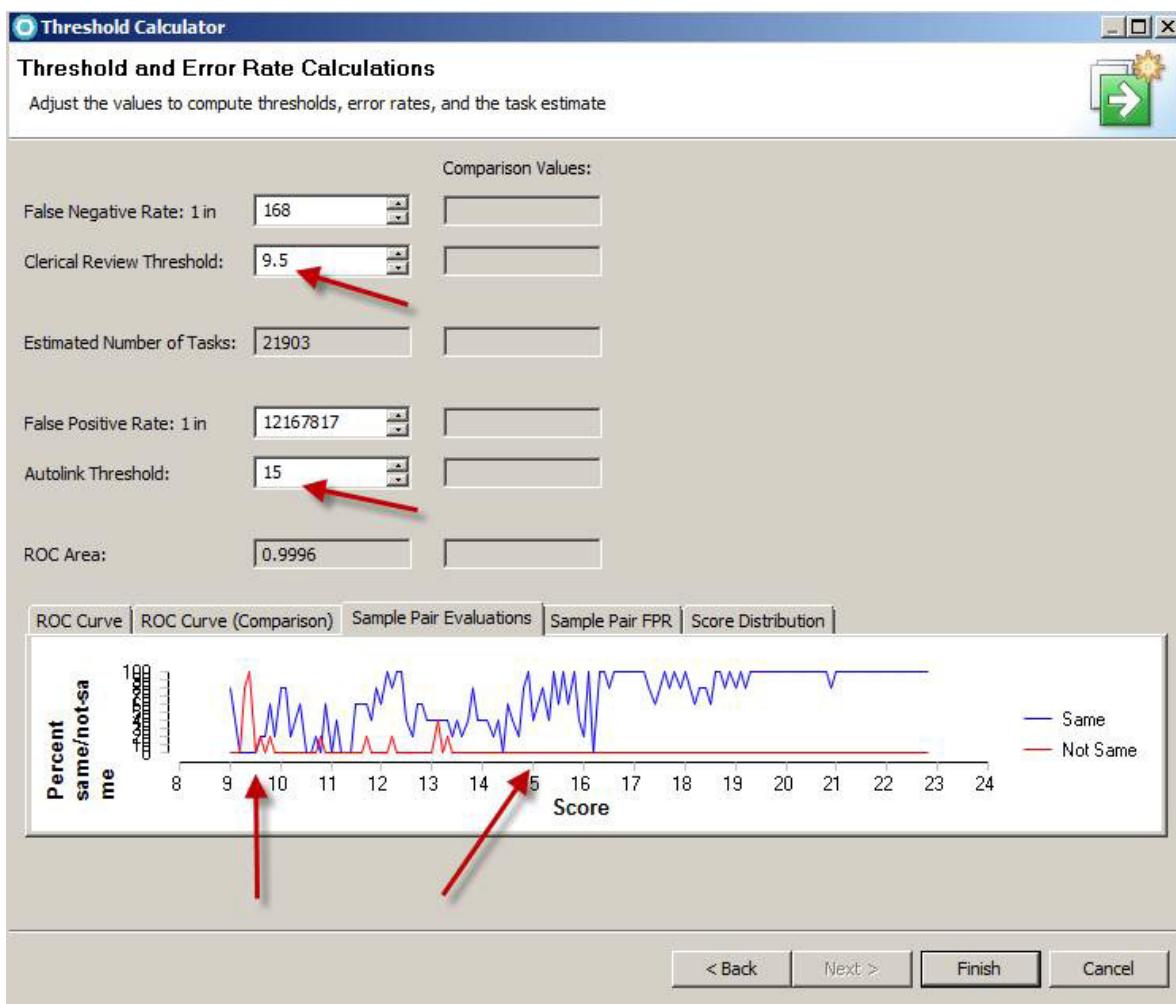




7. From here you can see the default values provide us with a bad False Positive Rate and False Negative Rate. Select the **Sample Pair Evaluations** tab to view the results of the Pair Manager output.



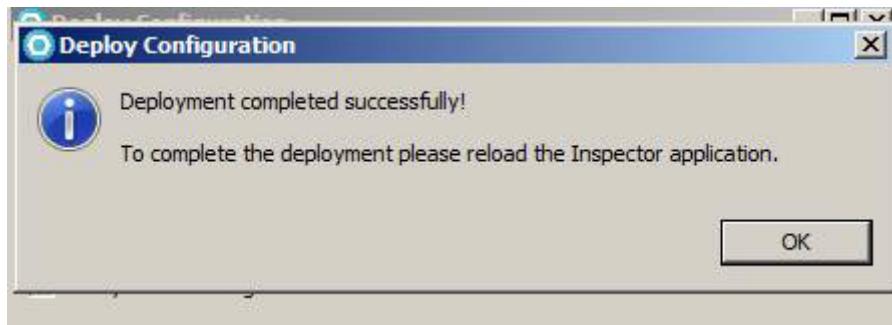
8. These graphs allow you to evaluate what would be a good threshold based on the sample data. Based on a quick analysis, it looks like between 9.5 and 14 there are a mix of "Same" and "Not Same", so we would like to keep that range as our Clerical Review. Set the property **Clerical Review Threshold** to **9.5** and the **Autolink Threshold** to **14** and click on the **Finish** button to set the Thresholds.



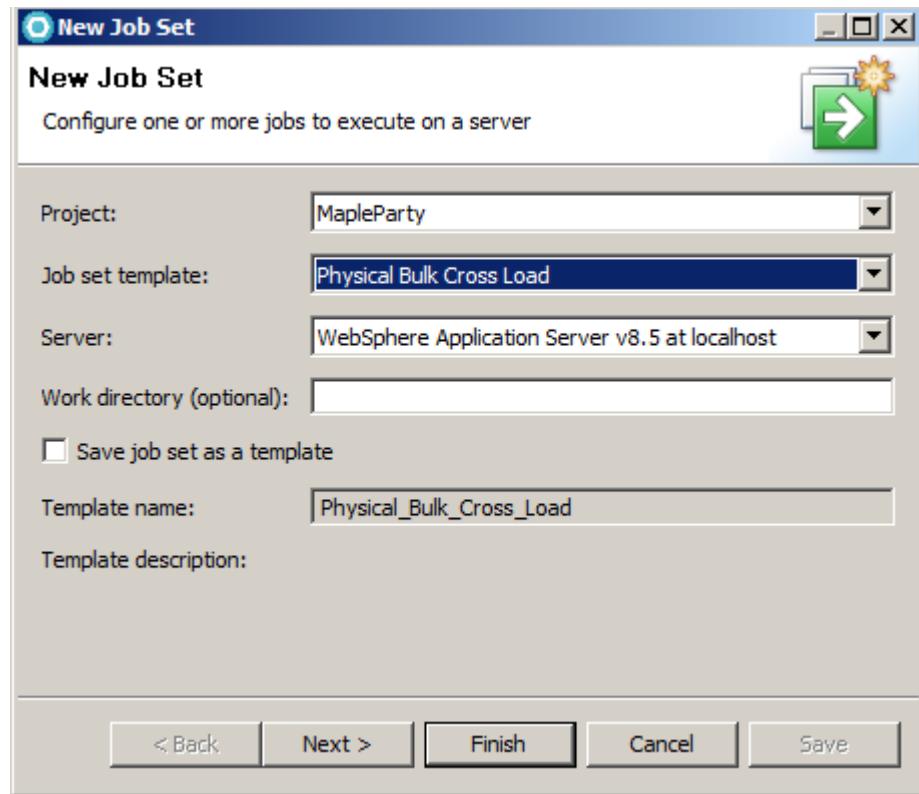
- 9. Save the project (CTRL+S)

Part 4: Re-deploy the configuration

- 1. Redeploy the configuration (from the RAD menu, select **Master Data Management > Deploy Hub Configuration....**)



- 2. Re-run the Physical Cross Load job (from the RAD menu, select **Master Data Management > New Job Set**)



End of exercise

Appendix E. Entity Analytics

What this exercise is about

This exercise covers viewing our Entities using the reporting tools available in the workbench.

What you should be able to do

At the end of this exercise, you should be able to:

- View an Entity distribution report
- View the Member comparison data of an Entity
- View the Entity Member breakdown

Introduction

Now that we have created our thresholds and re-ran the bulk cross load, we can view how Entities that were linked and created in our solution. We will use the analysis section of the workbench similar to our Bucket analysis exercise.

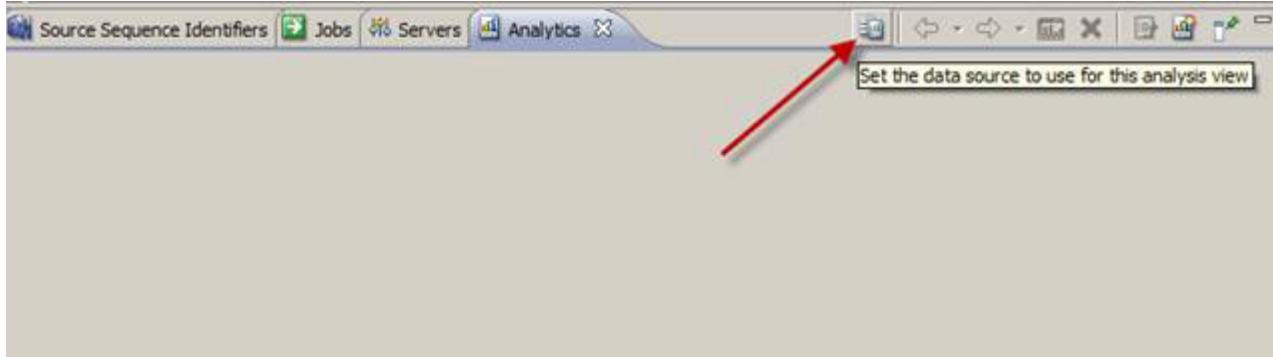
Requirements

Each of these exercise depend on the previous exercise being complete. For this exercise you must have the data model and algorithms defined in the configuration file. The configuration must be deployed with sample data loading into the database. The weights must be generated and a bulk cross match and load process must have been run on the sample data.

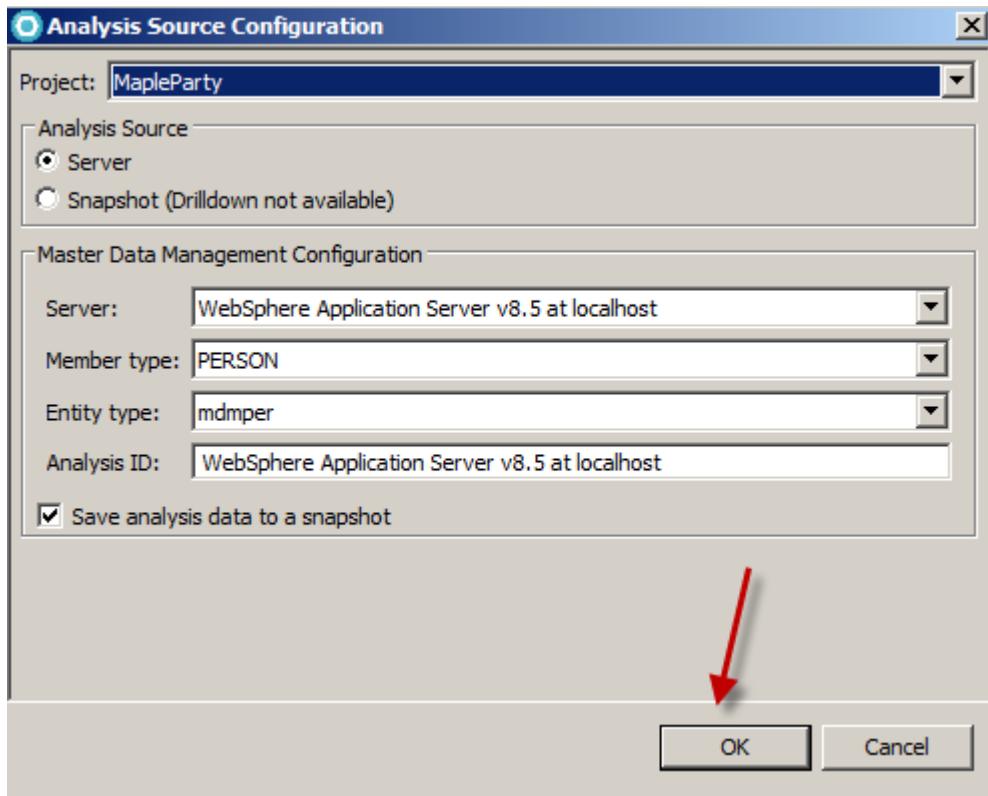
Exercise instructions

Part 1: Entity analysis overview

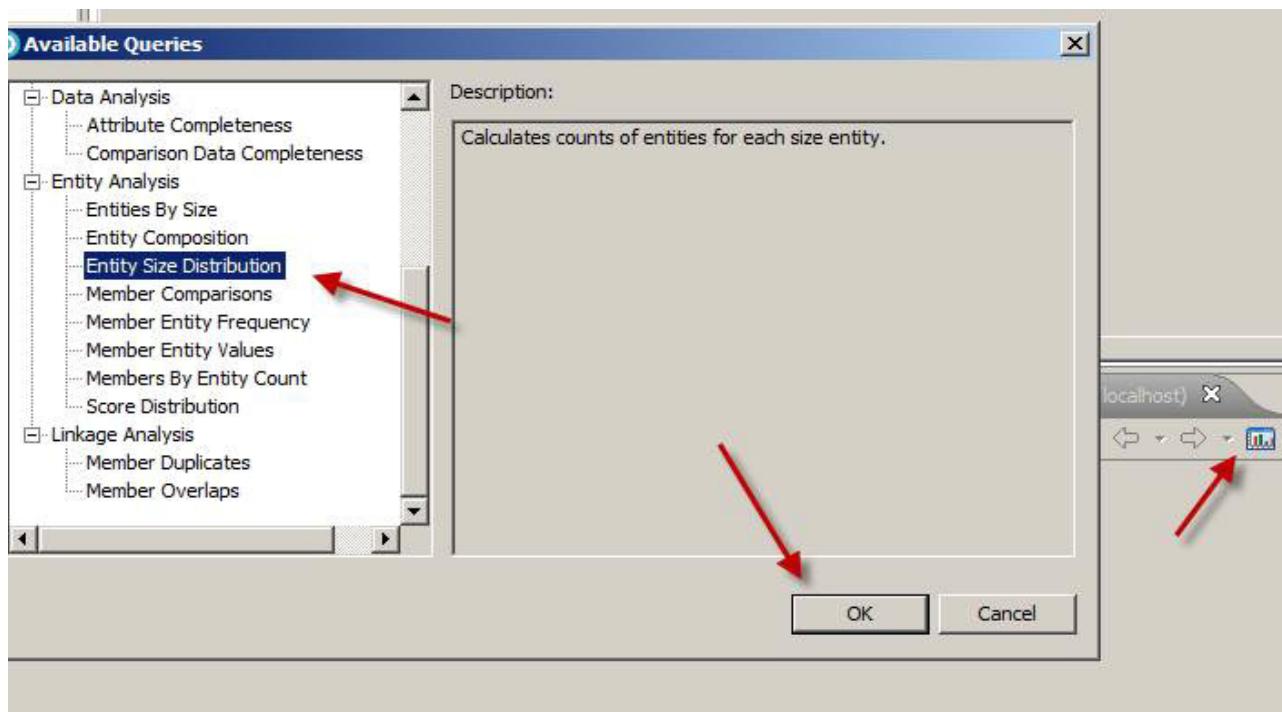
- 1. Open the **Analytics** window in RAD and if not already set, click on the **Set the data source to use for this analysis view** button.



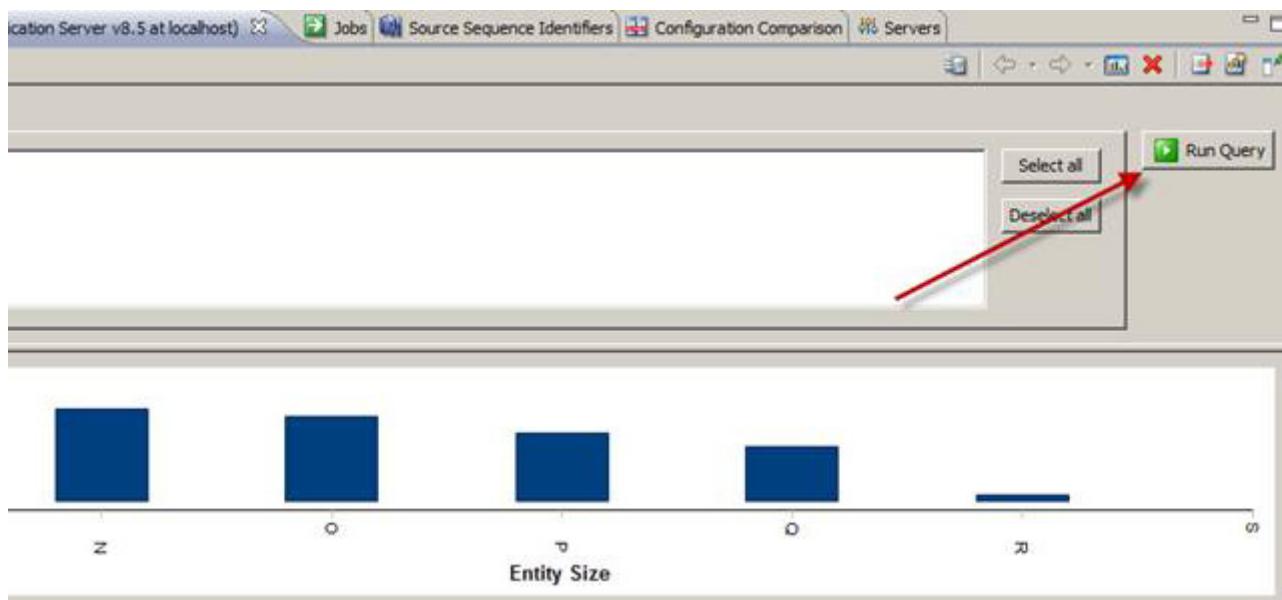
- 2. Accept the default values and click the **OK** button



- 3. Click the Add Query button and select the **Entity Size Distribution** query found under the Entity Analysis folder. Click the **OK** button.

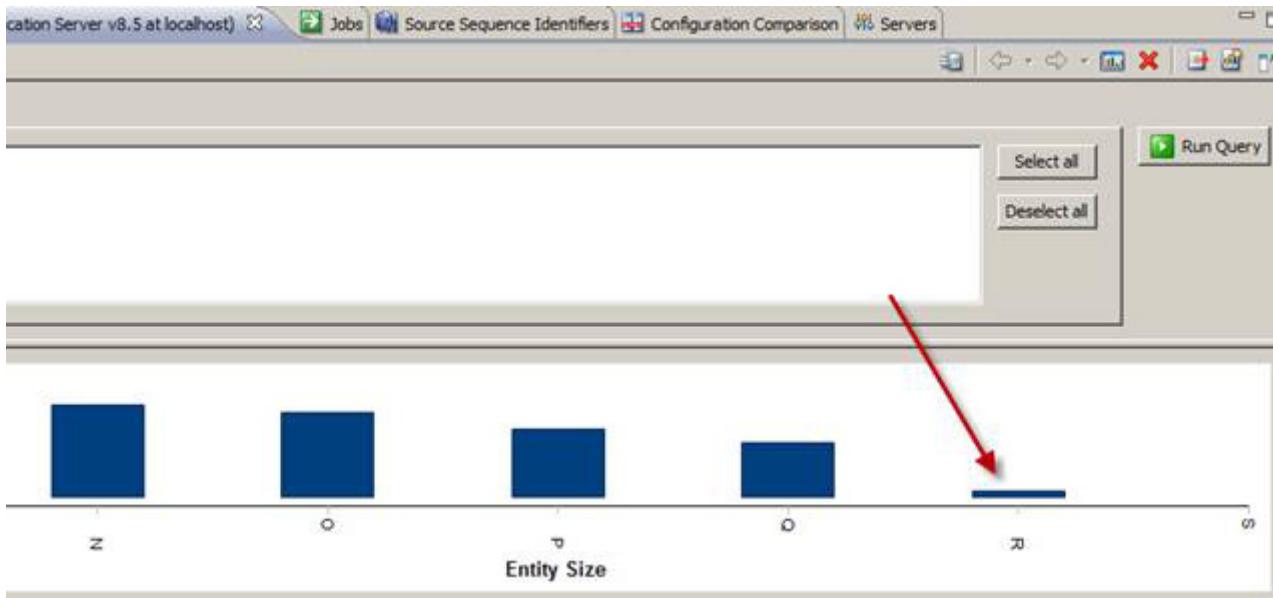


- 4. Click the **Run Query** button

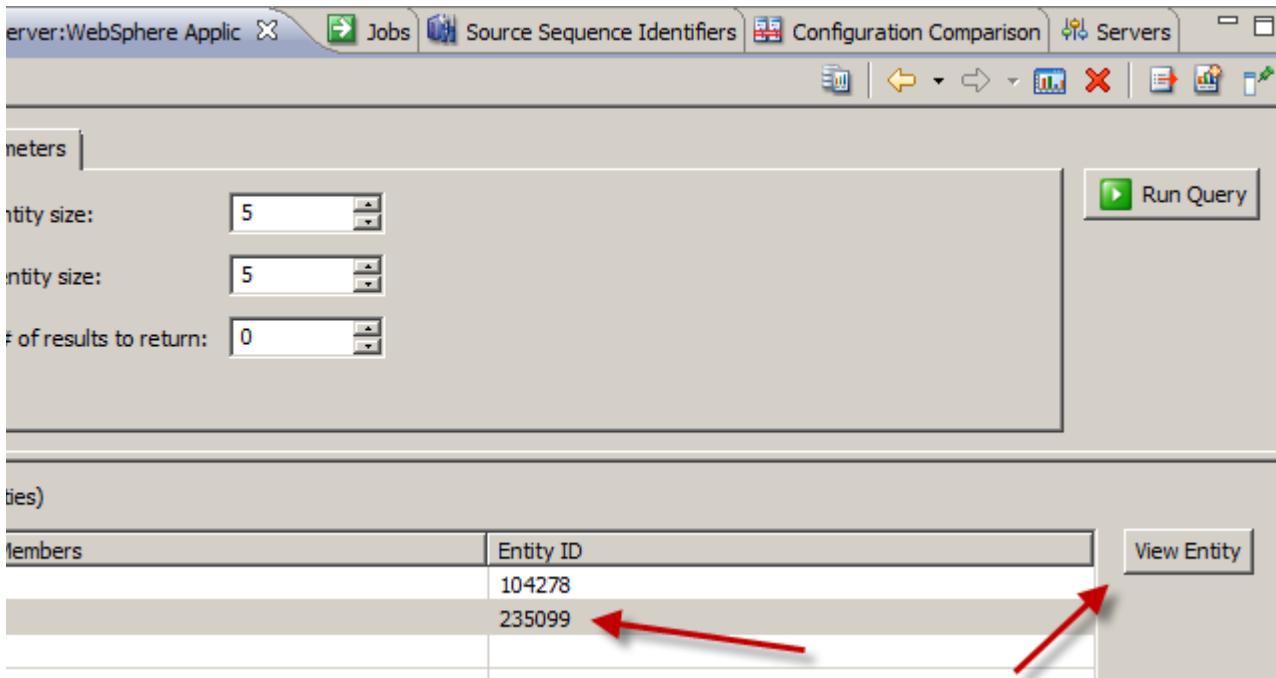


From this report you can see that we have a number of entities made up of 1 Member, 2 Members, 3 Members 4 Members and 6 Members. The Entities with 2, 3, and 4 have been created based on our algorithm and have the comparison score > 14. You can drive down the query by clicking on one of the bars to see the details.

- 5. Large entities (with many records) could be created based on thresholds that are too low (false positives). Click on the **5th** bar (indicating 5 members) to see the details.



6. In this screen we can see that we have 2 entities that are made up of 5 records. Select Entity **235099** and click the **View Entity** button.



From this screen you can see the details of each of the members and how they compare to each other.

7. Select the records **318509** and **433948** and click the **Compare Members** button.

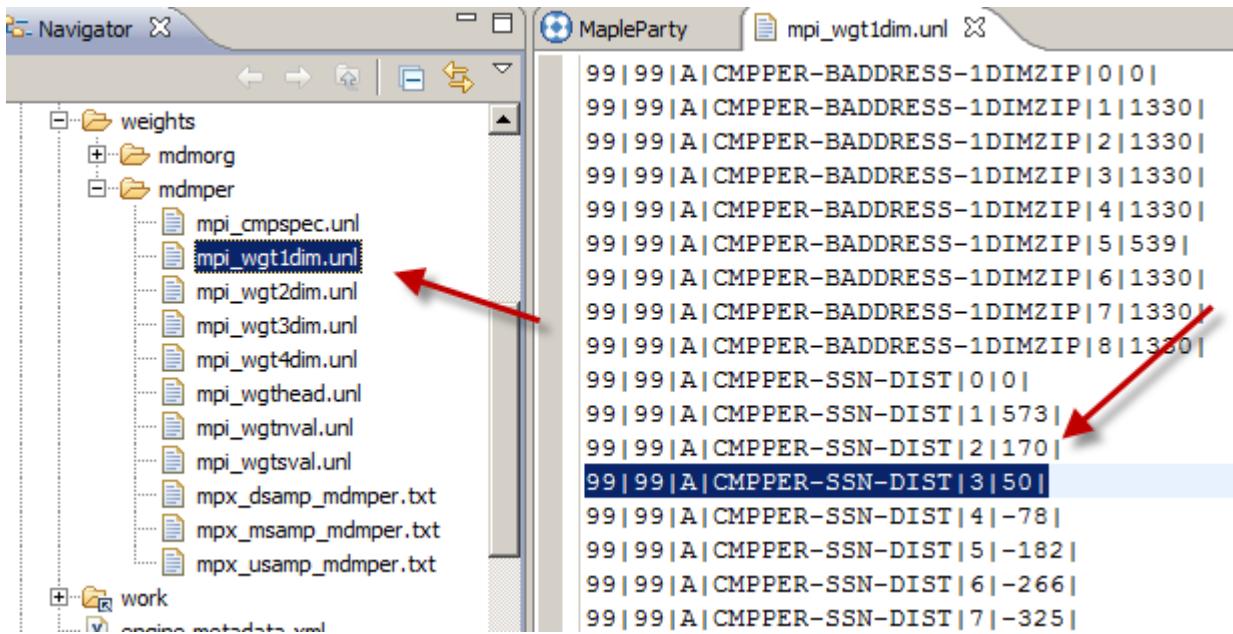
(Found 5 members)

M...	So...	1 (Name)	2 (Postal C...)	3 (Personal Address)	4 (SSN)	5 (...)	6 (...)	7 (...)
246843	1	ZAMZOW:BART:M..	85275	N-18668:S-PEACH:S-ST:.S-M...	247781514	22...	19...	M
248853	1	ZAMZOW:BART:M..	85275	N-18668:S-PEACH:S-ST:.S-M...	247781514	22...	19...	M
318509	1	ZAMZOW:BART:M..	85275	N-18668:S-PEACH:S-ST:.S-M...	247781514	22...	19...	M
433948	1	ZAMZOW:BART:M..			247718541	22...		M
434483	1	ZAMZOW:DOUG:B..				22...	19...	M

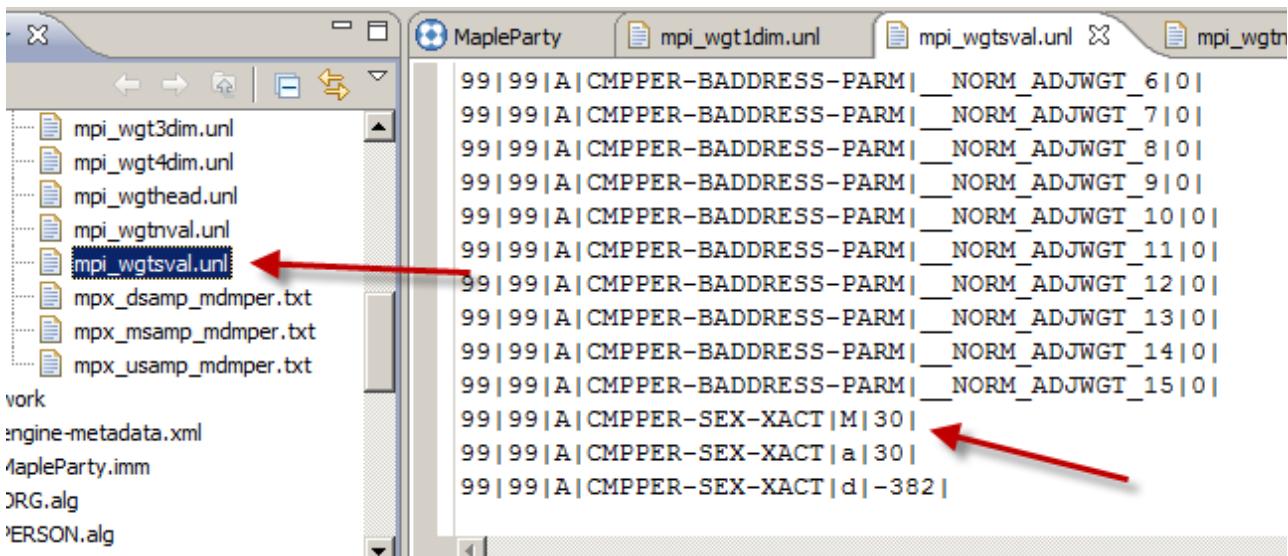
8. Remember in a previous exercise how we manually figured the Match score of our searches, in this view we can see the breakdown of the score between the members and how the match score is calculated.

Candidate ...	Score	cmpspecc...	Proband Value	Candidate Value	Matc...	Weight
318509	18.1					
		XNM	ZAMZOW BART M	ZAMZOW BART M	E	+7.17
		PADDRESS	2297012	2297012	P	+4.47
		BADDRESS	6839 WHITLEY ST PHOEN	-- 18668 PEACH ST MESA A	c	gt= -1
		SSN	247718541	247781514	D	+0.50
		DOB			M	+0.00
		SEX	M	M	E	+0.30
		SIN			M	+0.00
		DRLC	BZ247718541	BZ247781514	D	+1.89
		USRNM	BMZAMZOW99	BMZAMZOW67	D	+3.85

9. To see where the scores come from, open the **mpi_wgt1dim.uni** file from the **MapleParty > weights > mdmper** folder in RAD. In this file you'll find a match on the SSN with 2 edits provides a score of 0.50.



10. Under the **mpi_wgtsval.unl**, you'll find a match on M for the gender provide 0.30 value to the score.



11. For the name we match on all values BART (3.60), ZAMZOW (4.38) (NOTE: we won't find the name Zamzow so we go with agreement 'a') and M (1.75). We get an additional 20 points for a match and proper location for a total score of $3.60 + 0.20 + 4.38 + 0.20 + 1.75 + 0.20 = 10.13$. However since there is a **__FULLNAME_MAXWGT** of 657, we only receive $6.57 + (3 \times 20)$ for the positional bonus = 7.17

```

MapleParty X mpi_wgt1dim.unl mpi_wgtsval.unl X mpi_wg
99|99|A|CMPPER-XNM-XACT|FLOSSIE|420|
99|99|A|CMPPER-XNM-XACT|GILDA|420|
99|99|A|CMPPER-XNM-XACT|GINA|420|
99|99|A|CMPPER-XNM-XACT|HAYDEE|420|
99|99|A|CMPPER-XNM-XACT|HELENA|420|
99|99|A|CMPPER-XNM-XACT|a|438|←
99|99|A|CMPPER-XNM-XACT|d|-134|←
99|99|A|CMPPER-XNM-PARM|__FULLNAME_MAXWGT|657|
99|99|A|CMPPER-XNM-PARM|__INITIAL_ADJWGT|320|
99|99|A|CMPPER-XNM-PARM|__INITIAL_MINWGT|50|
99|99|A|CMPPER-XNM-PARM|__INITIAL_MAXWGT|438|
99|99|A|CMPPER-XNM-PARM|__PHONETIC_ADJWGT|151|
99|99|A|CMPPER-XNM-PARM|__PHONETIC_MINWGT|50|
99|99|A|CMPPER-XNM-PARM|__PHONETIC_MAXWGT|438|

```

- 12. Under the DOB, if we open the **mpi_wgtnval.unl**, we'll find that we get a score of 4.74 for matching the year 1942.

```

PartyConfiguration X mpi_wgtnval.unl X m
99|99|A|CMPID-DOB-YEAR|1951|474|
99|99|A|CMPID-DOB-YEAR|1995|474|
99|99|A|CMPID-DOB-YEAR|1938|474|
99|99|A|CMPID-DOB-YEAR|2003|474|←
99|99|A|CMPID-DOB-YEAR|1942|474|
99|99|A|CMPID-DOB-YEAR|1961|474|
99|99|A|CMPID-DOB-YEAR|1953|474|
99|99|A|CMPID-DOB-YEAR|1932|475|
99|99|A|CMPID-DOB-YEAR|1968|475|
99|99|A|CMPID-DOB-YEAR|2002|475|

```

End of exercise

IBM
®