MDM: Multidomain and Hierarchy Configuration

Lab Guide

Version: MDM10.4-MH-CONFIG-202008

Informatica™
University

MDM: Multidomain and Hierarchy Configuration

Version: MDM10.4-MH-CONFIG-202008

August 2020

Document Conventions

This guide uses the following formatting conventions:

If you see…	It means…	Example
>	Indicates a sub menu to navigate to.	Click Repository > Connect. In this example, you should click the Repository menu or button and choose Connect.
boldfaced text	Indicates text you need to type or enter.	Click the Rename button and name the new source definition **S_EMPLOYEE**.
UPPERCASE	Database tables and column names are shown in all UPPERCASE.	T_ITEM_SUMMARY
italicized text	Indicates a variable you must replace with specific information.	Connect to the Repository using the assigned *login_id*.
Note:	The following paragraph provides additional facts.	**Note:** You can select multiple objects to import by using the Ctrl key.
Tip:	The following paragraph provides suggested uses or a Velocity best practice.	**Tip:** The m_ prefix for a mapping name is…

Other Informatica Resources

In addition to the student and lab guides, Informatica provides these other resources:

- Documentation and Knowledge Base
- Global Customer Support
- Professional Certification

Accessing Documentation and Knowledge Base

To get the latest documentation and Knowledge Base for your product, go to https://network.informatica.com

Contacting Global Customer Support

You can contact a Customer Support Center by telephone or through the Online Support. Online Support requires a username and password. You can request a username and password at

https://www.informatica.com/services-and-training/support-services/contact-us.html

Obtaining Informatica Professional Certification

You can take and pass exams provided by Informatica to obtain Informatica Professional Certification. For more information, go to

https://www.informatica.com/services-and-training/certification.html

Table of Contents

Module 0: Getting Started

Lab 0-0: Starting the Services

Overview:

The “MDM: Multidomain and Hierarchy Configuration” course lab environment has 1 Virtual Machine (VM).

For the MDM environment to work, you need to have all the MDM services up and running. In this lab, you will start the VM and deploy all the MDM services.

Objectives:

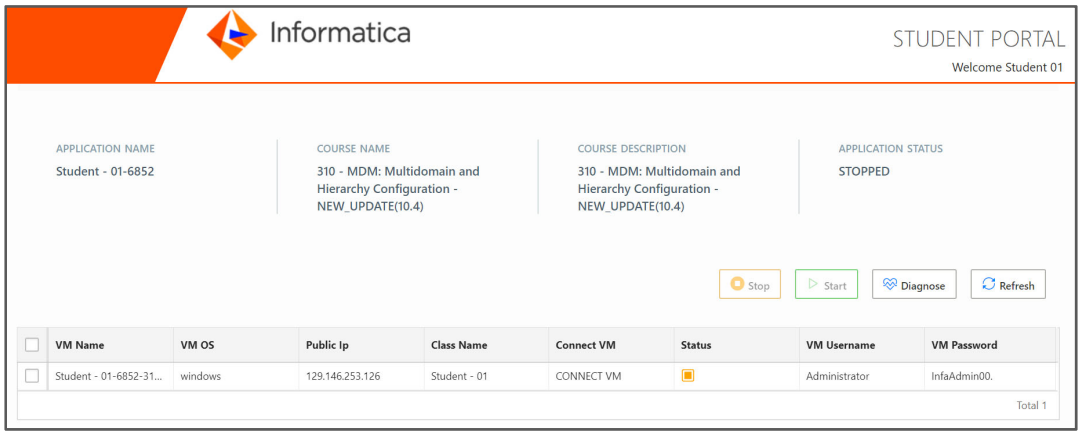
- Start the VM in the lab environment – Start Applications
- Start JBoss and monitor deployments

Duration:

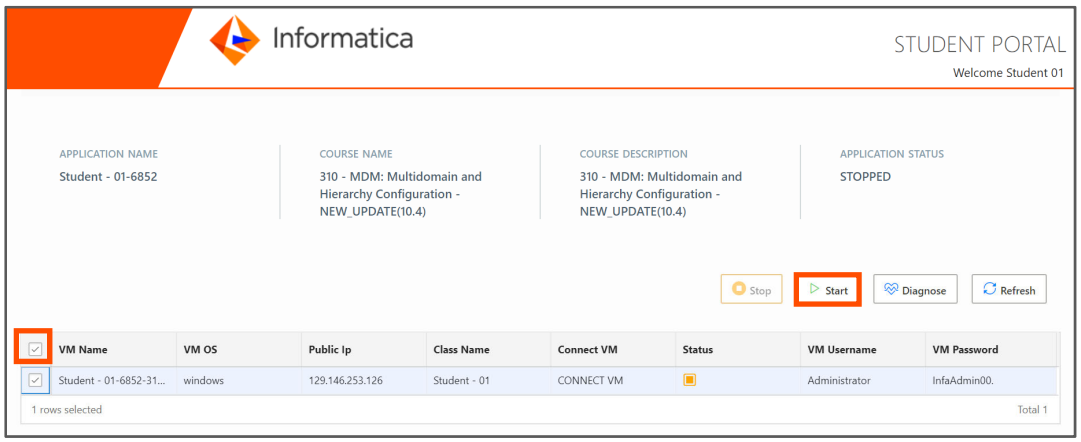
10 minutes

Tasks

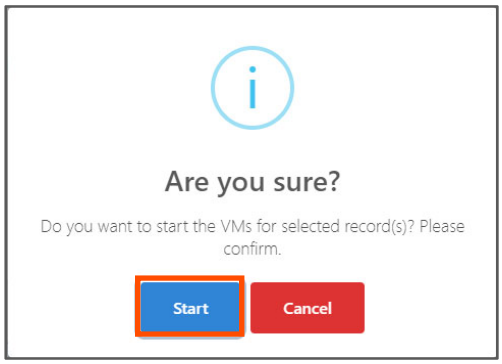
1. Open the lab environment link.



2. Select the VM and click **Start**.

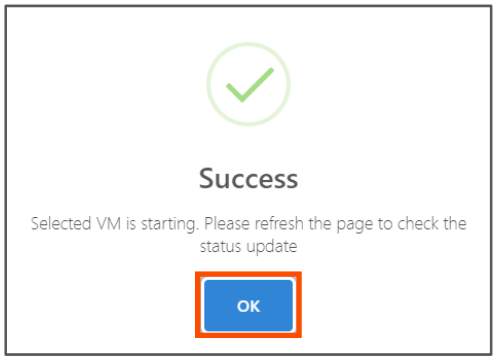


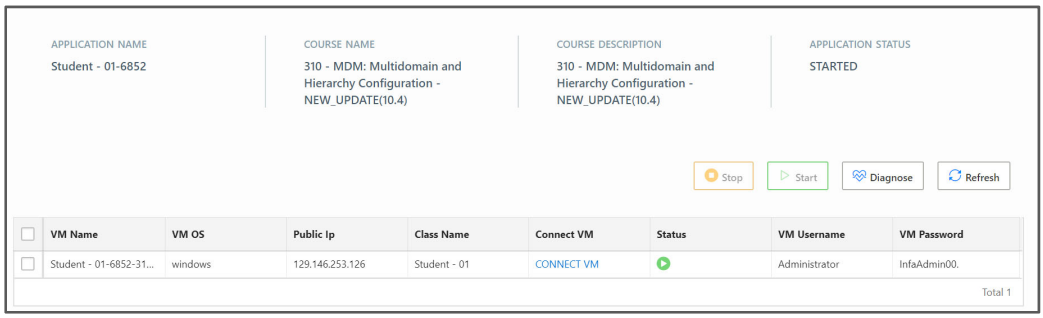
3. A confirmation window appears. Click **Start**.



Note: It will take a few minutes for the VM to start.

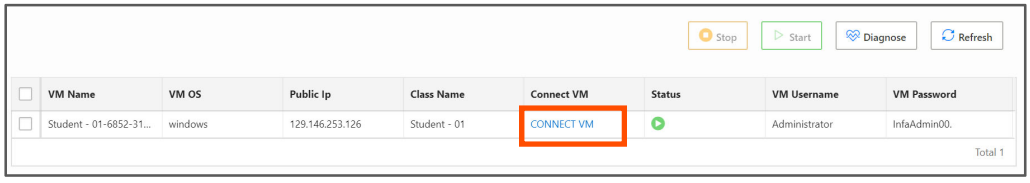
4. Click **OK**.





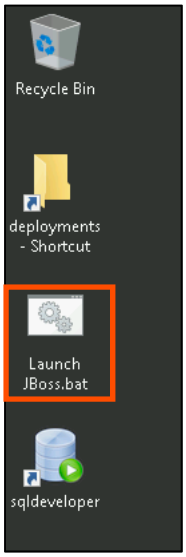
Monitor the application status. It will change from **Starting** to **Started**. Use the **Refresh** button or refresh the browser to monitor the status.

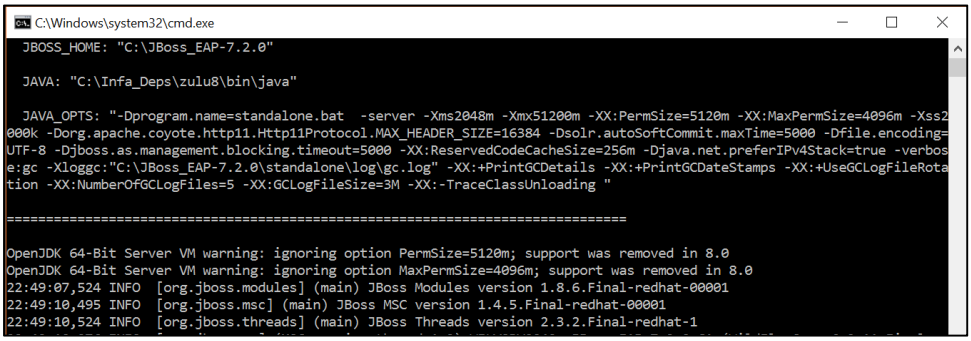
5. Click the **CONNECT VM** link.



6. Click **Connect**.

7. Double-click the **Launch JBoss.bat** script on the desktop.

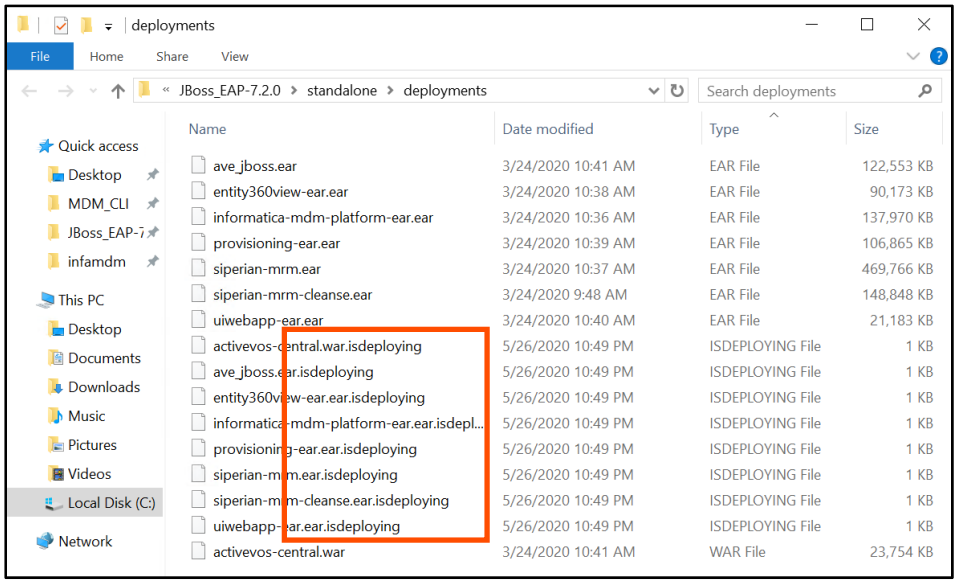




Note: This starts JBoss on your virtual machine. It takes about 5 minutes to start. The Oracle services are configured to start up with the virtual image. If you close the window, it will kill the JBoss process. Hence, we strongly recommend that you minimize the window to prevent closing it accidently.

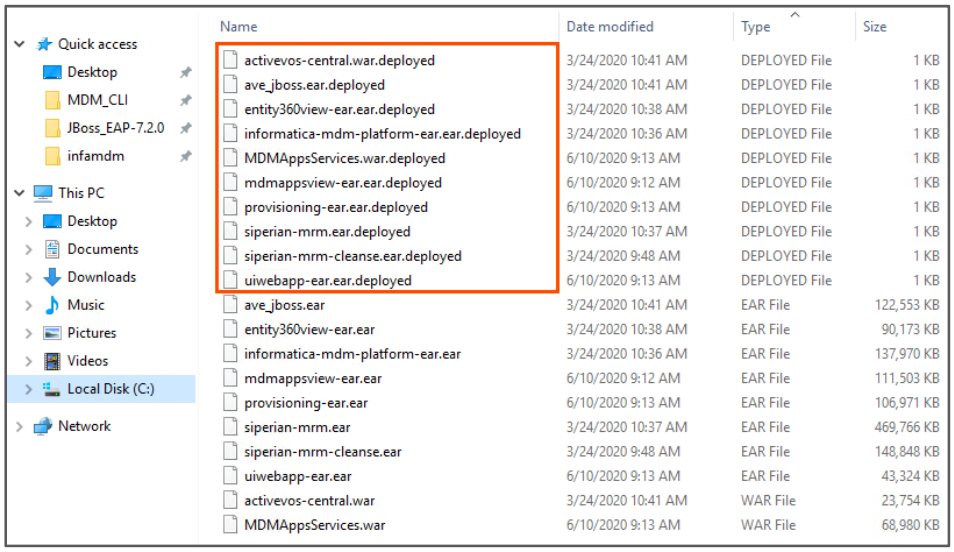
8. Double-click the Deployments folder (**deployments-Shortcut**) on the desktop.

9. You can see files with "**isdeploying**" file extension.



Note: These are the MDM deployments and it will take about 5-8 minutes to deploy.

10. After the deployment, the **isdeploying** file extensions will change to "**.deployed**" files. This indicates that MDM is up and running.



Note: Do not delete any of these files.

This concludes the lab.

Module 2: Define the MDM Data Model

Lab 2-1: Creating a Customer Data Model

Overview:

A Data model is an abstract model that describes how data is structured and organized. A Landing Table provides the entry point for records in the flow of data from source systems into Informatica MDM Hub. A Staging Table provides temporary, intermediate storage in the flow of data from landing tables into the base object tables. A Base Object Table is a table in the Hub Store that contains collections of records about individual entities. Each individual entity will ultimately have a single master record—the best version of the truth—for that entity. An individual entity might have additional records in the base object (contributing records) that contain the multiple versions of the truth that need to be consolidated into the master record.

Objectives:

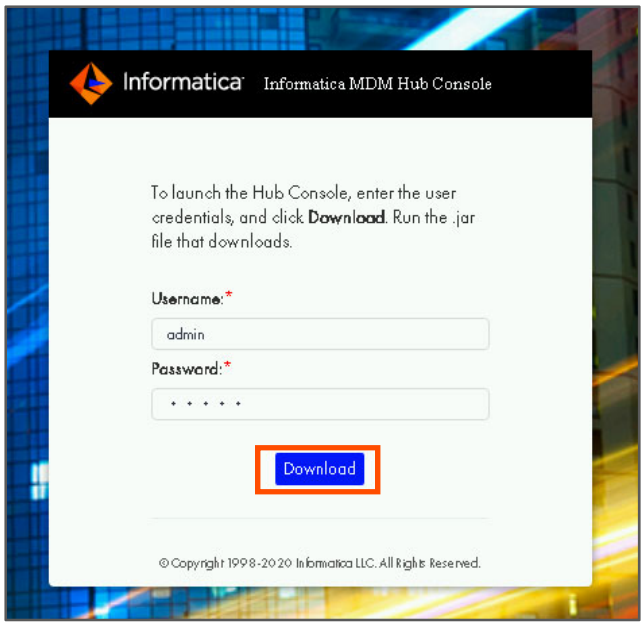
- Create and configure a Landing table for Customer data from the Sales source system
- Create and configure Base Object tables
- Create and configure Staging tables for Customer data from the Sales and CRM source systems

Duration:

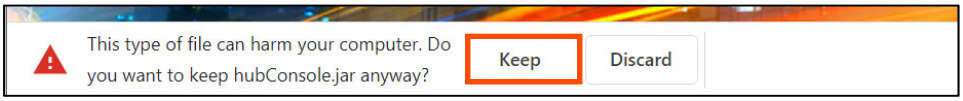
55 minutes

Tasks

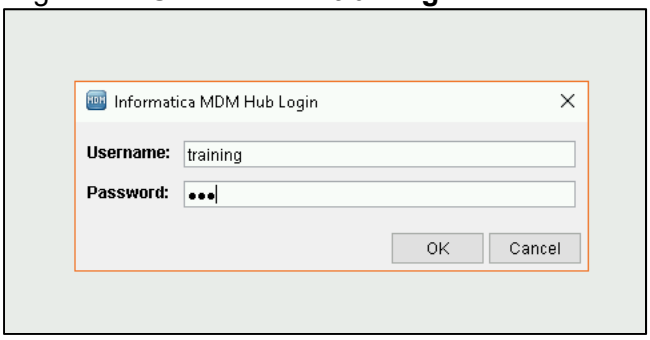
1. Open the Chrome browser and from the **Informatica** bookmarks, click **MDM Hub Console**.
2. Enter the Username and Password as **admin** and click **Download**.



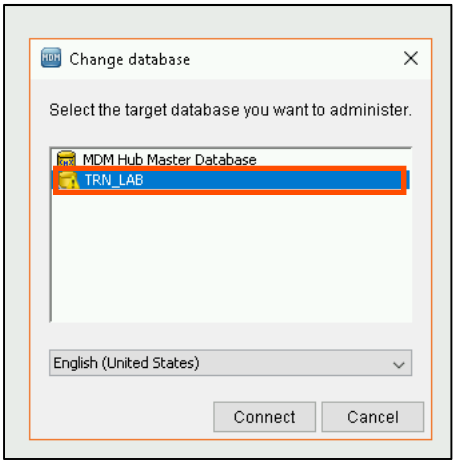
3. You will be prompted to download the hubconsole.jar file (just above the taskbar). Click **Keep** to download the file, and then open the file.



4. Log in with Username as **training** and Password as **mdm**.



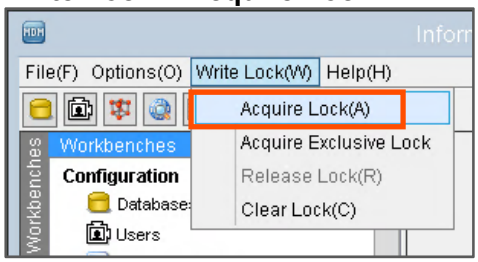
5. As your target database, select **TRN_LAB** and click **Connect**.



Create a Base Object

6. In the Workbenches left pane, select **Model** > **Schema**.
7. Acquire a write lock.

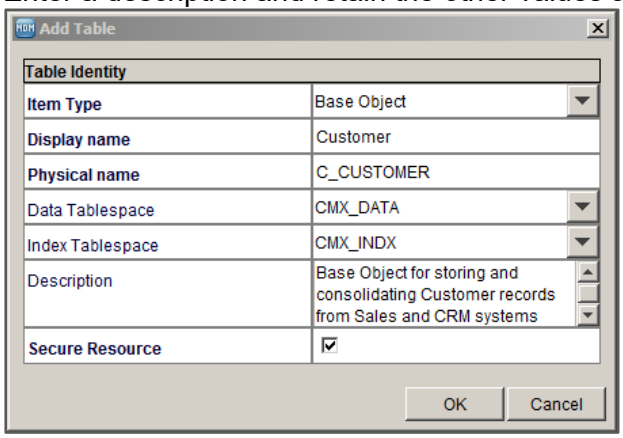
   Write Lock > **Acquire Lock**.



   Good to Know: The following types of write locks are possible from the MDM Hub Console:

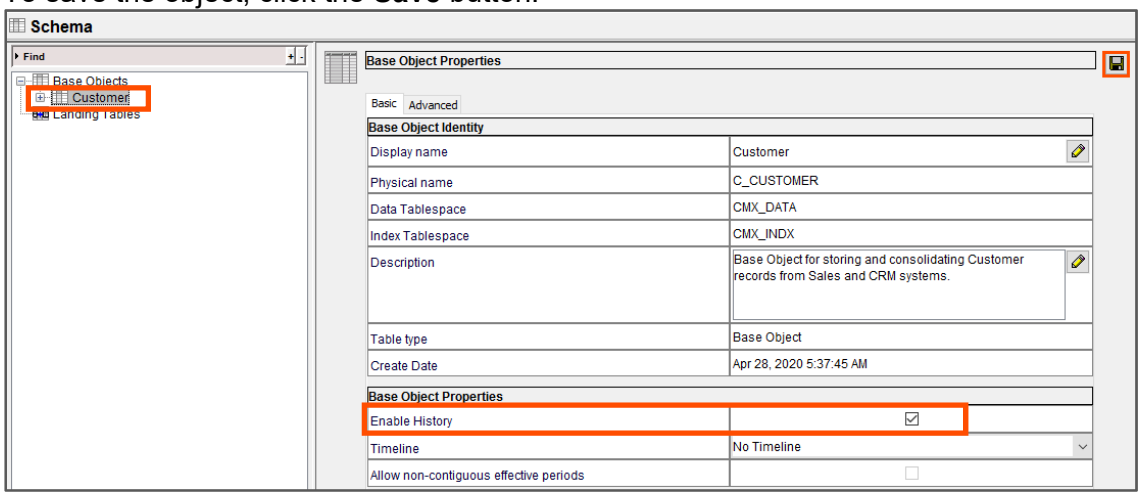
Exclusive lock	Allows only one user to make changes to the underlying Operational Reference Store, preventing any other users from changing the Operational Reference Store while the exclusive lock is in effect.
Write lock	Allows multiple users to make changes to the underlying metadata at the same time. Write locks can be obtained on the MDM Hub master database or on an Operational Reference Store.

8. Right-click the **Base Objects** node and select **Add Item**.
9. Enter the Display name as **Customer**.
10. Enter a description and retain the other values as shown in the following image.

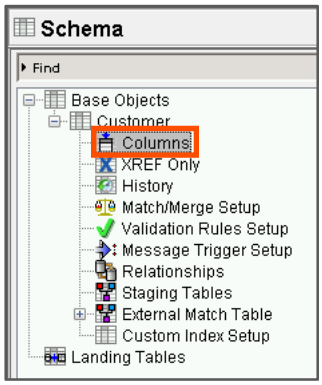


11. Click **OK**.
12. To check the Basic and Advanced properties of the base object, in the Schema tree, click the **Customer** node.
13. In the Basic properties tab, select the **Enable History** option.

14. To save the object, click the **Save** button.



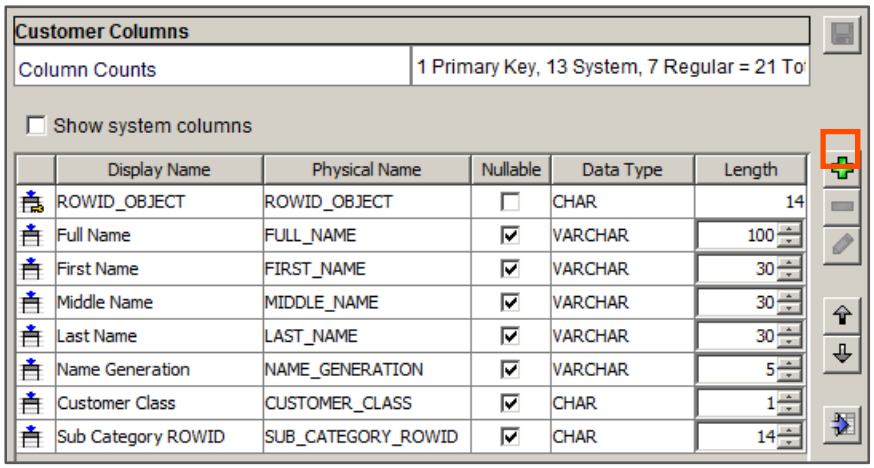
15. Expand the **Customer** base object node and select the **Columns** node.



16. To define columns for the Customer base object, use the **Add Column** button and add the columns as mentioned in the following table.
Note: To maintain the consistency with the predefined files used in this course, you must use the same Display and Physical Names as shown below. Physical Names are populated automatically by either pressing the <Tab> key or clicking in the Physical Name field.

Display Name	Physical Name	Data Type (Length)	Student Notes
ROWID_OBJECT	ROWID_OBJECT	CHAR (14)	You do not need to create this column. It is automatically created by MDM (Primary Key)
Full Name	FULL_NAME	VARCHAR (100)	
First Name	FIRST_NAME	VARCHAR (30)	
Middle Name	MIDDLE_NAME	VARCHAR (30)	

Last Name	LAST_NAME	VARCHAR (30)	
Name Generation	NAME_GENERATION	VARCHAR (5)	Jr., Sr., II, III, etc.
Customer Class	CUSTOMER_CLASS	CHAR (1)	Individual (I), Organization (O), or Region (R)
Sub Category ROWID	SUB_CATEGORY_ROWID	CHAR (14)	Foreign Key (FK) to the Sub Category base object. Note that Foreign Keys must be CHAR (14)



Note: Take a moment to check the System Columns that are normally hidden from the view. To review the columns, select the **Show system columns** checkbox. Some of the columns are not currently used and are reserved for future use. Observe the icons to the left that indicate the difference between the system columns and user-defined columns. To hide the system columns, clear the **Show system columns** checkbox.

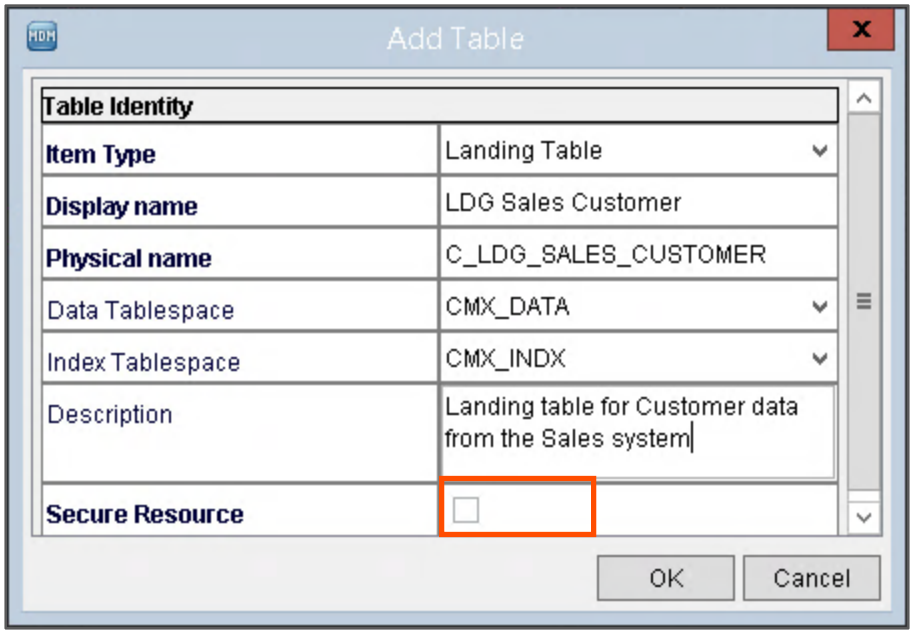
17. **Save** the object.
    Note: You cannot alter the Physical Name or the Data Type of a column once you have defined and saved the object. Also, you cannot shorten the column length. However, it can be increased. If you have defined any of the columns differently than they are in the figure above, you must delete and recreate the columns in error. Otherwise, the table definition will not be consistent with the rest of the course and may produce errors.

Create a Landing Table

18. Right-click the **Landing Tables** node and select **Add Item**.
19. For the Display Name, enter **LDG Sales Customer**.
20. Enter a **Description**.
21. Ensure that the Data Tablespace and Index Tablespace options are selected as **CMX_DATA** and **CMX_INDX** respectively.

Note: These are the Oracle tablespaces used in the training environment. Other Informatica MDM Multidomain Edition environments can have different tablespace names, depending on how the schema is set up in Oracle.

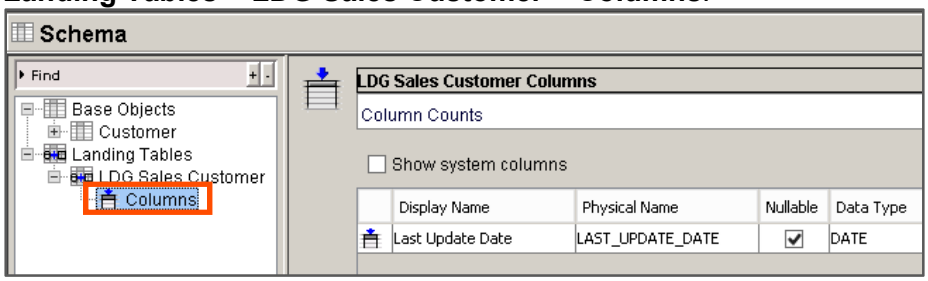
22. Ensure that the **Secure Resource** checkbox is cleared.



23. Click **OK**.

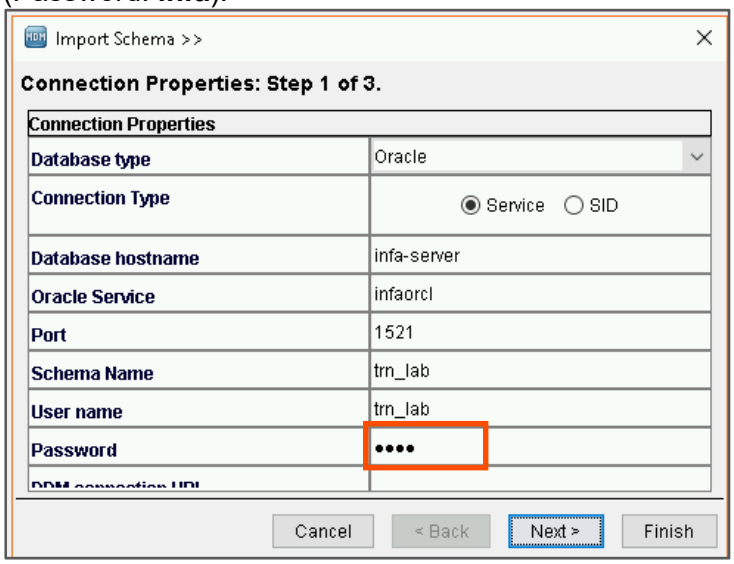
Import a Table Definition

24. For the landing table that you created, select the **Columns** node. That is, navigate to **Landing Tables** > **LDG Sales Customer** > **Columns**.

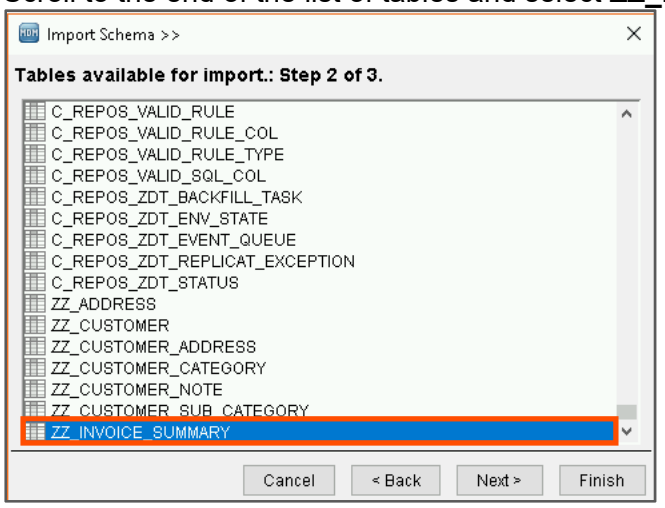


25. Click the **Import Schema** button  . This allows you to import column definitions from another Oracle table.

26. For the schema, enter the connection properties as shown in the screenshot (Password: **infa**):

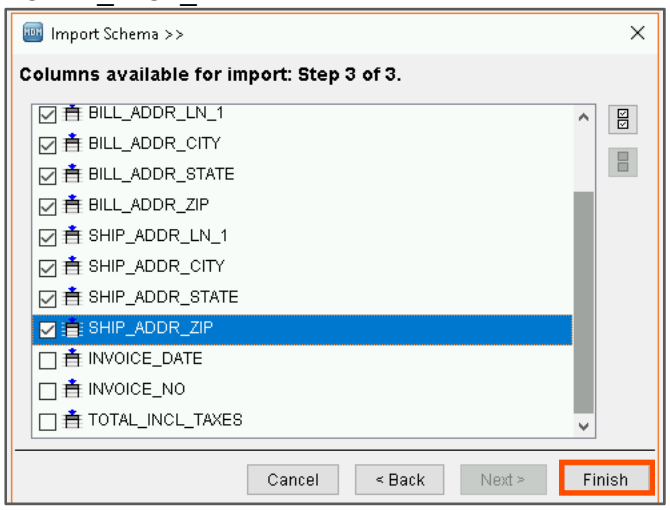


27. Click **Next**.
28. Scroll to the end of the list of tables and select **ZZ_INVOICE_SUMMARY**.



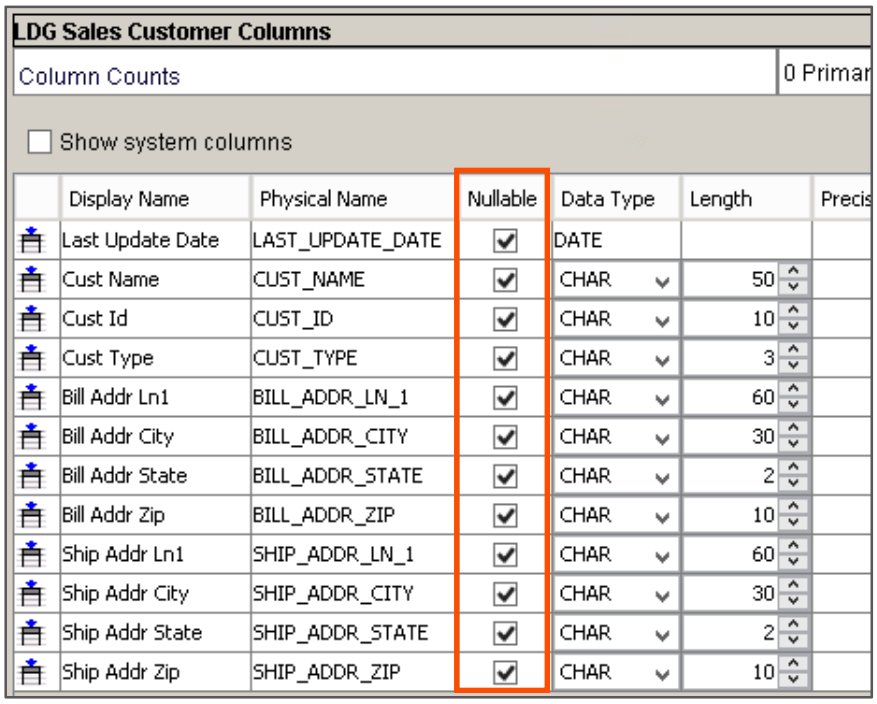
29. Click **Next**.

30. Select **all** the columns **except** INVOICE_DATE, INVOICE_NO, and TOTAL_INCL_TAXES.



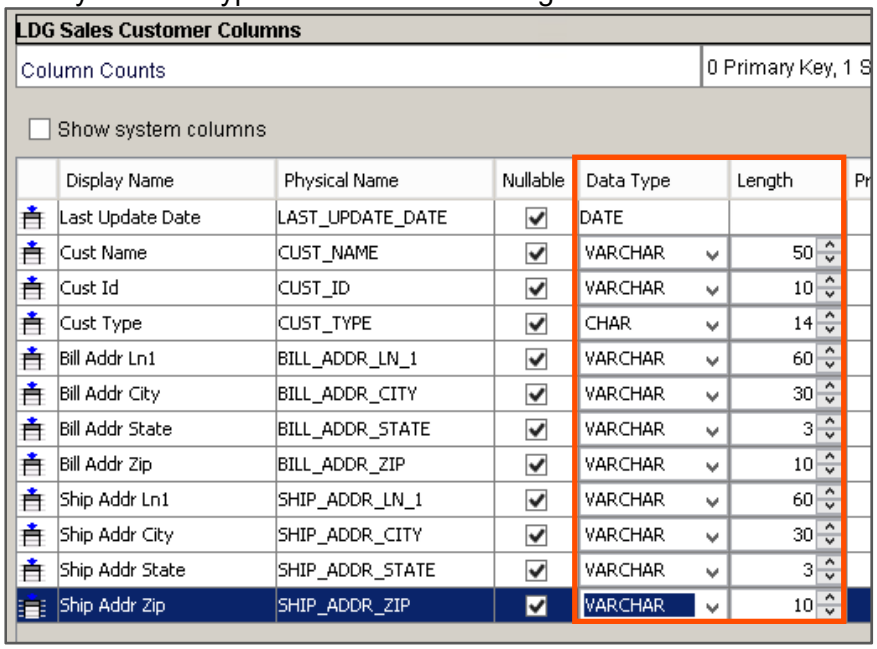
Note: The buttons on the right allow you to select or de-select all the items in the list. If you want to select or de-select more than half the items in the list, it is quicker to use these buttons and adjust the selections.

31. Click **Finish**.
32. Ensure that all the columns are set to **Nullable**.



	Display Name	Physical Name	Nullable	Data Type	Length	Precis
	Last Update Date	LAST_UPDATE_DATE	☑	DATE		
	Cust Name	CUST_NAME	☑	CHAR	50	
	Cust Id	CUST_ID	☑	CHAR	10	
	Cust Type	CUST_TYPE	☑	CHAR	3	
	Bill Addr Ln1	BILL_ADDR_LN_1	☑	CHAR	60	
	Bill Addr City	BILL_ADDR_CITY	☑	CHAR	30	
	Bill Addr State	BILL_ADDR_STATE	☑	CHAR	2	
	Bill Addr Zip	BILL_ADDR_ZIP	☑	CHAR	10	
	Ship Addr Ln1	SHIP_ADDR_LN_1	☑	CHAR	60	
	Ship Addr City	SHIP_ADDR_CITY	☑	CHAR	30	
	Ship Addr State	SHIP_ADDR_STATE	☑	CHAR	2	
	Ship Addr Zip	SHIP_ADDR_ZIP	☑	CHAR	10	

33. Modify the data types and the column lengths as shown in the screenshot.



Note: When the data type is changed, the field lengths are reset.

34. **Save** the table.

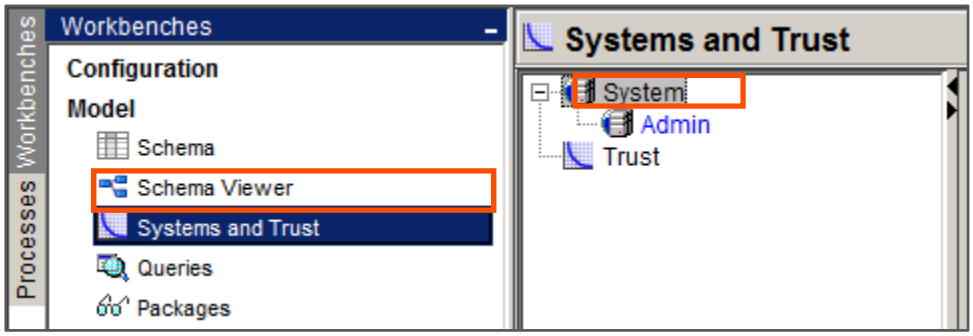
Create Two Staging Tables

Now, you will create a Staging table for Customer data from the Sales source system and another Staging table for the Customer data from the CRM source system.

When you define the columns for a staging table, choose the set of columns that the staging table's source system will provide to the base object. To do this, you must define the staging table in the base object and its source system.

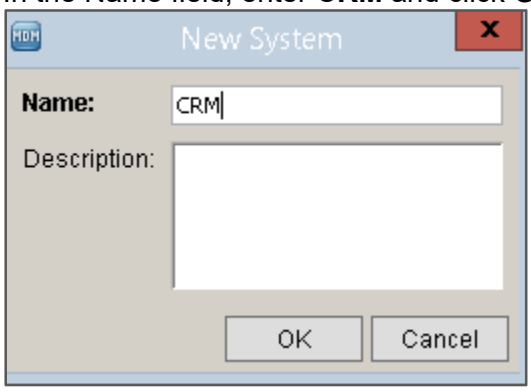
First, use the **Systems and Trust** tool to define the source systems.

35. To create the Sales source system, in the Workbenches pane, select the **Systems and Trust** tool.
36. In the middle pane, expand **System**. Observe that there is an existing system named **Admin**.

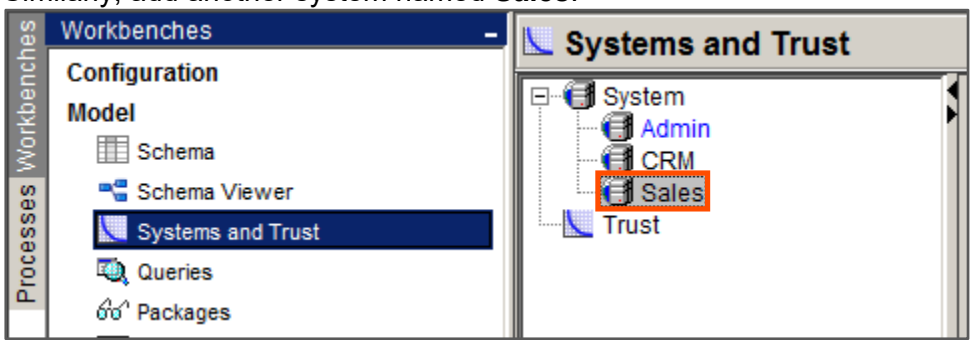


37. In the middle pane, right-click anywhere and select **Add System…**.

38. In the Name field, enter **CRM** and click **OK**.

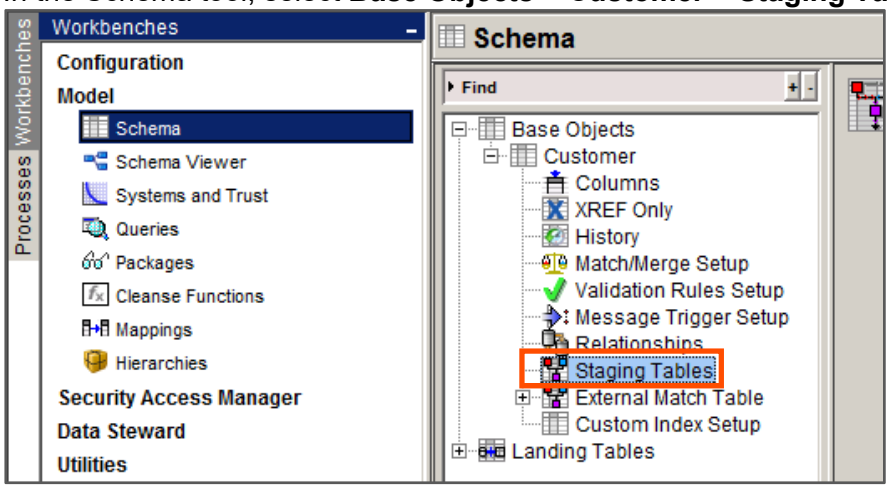


39. Similarly, add another system named **Sales**.



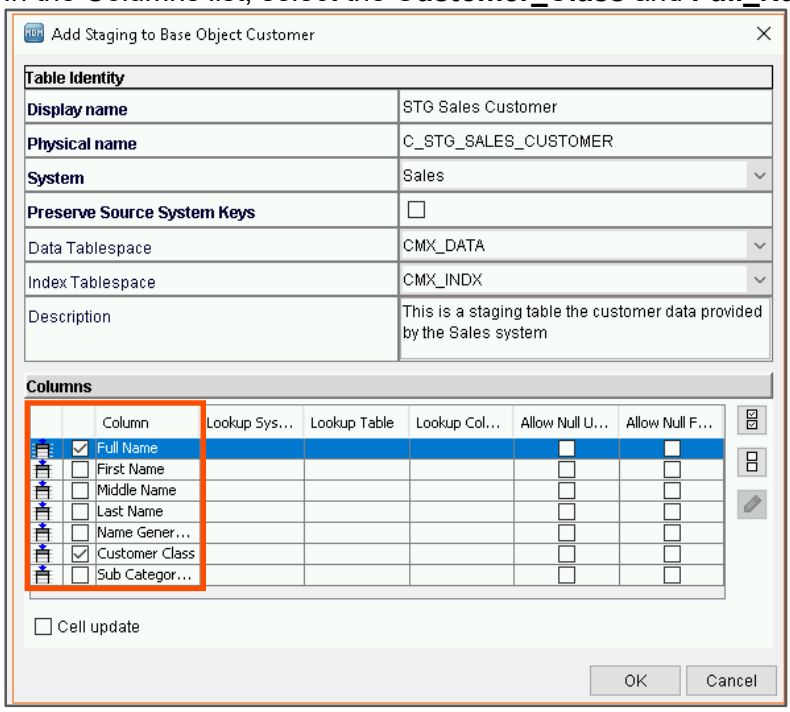
Defining the Staging Tables in the Base Object

40. In the Schema tool, select **Base Objects** > **Customer** > **Staging Tables**.

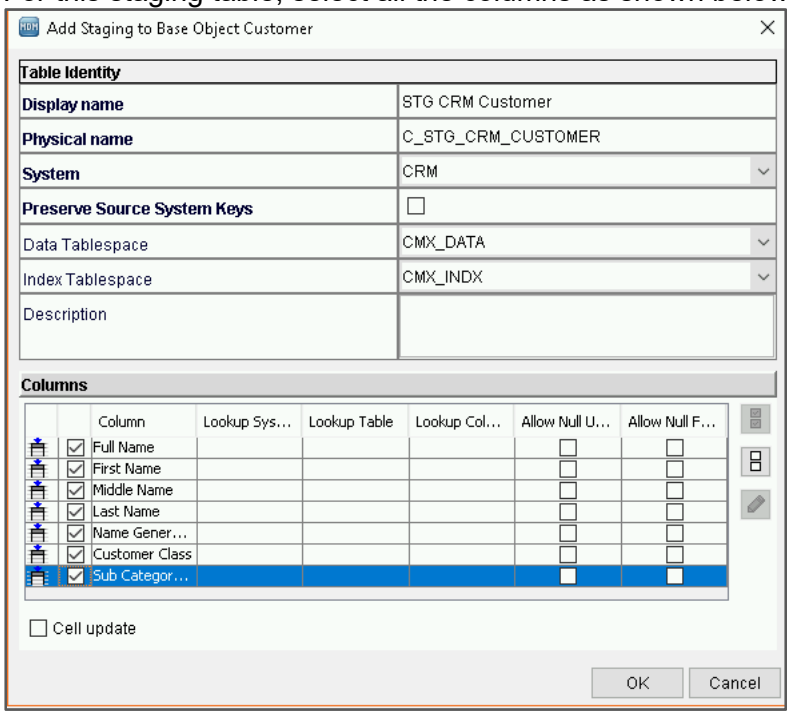


41. Right-click **Staging Tables** and select **Add Staging Table…**.
42. For the Display name, enter **STG Sales Customer**.
43. To create a Physical field, press the <Tab> key or click in the **Physical name** field. The system auto-generates the physical name.
44. For the System, select **Sales**.
45. Clear the **Preserve Source System Keys** option if it is selected.
46. Ensure that the Data Tablespace and Index Tablespace options are selected as **CMX_DATA** and **CMX_INDX** respectively.

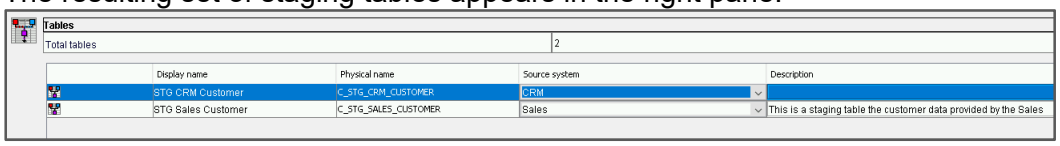
47. In the Description field, enter **This is a staging table for the customer data provided by the Sales system**.
48. In the Columns list, select the **Customer_Class** and **Full_Name** columns.



49. Click **OK**.
50. Similarly, for the Customer base object, add another staging table named **STG CRM Customer**.
    Note: This is based on the data from the CRM system (STG CRM Customer).
51. For this staging table, select all the columns as shown below:



The resulting set of staging tables appears in the right pane.



Create a Base Object with Three Staging Tables

52. Create a base object named **Address** and add its columns as per the table shown below:

Display Name	Physical Name	Data Type (length)	Nullable
ROWID_OBJECT	ROWID_OBJECT	CHAR (14)	no
Address Line 1	ADDRESS_LINE_1	VARCHAR (50)	yes
City	CITY	VARCHAR (50)	yes
State	STATE	VARCHAR (2)	yes
ZIP 10	ZIP_10	VARCHAR (10)	yes
ZIP 5	ZIP_5	VARCHAR (5)	yes
Valid Ind	VALID_IND	CHAR (1)	yes
Customer ROWID	CUSTOMER_ROWID	CHAR (14)	yes

53. In the Basic properties tab, select the **Enable History** checkbox.

Create Staging Tables for the Address Base Object

Staging Table - 1

54. For the Address base object, create a staging table named **STG CRM Address**.
55. Select the source system as **CRM**.
56. Select **all** the Address columns.

Staging Table - 2

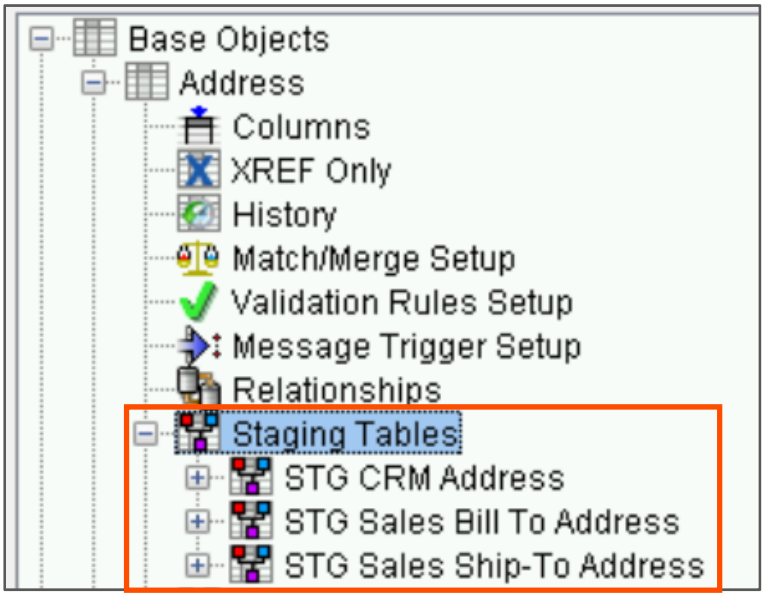
57. For the **Address** base object, create another staging table, and set the Display Name to **STG Sales Ship-To Addr**.
58. Ensure that the Physical Name is set to **C_STG_SALES_SHIP_TO_ADDR**.
59. After the physical name is created, change the display name to **STG Sales Ship-To Address**.

60. Select the source system as **Sales**.
61. Select all the base object columns.

Staging Table – 3

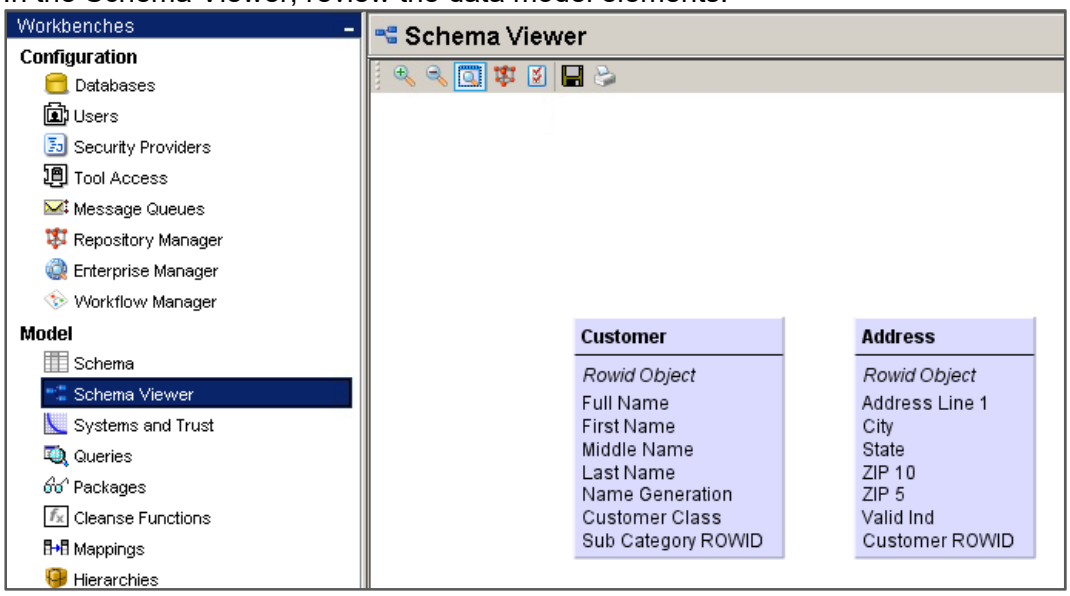
62. For the **Address** base object, create another staging table, and set the Display Name to **STG Sales Bill To Addr**.
63. Ensure that the Physical Name is set to **C_STG_SALES_BILL_TO_ADDR**.
64. After the physical name is created, change the display name **STG Sales Bill-To Address**.
65. Select the source system as **Sales**.
66. Select all the base object columns.

The Address base object should finally have three staging tables – one for the **CRM** system and two for the **Sales** system.



Verify that the **Source systems** are set correctly on your staging tables:

67. In the Schema Viewer, review the data model elements.



This concludes the lab.

ADDITIONAL INFORMATION

The following sections discuss editing and deleting base objects as they pertain to the labs.

Deleting Columns in a Base Object:

Deleting columns should always be approached with extreme caution. Any data loaded into a column is lost when the column is removed. Deleting a column can be a slow process due to the number of underlying tables that could be affected. At this stage of the class, no data has been loaded in the base object, so it is safe to delete a column if you have misconfigured it.

To delete a column from base objects and landing tables:

1. In the Hub Console, start the **Schema** tool.
2. Navigate to the column editor for the table that you want to configure.
3. Acquire a write lock.
4. Select the column that you want to delete from the list of column definitions in the Properties pane.
5. Click the **Remove Column** button. The column is analyzed, and the Impact Analyzer is displayed.
6. To delete the column, click **OK** in the Impact Analyzer. The Schema tool removes the column.
7. Click the **Save** button.

Deleting a Base Object:

At this stage of the class, no data has been loaded in the base object, so it is safe to delete and re-create a base object if you have misconfigured it.

To delete a base object:

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, select the base object that you want to delete.
4. Right-click the mouse and choose **Remove**. The Schema Manager prompts you to confirm the deletion.
5. Choose **Yes**. The Schema Manager asks you whether you want to view the impact analysis before deleting the base object.
6. Choose **No** if you want to delete the base object without viewing the impact analysis. The Schema Manager removes the deleted base object from the schema tree.

Editing Column Properties:

After the columns are added and saved, you can change certain column properties. Before you make any changes, note that once a table is defined and saved, you cannot:

- reduce the length of a CHAR, VARCHAR, NCHAR, or NVARCHAR2 field
- change the scale or precision of a NUMBER field

To change column properties:

1. Navigate to the column editor for the table that you want to configure.
2. Acquire a write lock.
3. For each column, you can change the following properties:

Property	Notes for Editing Values in This Column
Display Name	Name of this column as it will be displayed in the Hub Console.
Length	You can only increase the length of a CHAR, VARCHAR, NCHAR, or NVARCHAR2 field.
Default	Used if no value is provided for the column but the default value cannot be NULL. **Note:** Enabling **Default** does not have any effect on records that were loaded before **Default** was enabled. Existing NULL values remain NULL. Reload the data, preferably from the staging table, to ensure the values in the column are not NULL.

Trust	**Note:** You need to synchronize metadata if you enable trust. If you enable trust for a column on a table that already contains data, you will be warned that your trust settings have changed and that you need to run the trust "Synchronize" batch job in the **Batch Viewer** tool before doing any further loads to the table. Informatica MDM Hub will automatically make sure that the "Synchronize" job is available in the **Batch Viewer** tool. **Warning**: You must execute the Synchronize job before you run any more Load jobs. Otherwise, the trusted values used to populate the column will be incorrect. **Warning:** Beware and be very careful about disabling (unchecking) trust for columns that already contain data. Disabling trust results in the removal of columns from some of the underlying metadata tables and results in loss of data. If you inadvertently disable trust and save that change, you should correct your error by enabling trust again and immediately running the "Synchronize" job to recreate the metadata.
Unique	Enabling the **Unique** indicator will fail if the column already contains duplicate values. It is recommended that you avoid using the **Unique** option, particularly on base objects with records that might be merged.
Validate	**Warning:** Beware when disabling validation, which results in the loss of metadata for the associated column. This should be approached with caution and should only be done with certainty.
Putable	Enable this property for system columns into which you want to put data (insert or update) using SIF requests and by using batch jobs run through the Hub Console. Applies to any system column except: ROWID_OBJECT, CONSOLIDATION_IND, LAST_ROWID_SYSTEM, and CM_DIRTY_IND.

Note: Be sure to read about the implications of changing a property before you make the change. For more information about each property, see "Column Properties" in the Informatica MDM Multidomain Edition Configuration Guide.

Editing Landing Table Properties:

1. Select **Schema**.
2. Acquire a write lock.
3. Select the landing table that you want to edit.
   Schema Manager displays the Landing Table Identity pane for the selected table.
4. Change the landing table properties as required.
5. Click the **Save** button to save your changes.
6. Change the column configuration for your landing table, if required.

Removing Landing Tables:

1. Select **Schema**.

2. Acquire a write lock.
3. In the schema tree, expand the **Landing Tables** node.
4. Right-click the landing table that you want to remove and select **Remove**.
5. Schema Manager prompts you to confirm deletion.
6. Click **Yes**.
   Note: Schema Manager drops the landing table from the database, deletes any mappings between this landing table and any staging table (but does not delete the staging table), and removes the deleted landing table from the schema tree.

Module 2: Define the MDM Data Model

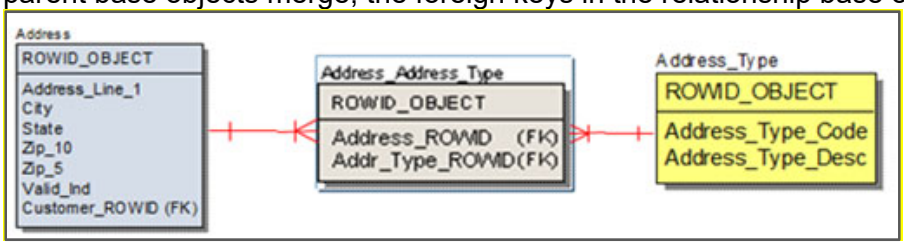
Lab 2-2: Defining Relationships between Base Objects

Overview:

MDM Hub supports one-to-many, many-to-many, and hierarchical relationships between records in an object.

One-to-many relationships – These parent-child relationships are implemented by foreign keys. Foreign keys are virtual and are known only to the repository. That is, there is no physical constraint created in the database, and you can define a Display Name column for the lookup. The contents will be displayed in a drop-down list in Add/Edit window in Data Manager and Merge Manager.

Many-to-many relationships – These relationships are constructed using Intersection tables (also called association tables) and are simply another base object. As the parent records in the parent base objects merge, the foreign keys in the relationship base objects are updated.



Objectives:

- Define the foreign-key relationship from the Address to the Customer base objects
- Create a many-to-many relationship between the Address and Address_Type base objects

Duration:

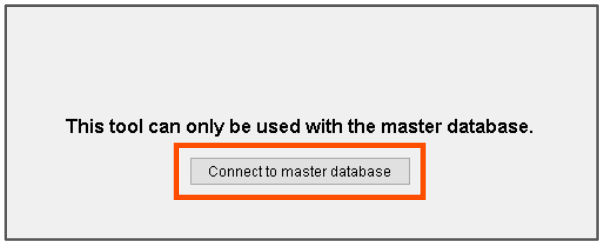
30 minutes

Before you Begin

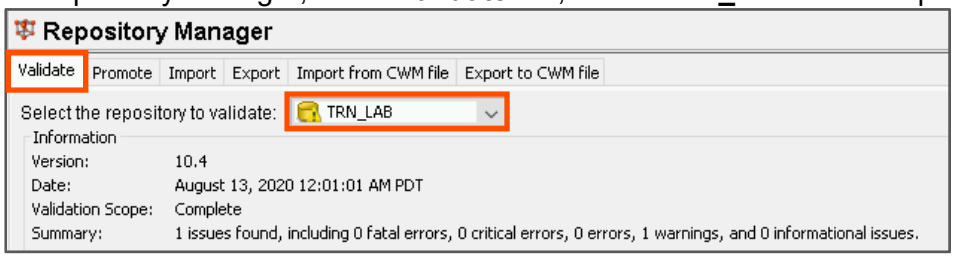
To complete the labs in this course, you must have more objects in your schema. To reduce the effort and avoid having students create all of them manually, the objects are already created in a **change list**, using MDM's Repository Manager tool. You need to use Repository Manager to promote the changes from that change list into your LAB schema.

Apply a Change List to Create Additional Objects:

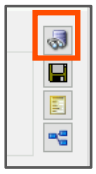
1. In the Workbenches pane, open the **Configuration** > **Repository Manager** tool.
2. You will be prompted to change to the Master Database. Click **Connect to master database**.



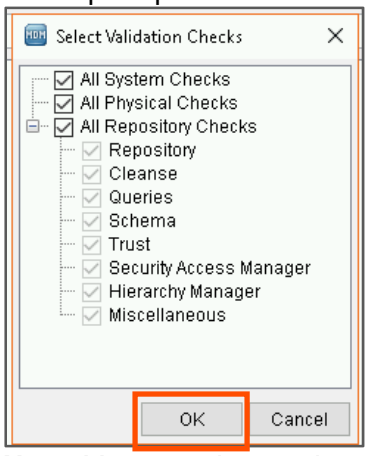
3. In Repository Manager, in the **Validate** tab, select **TRN_LAB** as the repository.



4. Click the **Validate** button on the right.



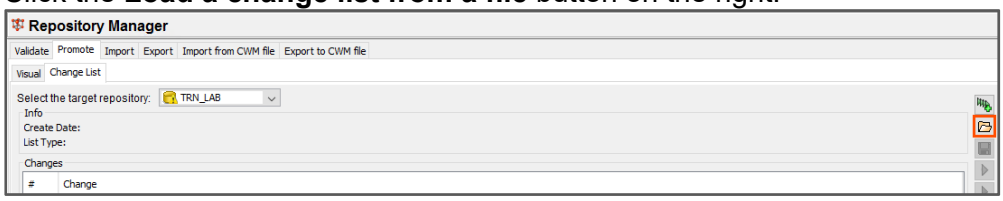
5. When prompted to select the validation checks, click **OK**.



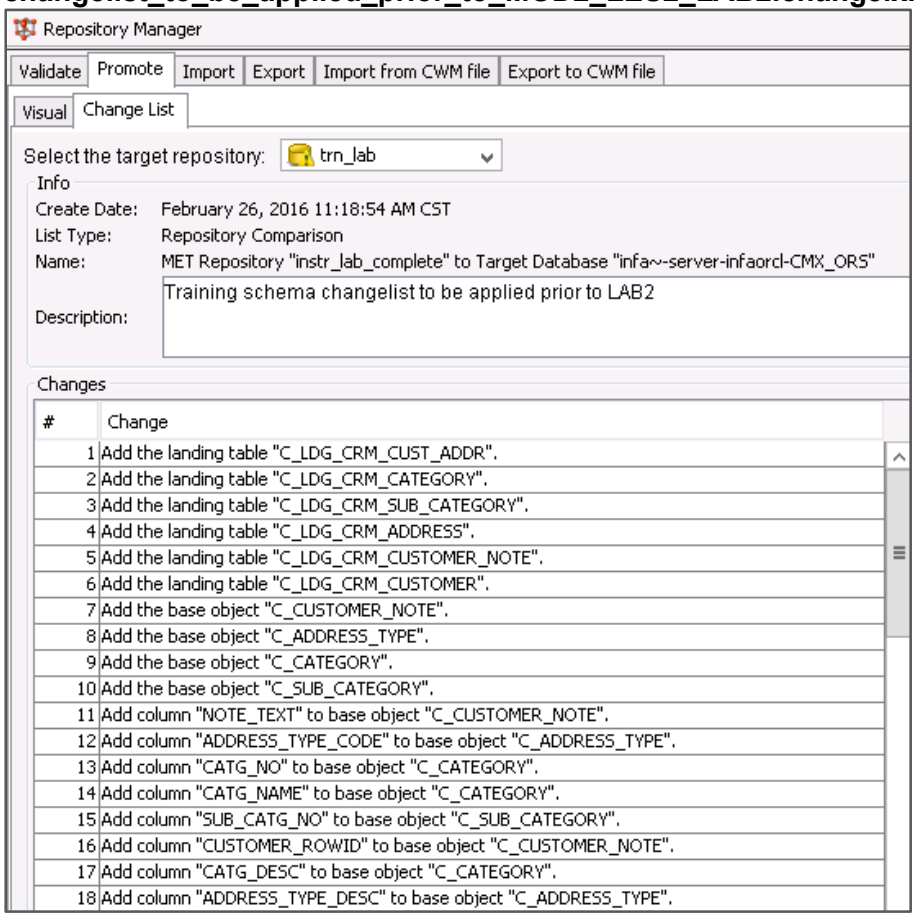
Note: You must ignore the warnings and information messages, if any. However, change lists cannot be imported if there are serious errors with an ORS. Errors and fatal messages must be resolved before an import is performed.

6. In the **Promote** tab, select the **Change List** sub-tab.
7. Select **TRN_LAB** as the target repository.

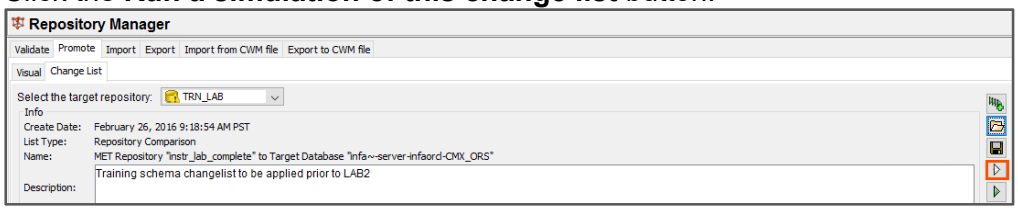
8. Click the **Load a change list from a file** button on the right.



9. Browse to **C:\MDM_Training\Training_MDE_Change Lists_10.2** and select the **changelist_to_be_applied_prior_to_MOD2_LES2_LAB2.change.xml** file.

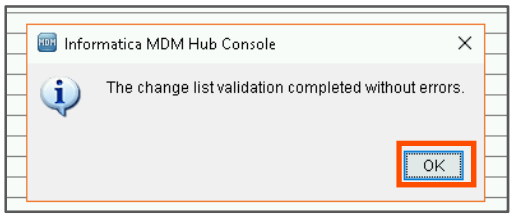


10. Click the **Run a simulation of this change list** button.

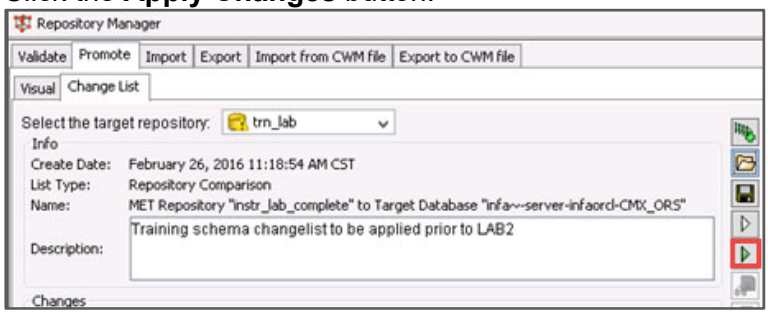


11. For the confirmation and Validate Data prompts, click **Yes**. No errors should be reported.

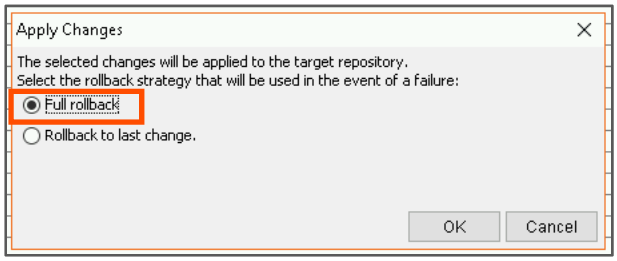
12. Click **OK**.



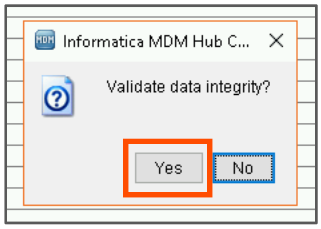
13. Click the **Apply Changes** button.



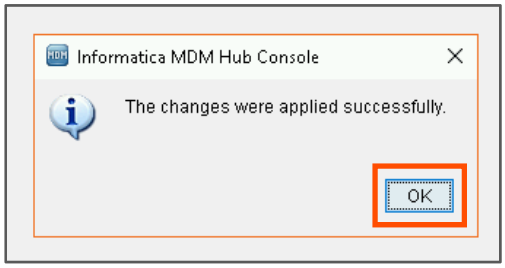
14. Select **Full rollback** and click **OK**.



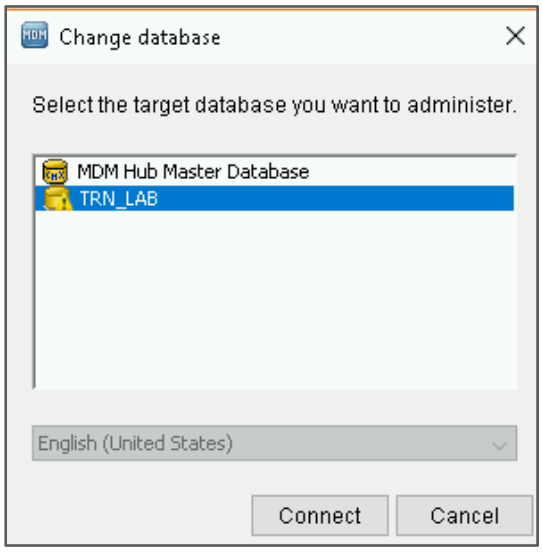
15. Click **Yes**.



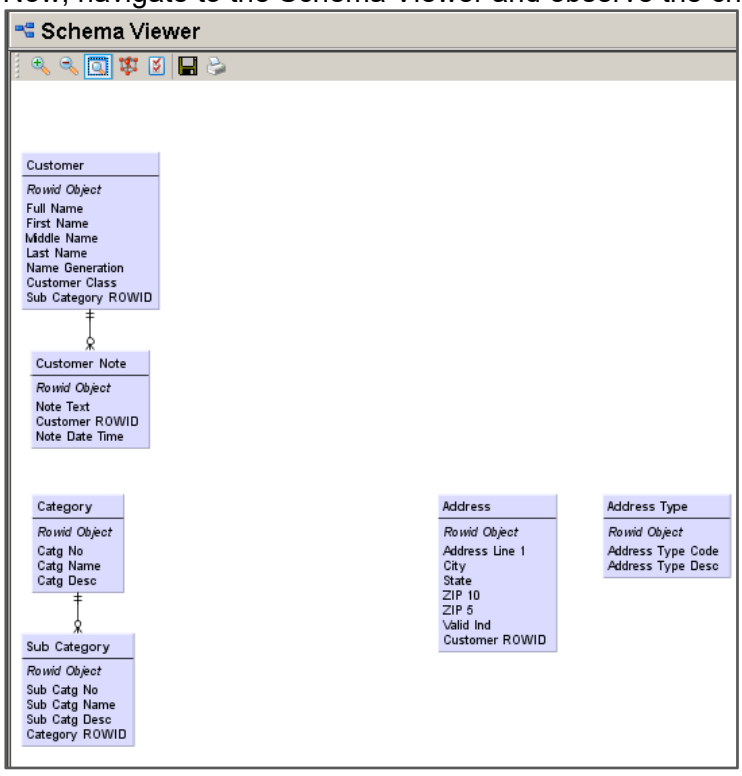
16. Click **OK**.



17. After the changes are applied, connect to the **TRN_LAB** ORS.



18. Connect to the **Schema** tool. Notice all the new objects and relationships that are added to your schema.
19. Now, navigate to the Schema Viewer and observe the changes.



The following base objects are added to the data model:

- Customer Note
- Address_Type
- Category
- Sub_Category

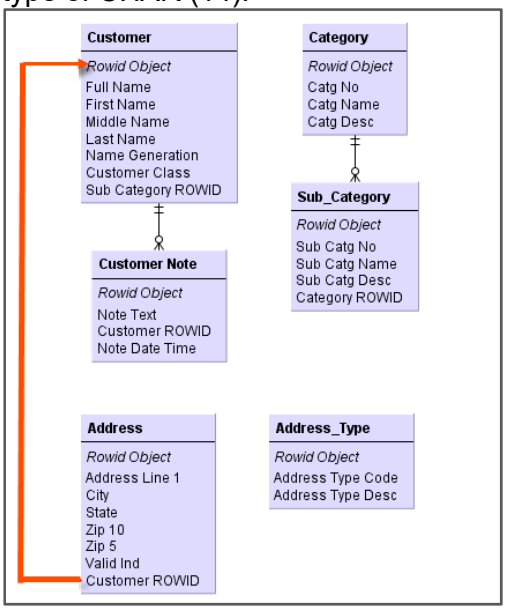
The following relationships are added:

- Customer – Customer Note
- Category – Sub_Category

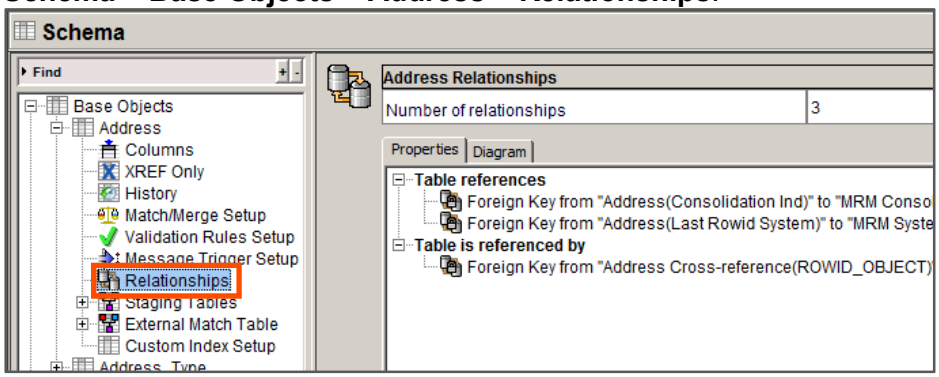
Define a Foreign Key Relationship Between Two Base Objects:

Now, you will define the foreign key relationship between the Address and Customer base objects. When you define a foreign key relationship in MDM, you define it from the child (referencing) table to the parent (referenced) table. In this lab, Address is the child table and Customer is the parent table.

The Customer_ROWID column in the Address table will be used for the foreign key from Address to Customer.ROWID_OBJECT. Columns that are used as foreign keys require a data type of CHAR (14).

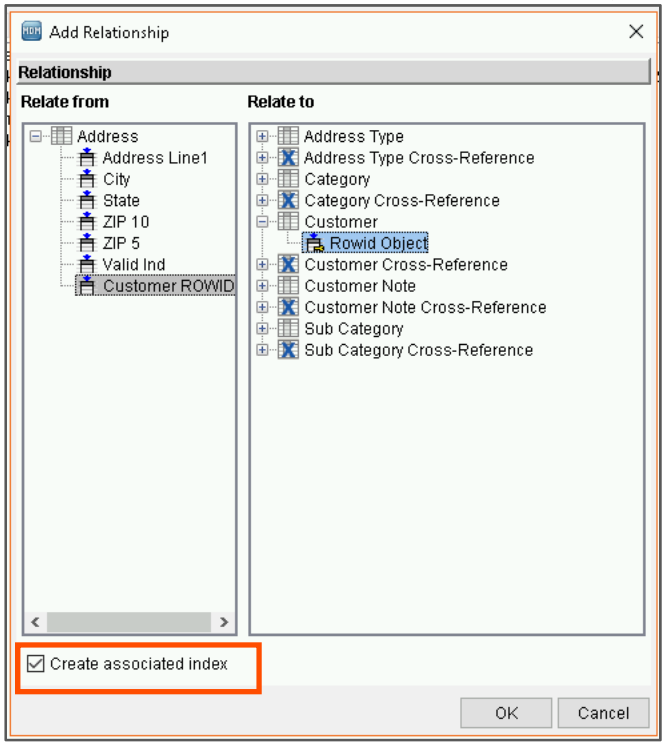


20. Ensure that you acquire the write-lock on your schema:
    Write Lock > **Acquire Lock**.
21. Navigate to the **Schema** tool and under the **Address** base object, browse to the **Relationships** node.
    Schema > **Base Objects** > **Address** > **Relationships**.



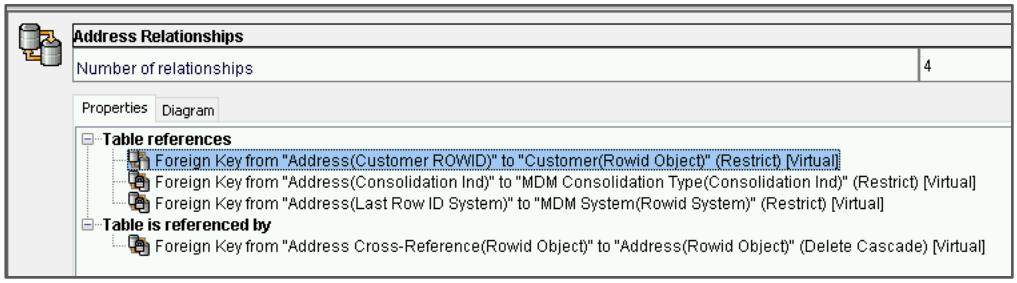
22. Right-click on the **Relationships** node and choose **Add Relationship…**.

23. Expand the **Address** node in the “Relate from” pane and select the **Customer ROWID** column.
24. In the “Relate to” pane, expand the **Customer** node.
25. Select the **Rowid Object** column.
26. Select the **Create associated index** checkbox, if not already selected.



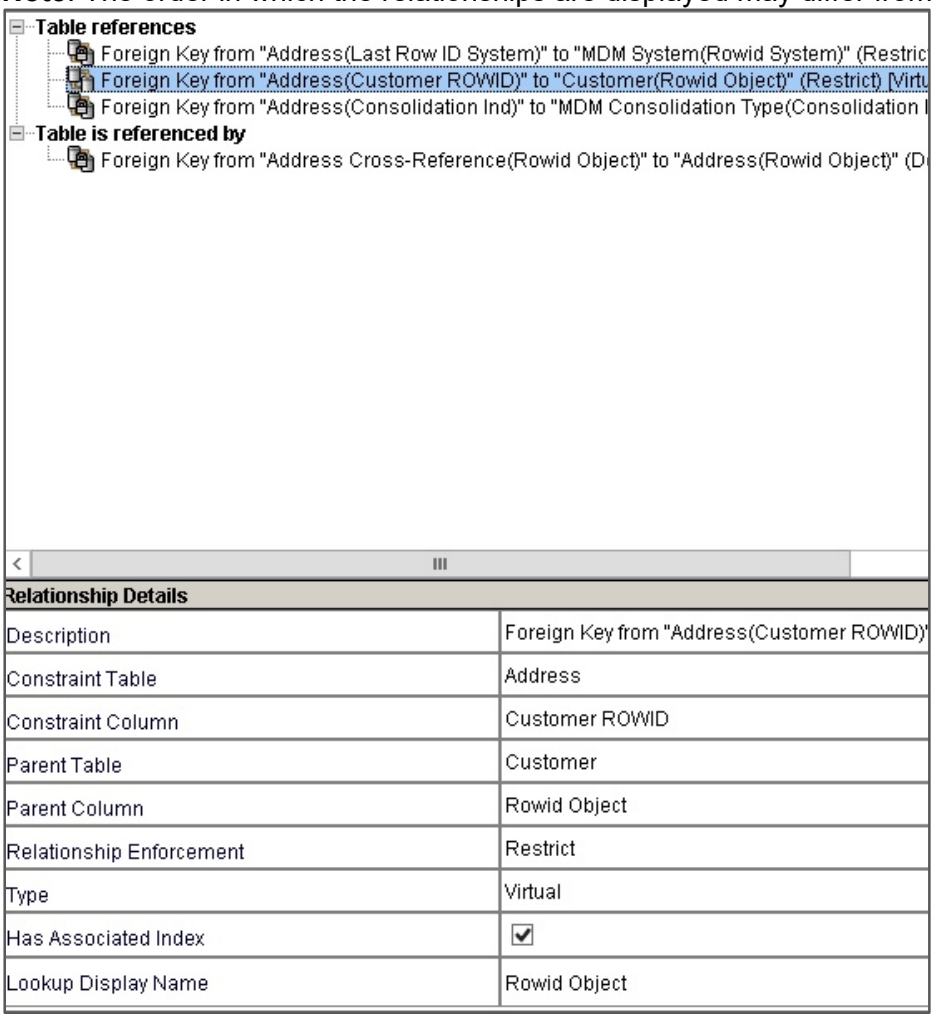
This creates an index on a foreign key relationship.

27. Click **OK**.

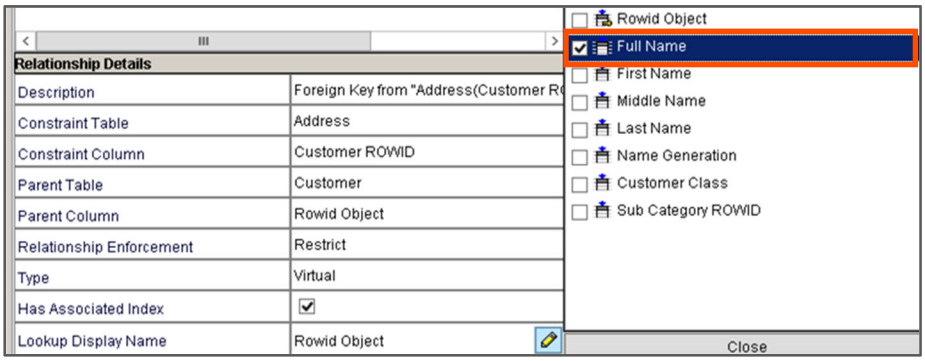


Note: The relationship – Foreign key from “Address (**Customer_ROWID**)” to “Customer (**Rowid Object**)”, is added to the Relationship properties pane under the **Table references** heading.

28. Click on the relationship name. The relationship details are displayed in the bottom pane.
Note: The order in which the relationships are displayed may differ from the image.



29. For the **Lookup Display Name** field, click the **Edit** button to the right of the column.
30. Select the **Full Name** column and click **Close**.



Note: Now, when a link to a Customer record is displayed with reference to an Address record, the Customer record will be labeled with the value of the Full Name attribute rather than the Rowid Object value. This is more intuitive to the human user.

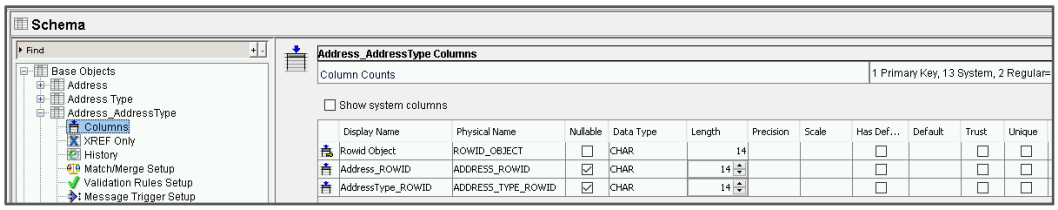
31. Click **Save**.

Define a many-to-many Relationship Between the Base Objects

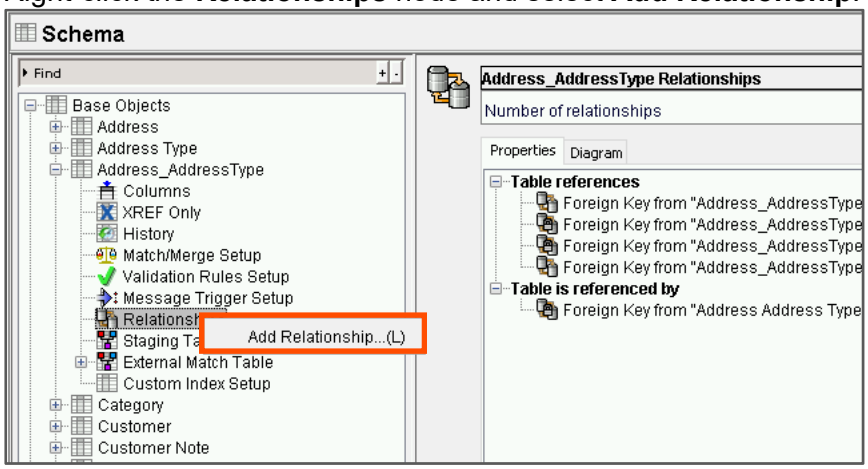
In this section, you will create the **Address_AddressType** relationship. This is a many-to-many relationship between the **Address** and **Address_Type** base objects. This relationship is used to manage the many-to-many link between Addresses and their Address Types – for example, one address can be both a billing address and a shipping address.

32. Add a new base object called **Address_AddressType**.
33. For this Address_AddressType base object, create the following two columns:

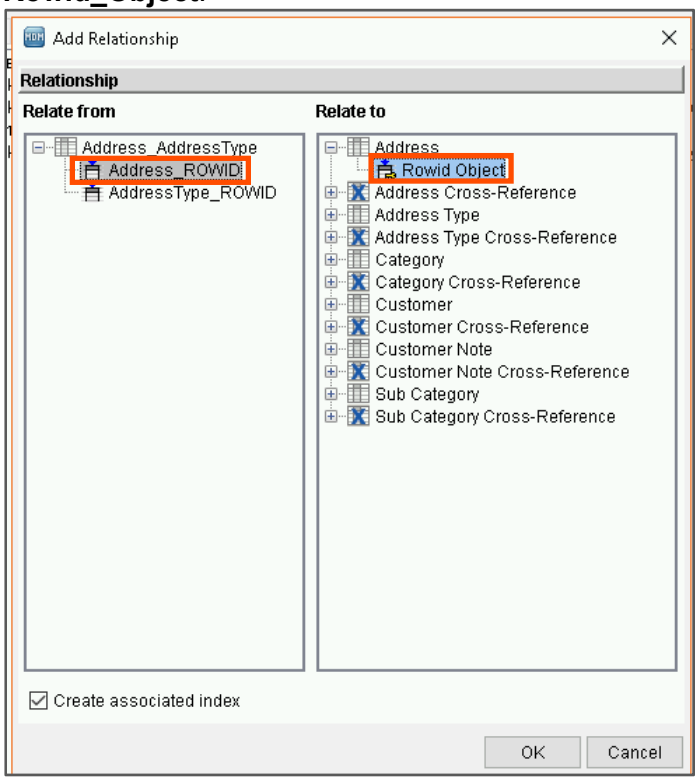
Column Name	Data Type	Description
Address_ROWID	CHAR(14)	This will contain the FK to the **Address** base object
AddressType_ROWID	CHAR(14)	This will contain the FK to the **Address_Type** base object



34. **Save** the changes.
35. Right-click the **Relationships** node and select **Add Relationship**.

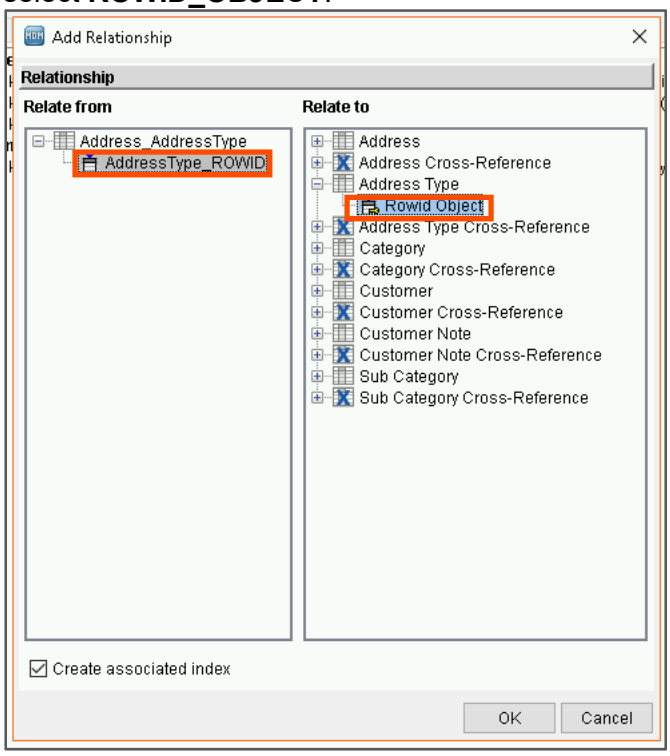


36. From Address_AddressType, select **Address_ROWID** and from Address, select **Rowid_Object**.

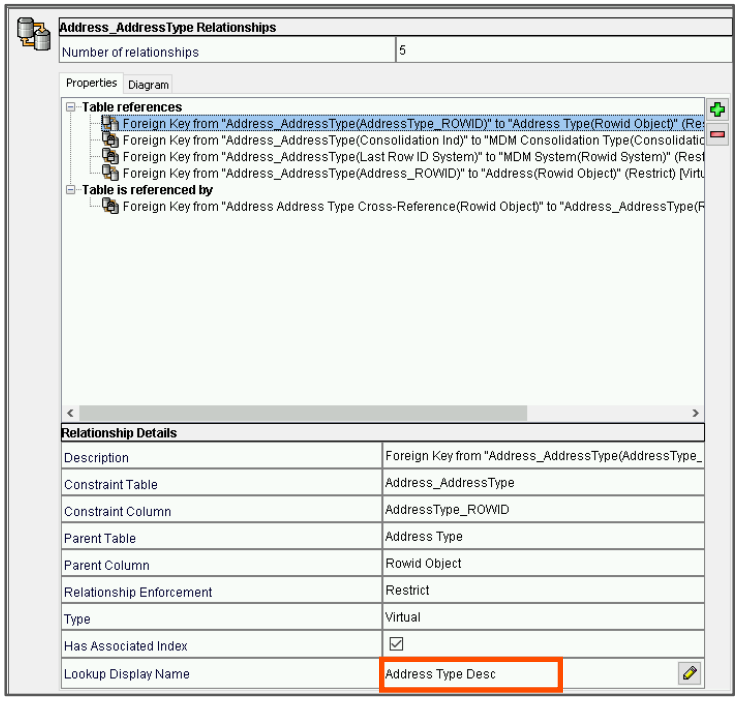


37. Click **OK**.
38. Right-click the **Relationships** node and select **Add Relationship**.

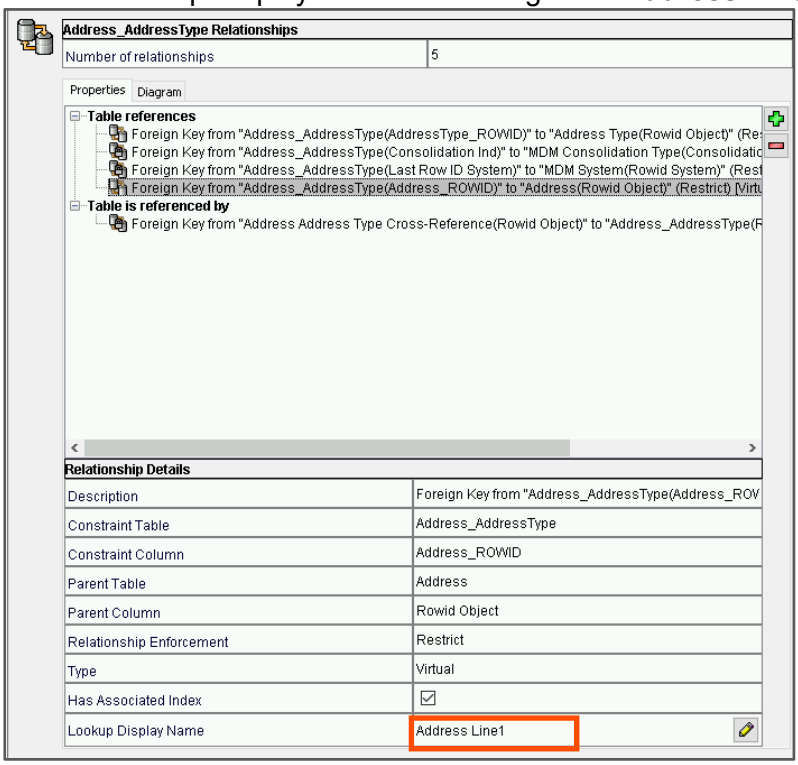
39. From Address_AddressType, select **AddressType_ROWID** and from Address_Type**,** select **ROWID_OBJECT**.



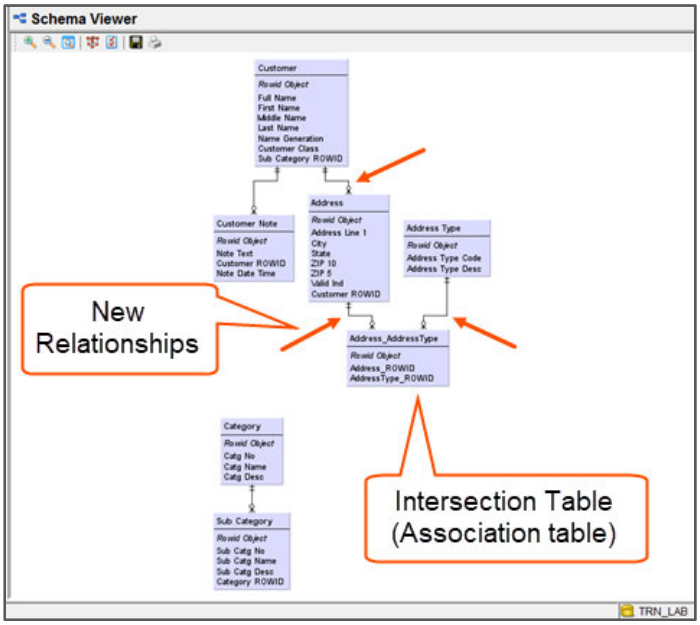
40. Click **OK**.
41. In the relationship Properties tab, select "**Foreign Key from "Address_AddressType(AddressType_ROWID)" to "Address_Type(ROWID_OBJECT)" (Restrict) [Virtual]**".
42. Edit the Lookup Display Name and change it to **Address Type Desc**.



43. **Save** the object.
44. In the relationship Properties tab, select “**Foreign Key from "Address_AddressType(Address_ROWID)" to "Address (ROWID_OBJECT)" (Restrict) [Virtual]**”.
45. Edit the Lookup Display Name and change it to **Address Line 1**.

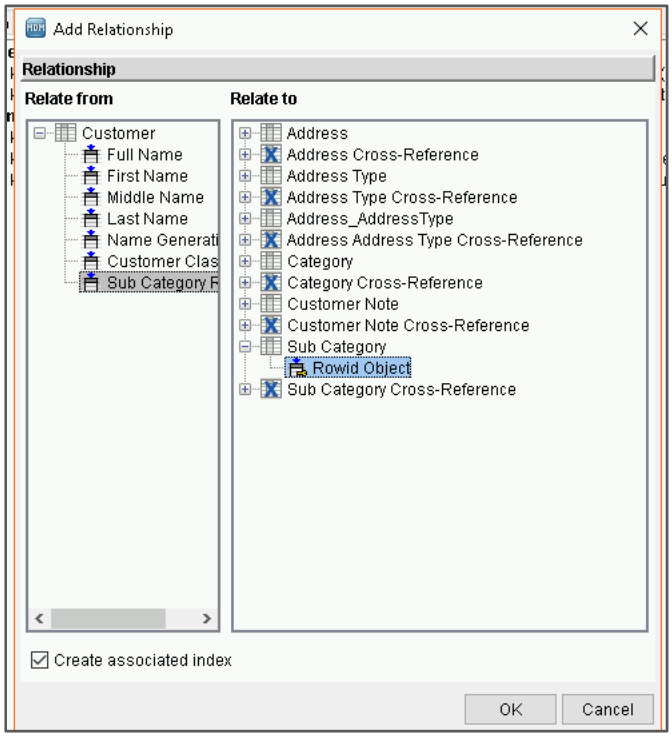


46. **Save** the object.
47. Navigate to the **Schema Viewer** tool and observe the modifications made to the schema.

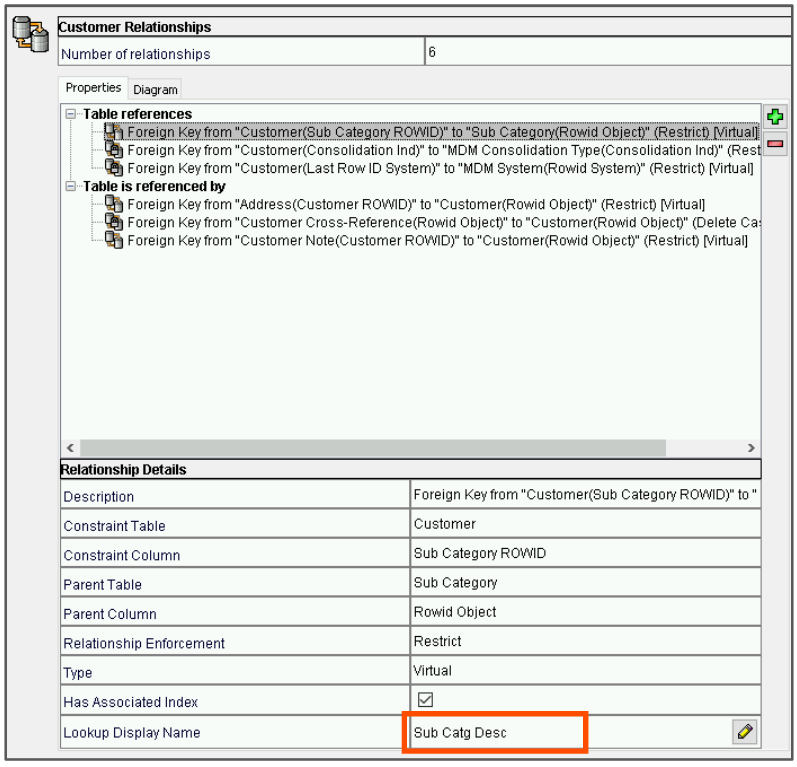


Create a Relationship From Customer to Sub-Category

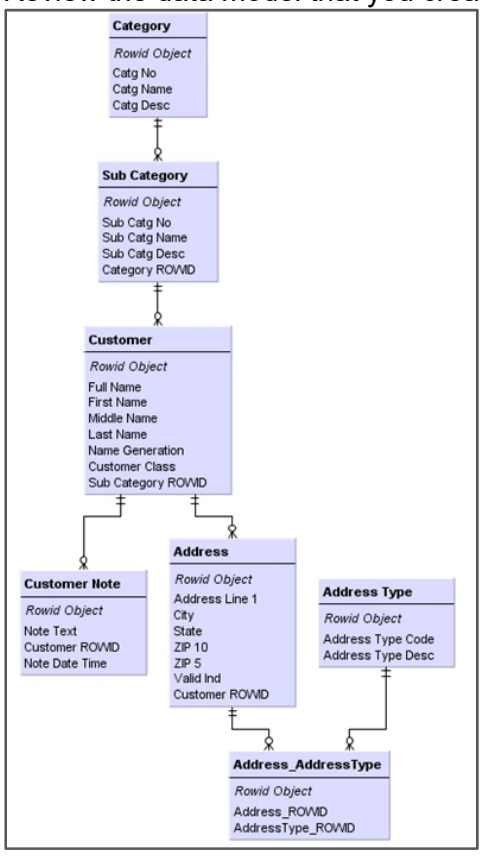
48. Add a new relationship to the **Customer** base object.
49. Select the referencing column as **Sub Category ROWID** and the referenced column as **Rowid_Object**.



50. Change the Lookup Display Name for the new relationship to **Sub Catg Desc**.



51. **Save** your changes and navigate to **Workbenches** > **Model** > **Schema Viewer**.
52. Review the data model that you created.



Note: If your schema view does not reflect the change, select the **Schema Viewer** > **Refresh** command. This command forces the viewer to recompute the diagram from the current schema.

This concludes the lab.

ADDITIONAL INFORMATION

Deleting Foreign Key Relationships:

You can delete any user-defined foreign key relationship that is added. You cannot delete the system foreign key relationships that Informatica MDM Hub automatically defines and enforces to protect the referential integrity of your schema. To delete a foreign key relationship between two base objects:

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, expand a base object and right-click **Relationships**.
4. On the Properties tab, click the foreign-key relationship that you want to delete.
5. Click the **Remove** button.
6. Schema Manager prompts you to confirm deletion.
7. Click **Yes**.
   Note: Schema Manager deletes the foreign key relationship.
8. Click the **Save** button.

Module 2: Define the MDM Data Model

Lab 2-3: Defining Lookups

Overview:

This lab is a continuation of the previous lab. In the previous labs, you defined the target data model with base objects and relationships that used foreign keys. You also created staging tables for the base objects, tied to specific source systems.

Now, for each foreign key column in a staging table, you need to define a lookup. During the Load process, when records are moved into base objects, the lookup will translate the incoming source foreign key value to a corresponding foreign key value in the target data model, thus preserving referential integrity.

For base objects, whose primary key is ROWID, you need to configure the MDM Hub to translate the source foreign key value to the corresponding ROWID, by locating the source foreign key value in the base object's XREF table.

Objectives:

- Define the lookup strategy to translate the CRM_ID on the Address record from the CRM system, into a Customer Rowid_Object value for the Customer ROWID foreign key

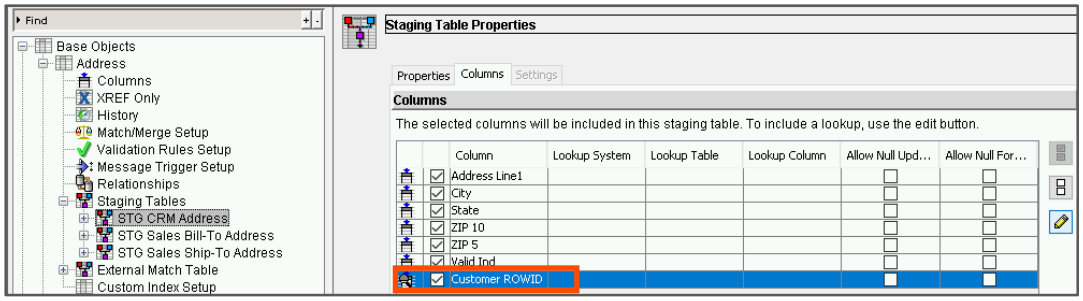
Duration:

15 minutes

Tasks

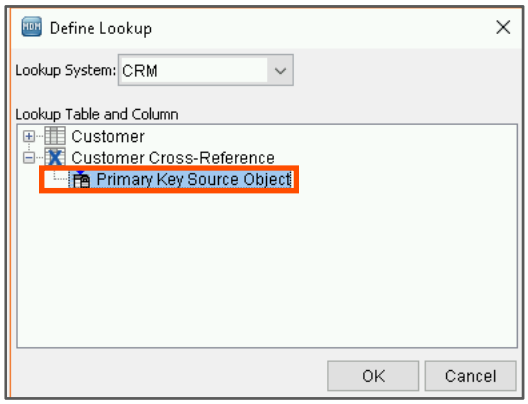
Define a Lookup

1. Go to the **Schema** tool.
2. Verify that you have acquired a **Write Lock**.
3. Navigate to the staging tables of the **Address** base object.
4. Select the **STG CRM Address** staging table.
5. Select the **Columns** tab.
6. Select the **Customer_ROWID** column.
   Note: The Edit Lookup Column button to the right of the columns list is enabled when you click on the Customer_ROWID column. This indicates that you can edit the lookup strategy for the selected column.

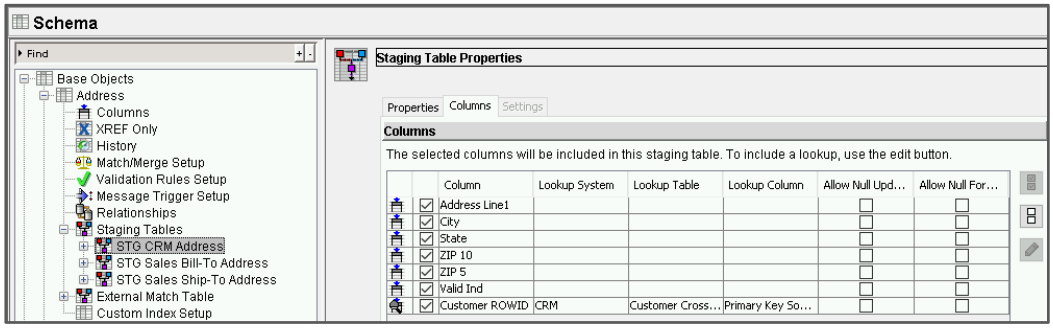


Note: If the relationship had not already been defined, you would not be able to define the lookup.

7. Click the **Edit Lookup Column** button. The **Define Lookup** dialog opens.
8. From the Lookup System drop-down, ensure that **CRM** is selected.
9. Under the **Customer Cross-Reference,** select **Primary Key Source Object**.



10. Click **OK**.



11. **Save** the object.
10. Ensure that lookups are defined for the following tables:
    - STG CRM Address
    - STG Sales Bill-To Address (selecting the Lookup System as **Sales)**
    - STG Sales Ship-To Address (selecting the Lookup System as **Sales)**
    - STG CRM Customer
    - CRM Customer Note
    - STG CRM Sub Category

This concludes the lab.

Module 4: Configure the Stage Process

Lab 4-1: Registering a Process Server

Overview:

The Process Server is a servlet that handles cleanse requests. This servlet is deployed in an application server environment. The servlet contains two server components:

- a cleanse server that handles data cleansing operations
- a match server that handles match operations

The Process Server is multi-threaded so that each instance can process multiple requests concurrently. It can be deployed on a variety of application servers. Informatica MDM Hub supports running multiple Process Servers for each Operational Reference Store (ORS). The cleanse process is generally CPU-bound. This scalable architecture allows you to scale your Informatica MDM Hub implementation as the volume of data increases.

Objective:

- Register a Process server

Duration:

5 minutes

Tasks

Register a Process Server

Each ORS needs to have metadata to enable it to connect to a process server in order to test the mappings and run Staging batch jobs.

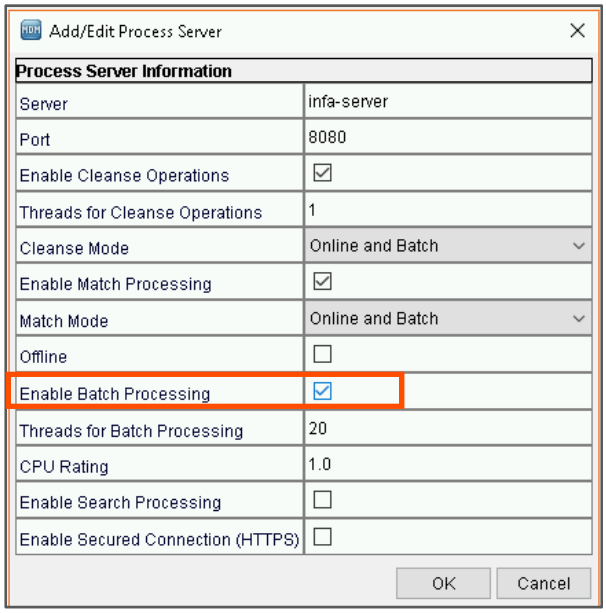
1. Ensure that you have acquired the write-lock on your schema – **Write Lock** > **Acquire Lock**.
2. Under the Utilities workbench, navigate to the **Process Server** tool.
3. Click the **Add** button.
4. In the Add/Edit Process Server dialog, enter **infa-server** as the **Server**.
5. Ensure that the port number is **8080**.
6. Ensure that the **Enable Cleanse Operations** and **Enable Match Processing** options are selected.

   Note

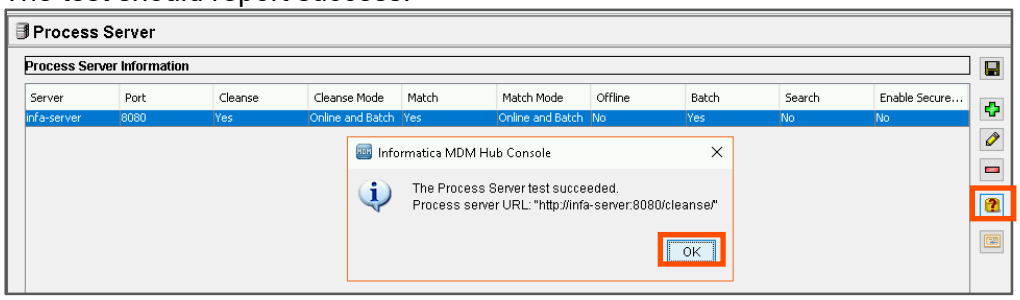
   If an ORS has multiple associated Process Servers, you can enhance performance by configuring each Process Server as either a match-only (**Enable Cleanse Operations** checked and **Enable Match Processing** unchecked) or a cleanse-only server (vice versa) per server setting. In this course, you have enabled both on the same server because your cleansing and matching loads will be very light.

7. Ensure that **Cleanse Mode** and **Match Mode** options are set to **Online and Batch**.

8. Ensure that the **Offline** checkbox is cleared.
   Note: If this option is selected, no cleanse jobs are sent to that Process Server.
9. Ensure that the **Enable Batch processing** checkbox is selected and retain the other settings.



10. Click **OK**.
11. Select the process server you have just added and click the **Test** button.
    The test should report success.



12. Click **OK**.
13. Click the **Save** button.

This concludes the lab.

Module 4: Configure the Stage Process

Lab 4-2: Creating a Basic Mapping

Overview:

A mapping defines movement of data from a landing table to a staging table. The simplest type of mapping only has **Copy Column** maps with no data standardization and no change in column values.

After the landing, staging, and base object tables have been defined; mappings need to be defined prior to executing Stage jobs. Mappings define the transformations performed in Stage jobs.

Objectives:

- Create a basic mapping
- Add a constant to the mapping
- Test the basic mapping
- Create a new mapping and use a conditional execution component
- Test the conditional execution component

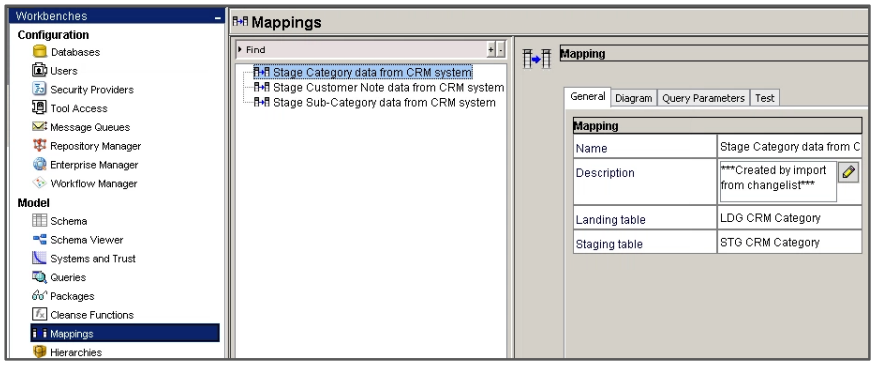
Duration:

80 minutes

Tasks

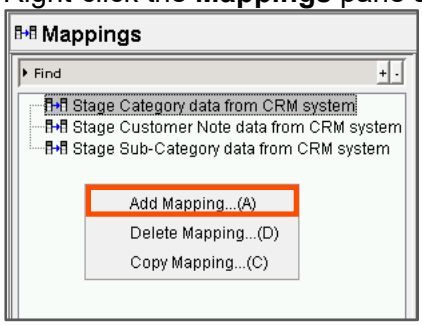
Create a Basic Mapping

1. Navigate to the **Model** > **Mappings** tool and acquire a **Write Lock** on your schema.

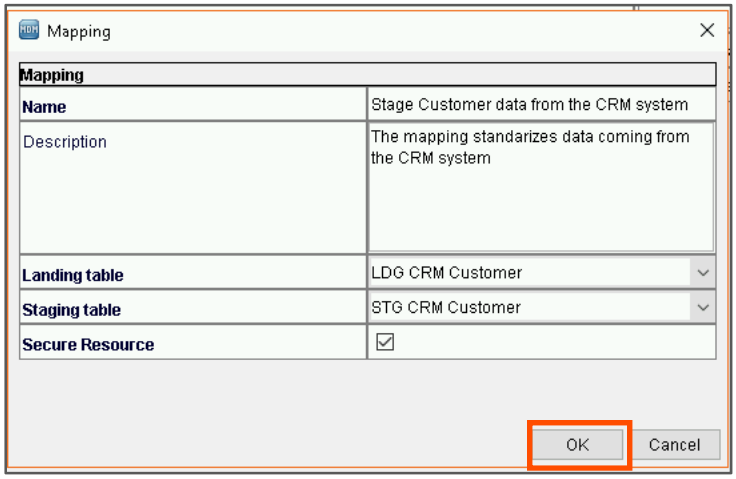


Note: Notice that there are already some mappings in your schema that you imported with the change file in the Relationships lab.

2. Right-click the **Mappings** pane and select **Add Mapping**.



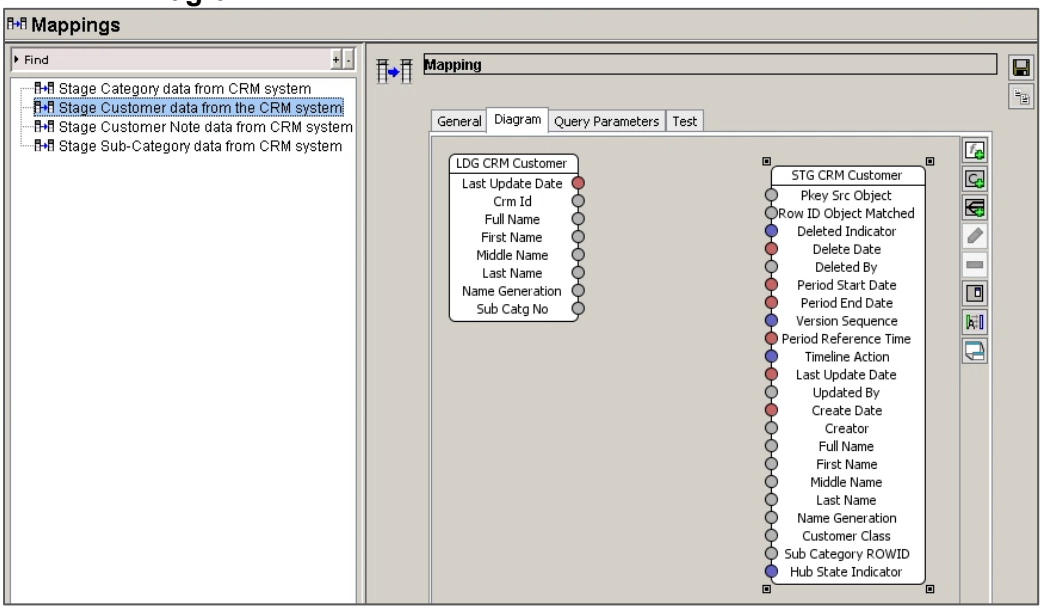
3. Enter the mapping name as **Stage Customer data from the CRM system**
4. Add a **Description**.
5. Select **LDG CRM Customer** as the landing table.
6. Select **STG CRM Customer** as the staging table.
7. Click **OK**.



Note: When you click OK, you cannot change the name of a mapping. So, ensure that you use the correct name before you proceed.

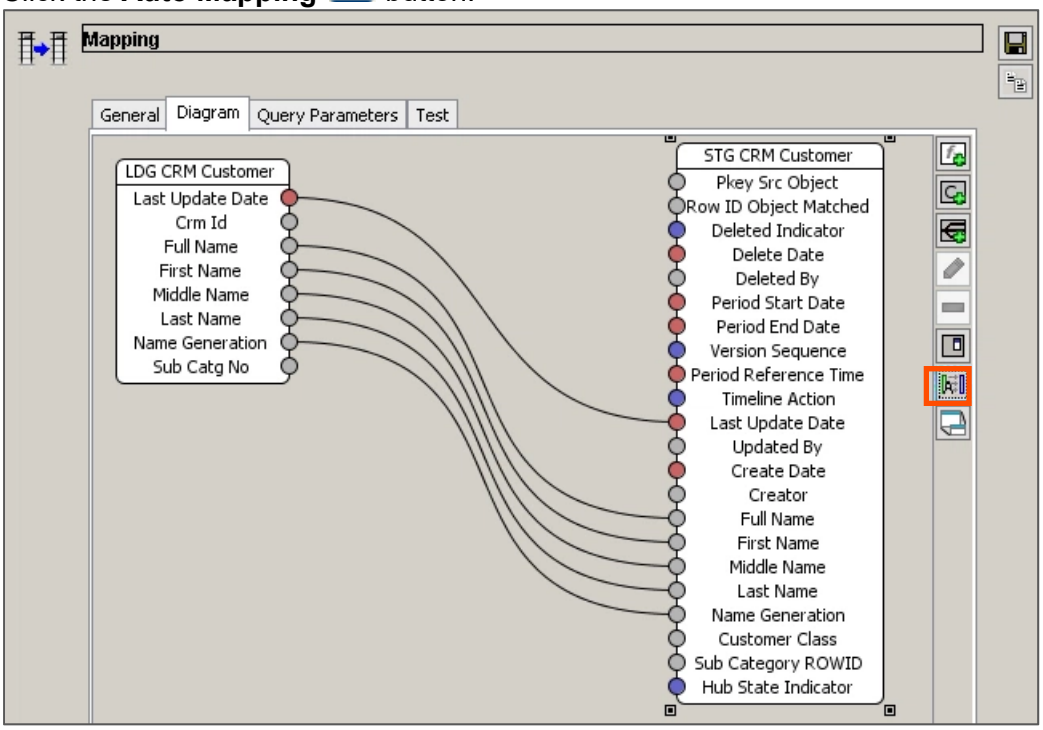
Also, if you do not see the new mapping created, release the lock and acquire it again.

8. Select the **Diagram** tab.



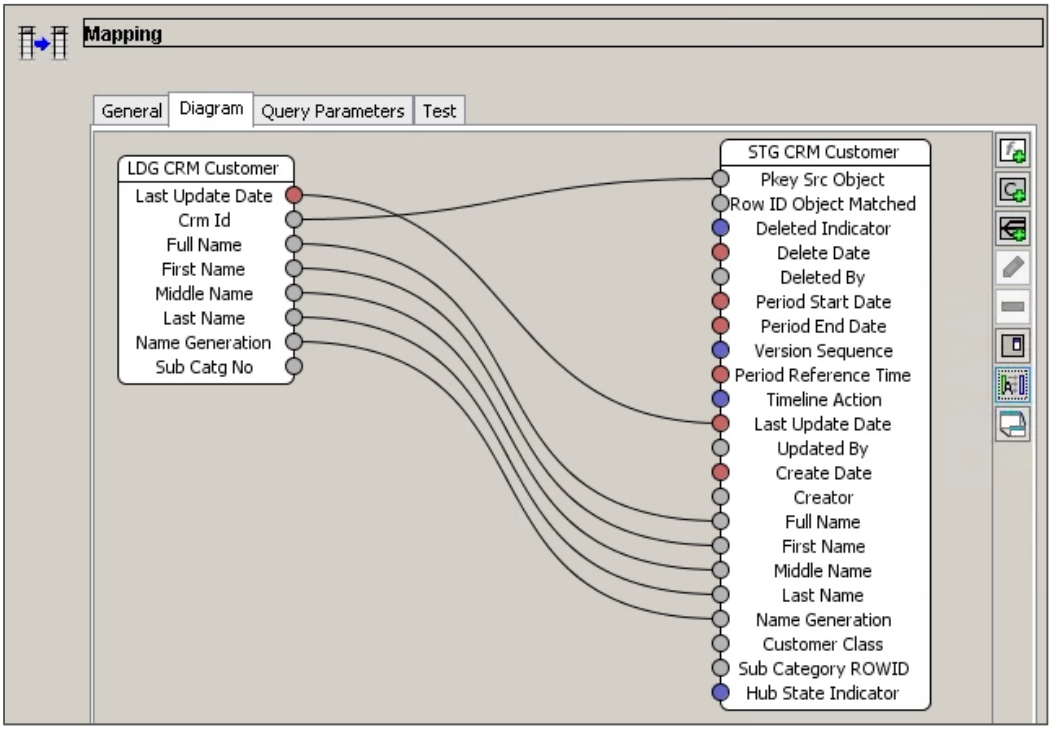
Note: The landing table shows on the left of the diagram and the staging table on the right. For these steps, maximize your window and drag pane borders in order to see the entire diagram. You can collapse the Workbenches pane to free up space.

9. Click the **Auto Mapping**  button.



Note: To map a column manually, you can select the column node in the landing table and drag it across the corresponding column node in the staging table.

10. Map the landing table's **Crm Id** column to the staging table's **Pkey Src Object** column.

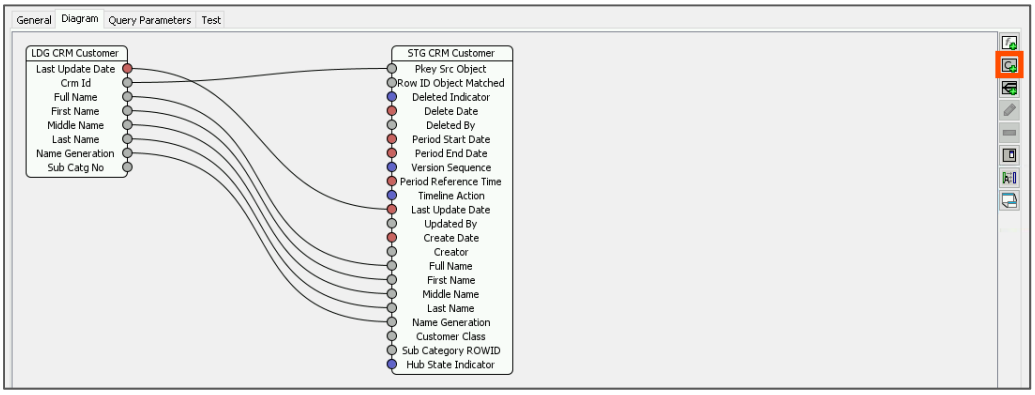


11. **Save** the mapping.

Use a Constant in a Mapping

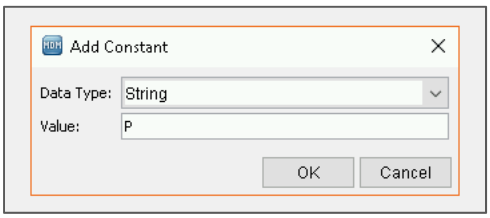
In this section, you will add a constant value of 'P' (Person) as the value for Customer Class for all CRM Customer records. Remember that all records coming from the CRM system represent individuals.

12. In the same mapping (**Stage Customer data from the CRM System**), click the **Add Constant** button.



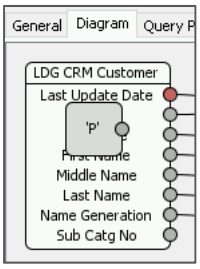
13. For the data type, select **String**.

14. In the Value field, enter **P**.

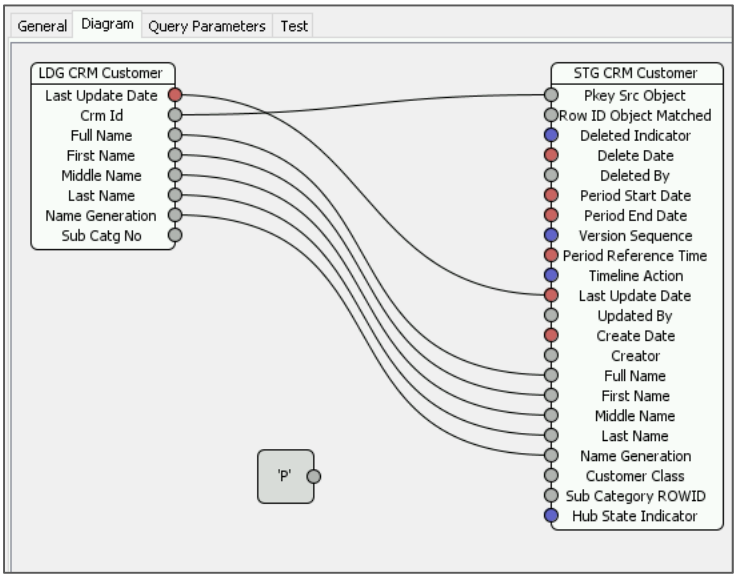


Note: Do not include any quotation marks around the value as it occurs automatically when you add a constant to the mapping.

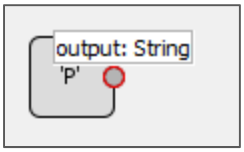
15. Click **OK**.



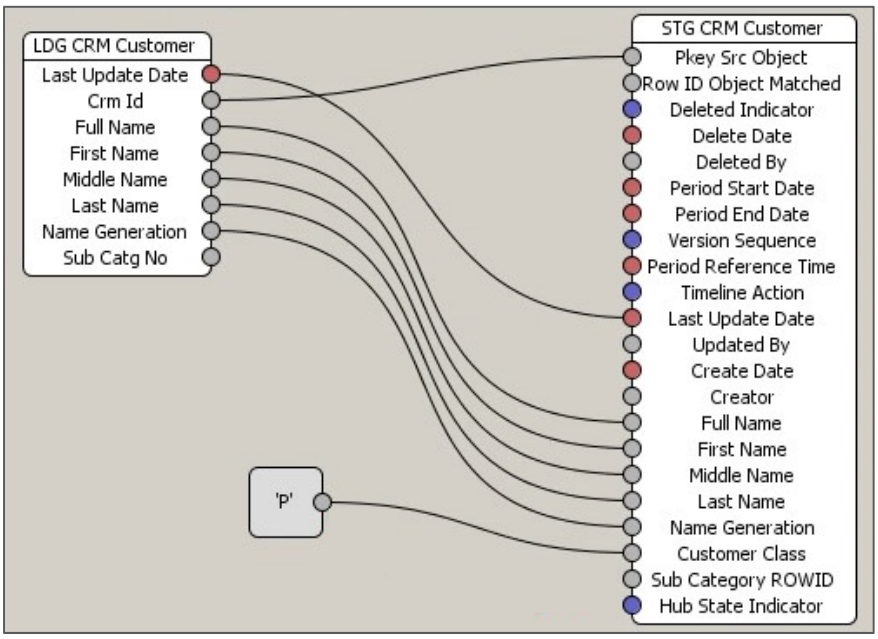
16. Drag the constant anywhere in the diagram where it does not overlap with any other diagram components.



17. Hover over the output node for the constant.



18. Drag the output area to the input area for **Customer Class** on the staging table.



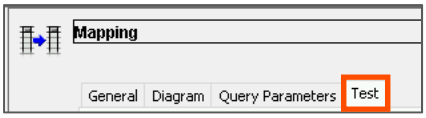
Note: Now, when the mapping is executed, every record in the staging table has the value ‘P’ in the **Customer Class** column.

19. **Save** the mapping.

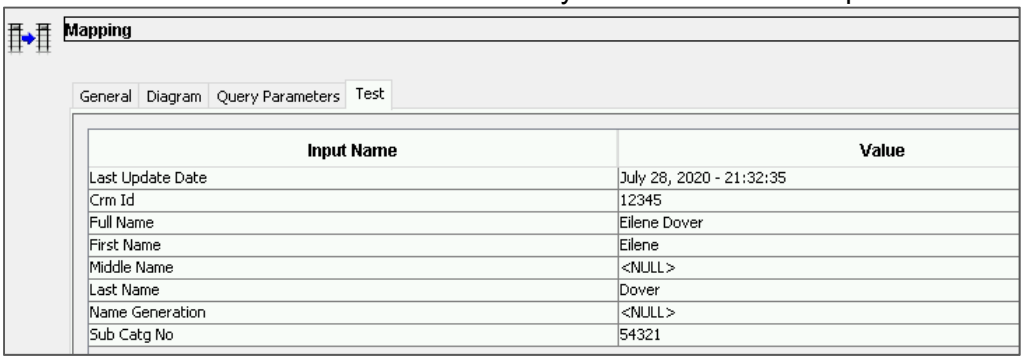
Test a Basic Mapping

In this section, you will test the basic mapping to ensure that input values produce the expected results.

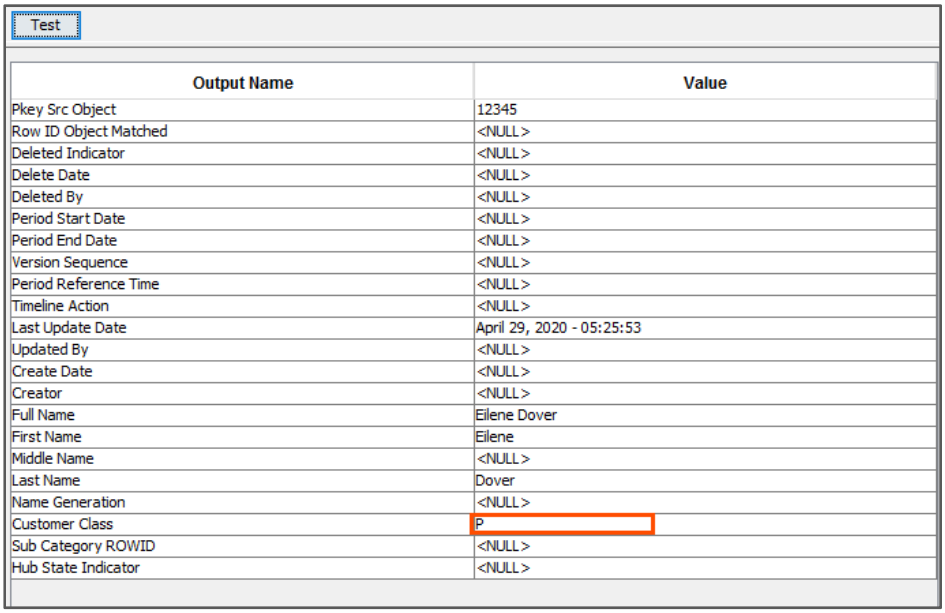
20. Select the **Test** tab.



21. Enter the test values as shown below:
Note: Use the date selector to choose today’s date for the Last Update Date field.



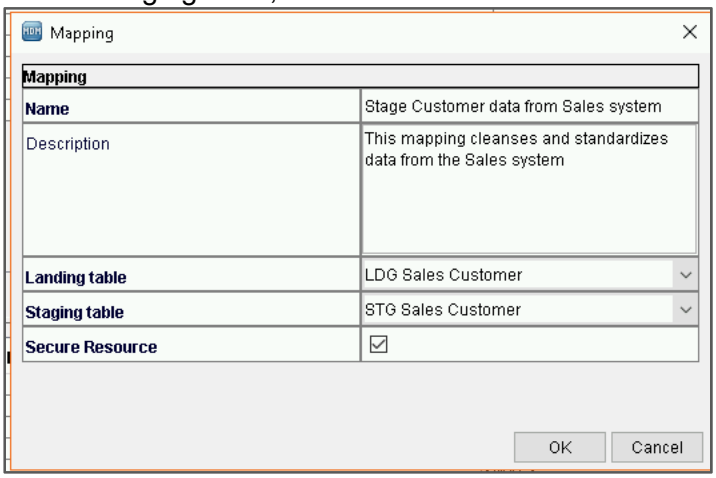
22. Click the **Test** button and review the outputs. Confirm that the output values are as expected from your mapping that includes the **Customer Class**, which should display the output value as **P**. This is because the input is a constant. If not, return to the diagram and correct the Mapping.



Use a Function and a Conditional Execution Component

In this section, you will create a new mapping from “LDG Sales Customer” to “STG Sales Customer”.

23. Right-click on **Mappings** and select **Add Mapping**.
24. In the Name field, enter **Stage Customer data from Sales system**.
25. Provide a description.
26. For the Landing table, select **LDG Sales Customer**.
27. For the staging table, select **STG Sales Customer**.



28. Click **OK**.

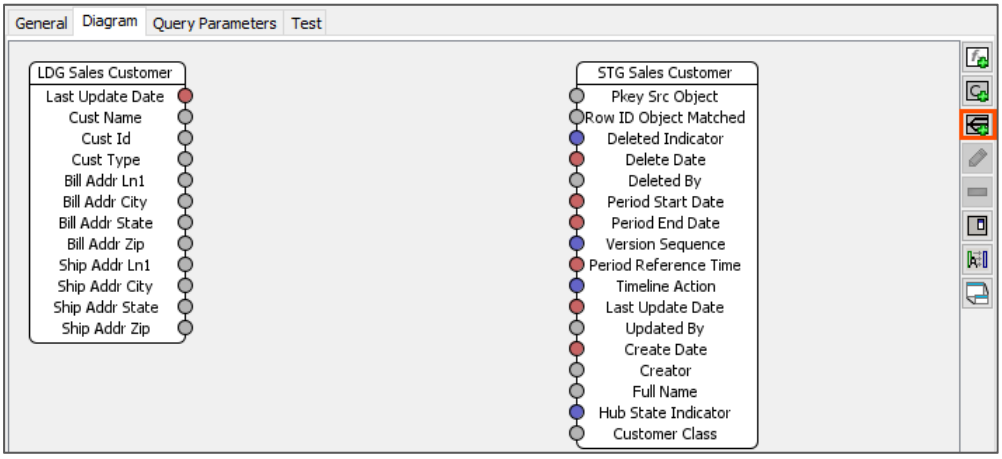
29. Click on the **Diagram** tab for the mapping.

Now, you will define the following logic:

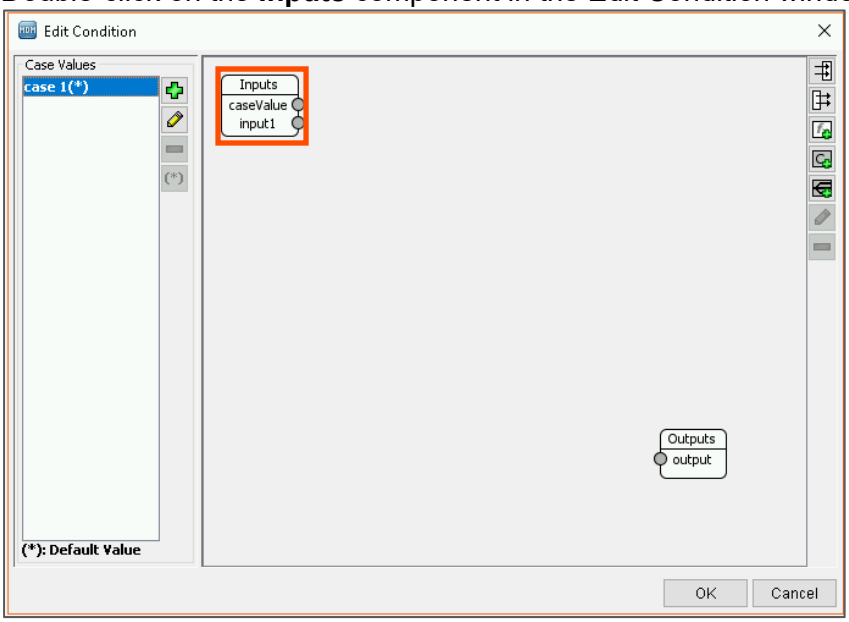
- If CaseValue = 'R', concatenate the word 'Region ' (note the space after the word) with the billing address ZIP code. Map the result to the conditional execution component output.
- If CaseValue = any other value, move the original Cust Name value to the conditional execution component output.

To achieve this, you must define the inputs and outputs for the conditional execution component, as well as the case values.

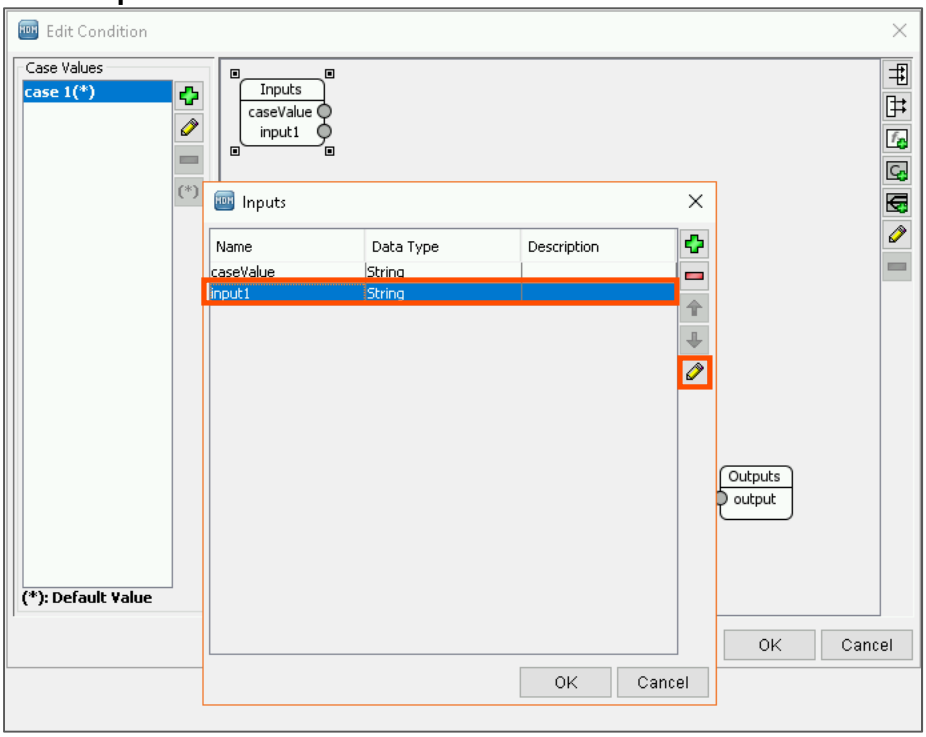
30. Click the **Add Conditional Execution Component** button.



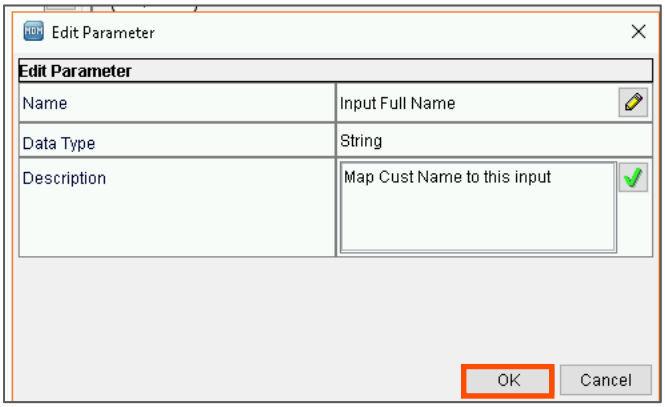
31. Double-click on the **Inputs** component in the Edit Condition window.



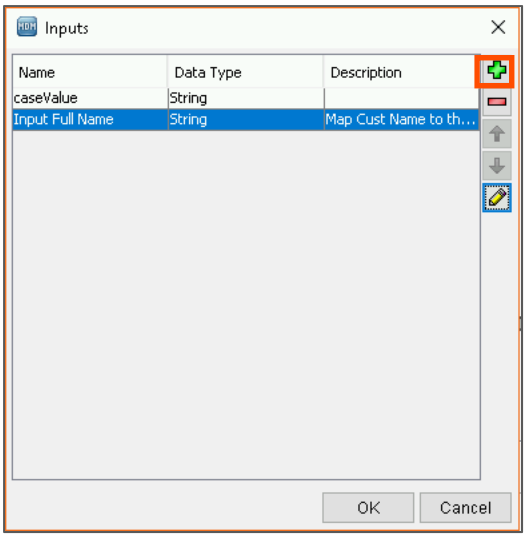
32. Select **input1** in the list and click the **Edit the Selected Parameter** button.



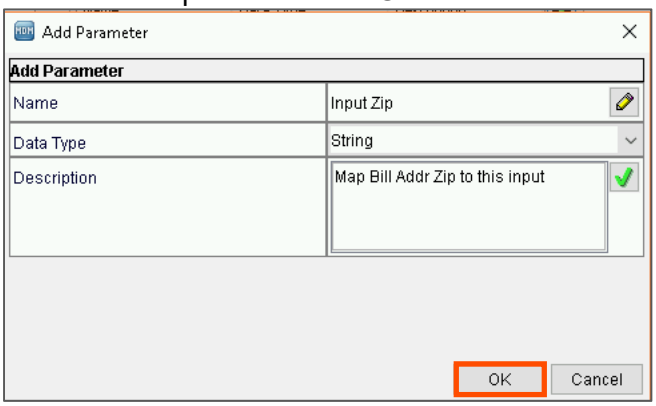
33. Change the name to **Input Full Name** and retain the data type as **String**.
34. Add a **Description** and click the green tick mark to save it.
35. Click **OK**.



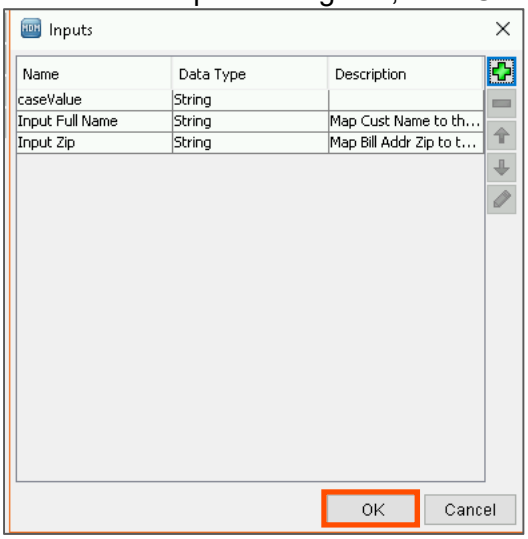
36. In the same Inputs dialog box, to add a new input for Zip, click the **Add a Parameter** button.



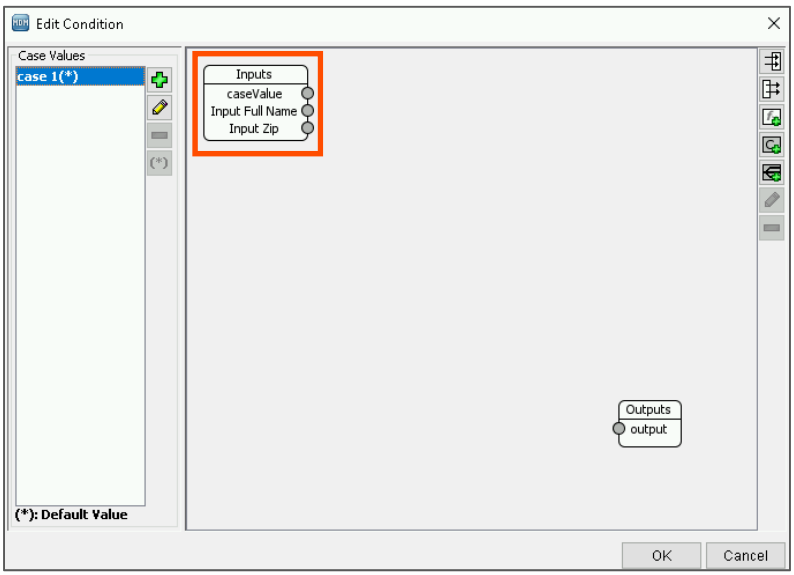
37. Enter the parameter name as **Input Zip** and retain the data type as **String**.
38. Enter a description and click **OK**.



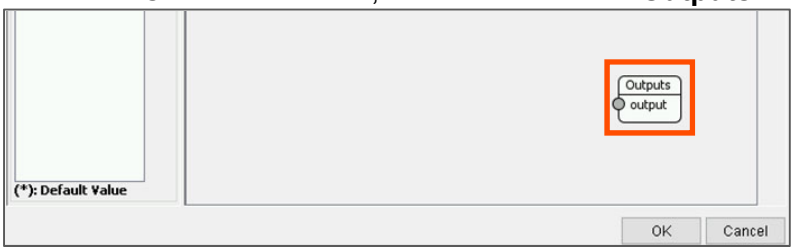
39. To close the Inputs dialog box, click **OK**.



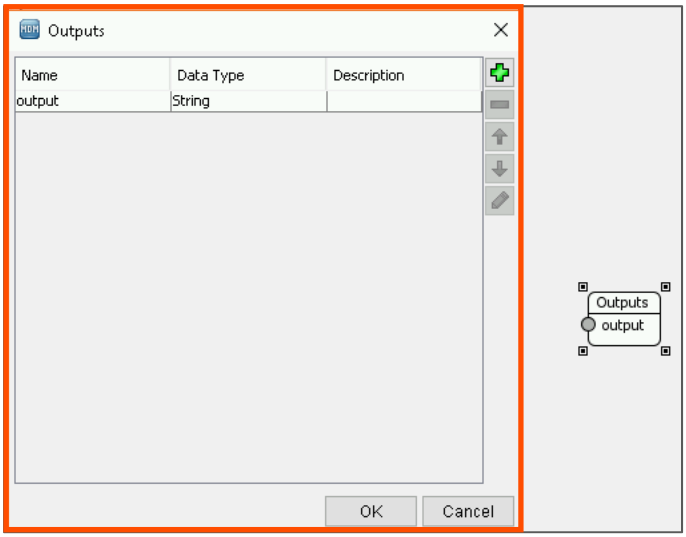
You will see the input as shown here:



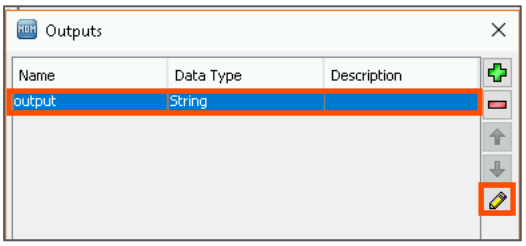
40. In the Edit Condition window, double-click on the **Outputs** component.



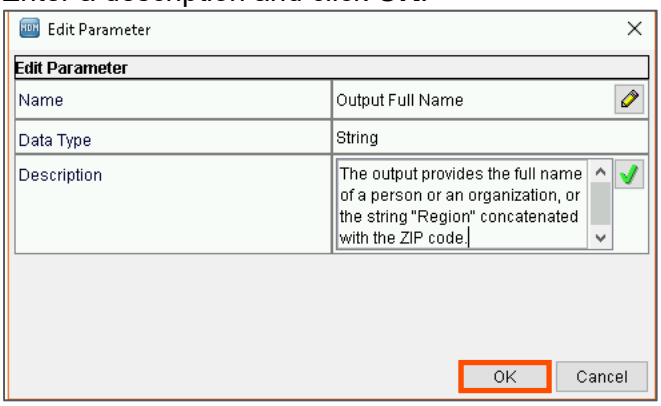
This opens the Outputs window.



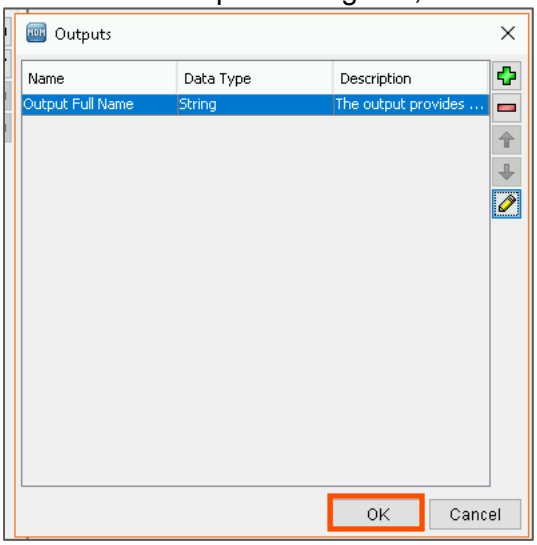
41. In the list, select **output** and click the **Edit the Selected Parameter** button.

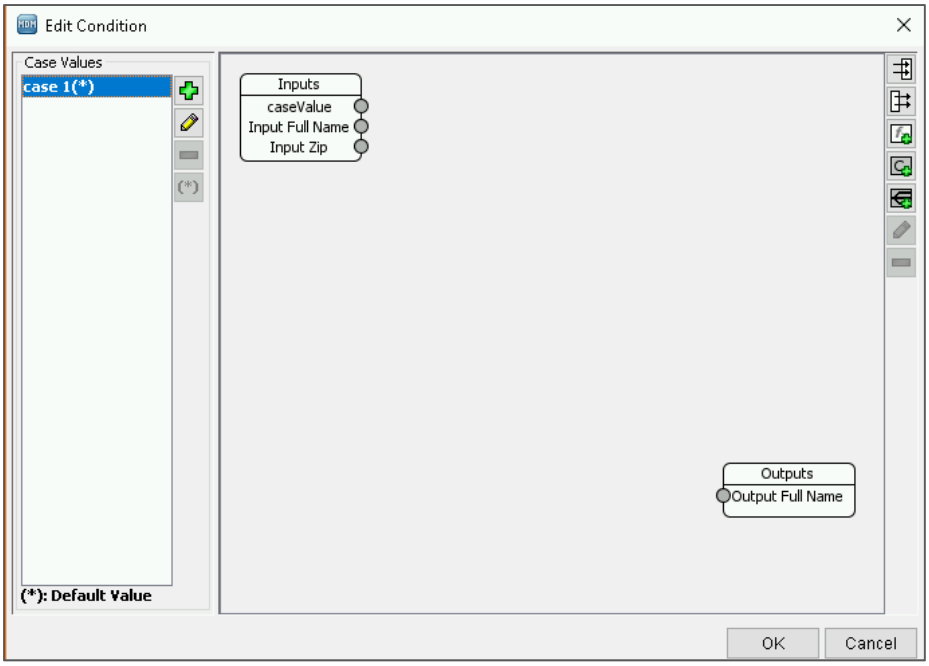


42. Change the Name to **Output Full Name** and ensure that the data type is **String**.
43. Enter a description and click **OK**.



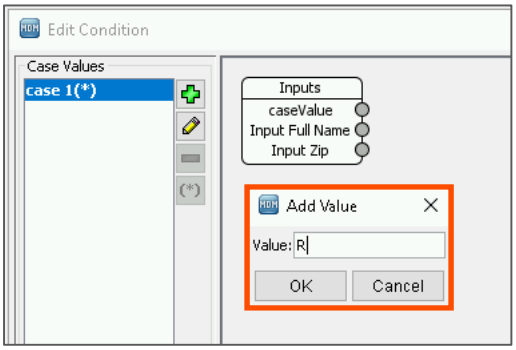
44. To close the Outputs dialog box, click **OK**.





You have defined the input and output parameters for the conditional execution component. Now, you will define the case values and the logic to be executed for each case value.

45. To add a new case value, click the **Add a New Case Value** button to the right of the case values list.
46. Enter the value as **R** and click **OK**.

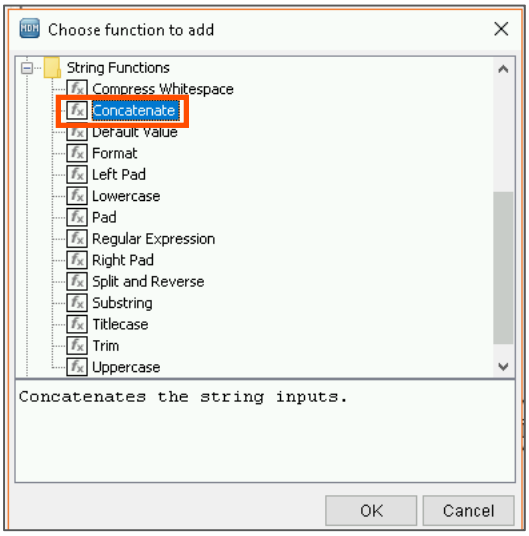


Note: Case values are always String values. The case values list now shows R and case 1(*).

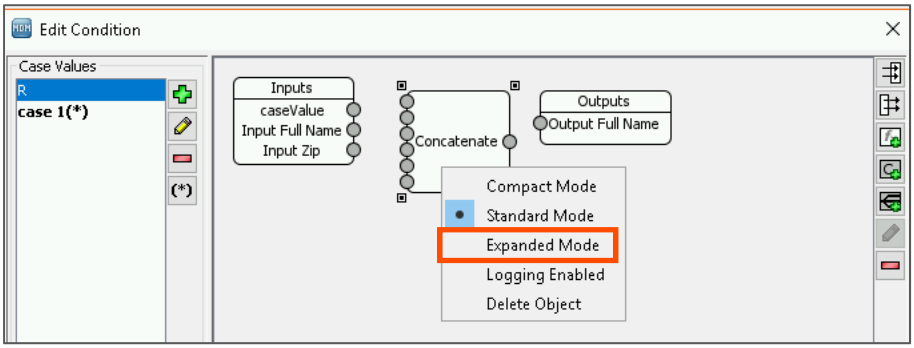
Now, you will define the logic for each case.

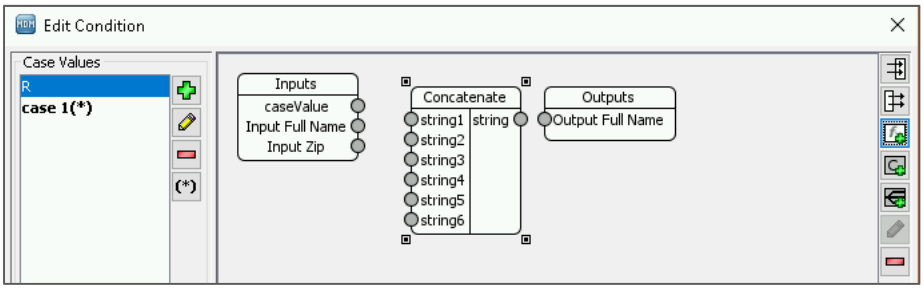
47. Select **R** in the case values list and click the **Add Function** button.

48. In the “Choose function to add” dialog box, from the **String Functions** list, select the **Concatenate** function.



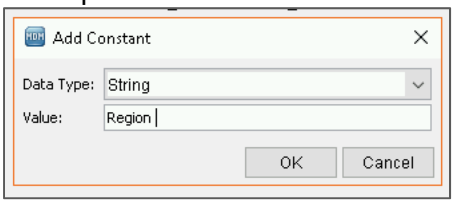
49. Click **OK**.
Note: The Concatenate function is added to the conditional execution component diagram.
50. Right-click the Concatenate function component and change its display mode to **Expanded Mode** so that you can see the input and output parameters better.



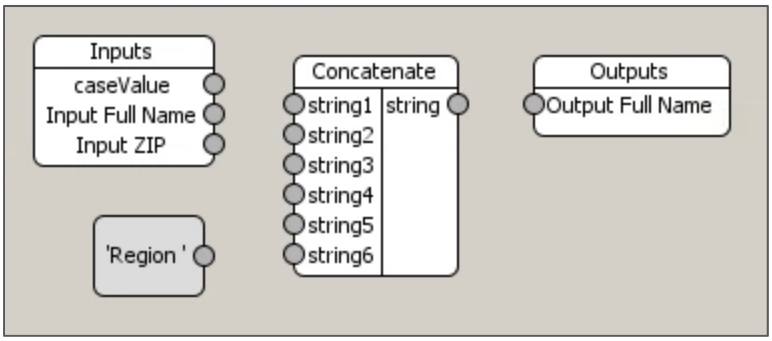


51. Click the **Add Constant**  button.
52. For the Data Type field, select **String**.

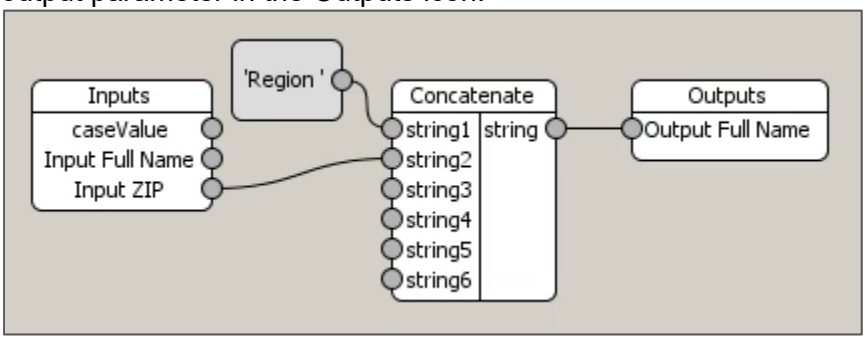
53. In the Value field, enter **Region** .
Note: Remember to include a space character at the end of the word so that there will be a space between the word and the ZIP code in the final output.



54. Click **OK**.

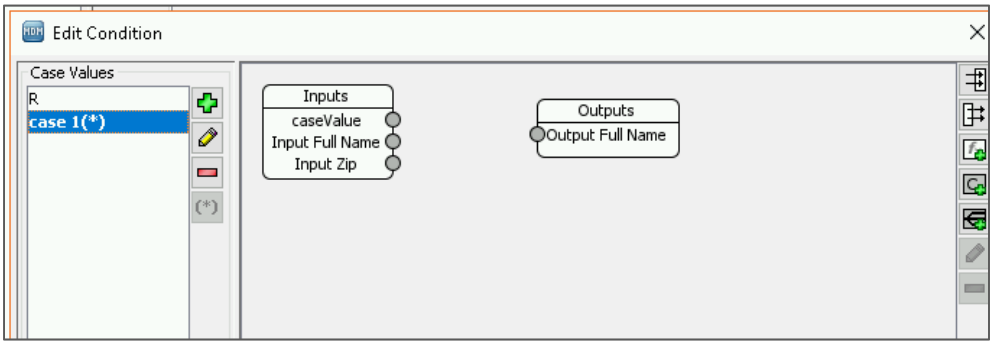


55. To create the execution logic for this case value, drag the **Region** output node to the input node for **string1** in the Concatenate function.
56. Map **Input Zip** to the **string2** input in the Concatenate function.
57. From the Concatenate function, drag the output string node to the **Output Full Name** output parameter in the Outputs icon.

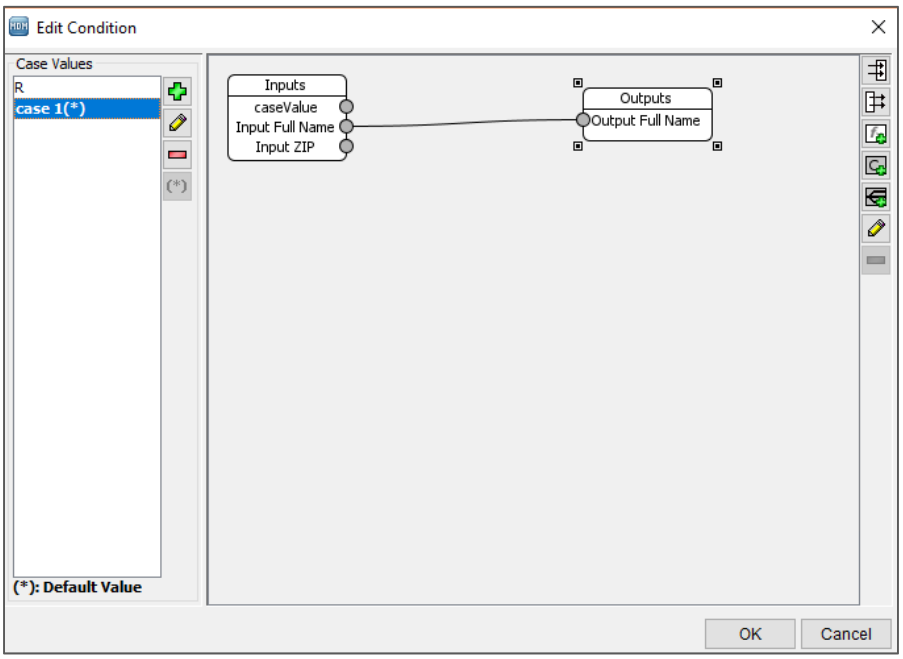


Define the Logic for all Other Values

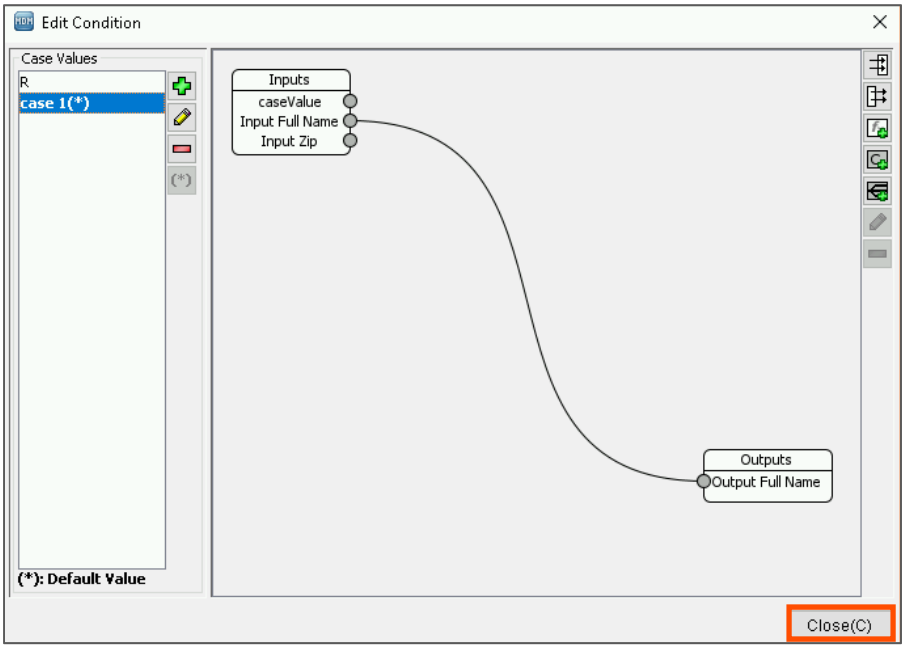
58. In the case values list, select **case 1 (*)**.



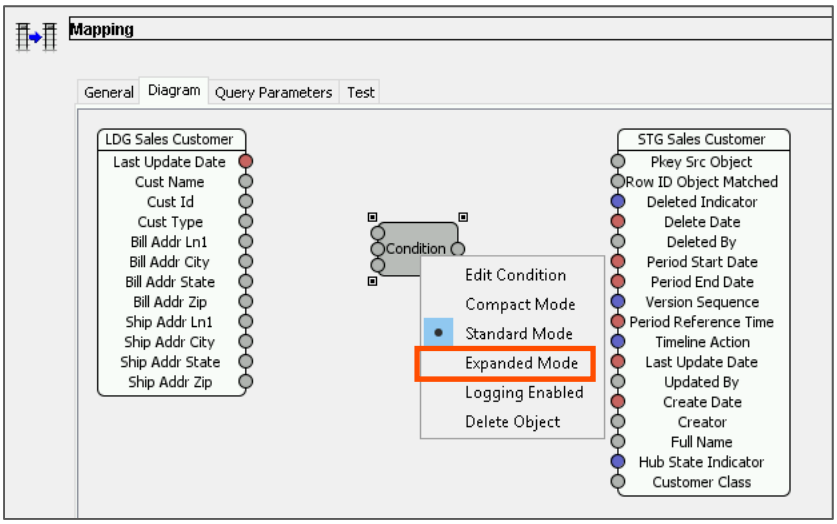
59. Drag the **Input Full Name** input parameter to the **Output Full Name** output parameter.

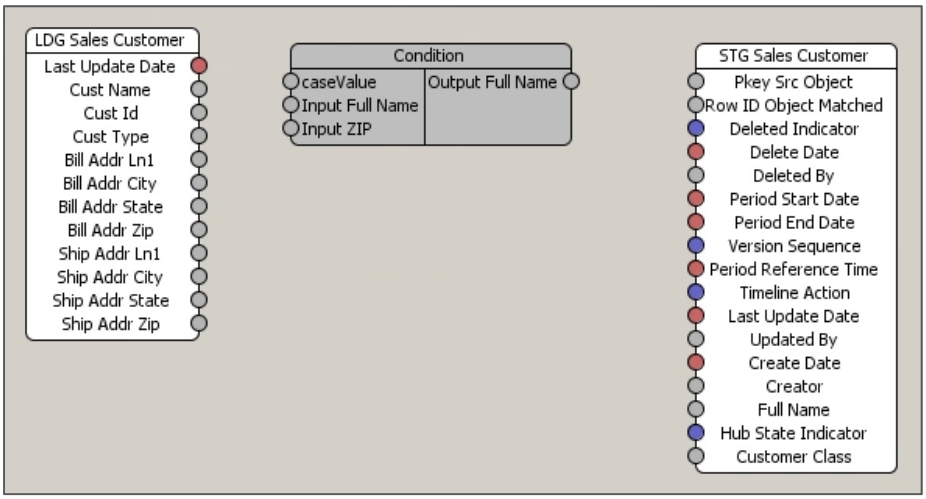


60. In the Edit Condition dialog box, click **Close**.
This will add the conditional execution component to the mapping.



61. Right-click the conditional execution component in your mapping and change its display mode to **Expanded Mode** so that you can see the input and output parameters you defined.

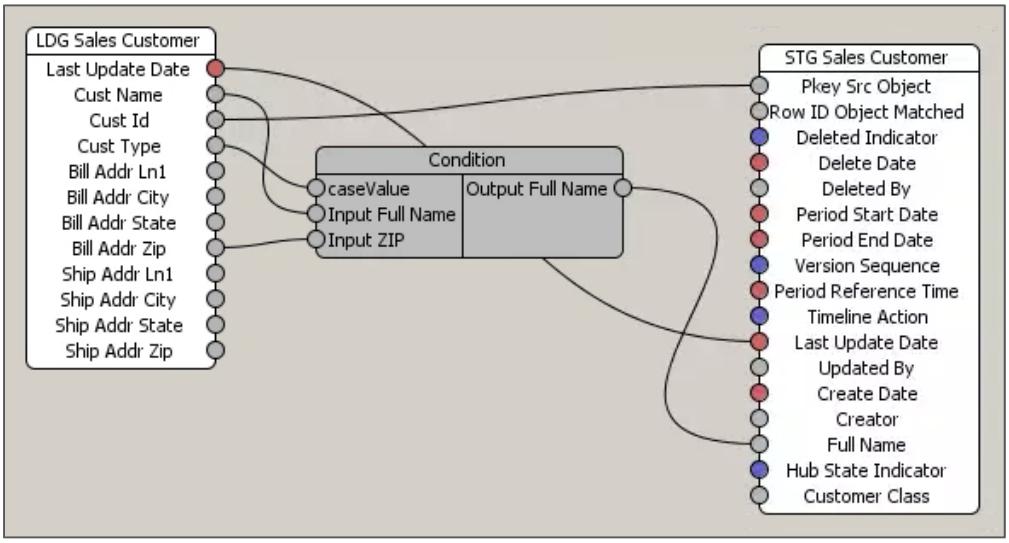




62. Map the conditional execution component parameters as follows:

Source	Node	Target	Node
Landing Table	Cust Type	Conditional Execution Component	caseValue
Landing Table	Cust Name	Conditional Execution Component	Input Full Name
Landing Table	Bill Addr Zip	Conditional Execution Component	Input ZIP
Conditional Execution Component	Output Full Name	Staging Table	Full Name

Landing Table	Cust Id	Staging Table	Pkey Src Object
Landing Table	Last Update Date	Staging Table	Last Update Date



63. **Save** the mapping.

Test a Conditional Execution Component in a Mapping

In this section, use what you have learned in the preceding walkthroughs to complete the Skills Application exercise.

In the previous walkthrough, you created a Conditional Execution Component in a new mapping to address source data where some customers are without name or address information other than the ZIP code (which you use for grouping sales that are reported per region).

If **Cust Type** in the landing table record is **R**, the record represents a region, and for these types of records, the mapping creates a value for the full name that consists of the word **Region** concatenated with the ZIP code.

64. Click the **Test** tab and provide values for the following input fields:

Case 1: Cust Type = R

- Last Update Date: <Enter Today's date>
- Cust Name: <blank>
- Cust ID: 12345
- Cust Type: R
- Bill Addr Zip: 98765

Check the resulting values in the output fields:

- Pkey Src Object: 12345
- Last Update Date: <Today's date>
- Full Name: Region 98765

Case 2: Cust Type = P or O

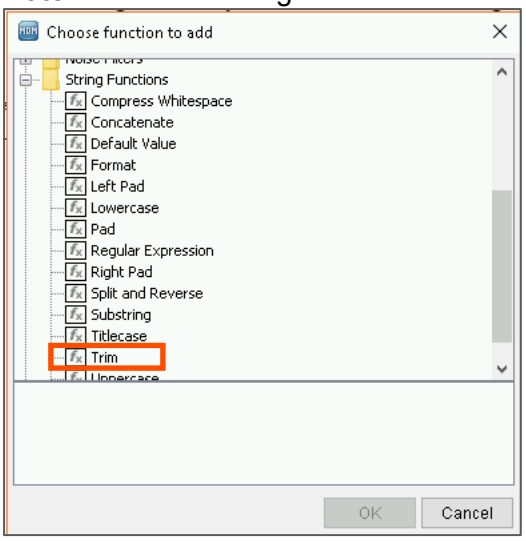
- Last Update Date: <enter Today's date>
- Cust Name: Angie O'Gram
- Cust ID: 12345
- Cust Type: P
- Bill Addr Zip: 98765

Check the resulting values in the output fields:

- Pkey Src Object: 12345
- Last Update Date: <Today's date>
- Full Name: Angie O'Gram

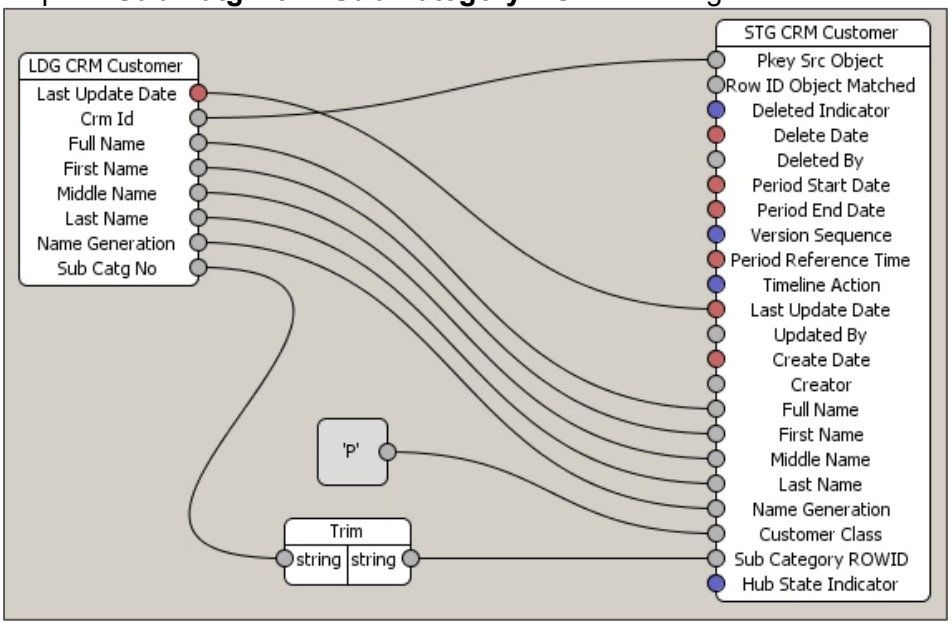
Create Additional Mappings

65. Navigate to the **Mappings** tool.
66. From the CRM system mapping, return to **Stage Customer Data from the CRM system**.
67. Add a **Trim** function to the mapping. To include a Trim function, click the **Add Function** icon on the right pane, and select **String Functions** > **Trim**.
    Note: You can also right click on a blank space in the mapping, and click **Add Function**.



The Trim function is a string function that trims white space from the beginning and end of a string.

62. Map the **Sub Catg No** to **Sub Category ROWID** through the **Trim** function.



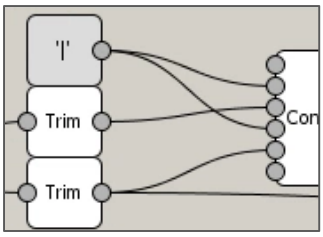
68. **Save** the mapping.
69. **Test** the results by giving some untrimmed values for the Sub Category No. (such as 3456).

Create a Mapping for Staging CRM Address Data

70. In the CRM system, add a mapping called **Stage Address Data from CRM System**.
71. Select the landing table as **LDG CRM Address**.
72. Select the staging table as **STG CRM Address**.
73. Create column mappings as mentioned in the table below:

Source Column(s)	Transformation	Target Column
Last Update Date	None: copy-column mapping	Last Update Date
CRM Id	Trim whitespace	Customer ROWID
CRM Id Addr Id Address Line	Concatenate CRM Id, Addr Id, and Address Line, separated with a '|' (pipe / vertical bar) character. For example: 1234|5678|99 Main St Trim whitespace from the three columns before concatenating.	Pkey Src Object
Address Line	Trim whitespace	Address Line 1
City Name	Trim whitespace	City
State Code	Trim whitespace	State
Zip	Trim whitespace	ZIP 10

Zip	The first 5 characters of the zip code (hint: use a String function for this)	ZIP 5

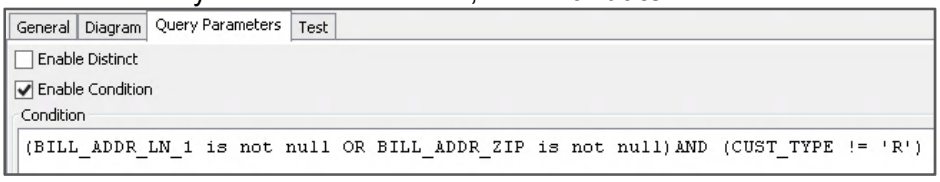


74. **Test** the mapping using values that you are familiar with, such as mailing and billing addresses, IDs, and so on.

Create a Mapping for Staging Sales Billing Address Data

75. Add a mapping **Stage Billing Address data from Sales system**.
76. From the Sales system, select the landing table as **LDG Sales Customer**
77. Select the staging table as **STG Sales Bill-To Address**.
78. In the **Query Parameters** tab, select **Enable Condition**.
79. In the **Condition** text field, enter **(BILL_ADDR_LN_1 is not null OR BILL_ADDR_ZIP is not null) AND (CUST_TYPE != 'R')**

    Recommended: Copy/paste of single quotes do not work. It is recommended that you hard code the condition text.
80. To check the syntax of the condition, click **Validate**.



81. Create column mappings as mentioned in the table below:

Source Column(s)	Transformation	Target Column
Last Update Date	None	Last Update Date
Cust Id	Trim whitespace	Customer ROWID
Cust Id Bill Addr Ln 1	Concatenate Cust Id and Bill Addr Ln 1, separated with a '|' (pipe, a vertical bar) character. For example: 1234|99 Main St Trim whitespace from both columns before concatenating.	Pkey Src Object
Bill Addr Ln 1	Trim whitespace	Address Line 1
Bill Addr City	Trim whitespace	City
Bill Addr State	Trim whitespace	State
Bill Addr Zip	Trim whitespace	ZIP 10

Bill Addr Zip	The first 5 characters of the zip code (hint: use a String function for this)	ZIP 5

82. **Save** the mapping.
83. **Test** the mapping with the help of values that you are familiar with – billing addresses, IDs, and so on.

Now, you need to create a mapping named **Stage Shipping Address data from Sales system**. Because the landing table and target base object for this mapping are the same as for the previous mapping you created, you can simply copy that mapping and use it as a starting point for this mapping:

84. Right-click the mapping **Stage Billing Address data from Sales system** and select **Copy Mapping** button.
85. Enter the name **Stage Shipping Address data from Sales system**.
86. Select the staging table as **STG Sales Ship-To Address** and click **OK**.
87. In the Query Parameters tab, in the Condition text field, change the text to **(SHIP_ADDR_LN_1 is not null)**.
88. To check the syntax of the condition, click **Validate**.
89. Create the column mappings as mentioned in the table below:

Source Column(s)	Transformation	Target Column
Last Update Date	None	Last Update Date
Cust Id	Trim whitespace	Customer_ROWID
Cust Id Ship Addr Ln 1	Concatenate Cust Id and Ship Addr Ln 1, separated with a '|' (pipe / vertical bar) character. For example: 1234|99 Main St	Pkey Src Object
Ship Addr Ln 1	Trim whitespace	Address Line 1
Ship Addr City	Trim whitespace	City
Ship Addr State	Trim whitespace	State
Ship Addr Zip	Trim whitespace	Zip 10
Ship Addr Zip	The first 5 characters of the zip code (hint: use a String function for this)	Zip 5

90. **Save** the mapping.
91. **Test** the mapping with the help of values that you are familiar with.

This concludes the lab.

ADDITIONAL INFORMATION

If you have created a mapping and need to correct the name of the mapping at this stage:

- Create a copy of the incorrectly named mapping
- Name the copy with the correct name
- Remove the incorrect mapping

Below are instructions on how to copy, remove, or edit mappings:

Copying Mappings

To copy a mapping:

1. Open the **Mappings** tool.
2. Acquire a write lock.
3. Right-click the mapping that you want to copy, and then choose **Copy Mapping**. The Mappings tool displays the Mapping dialog.
4. Specify the mapping properties.
5. Click **OK**.
6. Click the **Save** button to save your changes.

Removing Mappings

To remove a mapping:

1. Open the **Mappings** tool.
2. Acquire a write lock.
3. Right-click the mapping that you want to delete, and then choose **Delete Mapping**. The Mappings tool prompts you to confirm deletion.
4. Click **Yes**.

The Mappings tool drops supporting tables, removes the mapping from the metadata, and updates the list of mappings.

Note: You can edit some properties in a mapping, such as the description, diagram, and query parameters without any need to create a copy. Below are instructions on how to edit a mapping (you cannot edit the name).

Editing Mapping Properties

To create a new mapping by copying an existing one:

1. Open the **Mappings** tool.
2. Acquire a write lock.
3. Select the mapping that you want to edit.
4. Edit the mapping properties, diagram, and mapping settings as needed.
5. Click the **Save** button to save your changes.

Module 4: Configure the Stage Process

Lab 4-3: Creating a Cleanse List and a Graph Function in a Mapping

Overview:

Cleanse lists and graph functions validate, cleanse, standardize, and transform data in a mapping. In this lab, you will create a simple cleanse function to transform and standardize Customer Type values (Cust Type) to appropriate Customer Class values.

You will also create a simple graph function that will simulate the validation of an address. If the address contains a ZIP code, it will be considered valid.

Objectives:

- Create and use a cleanse list
- Create and use a graph function
- Define Delta Detection Options
- Define Raw Retention Periods

Duration:

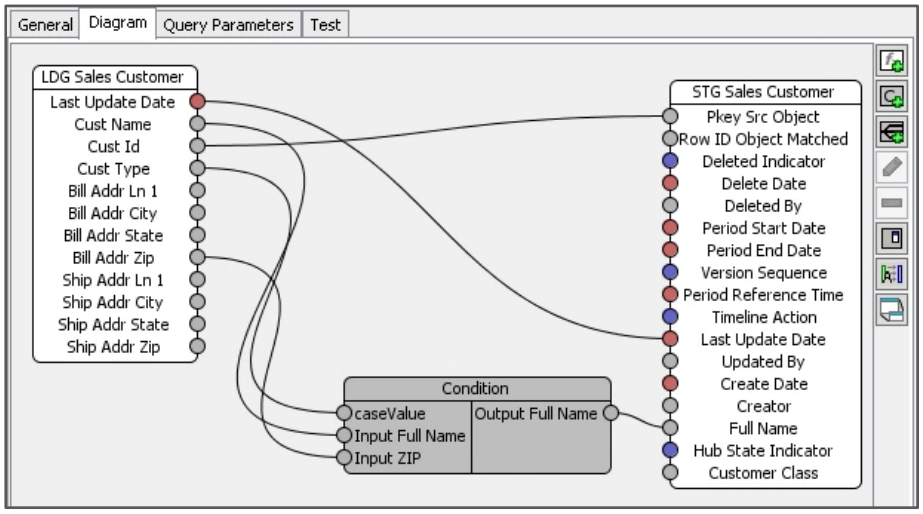
50 minutes

Tasks

Use a Cleanse List in a Mapping

In this section, you will add a Cleanse Function to the mapping “Stage Customer data from the Sales system”. The Cleanse Function replaces various values (IND, R, RLO, RSO, WO, WR, and any other unknown value) from the Customer Type column of the landing table to specific single-letter values (P, R, O, or U) for the Customer Class column of the staging table.

1. Open the **Mappings** tool.
2. Select the **Stage Customer data from Sales system** mapping and click the **Diagram** tab.



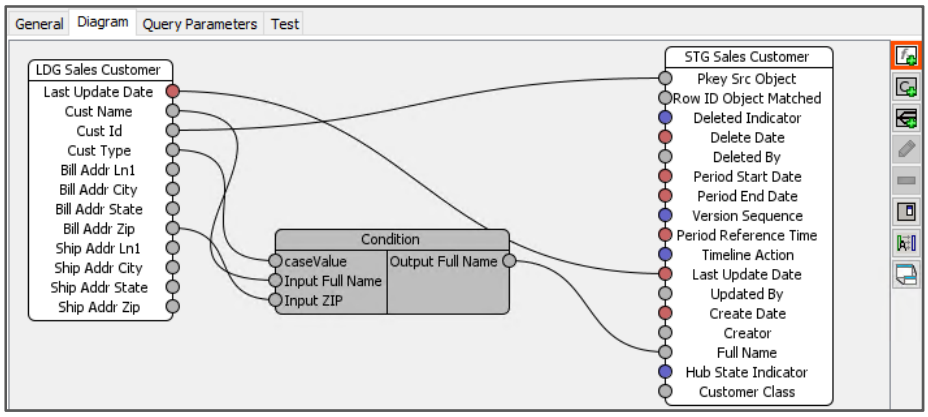
Before proceeding further with the mapping, observe the logic that the cleanse list executes.

3. In the Model workbench, open the **Cleanse Functions** tool.
4. Expand the **Training** cleanse function group and select the **CL - Sales Customer Type** function.
5. Select the **Details** tab and review the search and replace list:

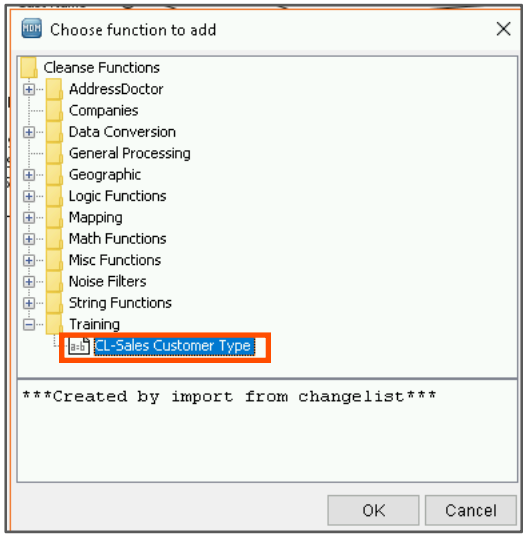
Search String	Output String	Match Type
IND	P	Exact Match
R	R	Exact Match
RLO	O	Exact Match
RSO	O	Exact Match
WO	O	Exact Match
WR	O	Exact Match

6. Navigate back to the **Mappings** tool and acquire a **Write Lock** on your schema, if needed.
7. Select the **Stage Customer data from Sales system** mapping and select the **Diagram** tab.

8. Click the **Add Function** button.

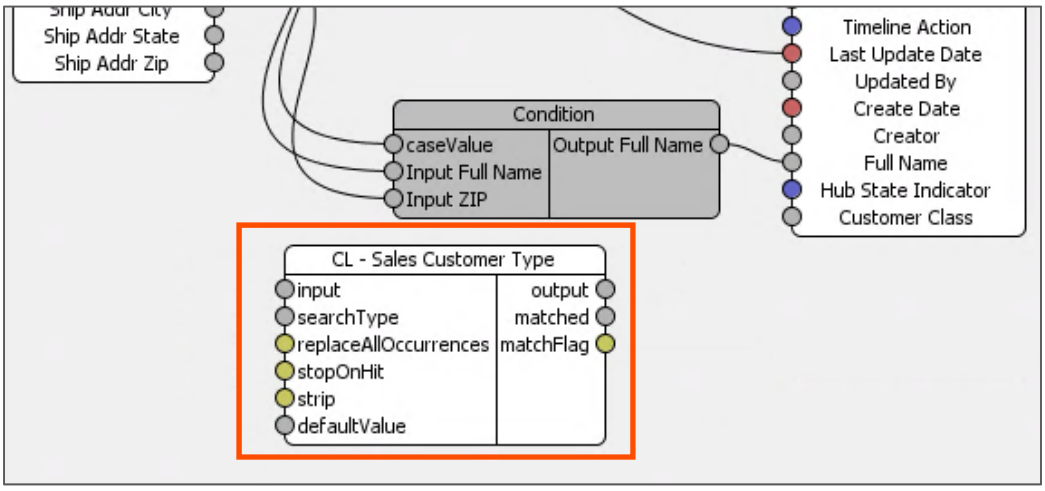


9. Expand the **Training** group and select **CL - Sales Customer Type**.



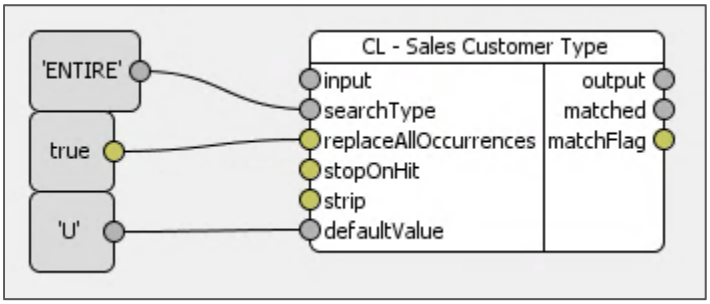
10. Click **OK**.

11. In the mapping, right-click on the cleanse list component and change its display mode to **Expanded Mode**.



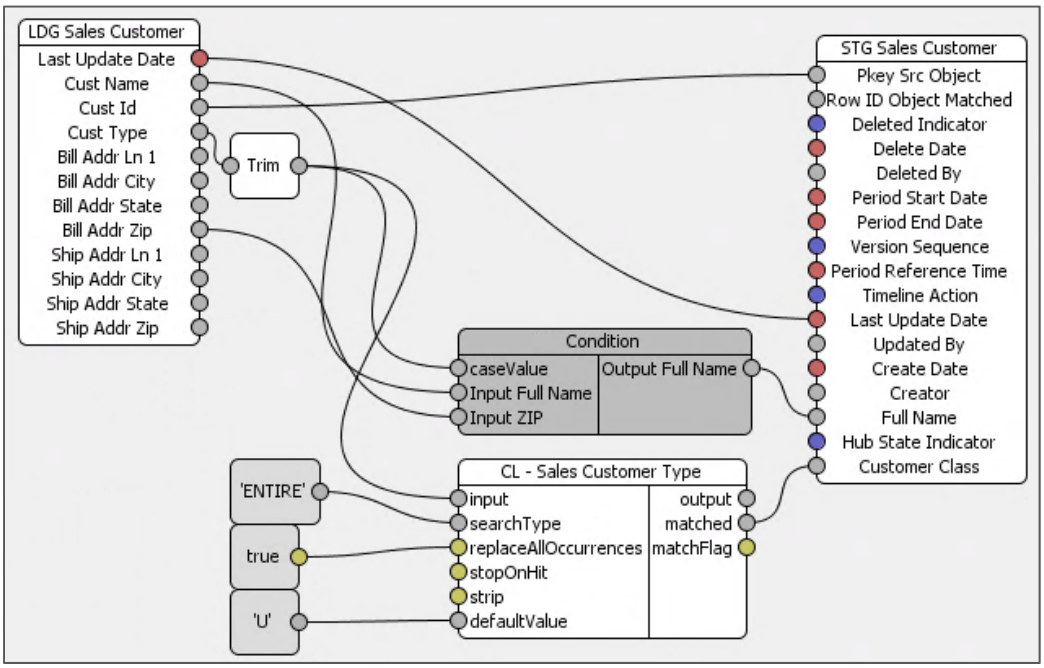
12. Add the following constants to the mapping and map them accordingly:
These constants will be used to provide values for the function's input parameters.

Constant Value	Output
ENTIRE (String)	CL – Sales Customer Type > searchType
True (Boolean)	CL – Sales Customer Type > replaceAllOccurences
U (String)	CL – Sales Customer Type > defaultValue



13. To remove unwanted spaces, use a **Trim** function between these **Cust Type** and **input** fields.
14. From the landing table, map **Cust Type** to **input** in the cleanse list function.
Note: Map the Cust Type to the Trim function, and from the Trim function to the input field of the cleanse list function.
15. **Delete** the existing link from Cust Type to caseValue.
16. Also, map the output of this Trim function (coming from Cust Type) to **caseValue** in the conditional function.

17. From the cleanse list function, map the **matched** output node to **Customer Class** in the staging table.



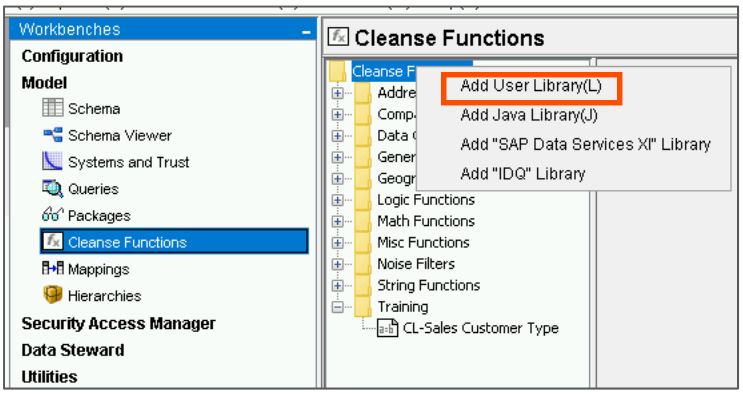
18. **Save** the mapping.
19. To test the mapping, select the **Test** tab.
20. Enter dummy input values in the **Last Update Date**, **Cust Name**, **Cust Id**, **Bill Addr Zip** fields.
21. For the **Cust Type** field, one by one, enter **IND**, **R**, **RLO**, **RSO**, **WO**, and **WR** and observe the resulting **Customer Class** value. Each of the above values should resolve as **P**, **R**, **O**.
    Note: 'U' is displayed when you provide any other value for Cust Type (other than IND, R, RLO, RSO, WO, WR).
22. While testing the Cust Type value for **R**, ensure that the Full Name field results with **Region <ZIP code>**.

Create a Cleanse List

You will create a Cleanse List that will be used to standardize Name Generation values (JUNIOR/JNR/JR, SENIOR/SNR/SR).

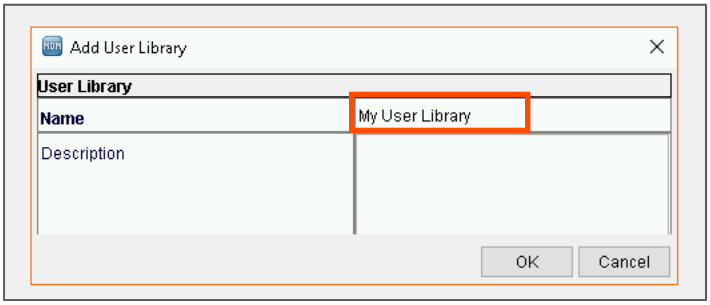
23. Ensure that you acquire the **Write Lock**.
24. From the Workbenches pane, select the **Cleanse Functions** tool.

25. At the top of the cleanse functions tree, select and right-click the **Cleanse Functions** node and select **Add User Library**.



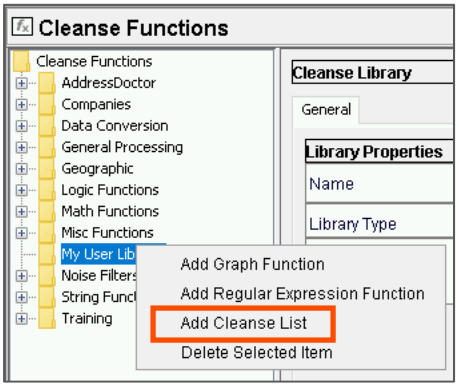
This creates a group under which you can create your own cleanse graphs and cleanse lists.

26. Name the user library **My User Library** and click **OK**.



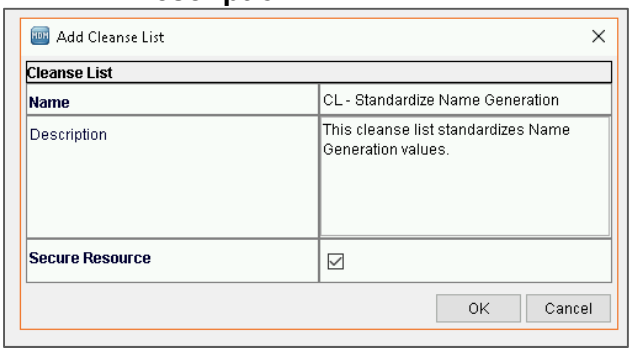
Note: If you do not see the library created, release and acquire the lock again.

27. In the cleanse functions tree, select the **My User Library** node.
28. Right-click on this node and select **Add Cleanse List**.

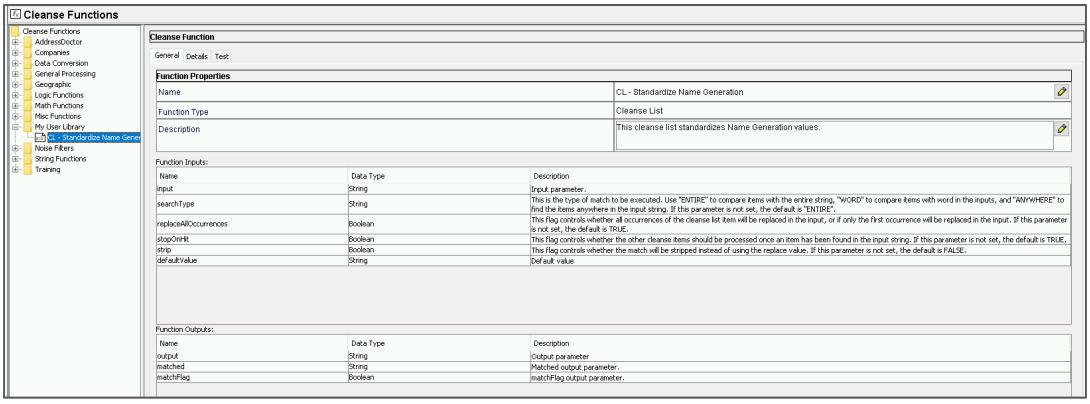


29. Enter the Name as **CL – Standardize Name Generation**.

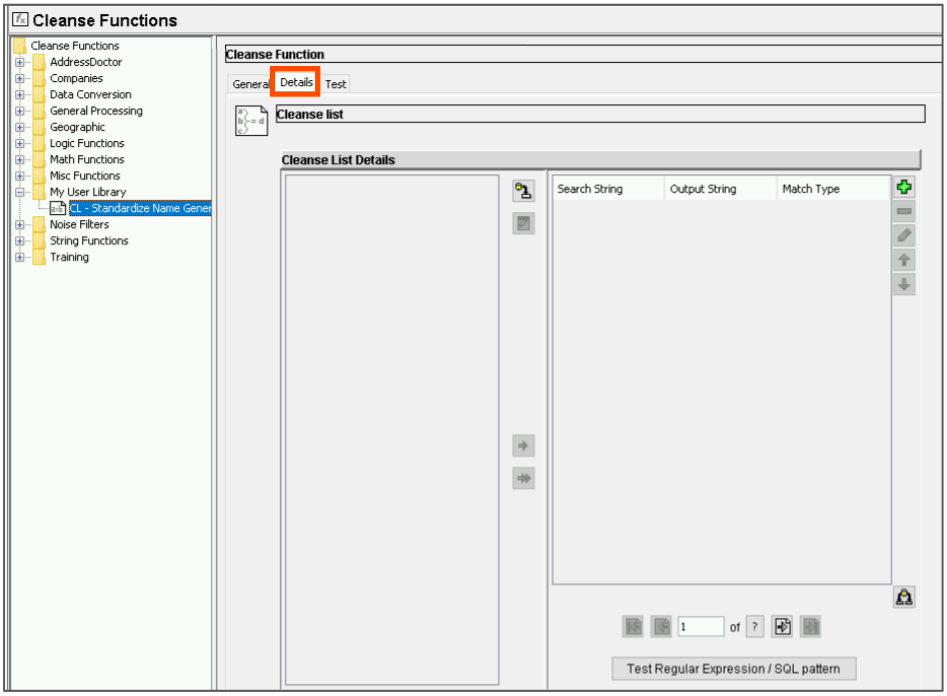
30. Provide a **Description**.



31. Click **OK**.
32. In the cleanse functions tree, click the **CL-Standardize Name Generation** node. The properties page on the right shows **General**, **Details**, and **Test** tabs.

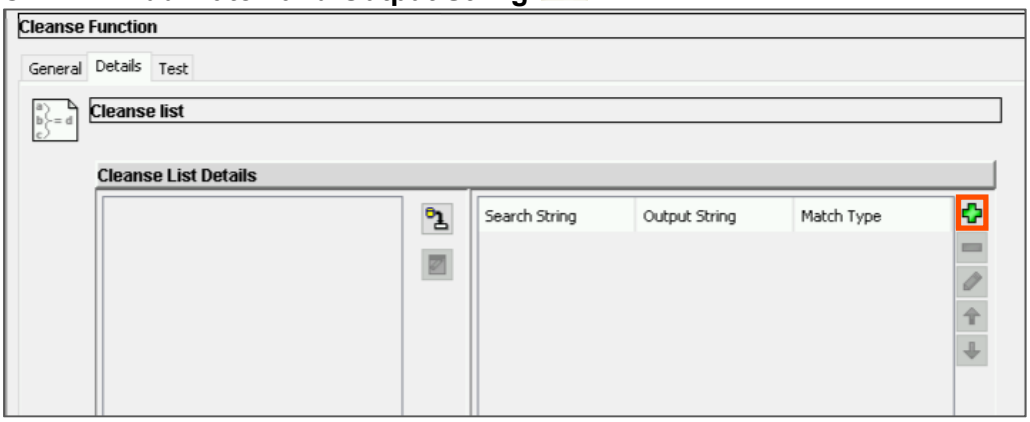


33. Select the **Details** tab.

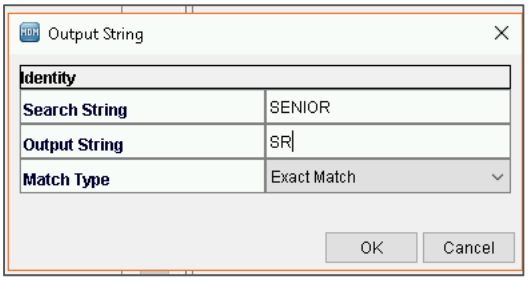


Note: This is where you define the cleanse list items. You can import them from the database tables, or you can enter them manually. As the cleanse list for this lab is short, you will enter it manually.

34. Click the **Add Match and Output String** button.



35. Enter the details as shown below:

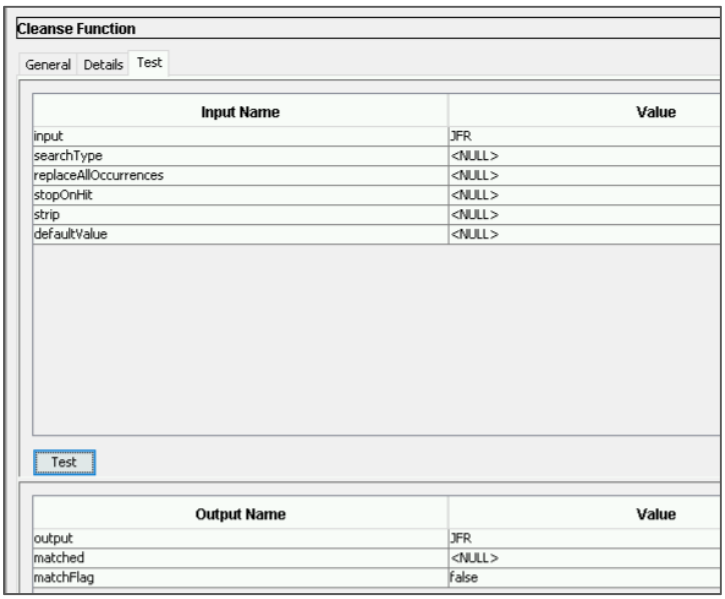


36. Click **OK**.
37. Similarly, add the following search and output string pairs:

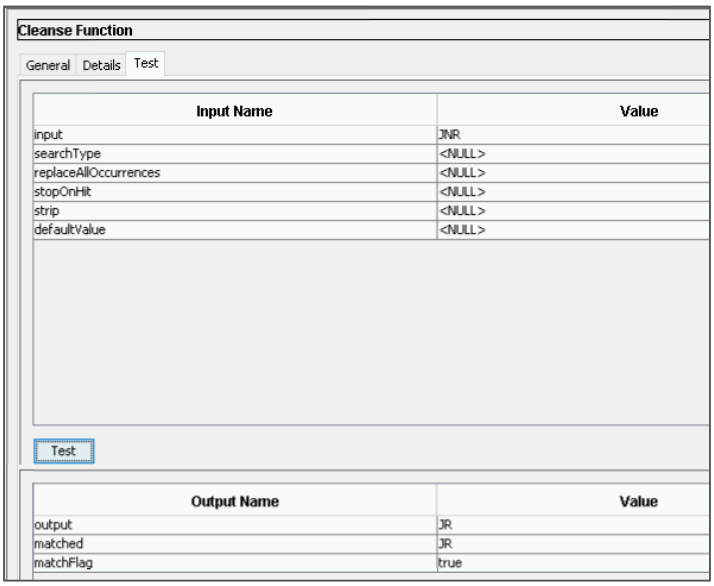
Search String	Output String
SNR	SR
SR	SR
JUNIOR	JR
JNR	JR
JR	JR

38. Select the **Test** tab.
39. For the input field, test the cleanse list with all the valid values. That is, provide the value for the input field as **SNR**, **SR**, **JUNIOR**, **JNR**, **JR**.
40. **Test** the cleanse list with a non-matching value. For example, provide the input value as **JFR**.
41. Notice the difference between the **output** and **matched** parameters, when you use a non-matching input value (not in the cleanse list).

Non-Matching example:



Matching Example:

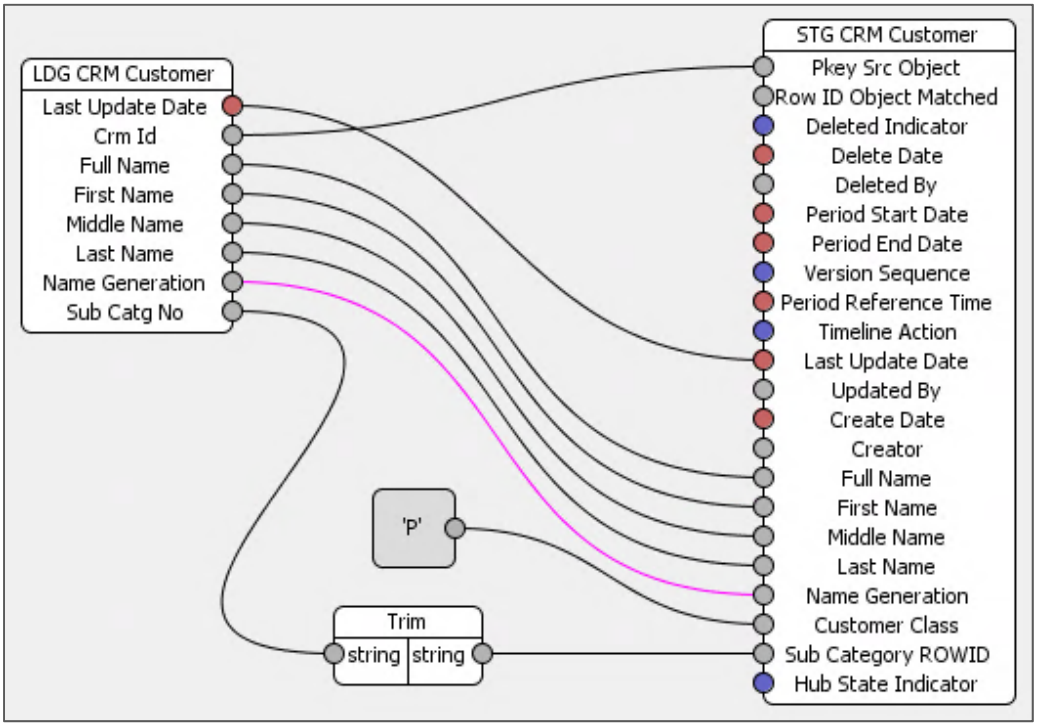


Use the New Cleanse List in a Mapping

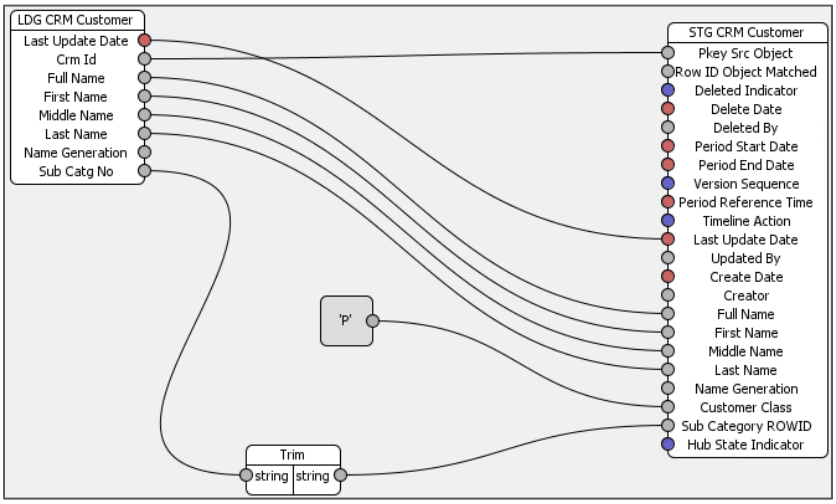
You will use the cleanse list that you just created in an existing mapping.

42. Navigate back to the **Mappings** tool.
43. Open the mapping **Stage Customer data from the CRM system** and click the **Diagram** tab.

44. For the Name Generation column, click on the mapping link.

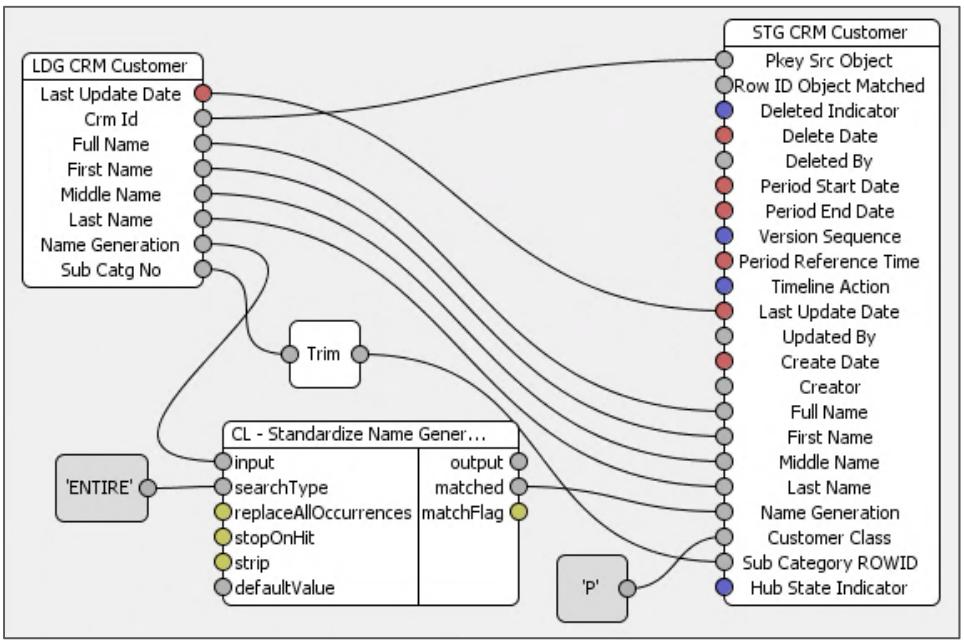


45. To delete the mapping link, right-click and select **Delete Link**.



46. Click the **Add Function** button and select **My User Library** > **CL - Standardize Name Generation**.
47. Set the component to **Expanded** Mode.
48. Add a string constant with the value **ENTIRE** and map it to the **searchType** node in the cleanse list.

49. From the landing table, map **Name Generation** to the **input** node in the cleanse list.

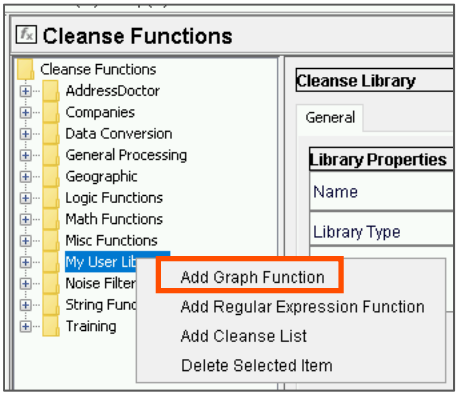


50. From the cleanse list, map the **matched** output node to the **Name Generation** node in the staging table.
51. **Save** the mapping.
52. **Test** it with values such as SNR, JNR, JUNIOR, and so on.

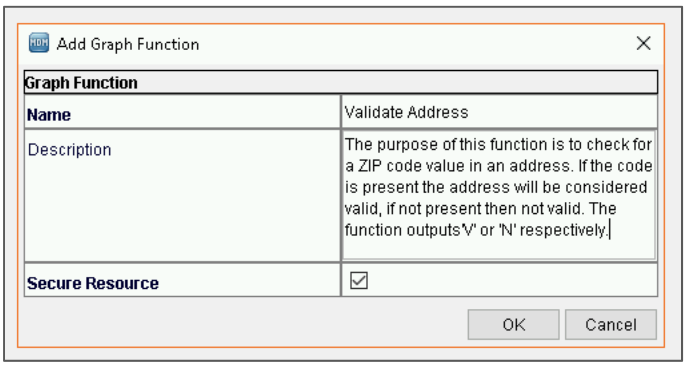
Create a Graph Function

You will create a reusable graph function. It will check whether an address has a ZIP code. If the ZIP code is present, the address is considered valid, if not, then invalid. It will return ‘Y’ for valid and ‘N’ for invalid.

53. Ensure that you acquire the **Write Lock**.
54. From the Workbenches pane, select the **Cleanse Function** tool.
55. In the cleanse functions tree, select the **My User Library** node.
56. Right-click on this node and select **Add Graph Function**.



57. Enter a **Name** and **Description** as shown below:

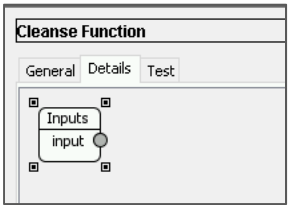


58. Click **OK**.

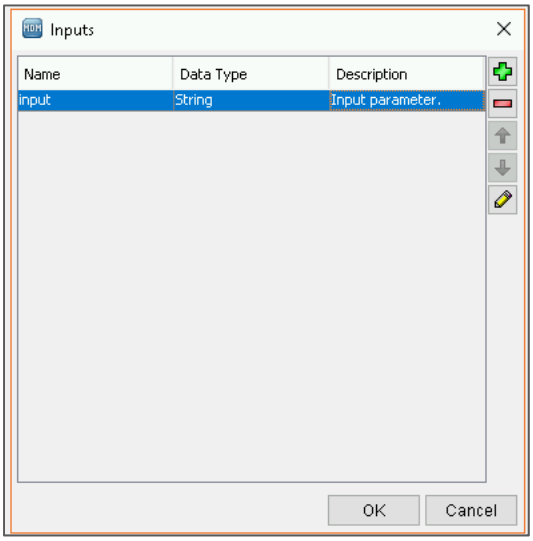
    The properties page on the right appears and shows **General**, **Details**, and **Test** tabs.

59. Select the **Details** tab.

60. To edit the input parameters, double click the **Inputs** component.

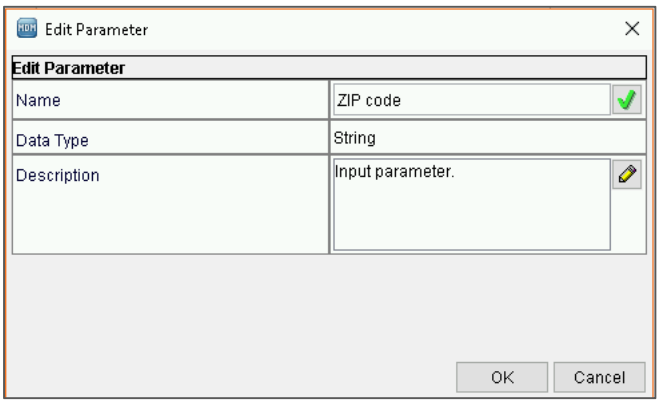


61. Select the **input** parameter.

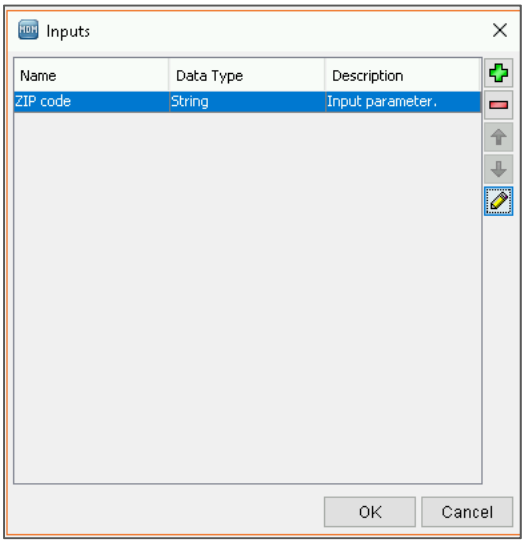


62. Click the **Edit** button  .

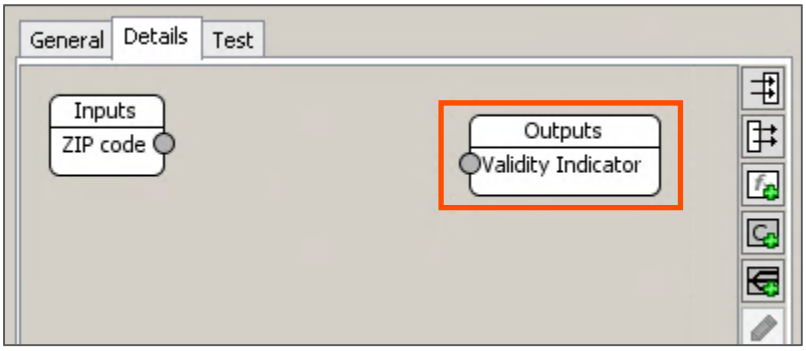
63. Rename **input** to **ZIP code** and click **OK**.



64. Click **OK**.

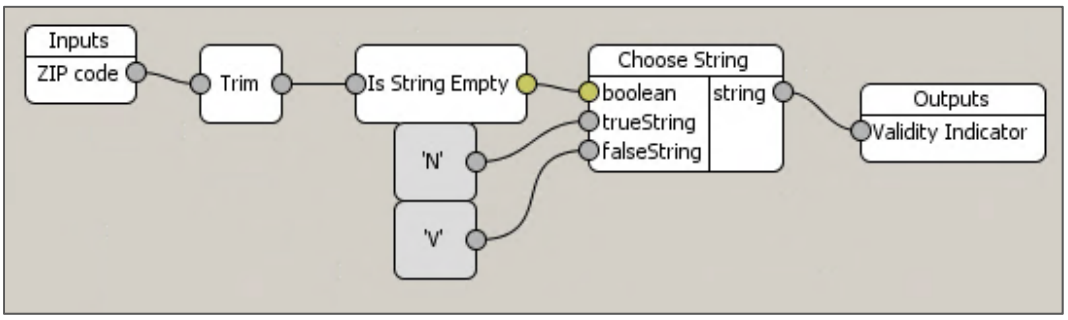


65. Similarly, edit the **Outputs** component and rename **output** to **Validity Indicator**.



66. Add a **Trim** function to the graph.
67. Add another function – **Is String Empty** from the Logic Functions group.
    Note: Use the **Add Function** icon to add the function.
68. From the same function group, add a **Choose String** function to the graph.
69. Add two string constants with the value **V** and **N**.

70. Map the components as shown below:

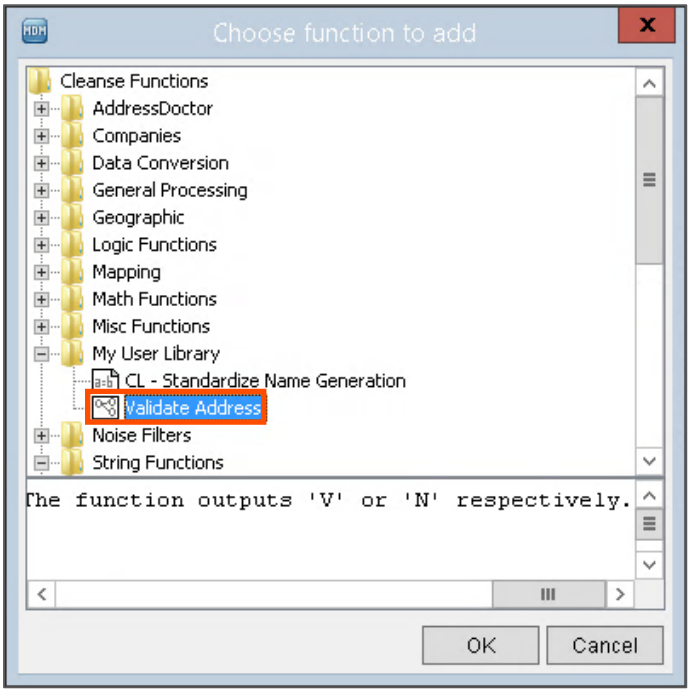


This graph function checks whether the ZIP Code is NULL, empty, or if a value exists. If empty or NULL, Is String Empty returns a true value, otherwise, it returns false. The Choose String function outputs N if the string is empty and V if it is not. Thus, if the ZIP Code exists, the Validity Indicator will be V, and N if it does not exist.

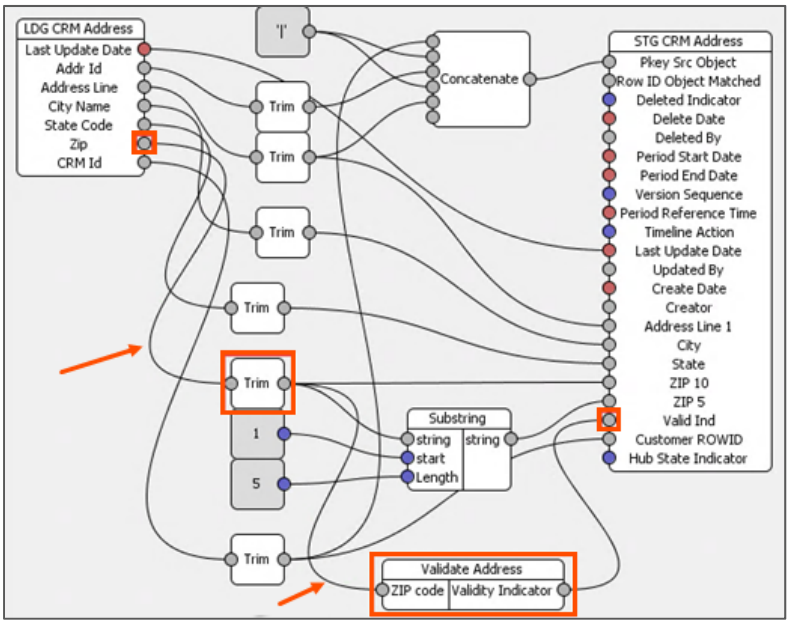
71. **Save** the cleanse graph.
72. Select the **Test** tab.
73. To test the graph, in the Zip Code field, enter **1234** and verify that the result is **V**.
74. Now, clear the Zip Code value and verify that the result is **N**.
75. To test the graph further, in the Zip Code field, enter 5 space characters and verify that the result is **N**.

Use the Graph Function in a Mapping

76. Navigate back to the **Mappings** tool.
77. Open the mapping **Stage Address data from CRM system** and click the **Diagram** tab.
78. Click the **Add Function** button and select **My User Library** > **Validate Address**.



79. Map the trimmed landing table column **Zip** to the input parameter for the **Validate Address** function.
80. From the Validate Zip Code, map the **output** node to the **Valid Ind** node in the staging table.



81. **Save** the mapping and perform a quick test.

    Note: Perform the same test as you did while creating the graph function, giving values for the Zip Code, or by providing blank spaces for it.

Use a Graph Function to Validate Fields in Existing Mappings

In this section, you will validate the address records using the **Validate Address** graph function for the Shipping and Billing Addresses of the Sales system.

82. Open the diagram view of the mapping **Stage Billing Address data from Sales system**.
83. Add the **Validate Address** graph function.
84. Map the landing table's trimmed **Bill Addr Zip** node column to the **ZIP code** of the Validate address function.
85. Then, map **Validity Indicator** to the **Valid Ind** node in the staging table.
86. **Save** and **test** the mapping.
87. Open the diagram view of the mapping **Stage Shipping Address data from Sales system**.
88. Add the **Validate Address** graph function.
89. Map the landing table's trimmed **Ship Addr Zip** column to the **ZIP code** of the Validate address function.
90. Then, map **Validity Indicator** to the **Valid Ind** node in the staging table.

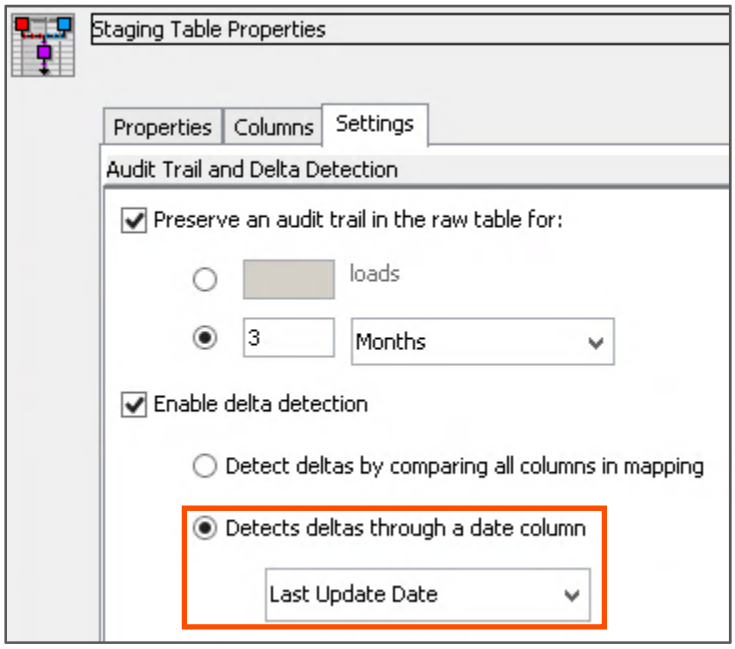
91. **Save** and **test** the mapping.

Define Delta Detection Options and Raw Data Retention Periods

Delta detection is the process of identifying the new and changed records in the data from a source system by comparing the current data set provided by the source with the previous data set provided by the source.

In this section, you will set options in a staging table to preserve data in the raw table for three (3) months. You will also learn how to enable delta detection when you compare all the columns in a mapping for Sales data versus using the Last Update Date column only to detect changes for CRM data.

92. In the Workbenches pane, navigate to the **Schema** tool.
93. Ensure that you acquire the **Write Lock**.
94. In the **Category** base object, select the **STG CRM Category** staging table.
95. To switch on the audit trail, in the **Settings** tab, select the **Preserve an audit trail in the raw table for** checkbox.
96. Specify the raw retention periods as 3 months.
97. To switch on delta detection, select the **Enable delta detection** checkbox.
98. Select the **Detects deltas through a date column** option and from the drop-down, select **Last Update Date**.



99. **Save** your work.

This concludes the lab.

Module 4: Configure the Stage Process

Lab 4-4: Run Staging Jobs

Overview:

In the previous labs, you configured the Stage process after you registered a Process server, created mappings, and defined audit trail and delta detection. This completes the Staging configuration.

In this lab, you will load records into the landing tables and run the Staging jobs to transform and move data from the landing tables to the staging tables.

Objectives:

- Load landing tables using an SQL script
- Run Staging batch jobs

Duration:

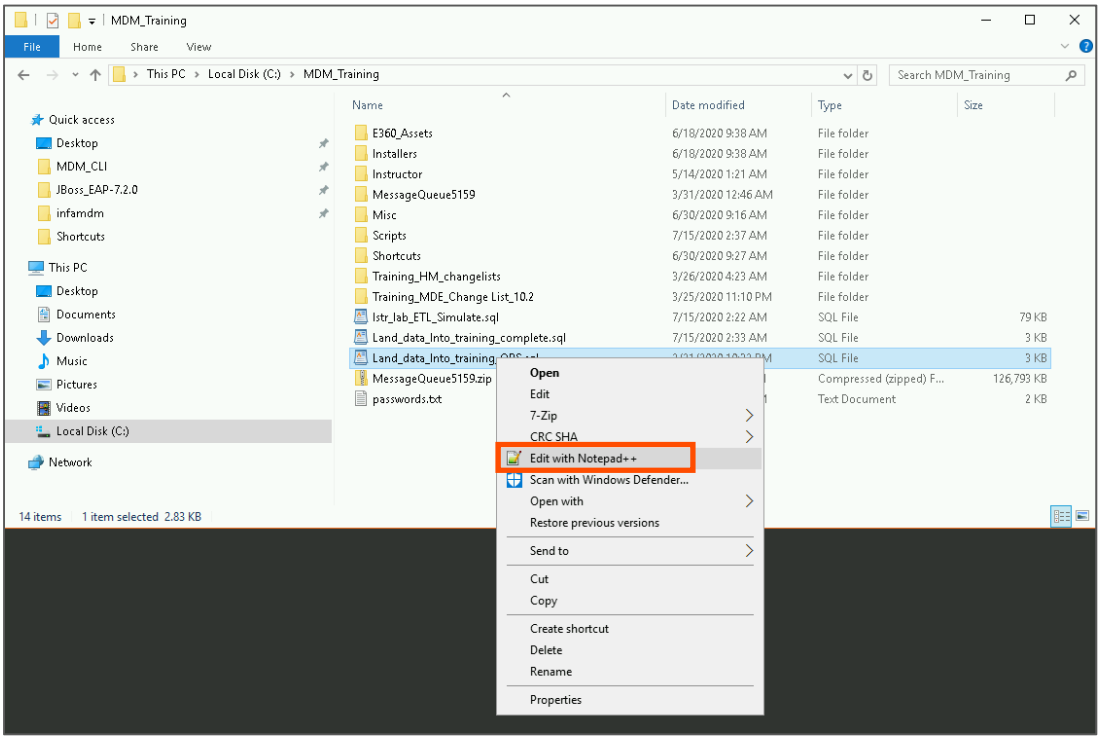
20 minutes

Tasks

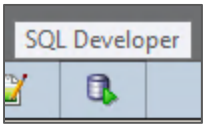
Simulate the Landing Process

The training image has a SQL script called **Land_data_Into_training_ORS.sql** that simply deletes all records in the target landing tables and then inserts records from the ZZ_* “source” tables into the landing tables. The ZZ_* source tables are available in the lab environment to simulate the Sales and CRM source systems. The SQL script is also created to support this course and is not part of the MDM product. Each installation of MDM will have its own processes for loading records into the landing tables.

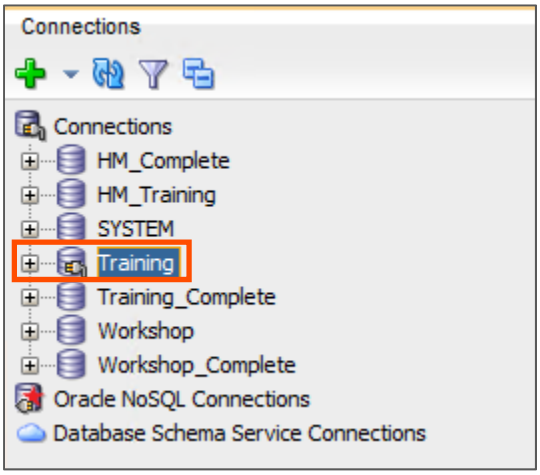
1. Navigate to the file below and open it using a Notepad++ command:
   C:\MDM_Training\Land_data_Into_training_ORS.sql



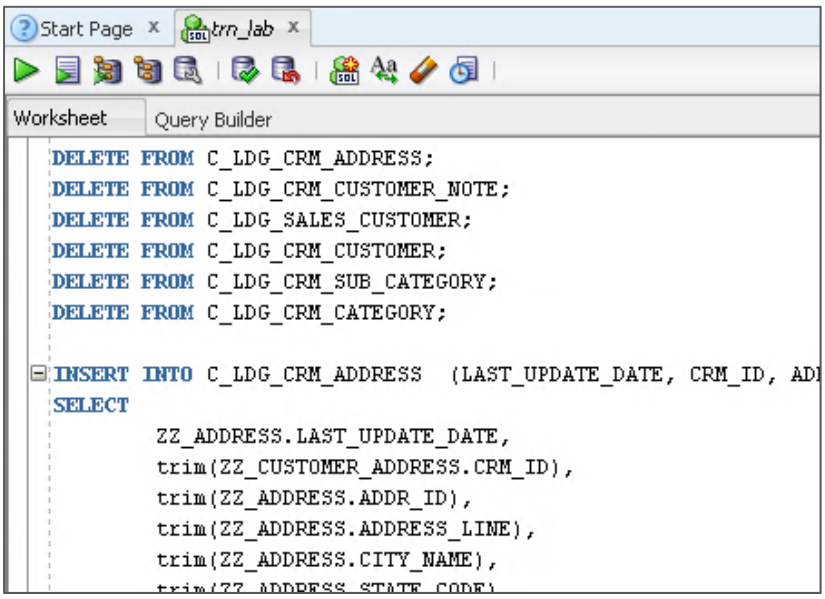
2. Copy all the contents of the file.
3. Click **Start** > **SQL Developer**.



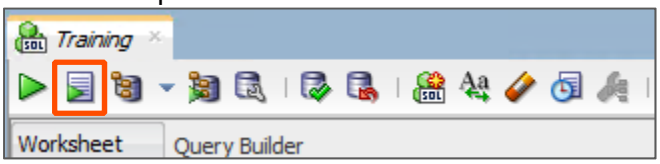
4. In the Connections pane, select **Training**, right-click and click **Connect**.
5. Enter the password as **infa**, if prompted.



6. Paste the text from the **TRN_Landing_process.sql** file into the worksheet.



7. Run the script.



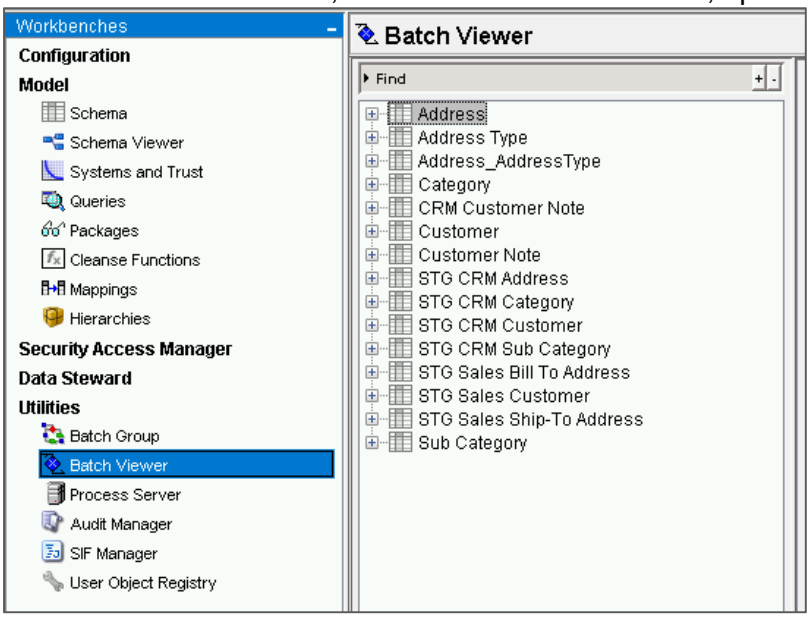
The result should be as follows:

0 rows deleted.
0 rows deleted.
0 rows deleted.
0 rows deleted.
0 rows deleted.
0 rows deleted.
45 rows inserted.
11 rows inserted.
84 rows inserted.
46 rows inserted.
12 rows inserted.
3 rows inserted.
committed.

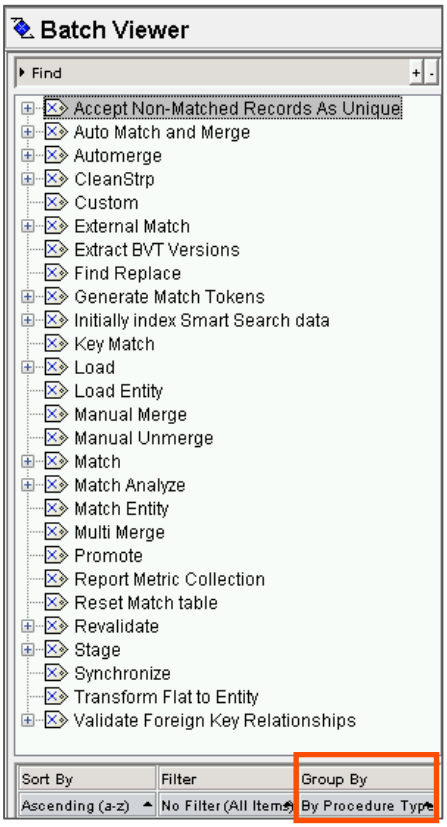
Note: If you use the landing table names and column names as recommended in the labs, the above script should run without any problems. However, if you use different names for your landing tables, the procedure gives a run time error. Additionally, if you use different physical names for the other tables (staging or base object tables) than in the lab instructions, you should also follow the procedure below as well.

Run a Batch Job

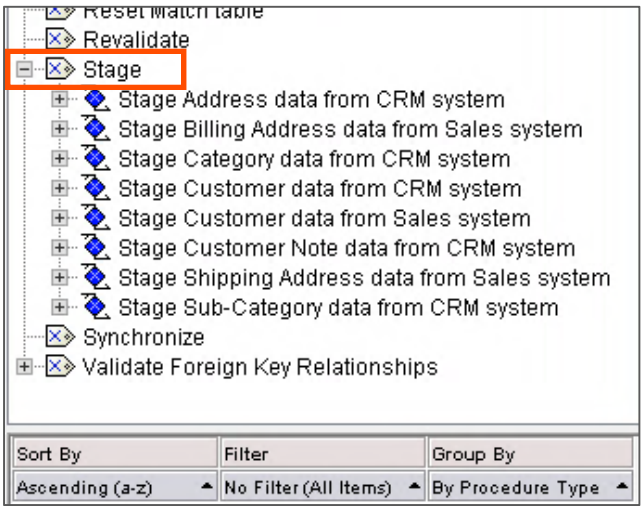
8. In the MDM Hub Console, from the Utilities workbench, open the **Batch Viewer** tool.



9. At the bottom of the Batch Viewer list, set the Group By attribute to **Procedure Type**, if not already set.

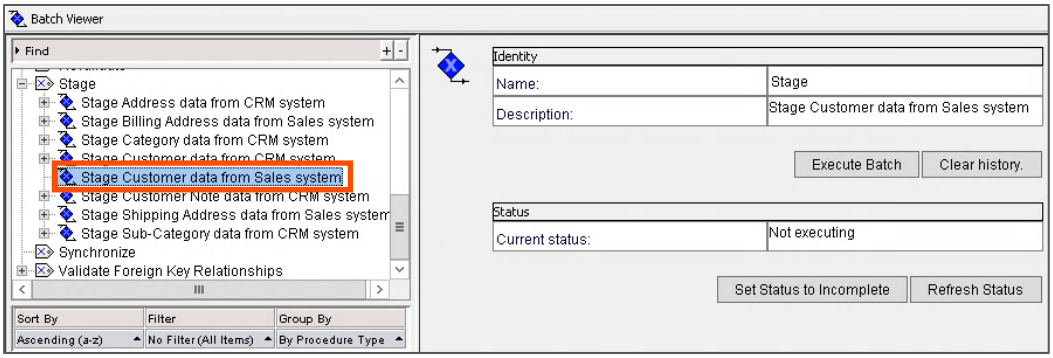


10. Expand the **Stage** node.

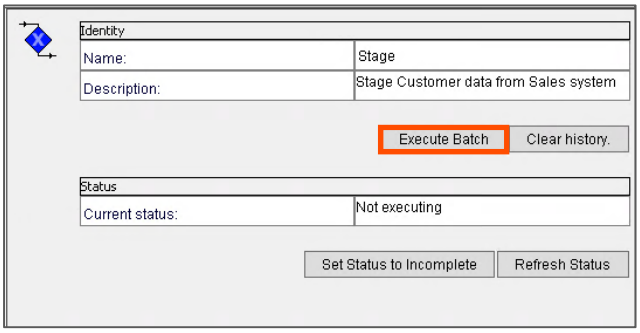


Note: You can see a stage job for every mapping in your schema. After you create the mappings, MDM automatically creates a Stage job for each mapping.

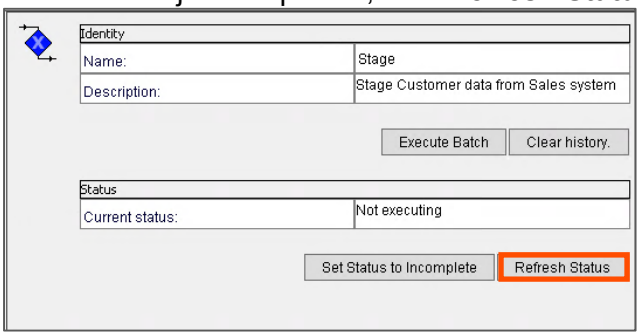
11. Select the **Stage Customer data from Sales system** job.



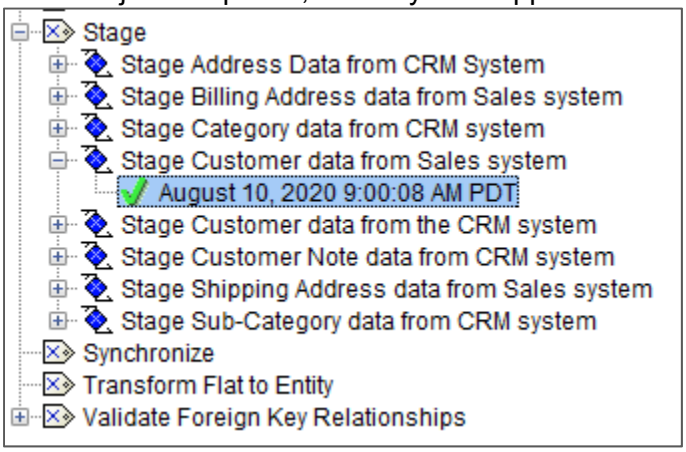
12. To run the stage job that executes the logic defined in your mapping, click the **Execute Batch** button.



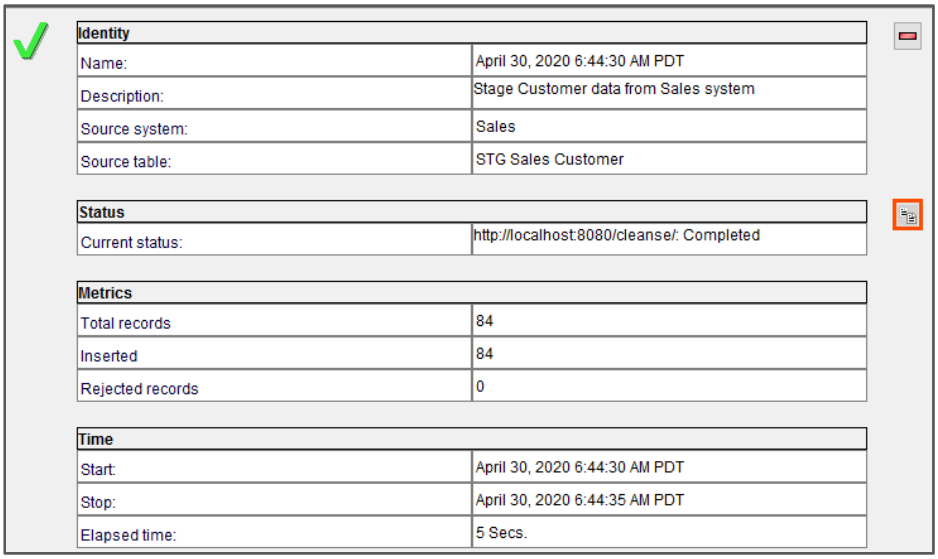
13. To check the job completion, click **Refresh Status**.



After the job completes, an entry for it appears in the Batch Viewer tree view.

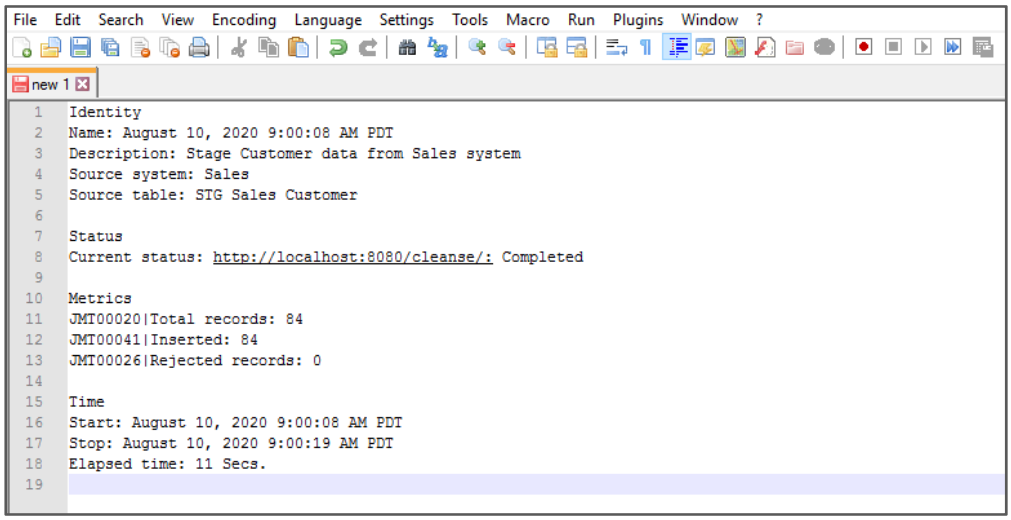


14. To check the job metrics, in the tree view, click the job entry.
15. In the right-hand pane, click the **Copies Properties to Clipboard** button.



Note: This allows you to paste the contents of the clipboard into a text application such as Notepad++, and easily read and save the status of any job. If there are any errors or warnings, the status messages can be quite long, and it is easier to read all the details in a text application than it is to scroll through the narrow status field.

16. Paste the content into a new page on **Notepad++**.



Run Stage Batch Jobs on Remaining Mappings

17. Run the jobs in the following order and verify the total number of records inserted as mentioned below:

Job	Number of records inserted
Stage Address data from CRM system	45
Stage Billing Address data from Sales system	82
Stage Category data from CRM system	3
Stage Customer data from CRM system	46
Stage Customer Note data from CRM system	11
Stage Shipping Address data from Sales system	26
Stage Sub-Category data from CRM system	12

This concludes the lab.

ADDITIONAL INFORMATION

To review the data that is inserted into the staging tables, use SQL Developer.

Note: When a batch job results in rejected records, those records are written to the reject table, and the job execution status pane displays a View Rejects button. If you click the View Rejects button, it opens a pop-up window with error messages that explain the reason behind the rejected records.

Recall that the Reject table associates with the staging table (named stagingTableName_REJ; For example: for the **C_STG_CRM_CUSTOMER_NOTE** table, the reject table is named as **C_STG_CRM_CUSTOMER_NOTE_REJ**.

About the Reject Table

The Reject table contains records that Informatica MDM Hub rejects for a specific reason. Records in these tables are not loaded into the staging tables. Data is rejected automatically during Stage jobs for the following reasons:

Future date or NULL date in the LAST_UPDATE_DATE column

LAST_UPDATE_DATE less than 1900

NULL value mapped to the PKEY_SRC_OBJECT of the staging table

Duplicates found in PKEY_SRC_OBJECT

If multiple records with the same PKEY_SRC_OBJECT are found, the surviving record is the one with the most recent LAST_UPDATE_DATE. The other records are sent to the REJECT table.

Invalid value in the HUB_STATE_IND field (for state-enabled base objects only)

Duplicate value found in a unique column

Module 5: Configure the Load Process

Lab 5-1: Set Trust, Validation Rules, Cell Update, and Null Update

Overview:

A validation rule downgrades trust for a cell value when the cell value matches a given condition. Each validation rule specifies a condition that determines whether the cell value is valid, and an action to take if the condition is satisfied.

In the previous labs, you configured the Stage process and ran various stage jobs for each of the mappings. In this lab, you will configure the Load process that includes setting trust and creating validation rules.

Objectives:

- Switch on Trust and Validation flags for columns in the Customer base object
- Define trust settings for the trusted columns in the Customer base object
- Define a validation rule for the Customer base object
- Use the Null Update and Cell Update properties of a staging table

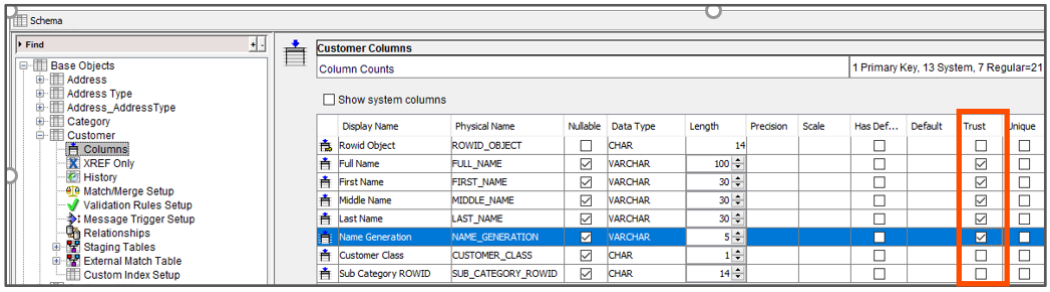
Duration:

30 minutes

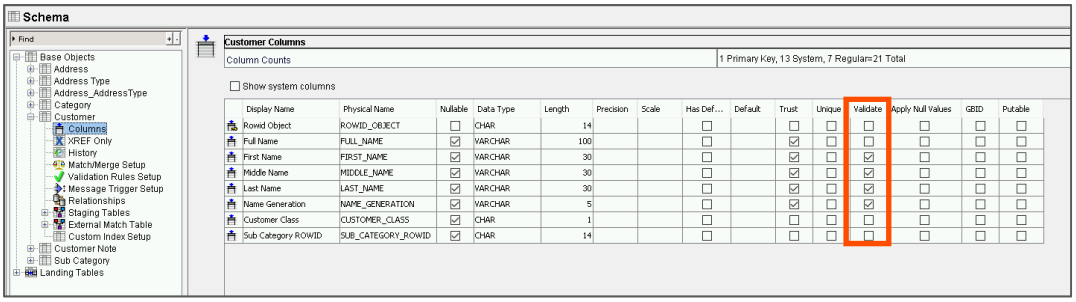
Tasks:

Switch on Trust and Validation Flags for Columns in a Base Object

1. Open the **Schema** tool and ensure that you acquire the **Write Lock** on your schema.
2. For the **Customer** base object, select the **Column** node.
3. In the Column properties, select the **Trust** option for the columns mentioned below:
   FULL_NAME
   FIRST_NAME
   MIDDLE_NAME
   LAST_NAME
   NAME_GENERATION



4. Similarly, select the **Validate** option for the columns mentioned below:
   FIRST_NAME
   MIDDLE_NAME
   LAST_NAME
   NAME_GENERATION



Note: Ensure that the Validate option is not selected for the FULL_NAME column.

5. **Save** the object.

Caution

If you receive a message about running the synchronization procedure, it is because you changed the trust settings after the data load into the base object.

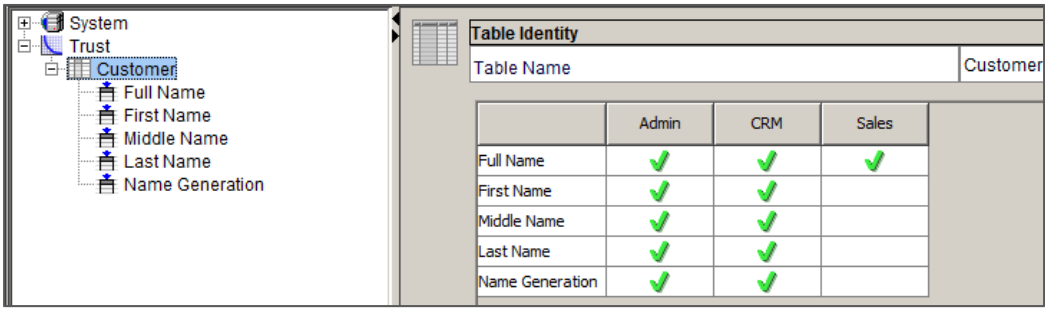
So, after you receive the message, do not delay running it. Go to **Utilities** > **Batch Viewer**. Click the **Execute Batch** button on the right. Expand the job node on the left and check for its successful run before you continue with anything else in Informatica MDM Multidomain Edition.

Define Trust Settings for the Trusted Columns in a Base Object

6. From the Model workbench, select the **Systems and Trust** tool.

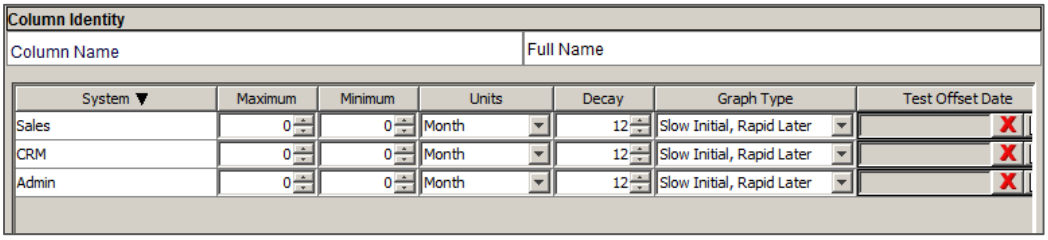
   Note: The source systems and the tables that have trust-enabled columns are listed as tree views. The **System** tree lists all the defined sources, and the **Trust** tree lists the trust-enabled tables.

7. Expand the **Trust** node and expand the **Customer** base object.

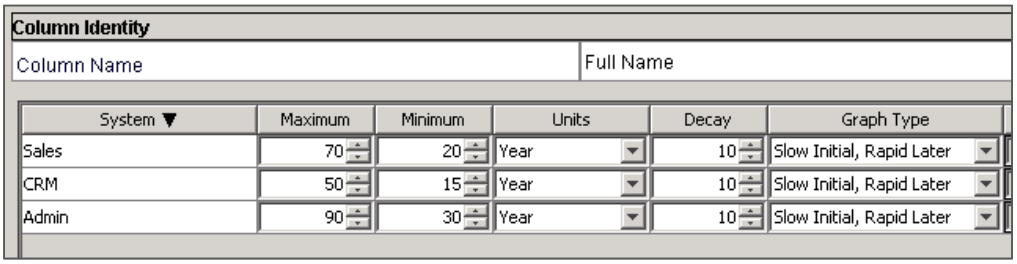


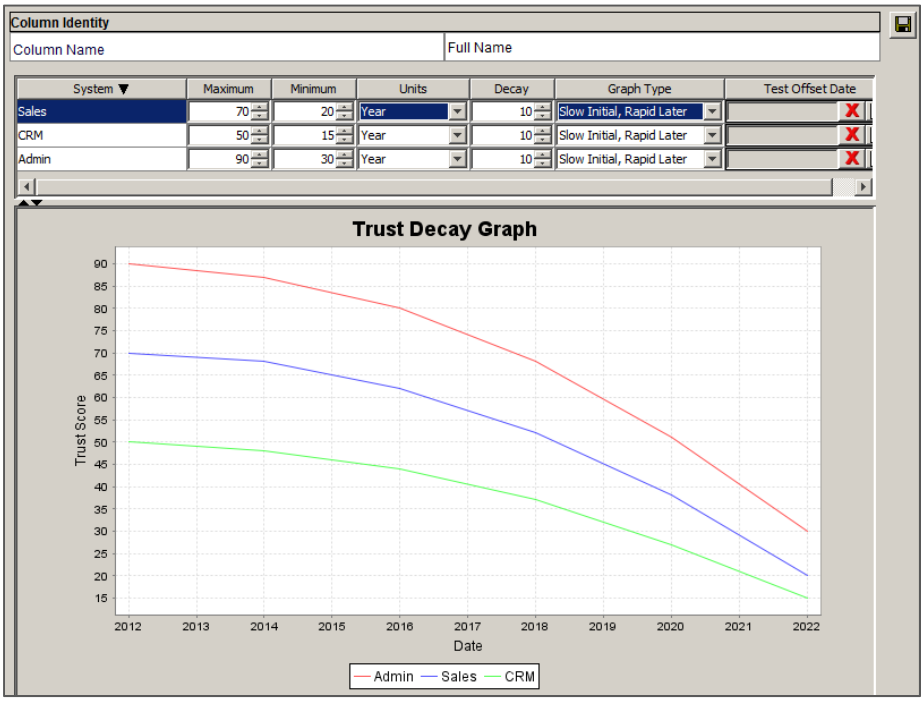
On the right, the Table Identity pane shows a list of the **Customer** columns that are enabled per source. The top pane on the right allows you to edit the values for the column for all sources that provide the column.

8. Click **Full Name**.



9. For the three systems, enter the trust values as mentioned below:



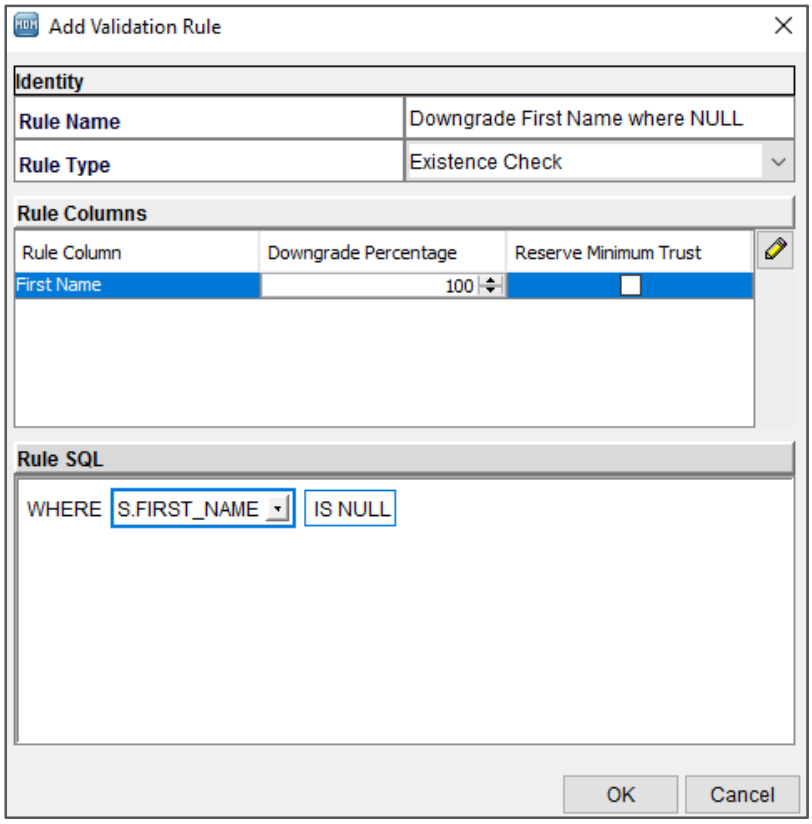


10. **Save** the trust settings.

Define Validation Rules for a Base Object

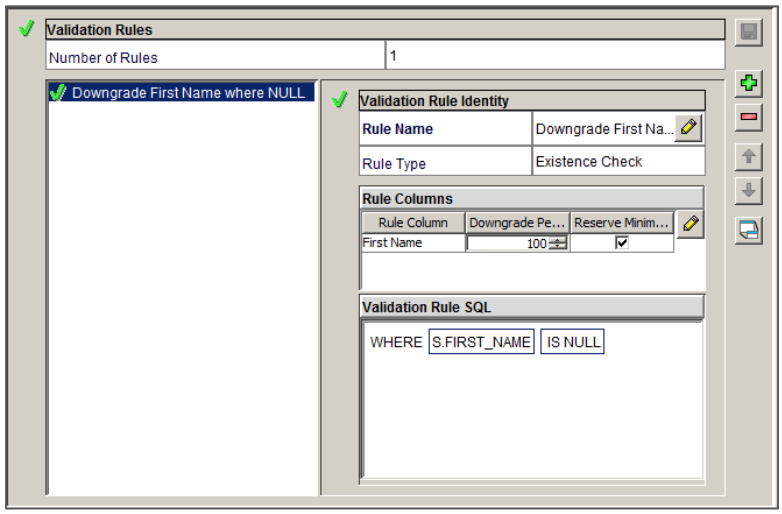
11. Return to the **Schema** tool.
12. In the Schema tree, expand the **Customer** base object node.
13. Click **Validation Rules Setup**.
14. Click the **Add Validation Rule** button.
15. Enter the rule name as **Downgrade First Name where NULL**.

16. Click the **Rule Type** drop-down.
    This provides a template for the rule in the Rule SQL area.
17. Select **Existence Check**.
18. If the Rule Columns list does not show the First Name column, click the **Edit Rule Column** button and select **FIRST_NAME** as the column that downgrades by the rule.
19. Change the Downgrade Percentage to **100** percent.
20. Change the **Rule SQL** to **WHERE S.FIRST_NAME IS NULL**.



Important: The validation rule specifies the bad condition, that is, the condition under which the trust must be downgraded.

21. Click **OK**. The validation rule will be saved.

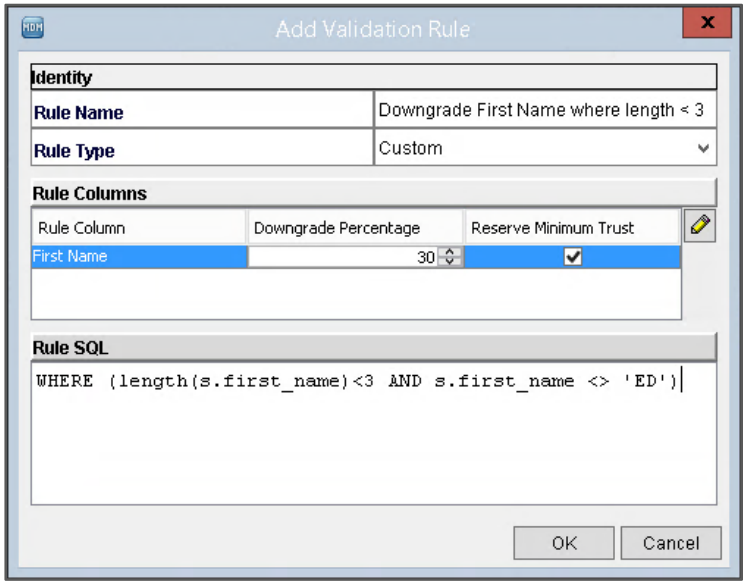


Note: The Validation Rules property page displays the rule you just added. You can edit the rule in the properties page.

Add Another Validation Rule

In this section, you will define a second validation rule that will downgrade first names that are shorter than 3 characters and not a valid 2-character first name. This validation rule intends to illustrate the syntax, and it is not a typical real-world validation rule.

22. Click the **Add Validation Rule** button.
23. In the Rule Name field, enter **Downgrade First Name, where length < 3**.
24. From the Rule Type drop-down, select **Custom**.
25. For the column that will be downgraded by the rule, select **First Name**.
26. Set the downgrade percentage to **30** percent and select the **Reserve Minimum Trust** checkbox.
27. In the Rule SQL field, enter **WHERE (length(s.first_name) < 3 AND s.first_name <> 'ED')**
28. Click **OK** to save the validation rule.

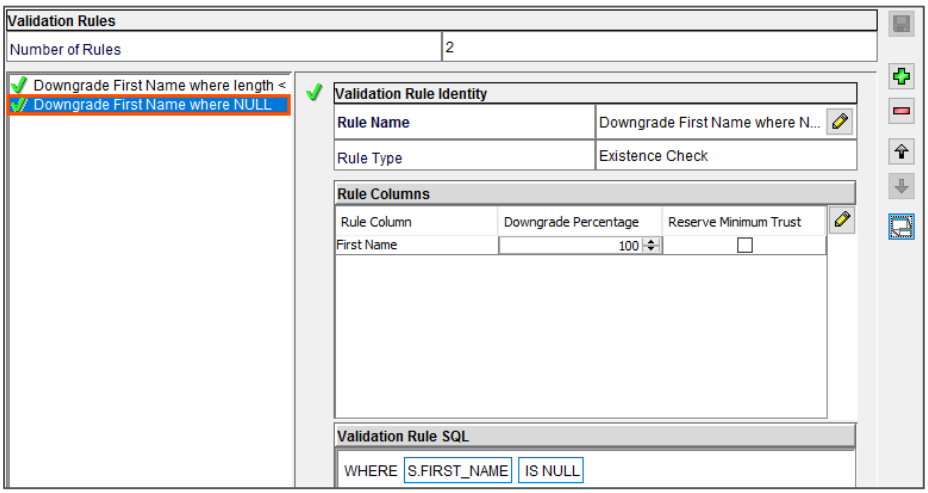


Sequence Validation Rules

The Validation rules are not cumulative for the downgrade percentage and their order is not important. The largest downgrade of the rules that tests true will be applied. However, you can reorder the rules, for example, to group similar rules together for display purposes.

29. Click the **Downgrade First Name where NULL** rule.

30. Click the **Move Down** arrow to move the **Downgrade First Name where NULL** rule down in the list.



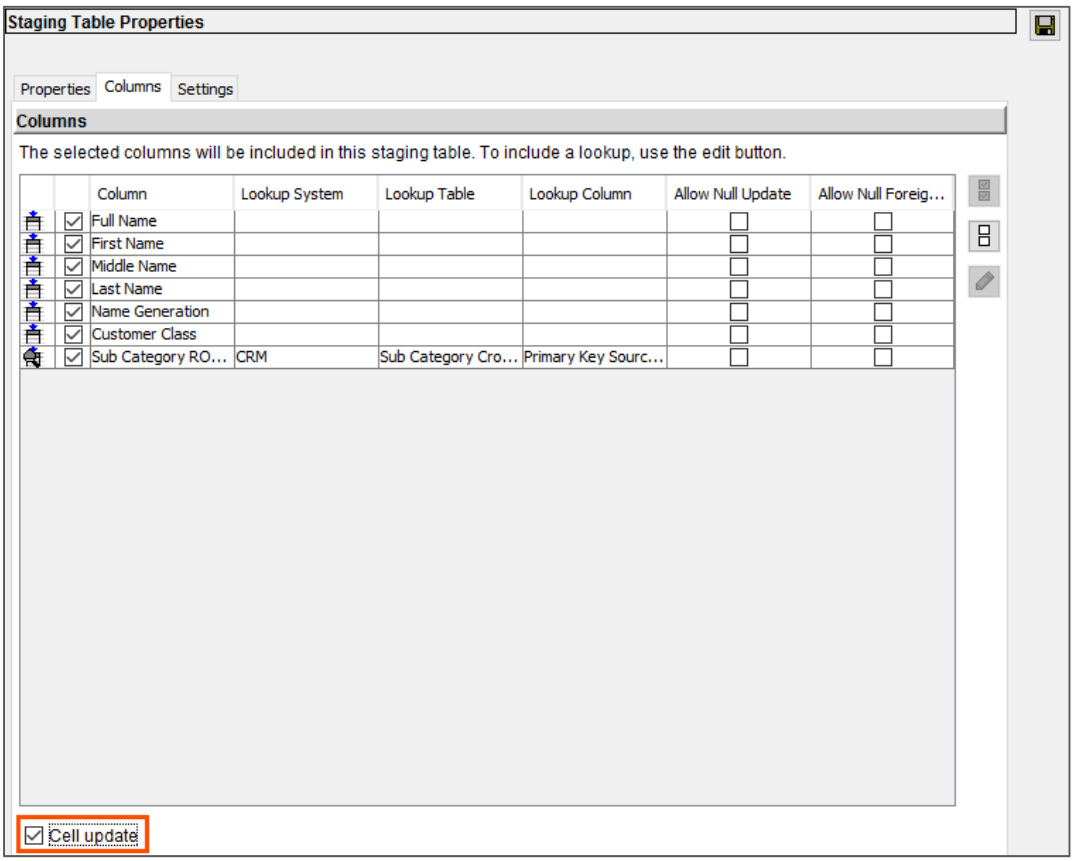
Use Null Update and Cell Update Options for a Staging Table

Cell Update

By default, during the stage process, for each inbound record of data, the MDM Hub replaces the cell value in the target base object whenever an incoming record has a higher trust level, even if the value it replaces is identical. If still after that, the value does not change, the MDM Hub updates the last update date for the cell to the date that associates with the incoming record and assigns to the cell the same trust level as a new value.

31. In the Schema tree, for the **Customer** base object, select the **Staging Tables** node.
    Schema > **Customer** > **Staging Tables**
32. Select the **STG CRM Customer** staging table.

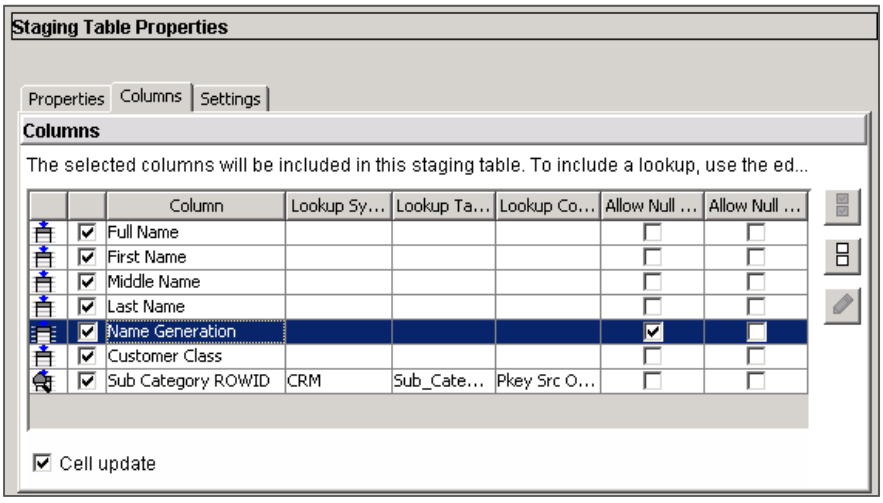
33. In the **Columns** tab, at the bottom of the page, select the **Cell Update** option.



34. Click **Save**, but do not close the properties page.

Null Update: When you select the Null Update for a column, it determines whether null updates are allowed when a Load job specifies a null value for a cell that already contains a non-null value.

35. In the Staging Table Properties page, navigate to the **Columns** tab.
36. For the **Name Generation** field, select the **Allow Null Update** option.



37. **Save** the properties.

Recap:

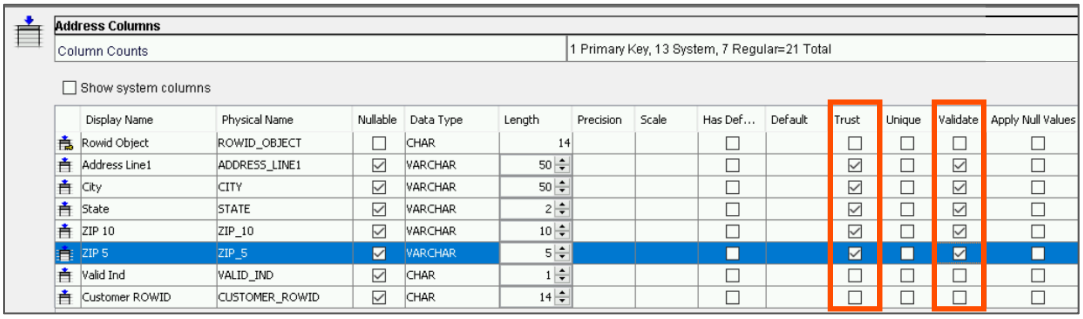
So far, you:

- Switched on Trust and Validation flags for columns in the Customer base object
- Defined trust settings for the trusted columns in the Customer base object
- Defined a validation rule for the Customer base object
- Used the Null Update and Cell Update properties of a staging table

Configure Trust and Validation for columns in the Address base object

Define the trust and validation rules for the **Address** base object as described below. Address elements should be treated as being part of the same logical trust group.

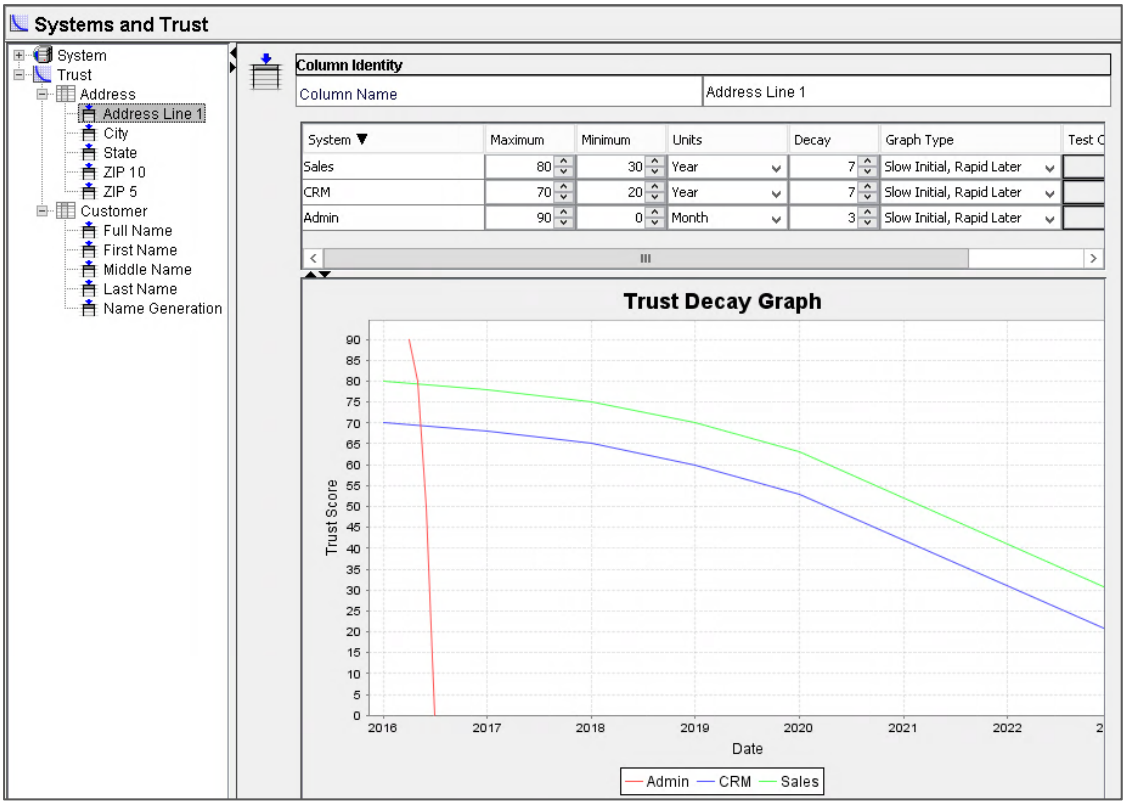
38. Open the **Schema** tool and ensure that you acquire the **Write Lock** on your schema.
39. For the **Address** base object, select the **Column** node.
40. Set **Trust** and **Validation** for **all address columns except** for primary and foreign keys, and **Valid_Ind** because it is a value computed by the mapping algorithm.



41. Click **Save**.
42. Click the **Systems and Trust** tool.
43. Open the **Address** Base Object and select the **Address Line 1** column.

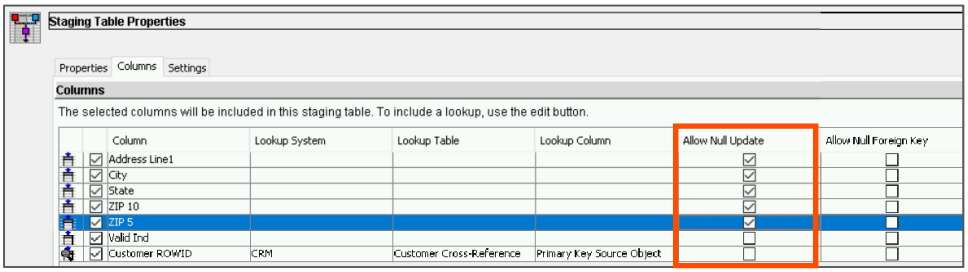
44. Configure the values as mentioned in the table below:

System	Maximum	Minimum	Units	Decay	Graph Type
Sales	80	30	Year	7	Slow Initial, Rapid Later (SIRL)
CRM	70	20	Year	7	SIRL
Admin	90	0	Month	3	SIRL



45. **Save** the changes.
Note: Addresses from the **Sales** system are more reliable than addresses from the CRM system. Also, addresses from the default **Admin** system are the most reliable. However, they should not survive for longer than 3 months, because within that period, the **Sales** system's support team must follow up and verify all addresses that are corrected in the Customer Master. On an average, addresses for our fictitious company's customers change after 7 years. Hence, Trust system values (Maximum, Minimum, Units, Decay, and Graph Type) must reflect similar values for all the columns in the three systems.
46. Navigate back to **Schema** > **Base Objects** > **Address** > **Staging Tables** > **STG CRM Address**.
47. Click the **Columns** tab.

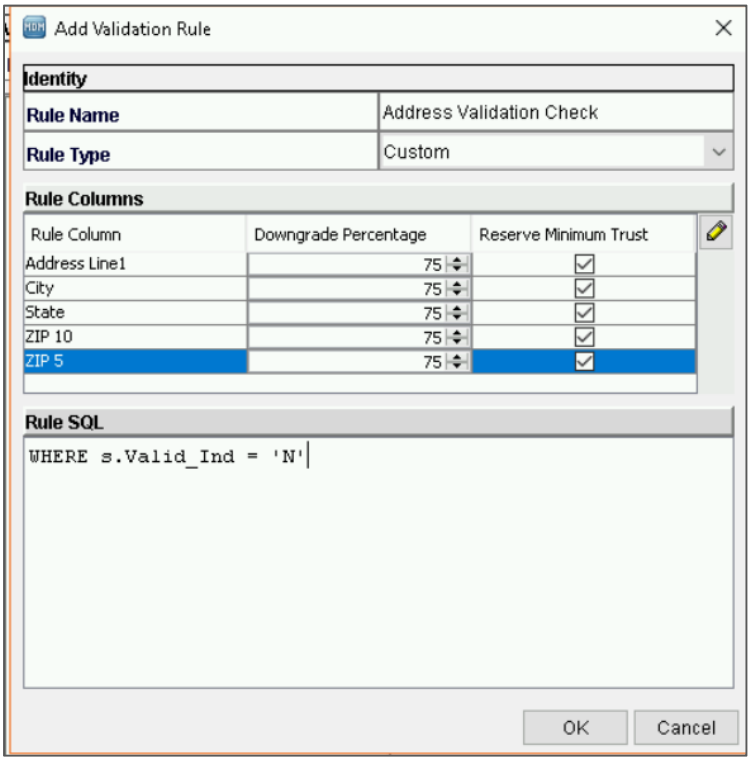
48. For the Address Line 1, City, State, ZIP 10, and Zip 5 columns, select the **Allow Null Update** option.



49. Click **Save**.

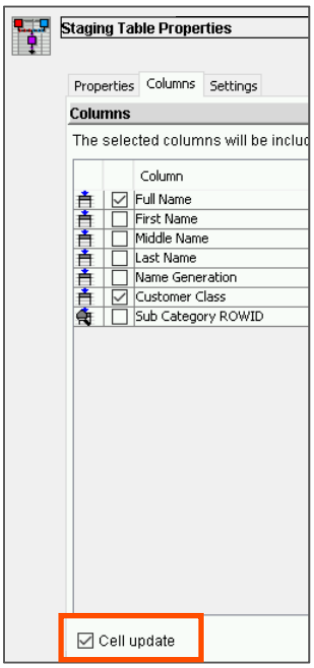
    Note: This allows incomplete address to load with the idea that a partial address is better than no address and that data from multiple sources may form a complete address.

50. Click **Validation Rules Setup** under the Address base object
51. Click the **+** icon to add a new validation rule.
52. For this Base Object, add the Validation Rule as shown, and click **OK**.



Note: Initially, if you do not see all the Rule columns, click the **Edit** icon and add the columns.

53. Now, for the **Customer** base object, open the staging table **STG Sales Customer** and select the **Cell Update** option.



54. Save the changes.

This concludes the lab.

ADDITIONAL INFORMATION

Assign trust settings relative to other systems. You may choose the trust values yourself or use the explicit values.

i. Addresses from the **Sales** system are more reliable than addresses from the CRM system.

ii. Addresses from the default **Admin** system are the most reliable, but it should not survive for longer than 3 months, because within that period the **Sales** system's support team has to follow up and verify all addresses that are corrected in the Customer Master.

iii. On average, addresses for our fictitious company's customers change after 7 years.

Module 5: Configure the Load Process

Lab 5-2: Running the Load Job

Overview:

In the previous lab, you defined trust and validation rules and populated the staging tables. So, you can go ahead and run the Load jobs now. These jobs will load the data from the staging tables into their target base objects.

Objective:

- Run the Load process

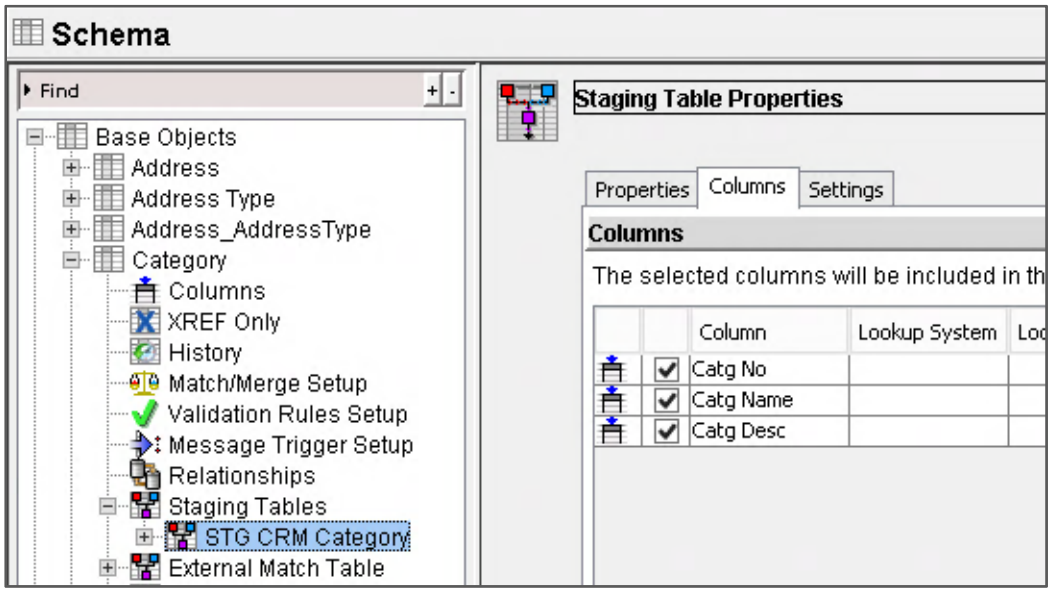
Duration:

30 minutes

Tasks

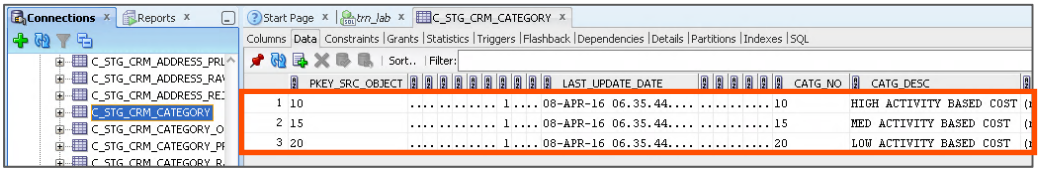
Review the Contents of a Base Object and its Staging Tables

1. In the **Category** base object, review the staging table used.

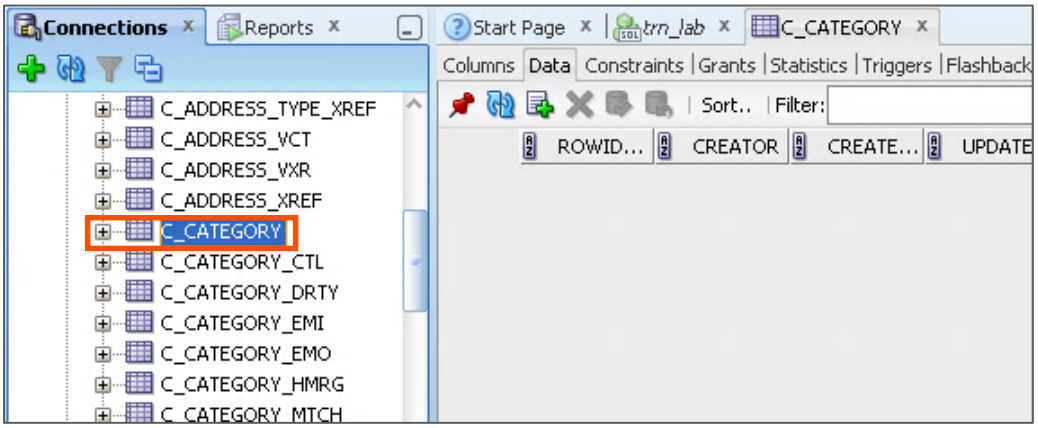


2. Navigate to **SQL Developer** and connect to **Training**.
3. Expand **Tables**.
4. Select **C_STG_CRM_CATEGORY**.

5. Click the **Data** tab and observe that there are 3 records in the right pane.



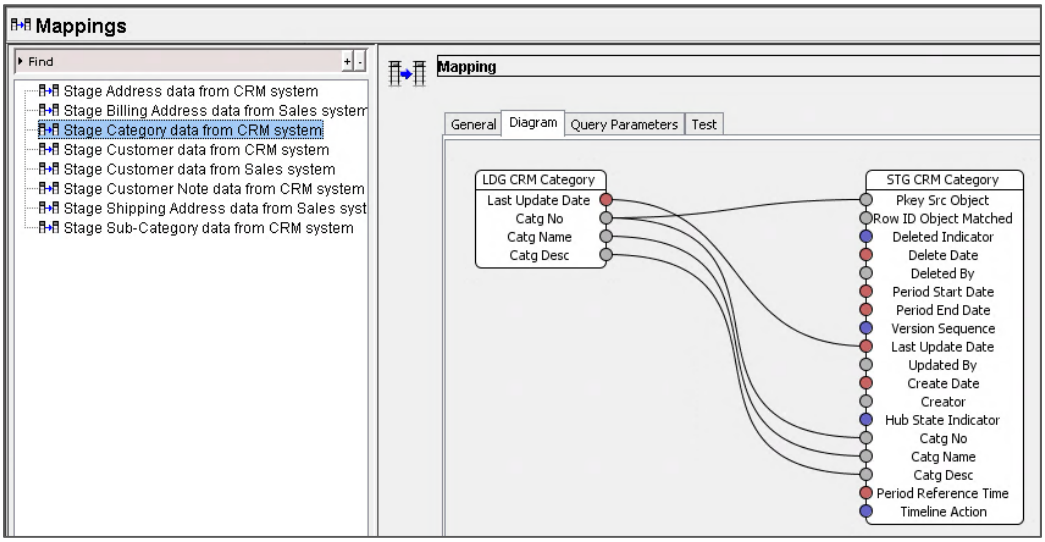
6. Select the base object table **C_CATEGORY**.



Note: When the load job is run, the base object is populated with the records from the staging table. This will currently be empty.

Run the Load Job for each Staging Table of a Base Object

7. In the MDM Hub Console, navigate to **Mappings** > **Stage Category data from CRM system**.
8. In the diagram view, observe the mapping associated with the **Category** base object.

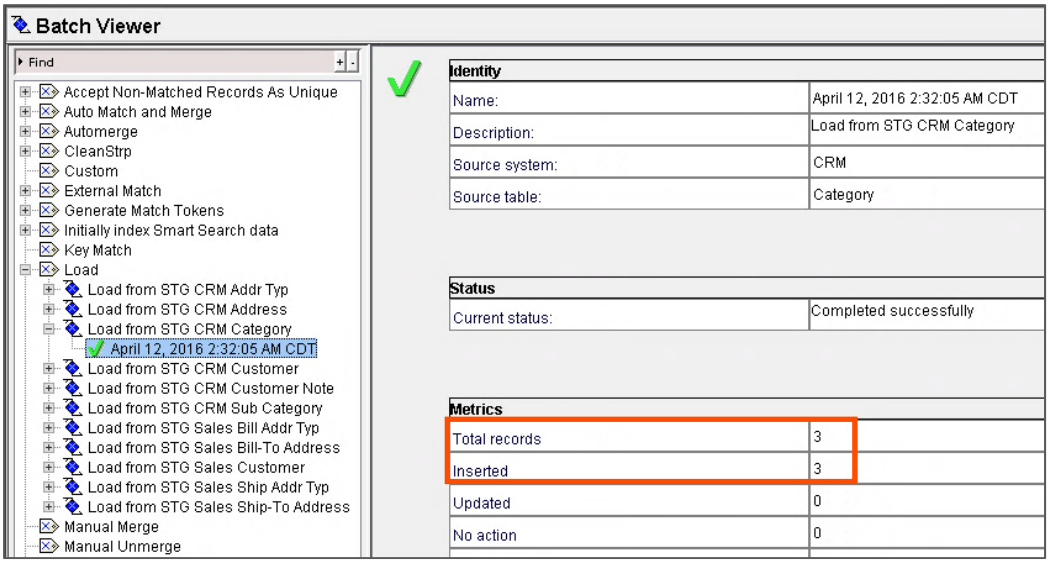


Important

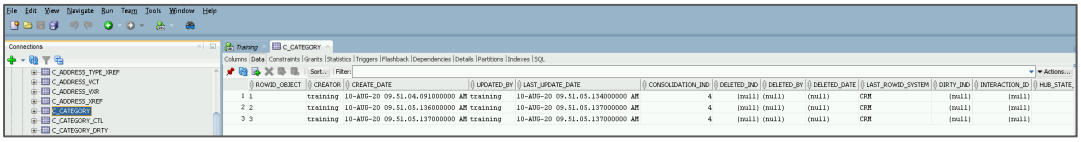
The **Sub_Category** base object (a child of the **Category** parent base object) also has a mapping that associates with its staging table **Stage Sub-Category data from CRM system**. The sub-category records have foreign keys that point to a category parent record.

Therefore, the sequence in which you run Load jobs matters. **Never load a base object before you load its parent base objects**; if you do, the load of the child base object will fail because the lookups on the parent base object do not exist.

9. Navigate to **Utilities** > **Batch Viewer**, to run the load jobs for the staging tables.
10. Change the Group By option to **By Procedure Type**.
11. Expand the **Load** node.
12. Select the **Load from STG CRM Category** job and click **Execute Batch**.
13. Check the status of the job and confirm that **3** records are loaded.



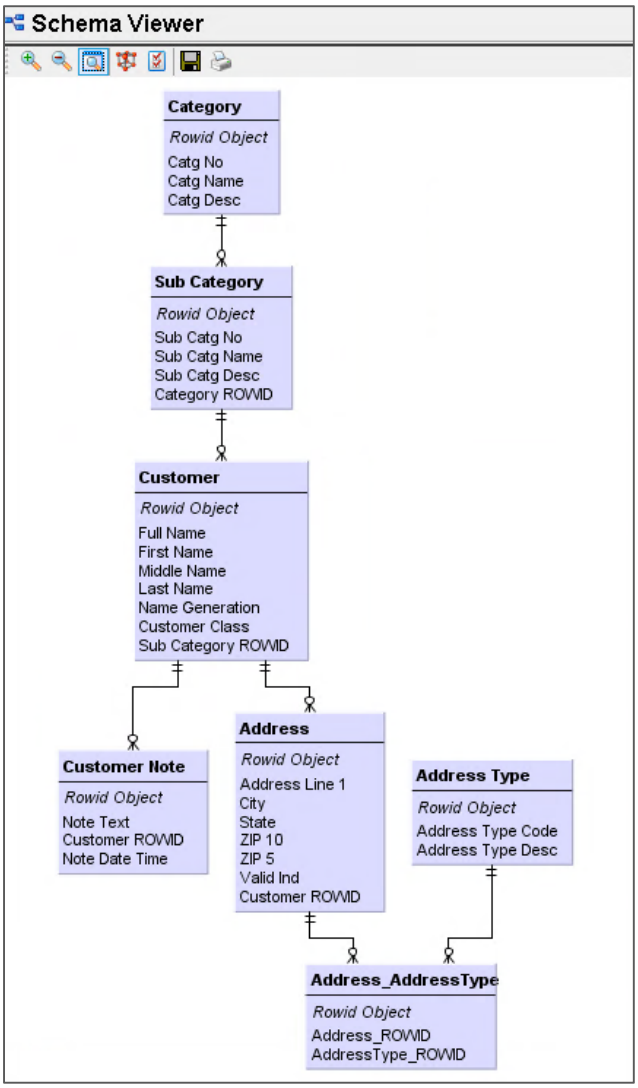
14. Open the **SQL Developer** and select the **C_CATEGORY** table.



Note: The three records that were in the **STG CRM Category** staging table are loaded into the base object. The **Consolidation_Ind** column for each record is set to **4**. This means that these are newly added records.

Load from Staging Tables in the Correct Order

15. Back in the MDM Hub Console, open the **Schema Viewer**.



Here, you can clearly see the dependency of the base objects and figure out that base objects must be loaded before their child base objects are loaded.

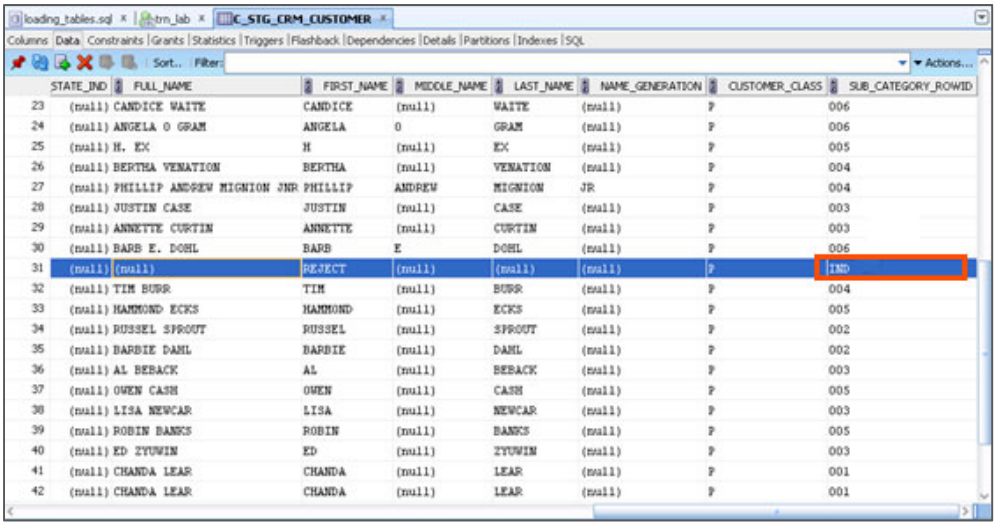
16. Navigate back to the Batch Viewer, run the load jobs in the following order and verify the number of records inserted as mentioned below:

Job	Number of records inserted
Load from STG CRM Sub Category	12
Load from STG CRM Customer	45; 1 reject
Load from STG CRM Customer Note	11
Load from STG CRM Address	45
Load from STG Sales Customer	84
Load from STG Sales Bill-To Address	82
Load from STG Sales Ship-To Address	21; 5 no action

17. Check the status of each job after it finishes running.
Note: For the Load from STG CRM Customer job, there is one rejected record.

Rejected Records in Load Jobs

18. To determine the reason why the single record was rejected, open **SQL Developer**.
19. In the **C_STG_CRM_CUSTOMER** table, in the FIRST_NAME column, find the record with the name **REJECT**.
20. For the **SUB_CATEGORY_ROWID** foreign key column, observe the invalid value of **IND**. That is the cause of the rejection – an invalid foreign key.



This concludes the lab.

ADDITIONAL INFORMATION

Rejected Records in Load Jobs

During the load process, records in the staging table can be rejected for the following reasons:

a. Future date or NULL date in the LAST_UPDATE_DATE column
b. LAST_UPDATE_DATE less than 1900
c. NULL value mapped to the PKEY_SRC_OBJECT of the staging table
d. Duplicates found in PKEY_SRC_OBJECT
e. Invalid value in the HUB_STATE_IND field (for state-enabled base objects only)
f. Invalid or NULL foreign keys

Module 6: Match and Merge Processes Overview

Lab 6-1: Setting Match Strategy, Match Columns, and Match Path

Overview:

Match Strategy specifies the reliability of the match versus the performance you require – fuzzy, or exact. An exact match/search strategy is faster. However, an exact match can miss some matches if the data is imperfect. A Match Path allows you to traverse the hierarchy between records – whether that hierarchy exists between base objects (inter-table paths) or within a single base object (intra-table paths). You can use match paths to configure match column rules that involves related records in either separate tables or in the same table.

Match Column is a column that you use in a match rule for comparison purposes. Each match column is based on one or more columns from the base object.

Objective:

- Set Match/Search Strategy for a Base Object
- Create Exact and Fuzzy Match Columns to use in Match Rules
- Define a Match Path

Duration:

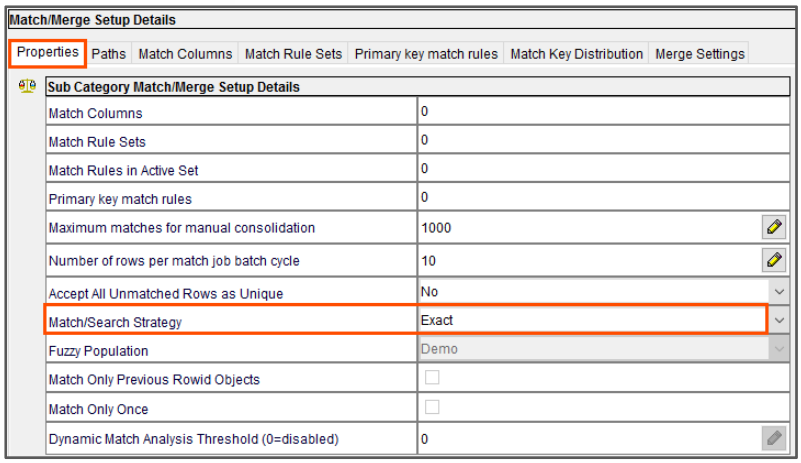
40 minutes

Tasks

Define an Exact Match Column for a Base Object

The Sub Category base object uses an Exact Match/Search Strategy to match subcategories that have the same values in Sub Catg Name.

1. Open the **Schema** tool and ensure that you acquire the **Write Lock**.
2. Expand the **Sub_Category** base object and click on its **Match/Merge Setup** node.
3. Select the **Properties** tab.
4. From the Match/Search Strategy field, ensure that **Exact** is selected.

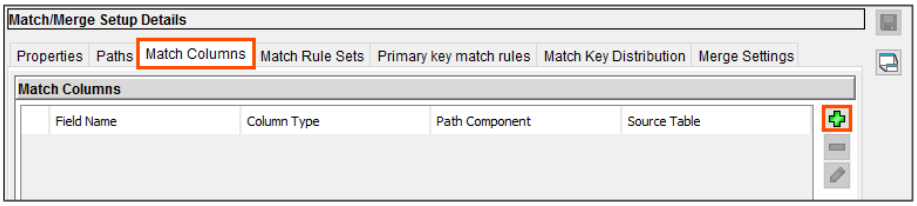


5. **Save** the details, if you have made any changes.

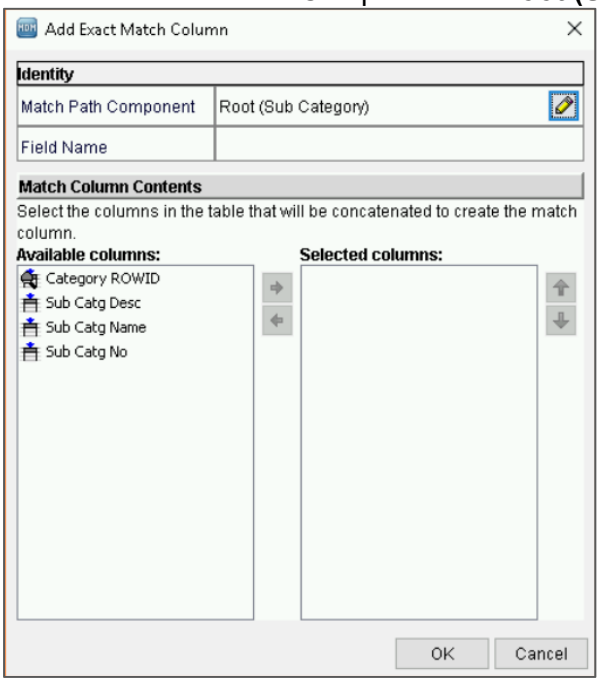
Define an Exact Match

You can use Exact Match Columns when the Match/Search Strategy for the base object is set to Exact or Fuzzy. In this case, the Sub Category base object uses Exact as the strategy. It is not possible to use a Fuzzy Match Column with an Exact Match/Search Strategy.

6. Select the **Match Columns** tab.
7. Click the **Add Exact Match Column** button.

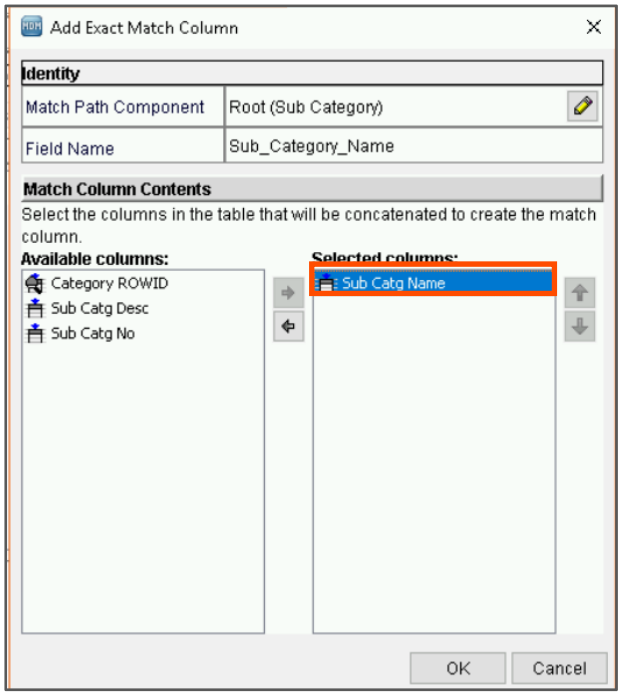


8. Retain the Match Path Component as **Root (Sub Category)**.

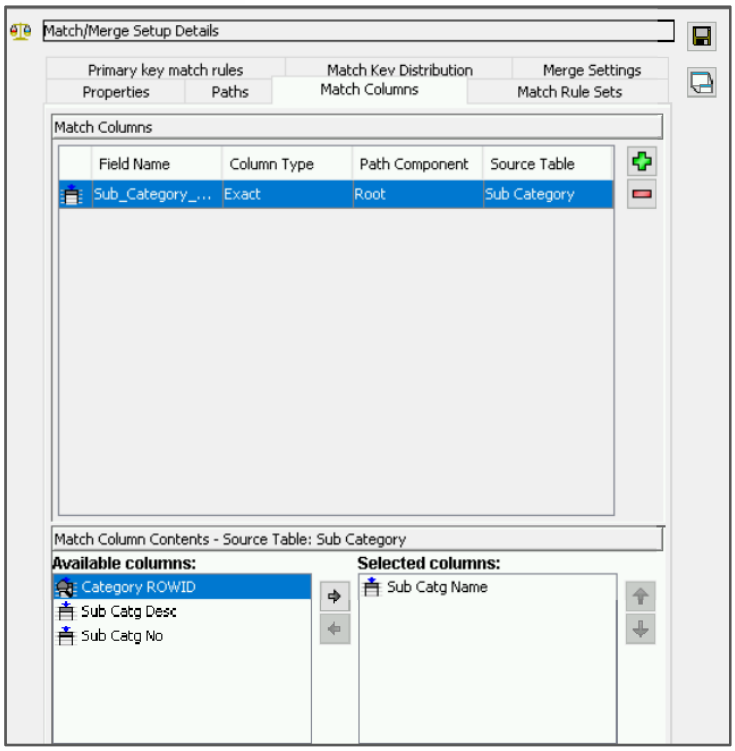


9. Enter the **Field Name** as **Sub_Category_Name**. This provides a label for this field when you use it in match rules.

10. From the list of available columns, select **Sub Catg Name** and click **Select Column**  to move it to the Selected Columns list.



11. Click **OK**.
12. **Save** the details.



Note: The exact match column **Sub_Category_Name** is added to the list of match columns and is tied to the **Sub Catg Name** physical column. We will come back to the Sub Category base object later in the lab.

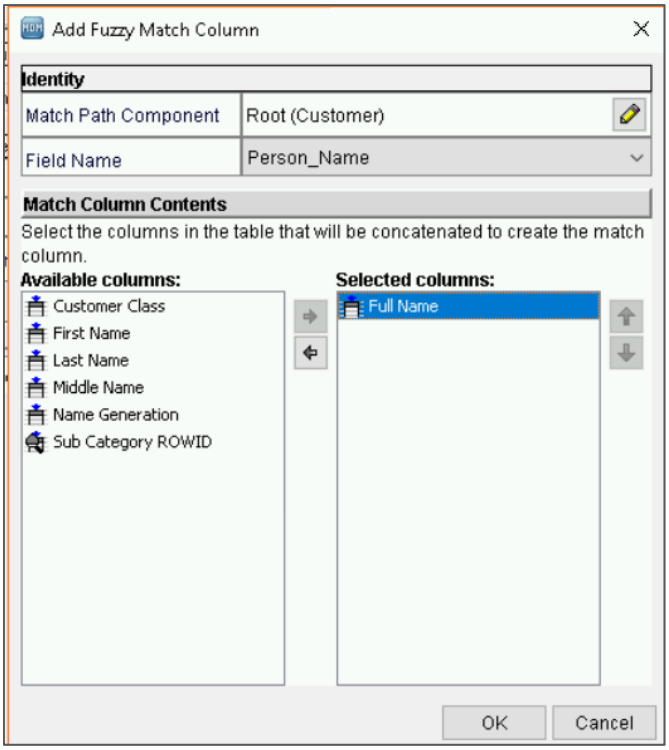
Define a Fuzzy Match Column for the Customer Base Object

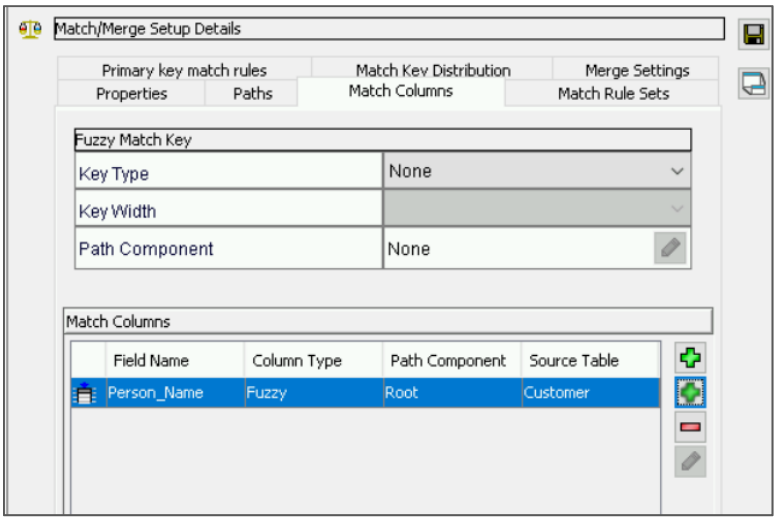
13. Click **Base Objects** > **Customer** > **Match/Merge Setup**.
14. In the **Match Columns** tab, click **Add Fuzzy Match Column** .

Important

Be careful to note the difference between **Add Fuzzy Match Column** (fuzzy-edged) button and **Add Exact Match Column** (sharp-edged) button.

15. Retain the Match Path Component as **Root (Customer)**.
16. From the **Field Name** drop-down, select **Person_Name**.
17. From the Available columns list, move **Full Name** to the **Selected columns** list.
18. Click **OK**.



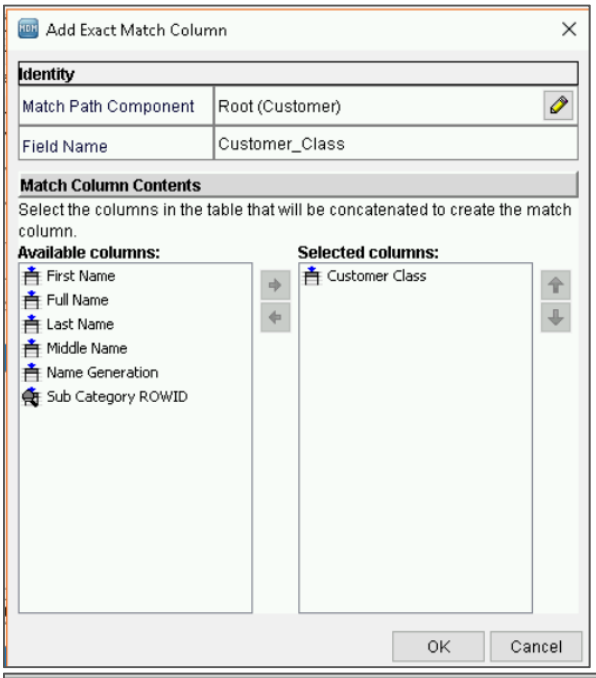


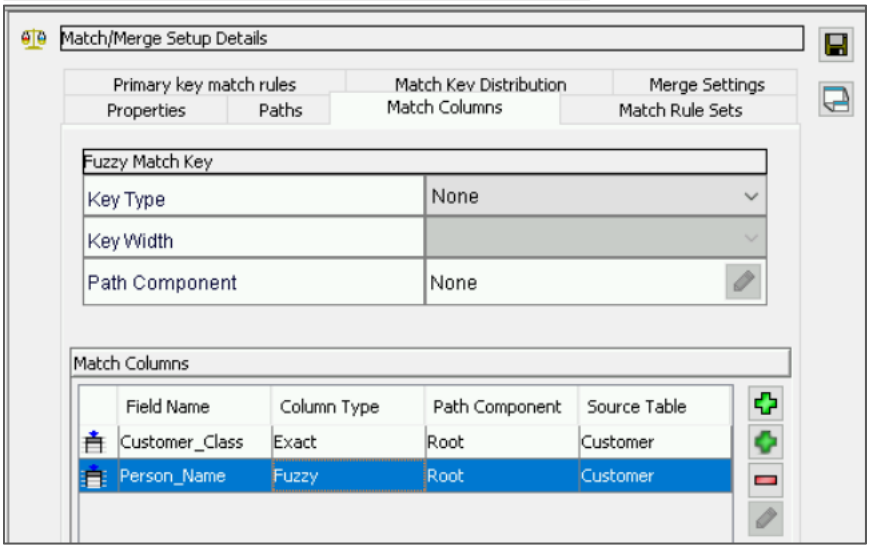
Define an Exact Match Column for the Same Base Object

You can include exact match columns even after you select **Fuzzy** as the overall **Match/Search Strategy** for the **Customer** base object.

19. For the Customer base object, in the Match Columns tab, click **Add Exact Match Column**  .
20. Ensure that the Match Path Component is set to **Root (Customer)**.
21. In the **Field Name** field, enter **Customer_Class**.
    Note: For exact match columns, there is not a predefined list of match columns, so you must provide a name for the match column yourself.

22. From the Available Columns list, move **Customer Class** to the Selected columns list.
23. Click **OK**.

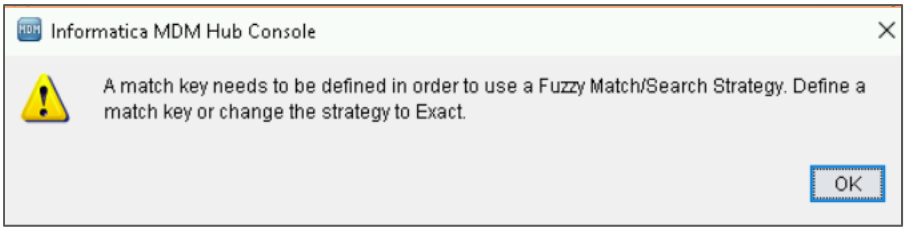




Note: For the Customer base object that uses Fuzzy as the Match/Search Strategy, you have two Match Columns defined, one Fuzzy and the other Exact.

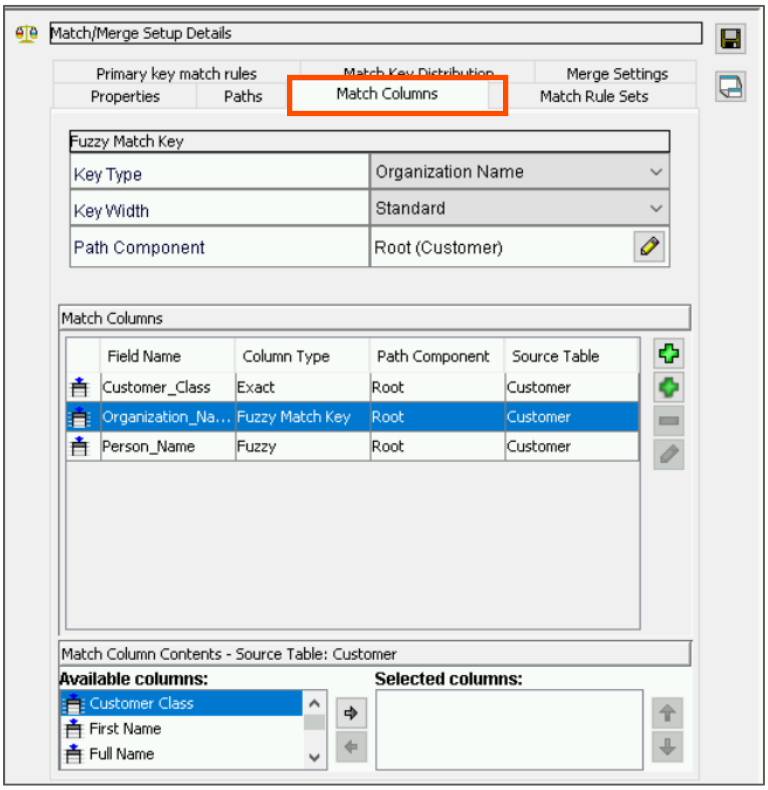
24. **Save** the details.

Note: You may get an error window. You need to provide a Fuzzy key before saving. Click **OK** to continue.

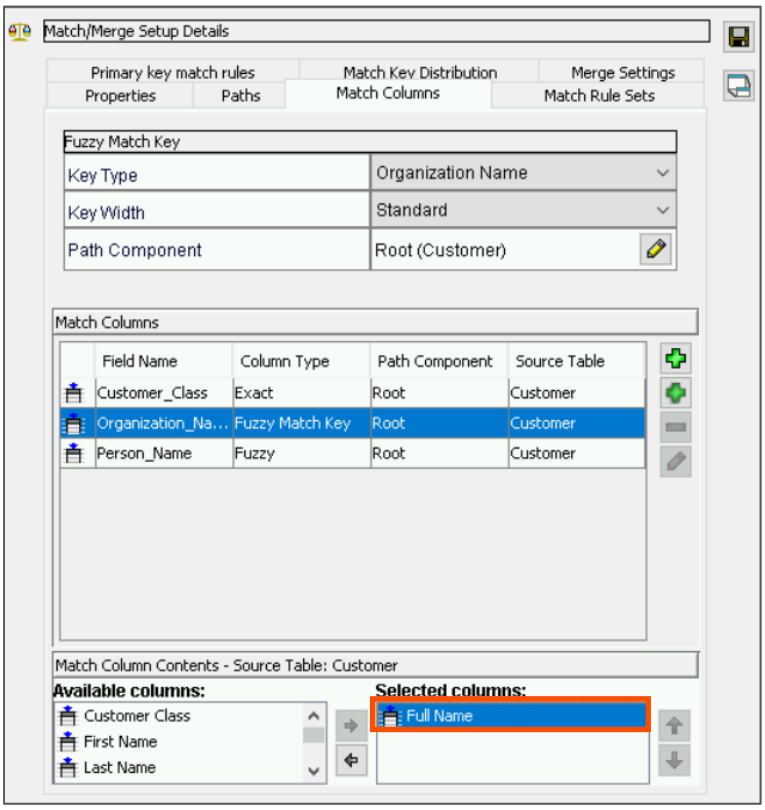


Define the Match Key for the Base Object

25. For the Customer base object, navigate to **Match/Merge Setup** > **Match Columns** tab.
26. In the Fuzzy Match Key section, from the **Key Type** drop-down, select **Organization_Name**.
27. Retain the Key Width as **Standard**.
28. Ensure that the **Path Component** is set to **Root (CUSTOMER)**.



29. Under the **Match Column Contents** section, from the Available columns list, select **Full Name** and move it to the Selected columns list.



30. **Save** the details.

    Note: Ignore any warning message that says no fuzzy rules have been defined.

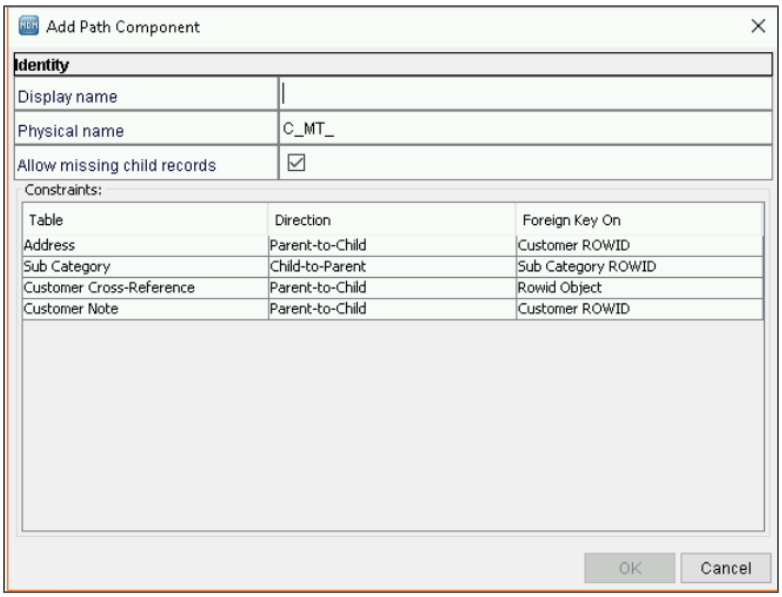
31. Click **OK** to continue.

Define a Match Path

Now, you will include data from the Address base object that helps to make more accurate matches for the customer records. You must define a match path to include address data.

32. Select the **Paths** tab.

33. In the Path Components section, click on a row and click **Add**  . A dialog box opens.

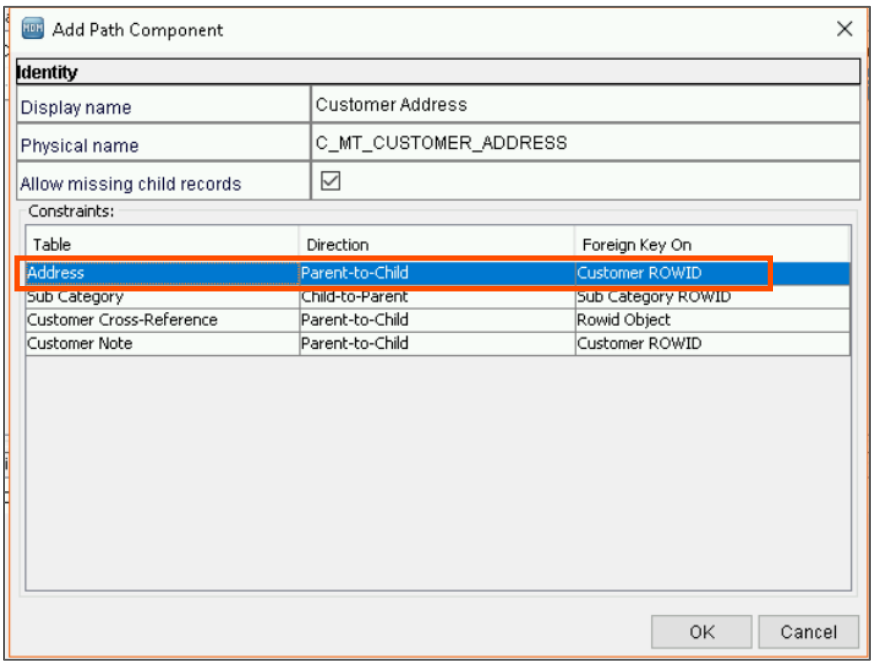


34. In the Display Name field, enter **Customer Address**.
35. To provide a default physical name, press the **Tab** key or click on the Physical Name field.
36. Ensure that the **Allow missing child records** checkbox is selected.

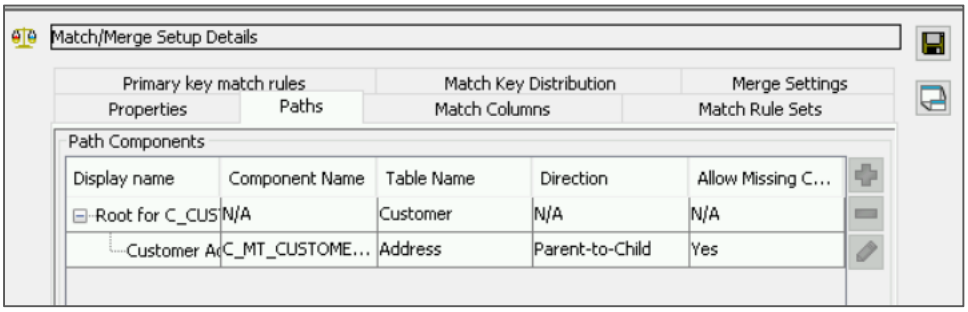
Allow missing child records
If you select **Allow missing child records** option, an outer join is created between Customer and Address, so that any customer who does not have an address can also be included in the Match job. Without this option, an inner join is created, and any Customer without an Address is eliminated from the data sent to the match engine.

37. In the Constraints section, select **Address** as the child table.

38. Click **OK**.



Note: In the Paths tab, Customer Address now shows as a match path.



39. **Save** the details.

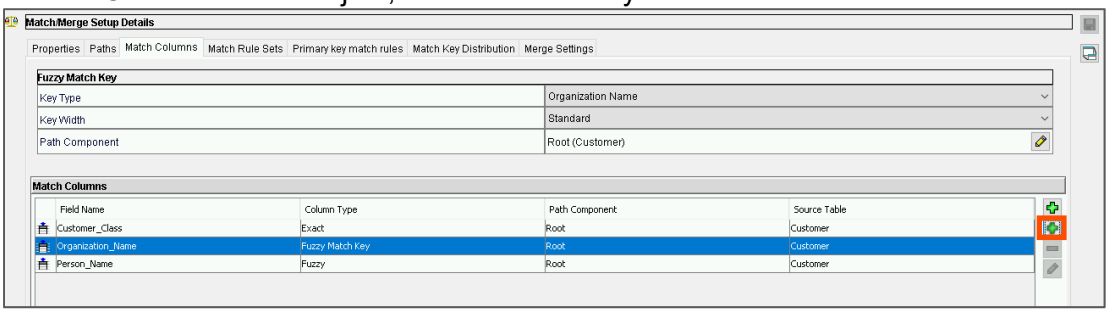
    Note: Ignore the warning message that says no fuzzy rules have been defined.

40. Click **OK** to continue.

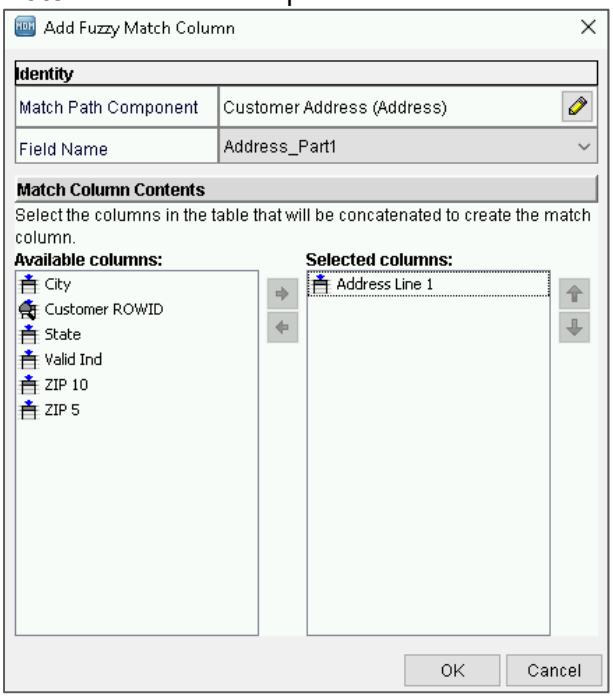
Define Match Columns for the Customer Base Object

You will now define more match columns for the Customer base object.

41. For the Customer base object, add another fuzzy match column.



42. Click the **Edit** (pencil) icon and set the Match Path Component to **Customer Address**.
43. From the **Field Name** field, select **Address_Part1**.
44. Move the **Address Line 1** column from the Available columns section to the Selected columns section.
    Note: Address Line 1 provides the data for the match column.



45. Add another fuzzy match column **Address_Part2**.
46. You should provide the data for the match column by the **City**, **State**, and **ZIP 10** columns of the **Address** base object.
    Note: Move the City, State, ZIP 10 columns to the Selected columns section.
47. Add an Exact match column as mentioned below:

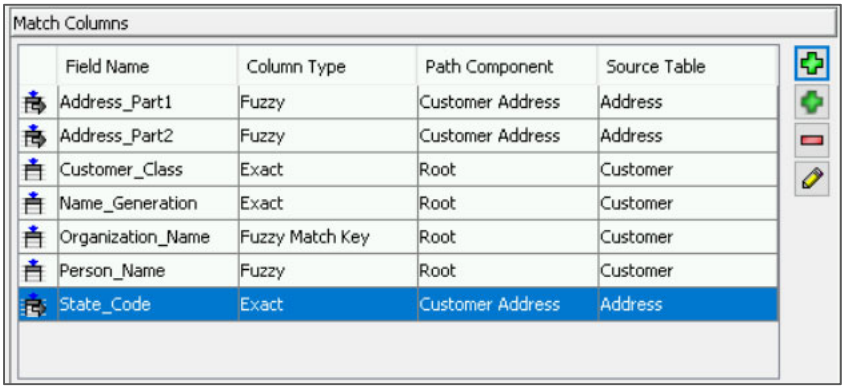
Exact Match Column	Customer Match Column
Match Path Component	Root (Customer)
Field Name	Name_Generation
Selected Columns	Name Generation

48. Similarly, add another Exact match column as mentioned below:

Exact Match Column	Customer Match Column
Match Path Component	Customer Address
Field Name	State_Code
Selected Columns	State

49. **Save** your settings.
    Note: Ignore warnings, if any.

50. Ensure that all the columns are added as shown below:



This concludes the lab.

Module 7: Exact Match

Lab 7-1: Configuring an Exact Match Rules Set and a Range Search

Overview:

Automerge is the process to merge records automatically. Match rules can result in automatic merging or manual merging. A match rule that instructs Informatica MDM Hub to perform an automerge combines the two or more records of a base object table automatically, without manual intervention.

In this lab, you will use an exact match column in a match rule for the **Sub Category** base object. In this rule, records will match and will be flagged for automerge when they have the same values for Sub Category Name. In the Schema tool, you can configure a range search that identifies candidate set of records based on similar tokens for the match key.

Objective:

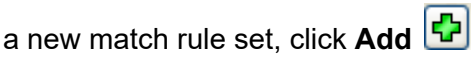
- Define match rules with the use of exact match columns and Schema tool
- Select the Match/Search Strategy for a base object
- Select the Match Population for a base object
- Create a Match Rule Set
- Configure a Range Search for the Match Rule Set

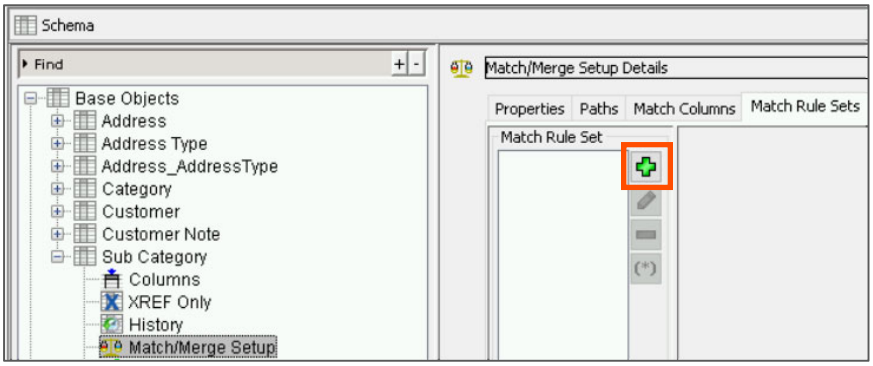
Duration:

15 minutes

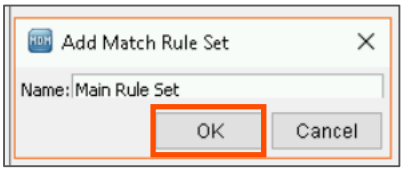
Tasks

Define a Match Rule for a Base Object with an Exact Match Column

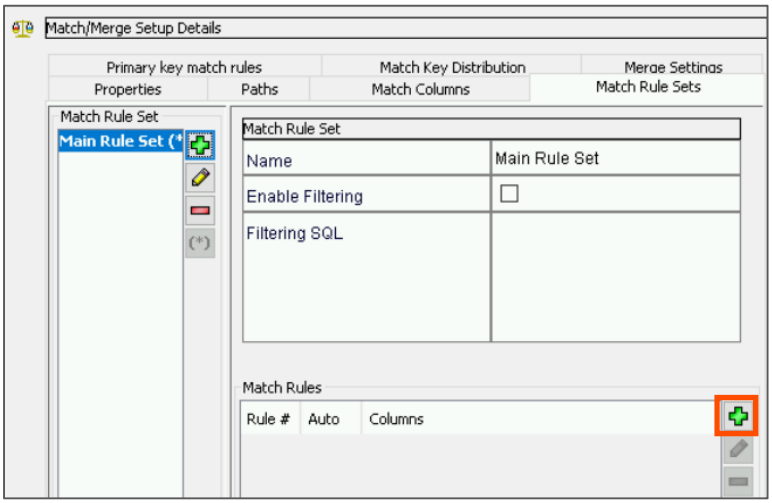
1. In the Schema tool, expand the **Sub Category** base object.
2. Select **Match/Merge Setup**.
3. Click the **Match Rule Sets** tab.
4. To create a new match rule set, click **Add**  .



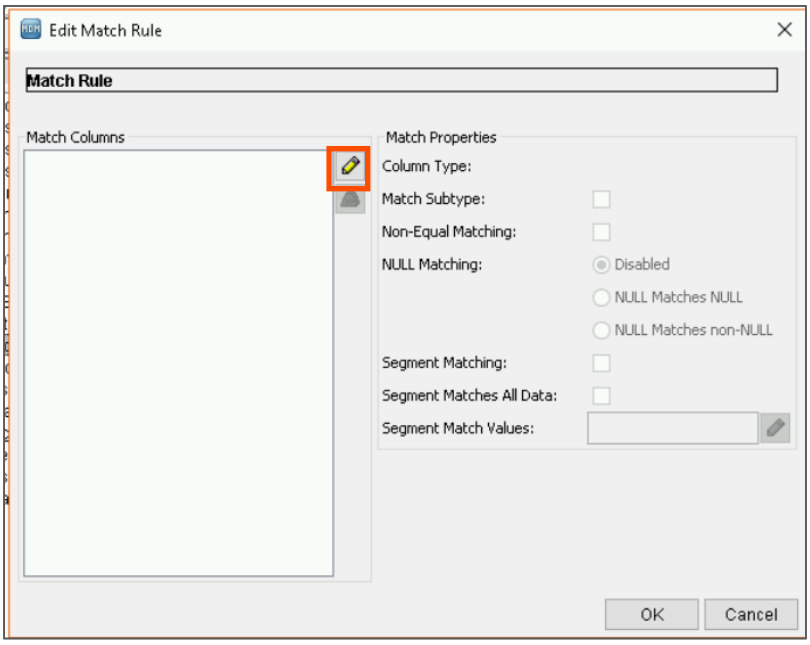
5. In the Add Match Rule Set dialog box, for the rule set name, enter **Main Rule Set**.
6. Click **OK**.



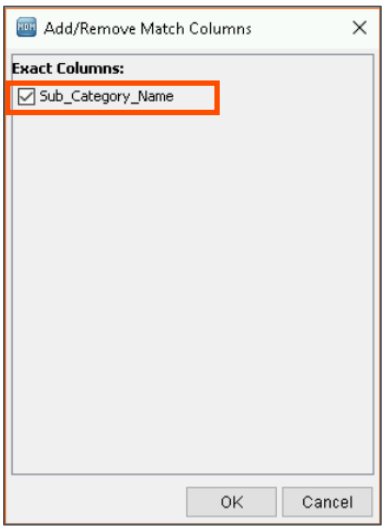
7. To add another match rule to the set, to the right of the **Match Rules** list, click **Add**  .



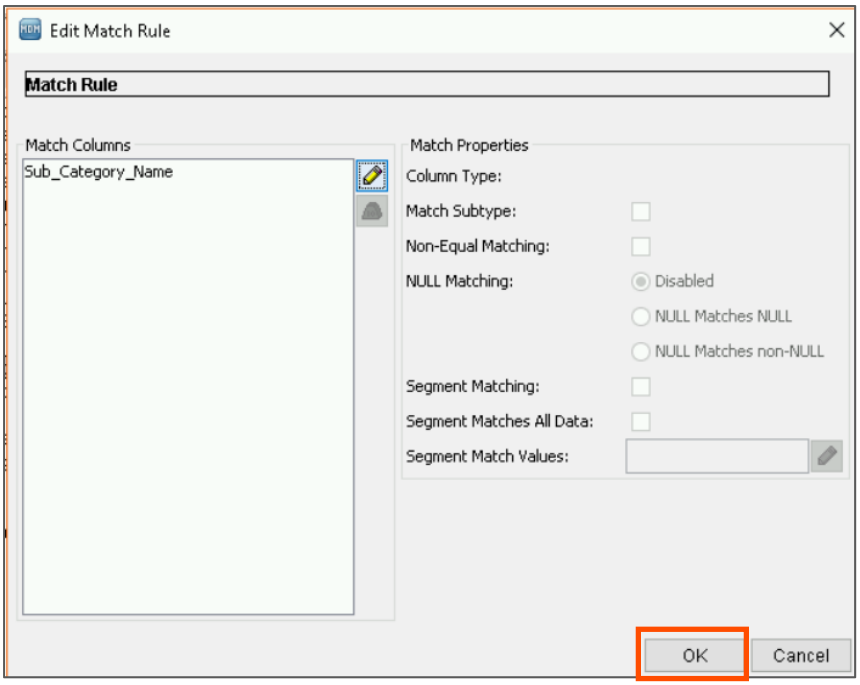
8. Click **Edit**  .



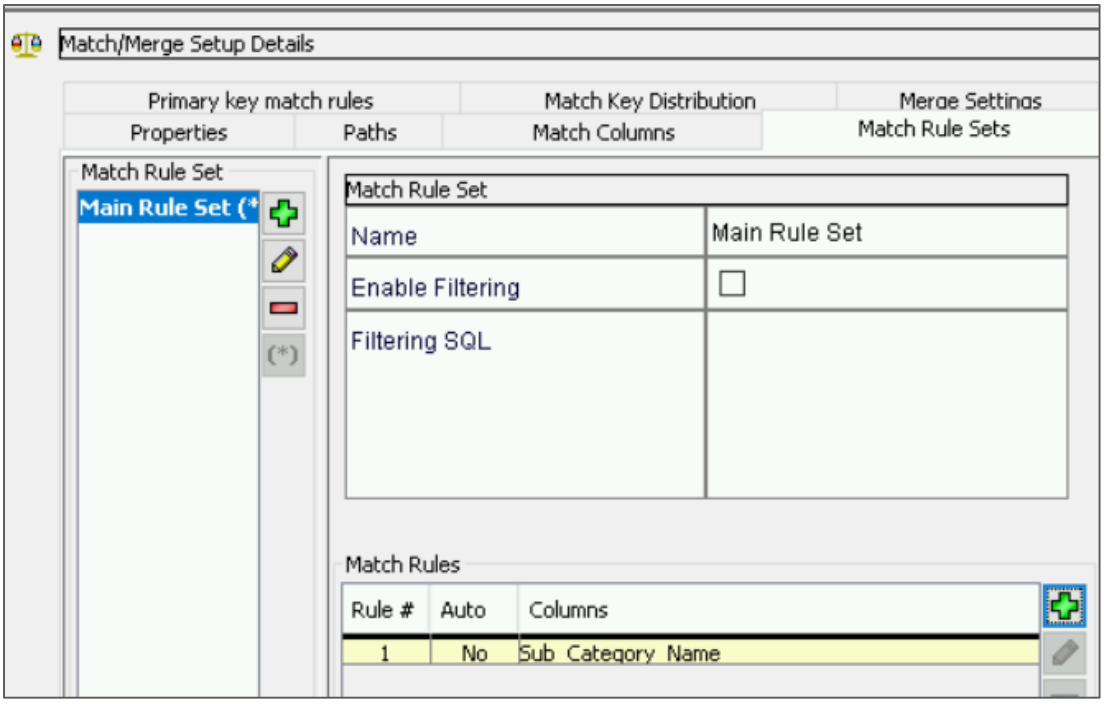
9. In the **Add/Remove Match Columns** dialog box, select the **Sub_Category_Name** match column.



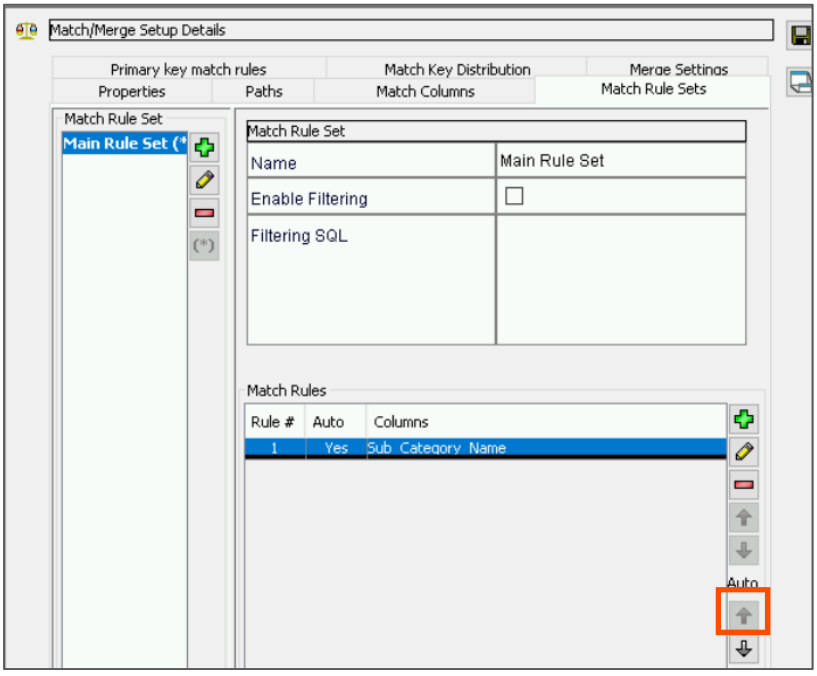
10. Click **OK**.



Note: The **Sub_Category_Name** column is added to the match columns for the match rule. For this match rule, you will not use any of the optional match properties, such as NULL Matching or Non-Equal Matching. By default, a match rule is added as a manual merge rule.



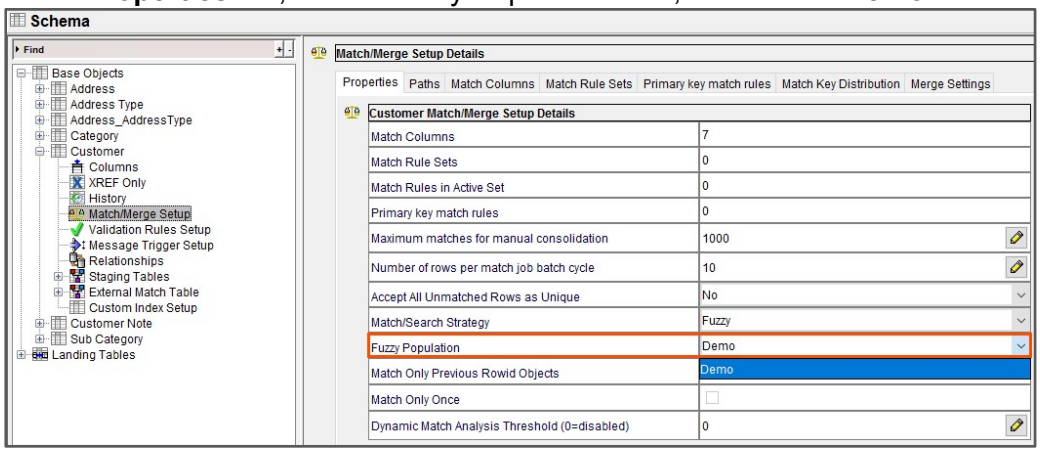
11. To change the match rule from a manual merge rule to an automerge rule, select the match rule and click the **Auto** button.



12. **Save** the rule.

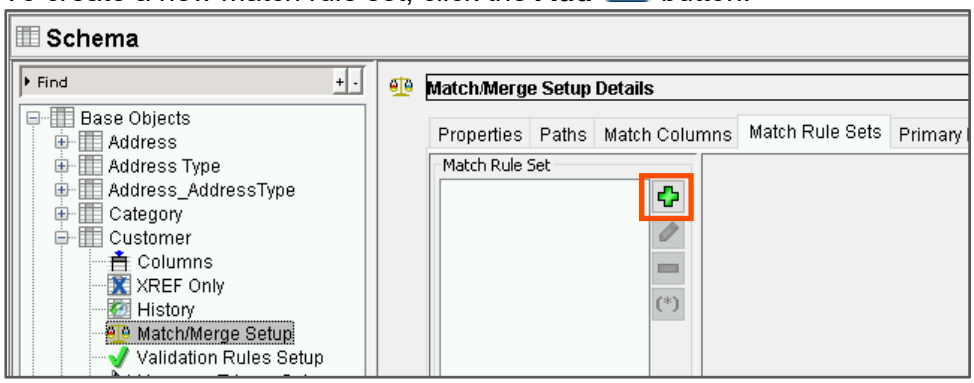
Select the Match/Search Strategy and the Match Population for an Object

13. Open the **Schema** tool and ensure that you have acquired the **Write Lock**.
14. Expand the **Customer** base object and browse to its **Match/Merge Setup** node.
15. In the **Properties** tab, for the Fuzzy Population field, ensure that **Demo** is selected.



Create a Match Rule Set

16. For the **Customer** base object, navigate to **Match/Merge Setup** > **Match Rule Sets** tab.
17. To create a new match rule set, click the **Add** button.



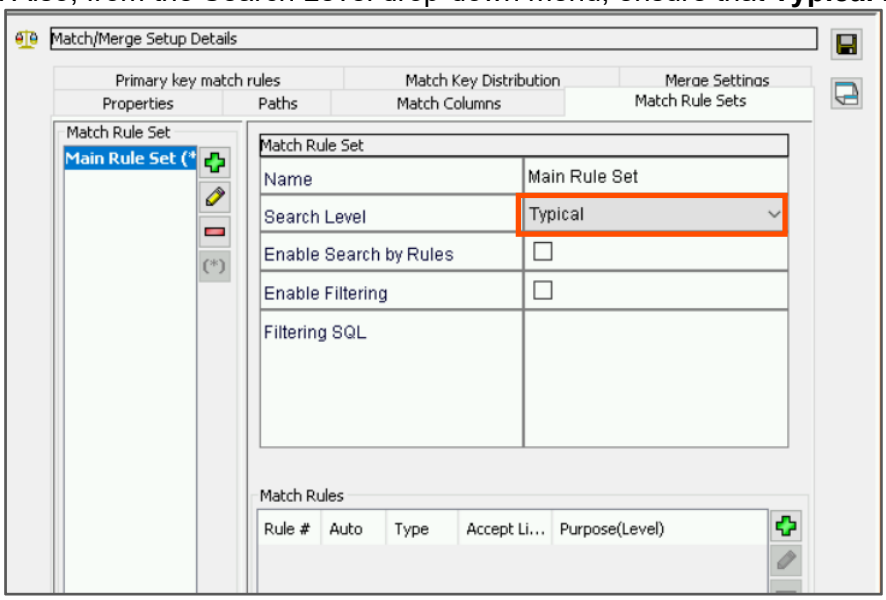
18. In the Name field, enter **Main Rule Set**.

19. Click **OK**.

Configure a Range Search

In this section, you will perform the Fuzzy Match on the Customer base object. When you configure a Range Search, your goal is to identify a candidate set of records based on similar match keys. For this purpose, you must configure a Fuzzy Match Key and the Search Level.

20. From the Match Rules Set tab, ensure that **Main Rule Set** is selected.
21. Also, from the Search Level drop-down menu, ensure that **Typical** is selected.



22. **Save** the rule.
    Note: A warning message appears.
23. Click **OK**.

This concludes the lab.

Module 8: Fuzzy Match

Lab 8-1: Configuring a Fuzzy Match Rule Set

Overview:

Various Match Purposes are optimized to focus on specific properties of records to determine the best matches using those properties. For example, Wide Contact focuses on the names and identification numbers, Resident focuses on address, telephone, and identification numbers. The specific match purposes available are a function of the population file. You can configure fuzzy match rule sets for appropriate base objects.

In this lab, from the Schema tool, you will define match rules using fuzzy match columns.

Objectives:

- Configure Filtering (Filtering SQL) for a Match Rule Set
- Add Columns to a Fuzzy Match Rule in the Match Rule Set
- Configure Match Filtering with an Exact Match Column in the Fuzzy Match Rule
- Configure the Match Purpose and Match Levels for a Fuzzy Match Rule

Duration:

40 minutes

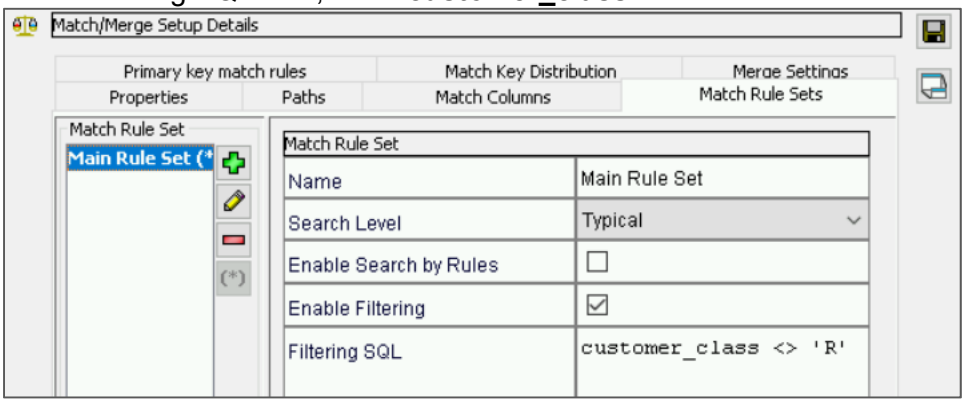
Tasks

Configure Filtering for a Match Rule Set

The Customer base object contains data for Organizations, Individuals, and Regions. The records for Regions do not have complete addresses and have very similar names that can result in overmatching between Region records. You will therefore specify an optional filter (Filtering SQL) in the Match Rule Set to remove Region records from the records to be matched. In an actual implementation, you will define another Match Rule Set specifically tailored to match Region records.

1. Navigate to the MDM Hub Console and ensure that you acquire a **Write Lock**.
2. Select Schema > Base Objects > Customer > Match/Merge Setup > Match Rule Sets.
3. Select Main Rule Set.
4. Select the **Enable Filtering** option.

5. In the Filtering SQL field, enter **customer_class <> 'R'**.



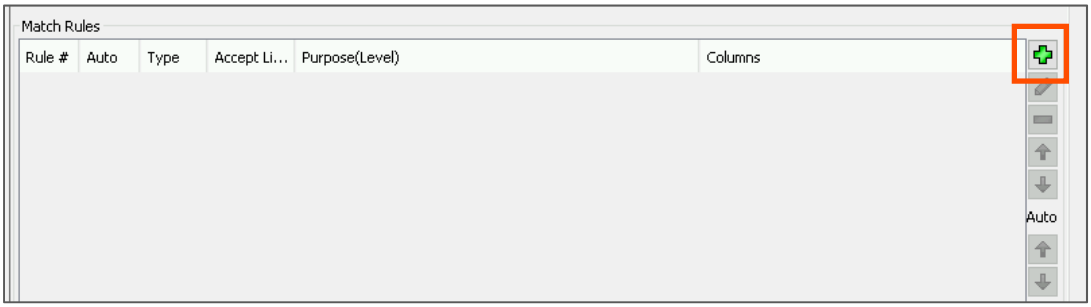
Note: This limits the records to match to those where customer_class is not 'R', that is, persons and organizations.

6. Click **Save** and acknowledge the warning message.

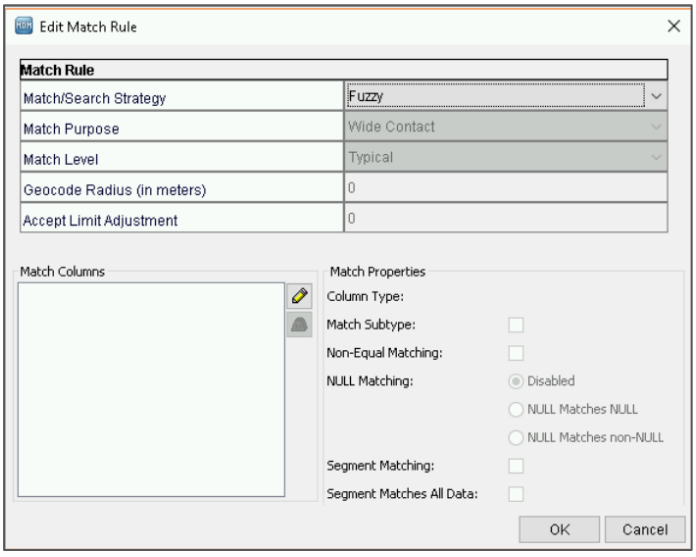
Add Columns to a Fuzzy Match Rule

Now, you will create a fuzzy match rule and add columns to it.

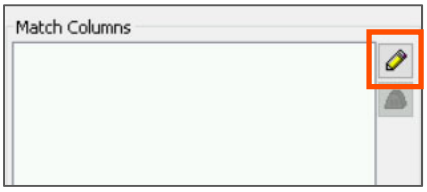
7. To add a new match rule to the match rule set, click **Add**  in the Match Rules section.



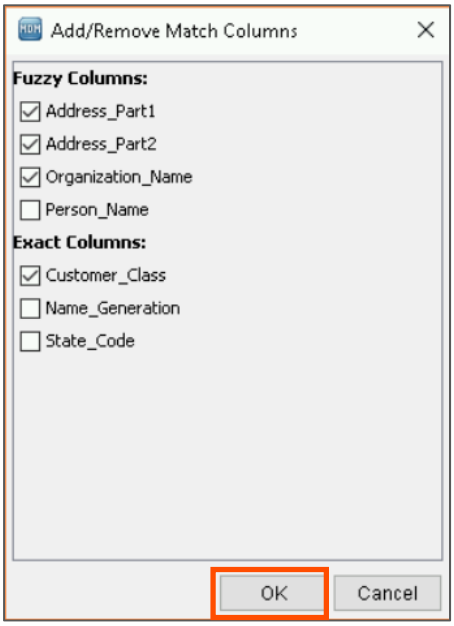
Recommended: Expand the bottom edge to check all the settings in the window.

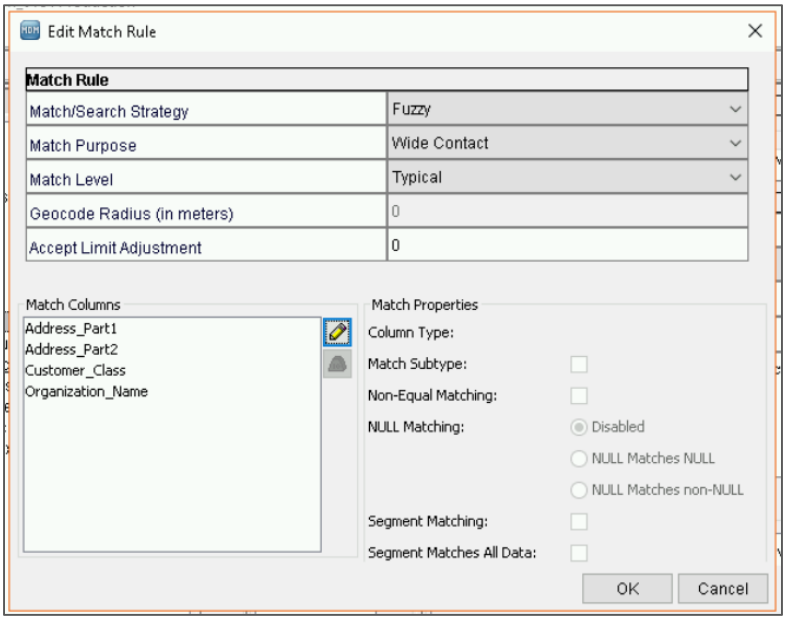


8. Click **Edit**  .



9. In the Add/Remove Match Columns window, from the Fuzzy Columns list, select **Address_Part1**, **Address_Part2**, and **Organization_Name**.
10. From the Exact Columns list, select **Customer_Class**.
11. Click **OK**.

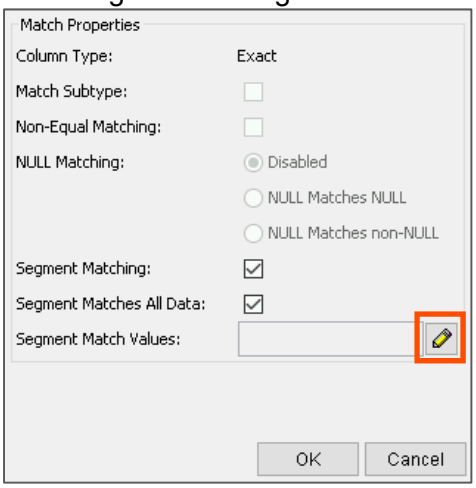




Note: For the Edit Match Rule window, **do not** click OK or close the window.

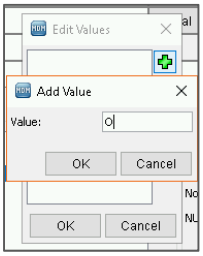
Configure Match Filtering using an Exact Match Column

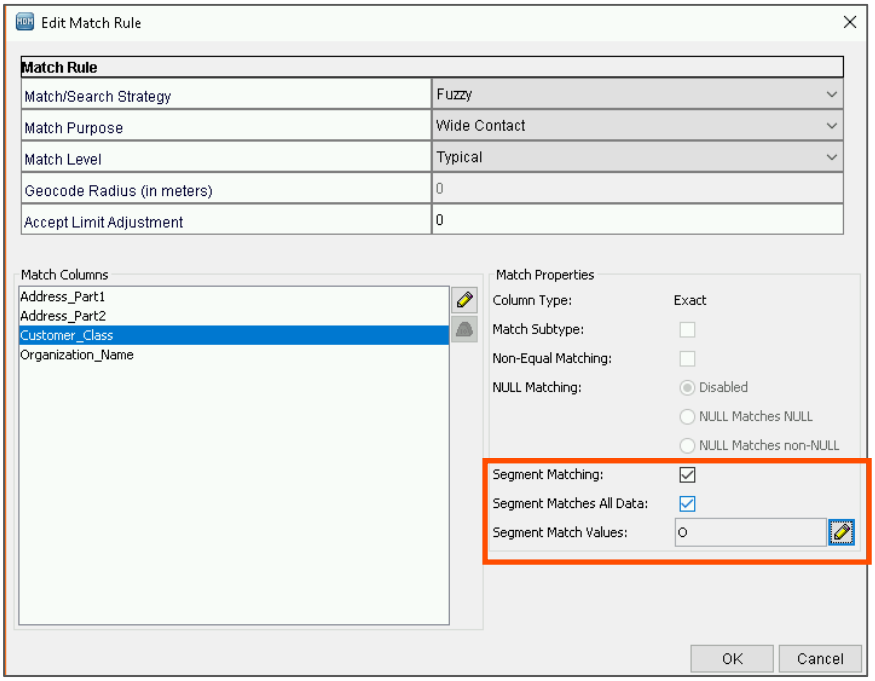
12. In the list of match columns, select **Customer_Class**.
13. Under Match Properties, select the **Segment Matching** and **Segment Matches All Data** options.
14. To the right of the Segment Match Values field, click the **Edit** button.



Note: If you cannot see it, expand the window by dragging the bottom right edge.

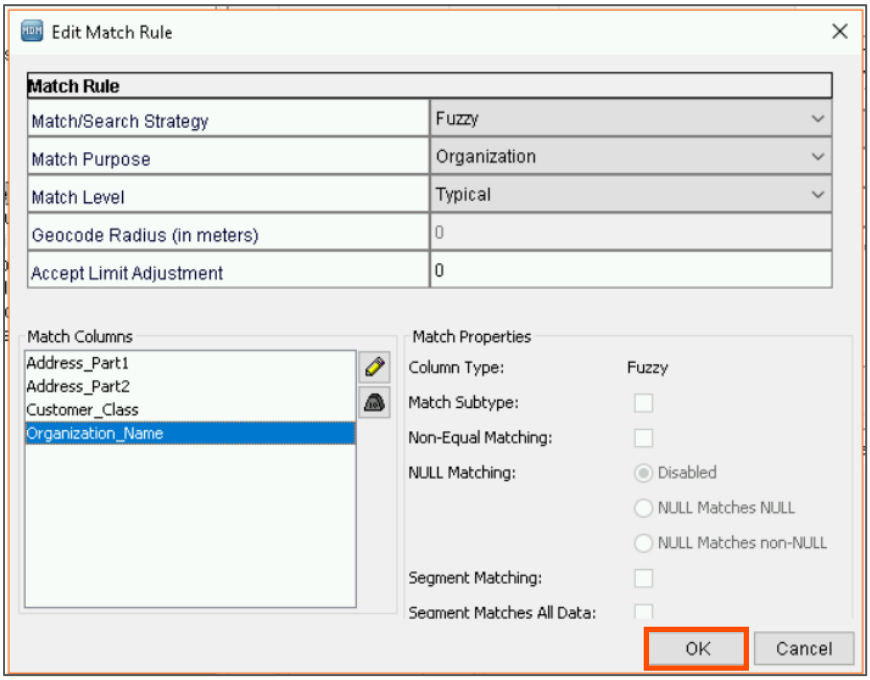
15. In the **Edit Values** dialog box, add the value **O**, and click **OK**.



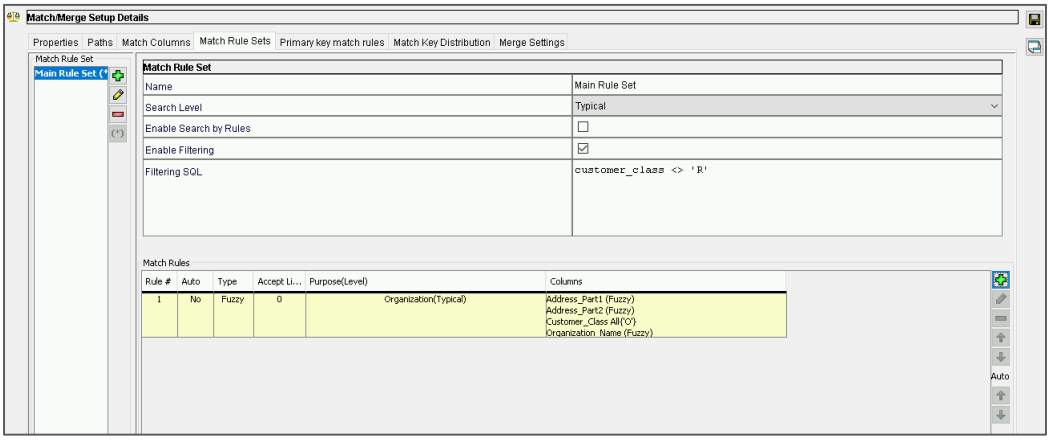


Configure the Match Purpose and Match Level

16. From the Match Purpose drop-down, select **Organization**.
17. From the Match Level drop-down, ensure that **Typical** is selected.
    Note: You cannot set the Match Purpose and Match Level until you add the Match Columns.
18. To close the Edit Match Rule window, click **OK**.



The new rule is added to the list of **Match Rules** for the selected **Match Rule Set**.



Note: This fuzzy match rule uses the fuzzy key column (Organization_Name) as a key to come up with a list of match candidates. The exact match column Customer_Class is used to designate that the focus will be on records that match in the Organization segment with the possibility of records from other segments that match an organization record.

19. **Save** the Match Rule Set.

Add two more Match Rules to the Customer Main Rule Set

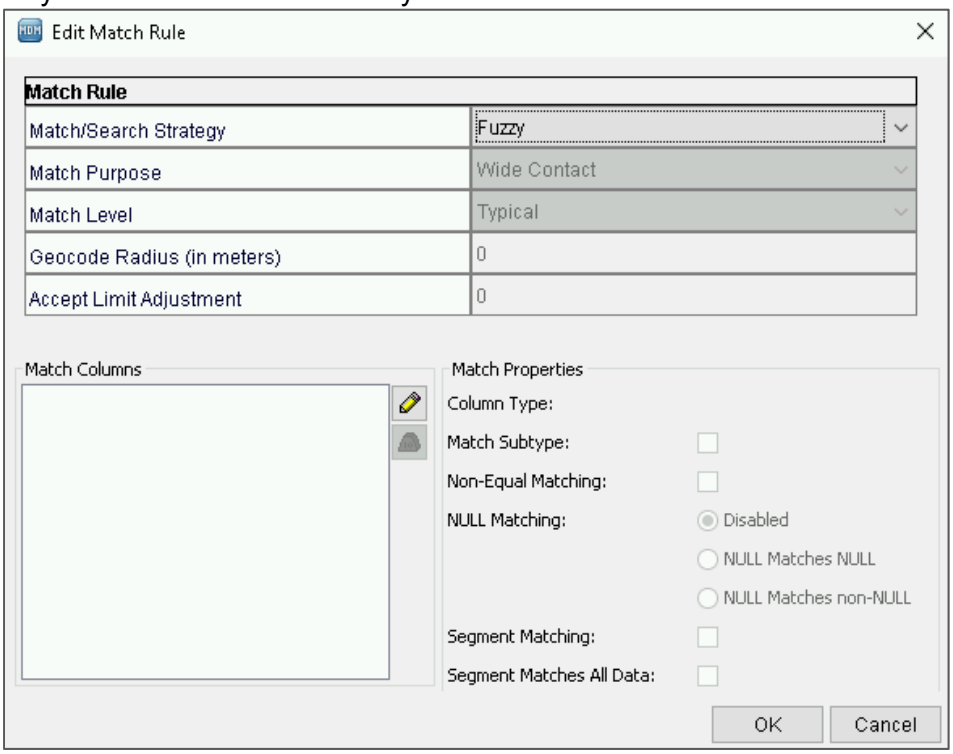
20. Add another match rule to the **Customer** base object's **Main Rule Set**.

    Note: This match rule will apply to Persons (customer_class = P) only and will match them mainly on name and address.
21. Add the **Address_Part1**, **Address_Part2**, **Person_Name**, **Customer_Class**, and **Name_Generation** match columns.
22. From the Match Purpose drop-down, select **Resident** and from the Match Level drop-down, select **Typical**.
23. For **Customer_Class**, select **Segment Matching** and enter **P** as the segment match value.

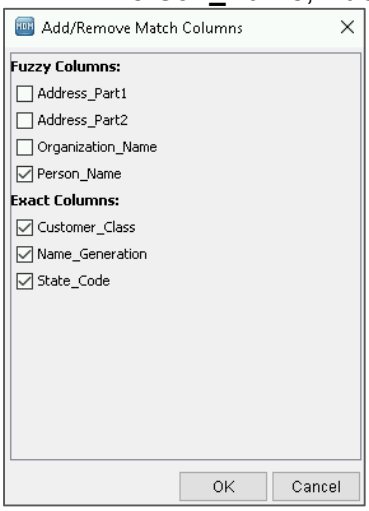
    Recommended: Expand the bottom edge of the window to check the Segment Match Values attribute.
24. For the **Name_Generation** column, select the **NULL Matches NULL** option.

2	No	Fuzzy	0	Resident(Typical)	Address_Part1 (Fuzzy) Address_Part2 (Fuzzy) Customer_Class {'P'} Name_Generation (Ø↔Ø) Person_Name (Fuzzy)

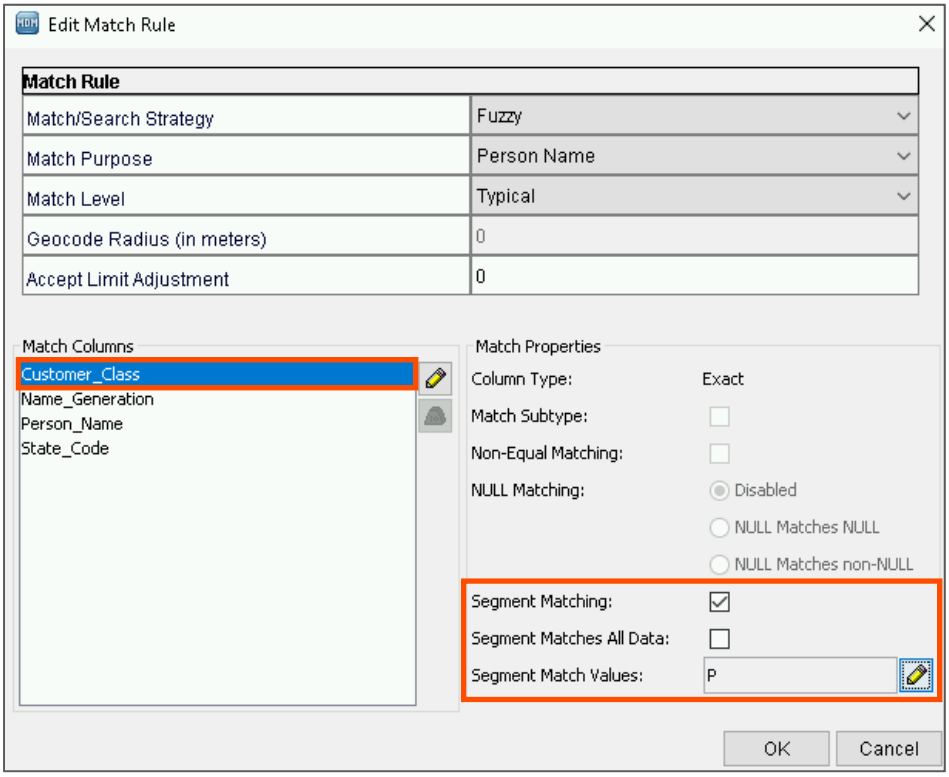
25. Add one more match rule. This match rule will apply to Persons (customer_class = P) only and will match them mainly on name and state.



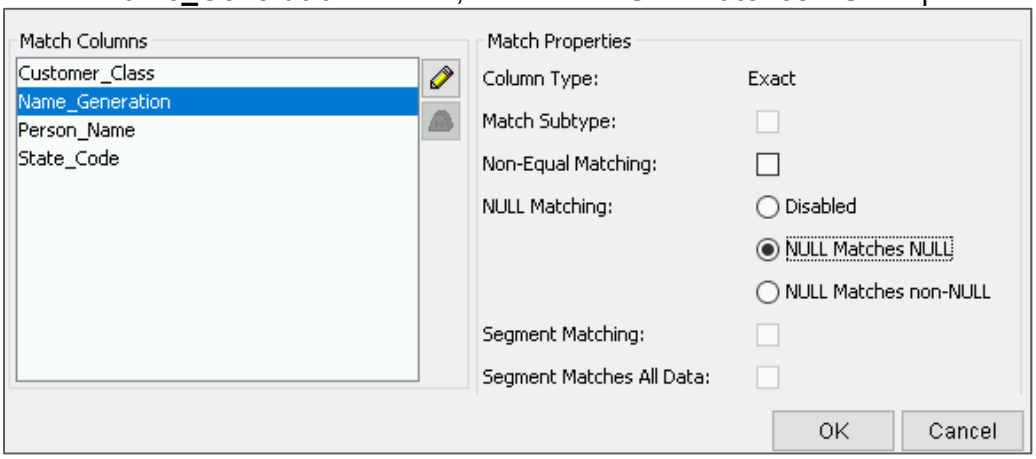
26. Add the **Person_Name**, **Customer_Class**, **Name_Generation**, **State_Code** columns.



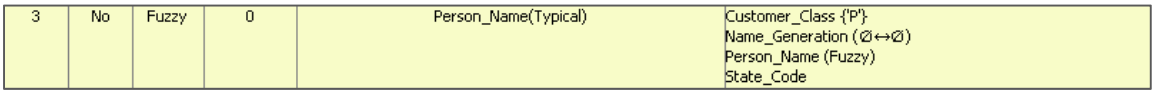
27. From the Match Purpose drop-down, select **Person Name** and from the Match Level drop-down, select **Typical**.
28. For **Customer_Class**, select **Segment Matching** and enter **P** as the segment match value.



29. For the **Name_Generation** column, select the **NULL matches NULL** option.



30. Click **OK**.



Note: You can leave these new match rules as manual merge rules.

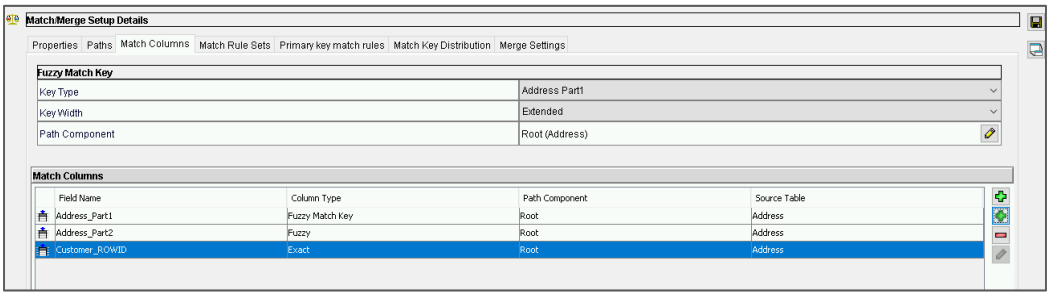
31. **Save** the Match Rule Set.

Add a Match Rule Set and Match Rule to the Address Base Object

Now, you will add a match rule to the Address base object to match addresses if their foreign key, Customer_ROWID, values are identical, and their addresses are considered similar when you match them loosely.

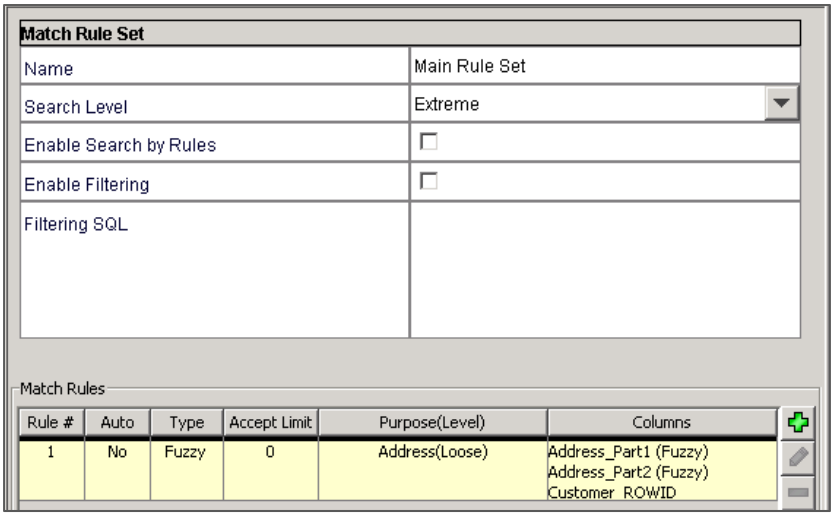
32. Navigate to the **Address** base object and select the **Match/Merge Setup** node.
33. Under the Properties tab, set the **Number of rows per match job batch cycle** property to **200**.
    Note: As we have 148 addresses in our base object, this will ensure that they all are processed in one batch job.
34. Ensure that the Match/Search Strategy is set to **Fuzzy** and the Population is set to **Demo**.
35. Under the Match Columns tab, select the Key Type as **Address_Part1**.
36. Set Key Width to **Extended**.
37. Create an exact match column on **Customer_ROWID**.
    Tip: Match Path = **Root (Address)**, Field Name = **Customer_ROWID**, Selected columns = **Customer ROWID**.
38. Add a fuzzy match column on **Address_Part2**.
    Tip: Match Path = **Root (Address)**, Field Name = **Address_Part2**.
39. As the base object column, select **State**.
    Tip: Selected columns = **State**.

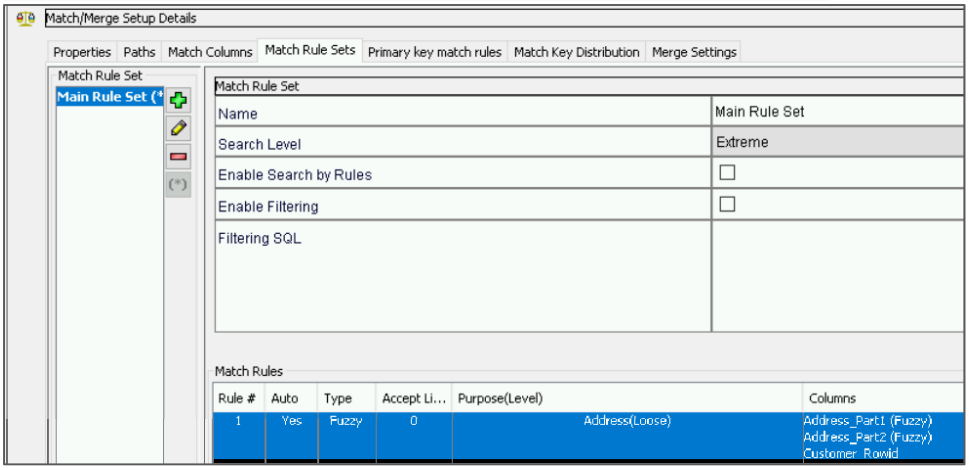
40. Ensure that the match columns are as shown in the following image:



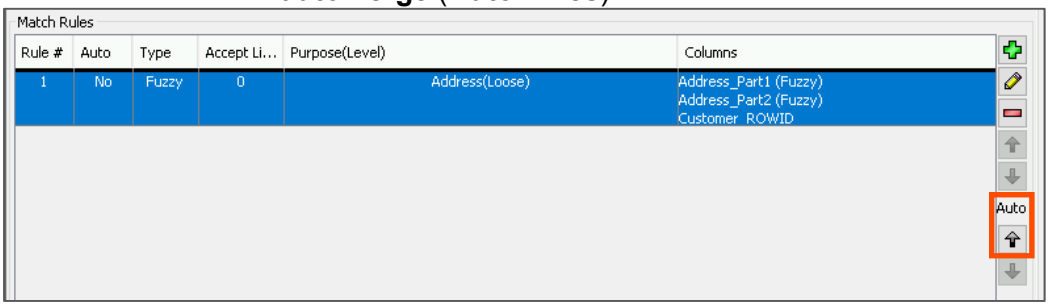
41. Navigate to the Match Rule Sets tab.
42. Create a match rule set named **Main Rule Set**.
43. Set the Search Level to **Extreme**.
44. Add a match rule and select **Address_Part1**, **Address_Part2**, and **Customer_ROWID** match columns.
45. From the Match Purpose drop-down, select **Address** and from the Match Level drop-down, select **Loose**.
46. Click **OK**.

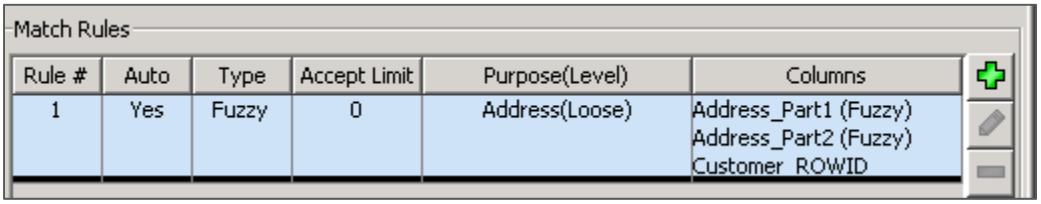
    The match rule is created as shown in the following image.





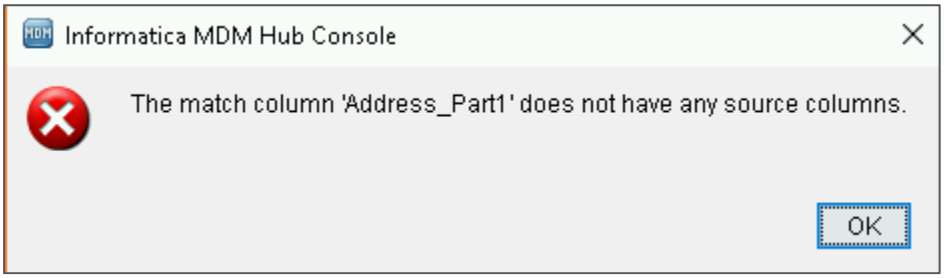
47. Set the match rule to **automerge** (**Auto = Yes**).



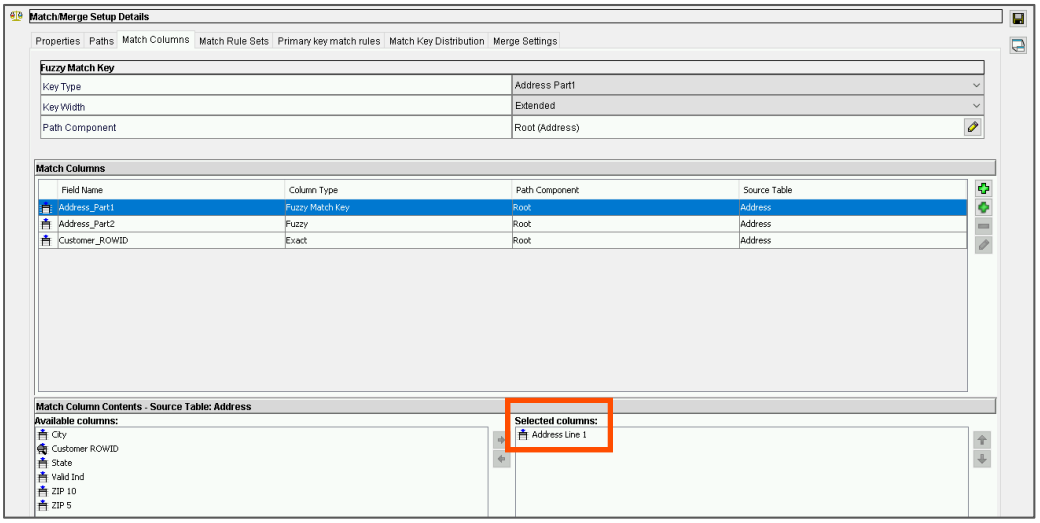


44. **Save** the changes.

Note: You may encounter an error:



If so, select **Address Line 1** column.



48. Navigate to the Batch Viewer tool and under the **Match** node, select the **Match for Address** job.
49. Execute the job.

50. Verify that the results are as mentioned below:

    Note: The number of records may vary.

    Matched records: **1**

    Tokenized: **148**

    Queued for automerge: **1**

    Queued for manual merge: **0**

This concludes the lab.

ADDITIONAL INFORMATION

Configure the Match Purpose and Match Level

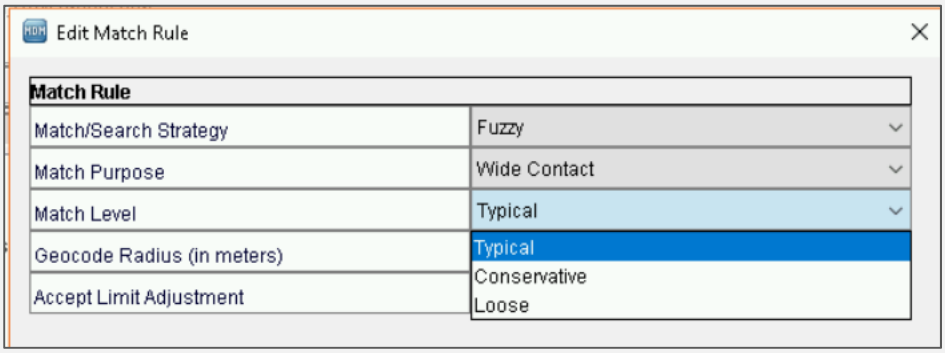
When you select some match columns, the **Match Purpose** attribute is enabled. The Match Purpose determines:

- How match rules behave
- Which columns you need
- How much weight MDM puts on each of the match columns used in the match rule.

See the product documentation as well as the Informatica Customer Support website for details on the various match purposes.

You must use the **Match Level** along with the **Match Purpose**. Three Match Levels determine how similar the records should be before they are considered as a match:

- Typical – produces typical matches
- Conservative – requires records to be more similar
- Loose – allows a higher degree of variation among records



Module 9: Merge Process

Lab 9-1: Setting Merge Options for Match Rules

Overview:

Given a match rule, you can configure the rule for automatic match and merge functionality. With the configured match rule, you can run an automatic match and merge job. With a configured match rule and matched records, you can run an automatic merge job.

Objectives:

- Set merge options for match rules
- Configure and run the Auto Match and Merge job for a base object
- Run the Automerge job for a base object

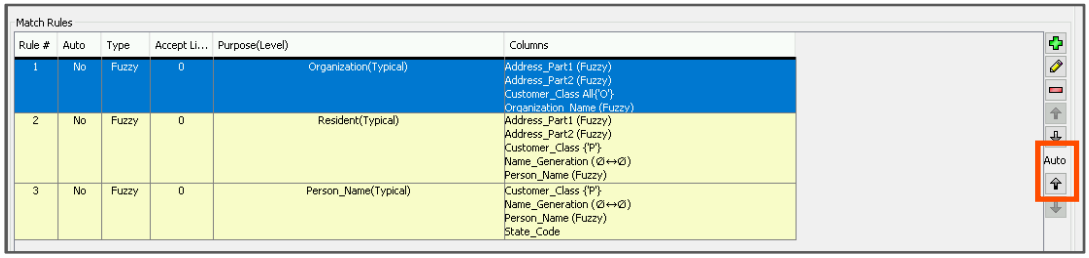
Duration:

35 minutes

Tasks

Configure Automerge for Match Rules

1. Navigate to the **Model** > **Schema** tool.
2. Ensure that you acquire a **Write Lock**.
3. Expand the **Customer** base object and select **Match/Merge Setup**.
4. Open the Match Rule Sets tab and select Main Rule Set.
5. Select the **Fuzzy** match rule with the **Organization** purpose level.
6. To set the match rule to merge automatically, click the **Auto**  button.



7. **Save** the Match Rule Set.

Run the Auto Match and Merge Job

8. Navigate to the **Batch Viewer** tool.
9. Under the **Auto Match and Merge** node, select the **Match and merge for Customer** job.

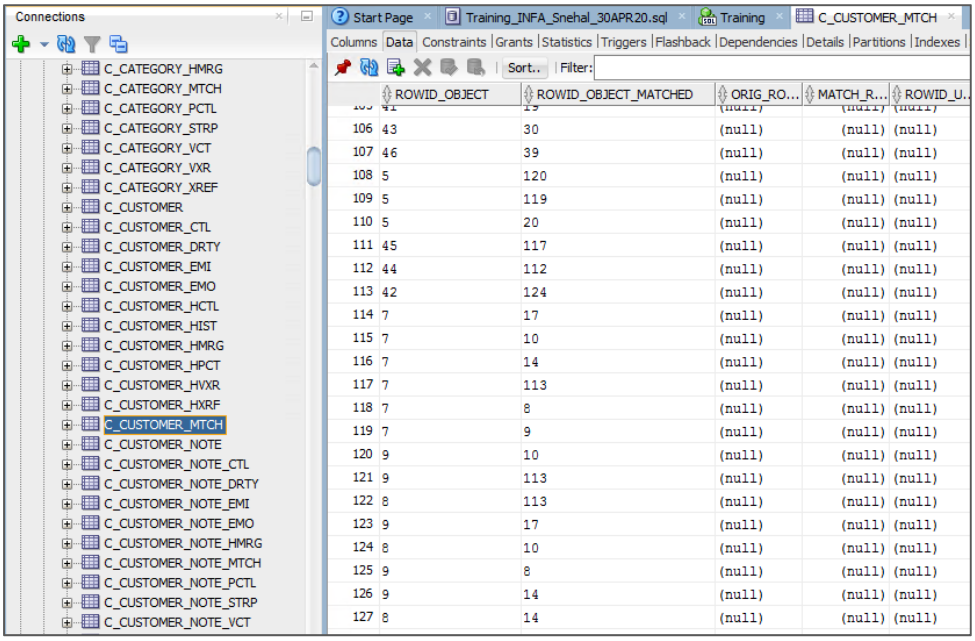
10. **Execute** the job.

    Note: The number of records may vary for you. Check for the “Match and Merge Successful” status and proceed.

Identity	
Name:	May 5, 2020 1:43:38 AM PDT
Description:	Match and merge for Customer
Source table:	Customer
Status	
Current status:	Match and Merge Successful.
Metrics	
Matched records	130
Records tokenized	1
Automerged records	0
Accepted as unique records	0
Queued for automerge	0
Queued for manual merge	130
Time	
Start:	May 5, 2020 1:43:38 AM PDT
Stop:	May 5, 2020 1:44:18 AM PDT
Elapsed time:	40 Secs.

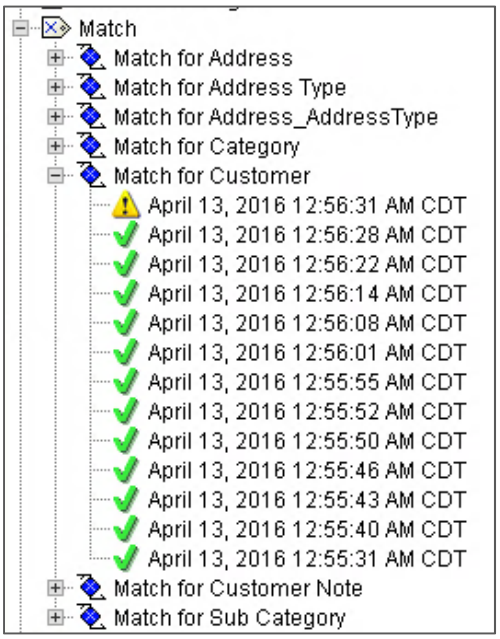
    Note: You can refresh the display with either the **Refresh Status** button or from the menu bar above **Batch Viewer** > **Refresh** command.

11. Open **SQL Developer**, and for the Customer base object, observe the **C_CUSTOMER_MTCH** table.



    Note: You should see the same number of rows in the match table as the number of matched records queued for manual merge from the Auto Match and Merge job.

12. In Batch Viewer, under the **Match** node, scroll down to the **Match for Customer** node, and expand it.



Note: There are several jobs listed as in the Customer Match/Merge Setup, the number of rows per match job batch cycle is set to 10.

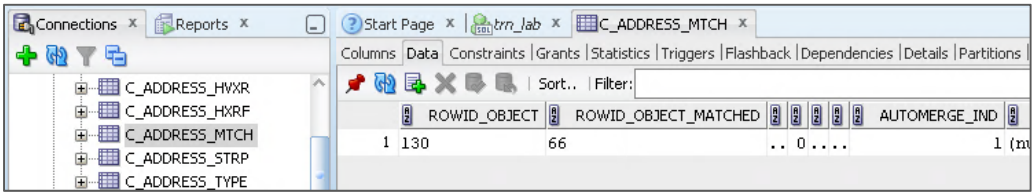
13. For the Customer base object, select and observe the results of a few of the Match jobs.

Run the Automerge Job

In the previous lab, you ran the Match job on the Address table that resulted in a single match.

Identity	
Name:	April 13, 2016 12:41:46 AM CDT
Description:	Match for Address
Source table:	Address
Status	
Current status:	"C_ADDRESS" - Match on ruleset "Main Rule Set"
Metrics	
Matched records	1
Records tokenized	148
Queued for automerge	1
Queued for manual merge	0
Time	
Start:	April 13, 2016 12:41:46 AM CDT
Stop:	April 13, 2016 12:41:52 AM CDT
Elapsed time:	6 Secs.

14. In **SQL Developer**, observe the resulting matches in the **C_ADDRESS_MTCH** table.



Note: Observe that the **AUTOMERGE_IND** value is set to **1**. This means that the record will be automatically merged when the Automerge job runs.

15. Click the **Batch Viewer** tool.
16. Under the **Automerge** node, select the **Merge data in Address** job.
17. **Execute** the job.

Identity	
Name:	April 13, 2016 1:16:14 AM CDT
Description:	Merge data in Address
Source table:	Address
Status	
Current status:	Completed successfully
Metrics	
Automerged records	1
Accepted as unique records	0
Records failed to be processed	0
Time	
Start:	April 13, 2016 1:16:14 AM CDT
Stop:	April 13, 2016 1:16:20 AM CDT
Elapsed time:	6 Secs.

Note: The matched records are automerged.

18. In **SQL Developer**, refresh the display and observe the **C_ADDRESS_MTCH** table. The record is no longer in the table as the records with the AUTOMERGE_IND that are set to 1 are removed from the table when you merge those.

This concludes the lab.

Module 10: Configure Data Access Views

Lab 10-1: Queries and Packages

Overview:

In the Informatica MDM Hub, a query is a request to retrieve data from the Hub Store. A package is a public view of one or more underlying tables in the Informatica MDM Hub. A package is based on a query that can select records from a table or from another package.

Queries and packages go together. Queries define the criteria for selecting data, and packages are views that users use to operate on that data. You can use a query in multiple packages.

Given a query and the Packages tool, you can create a simple query and use that query to create a PUT-enabled package to support data edits and merges. You can also create complex queries.

Objective:

- Create a simple query
- Create a PUT-enabled package to support data edits and merges
- Create a complex query

Duration:

40 minutes

Important

To execute this lab, you are provided with 2 options:

Option 1. You can execute the complete lab by following the instructions provided.

Option 2. You can import the completed queries and packages from another ORS - **Training_Complete**

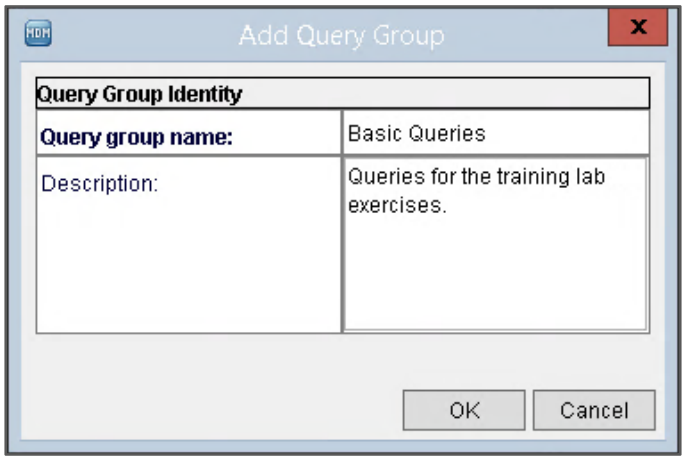
If you choose **option 2**, scroll to the end of this lab.

Option 1 Tasks

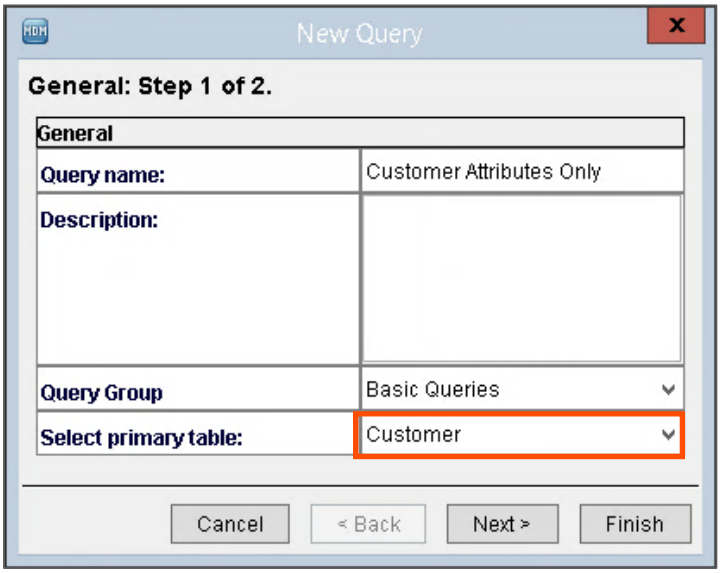
Create a Simple Query

1. From the **Model** workbench, select the **Queries** tool, and verify that you still have a **Write Lock**.
2. Right-click anywhere in the **Queries** pane and click **New Query Group**.
3. In the Query Group Name field, enter **Basic Queries**.

4. Optionally, enter a description.

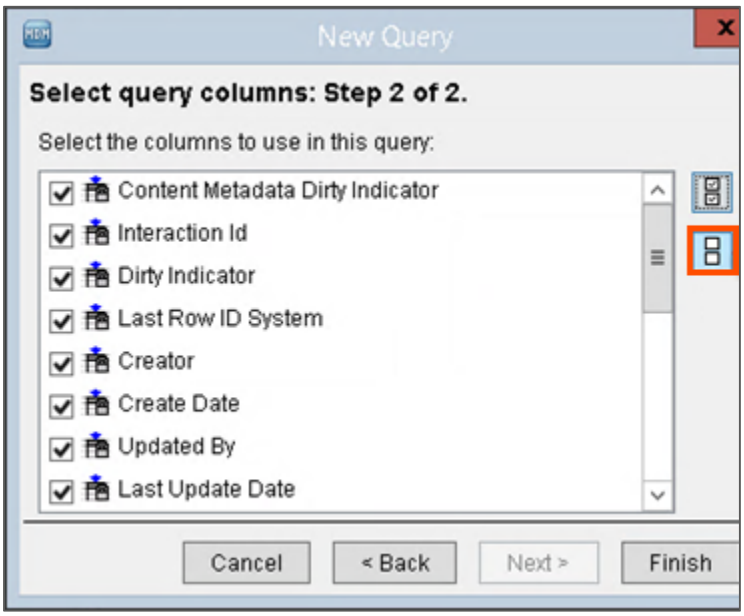


5. Click **OK**.
6. Right-click on the **Basic Queries** group and select **New Query**.
7. Click **Next** and in the Query name field, enter **Customer Attributes Only**.
8. From the Query Group drop-down, ensure that **Basic Queries** is selected.
9. As the primary table, select **Customer**.

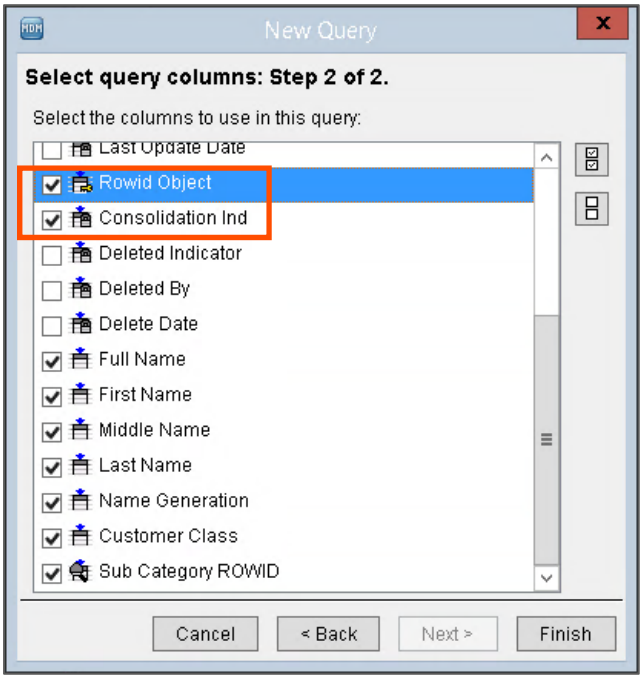


10. Click **Next**.

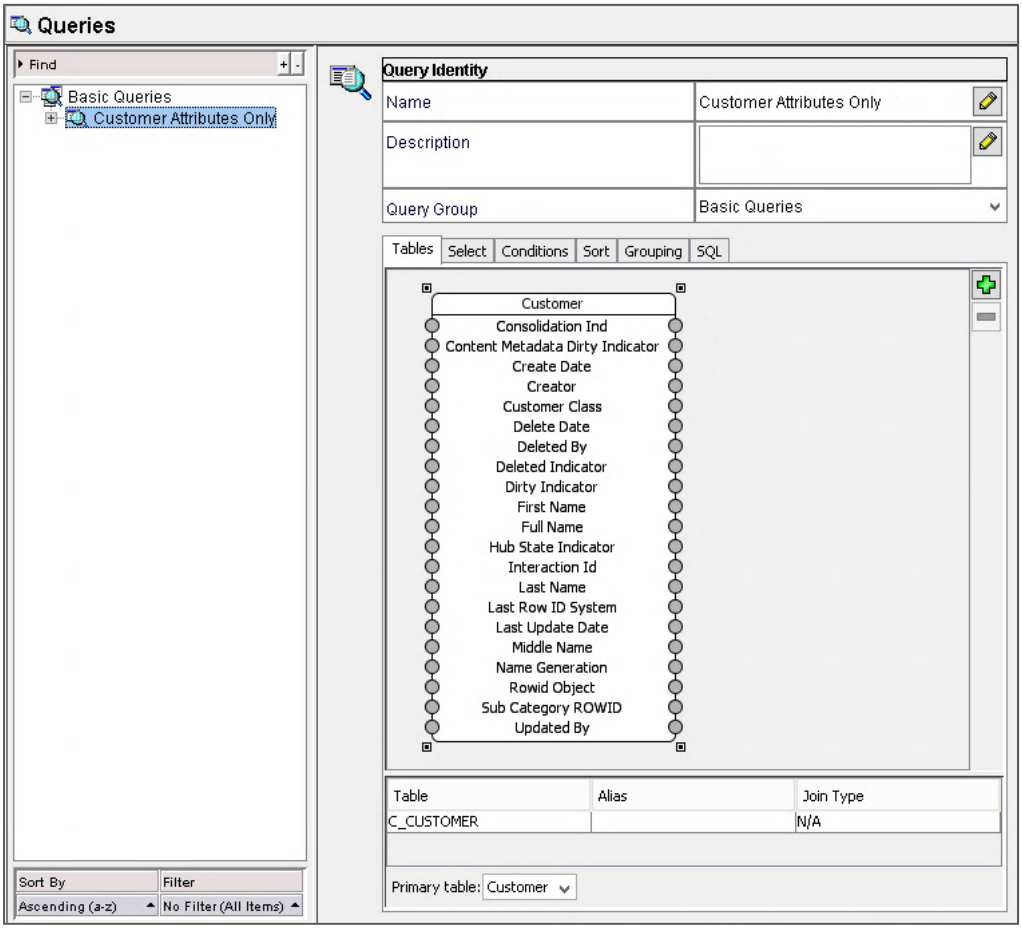
11. To clear all the options, click the **Select None** button.



12. Scroll down and select all the user-defined columns along with the **Rowid Object** and **Consolidation_Ind** system columns.

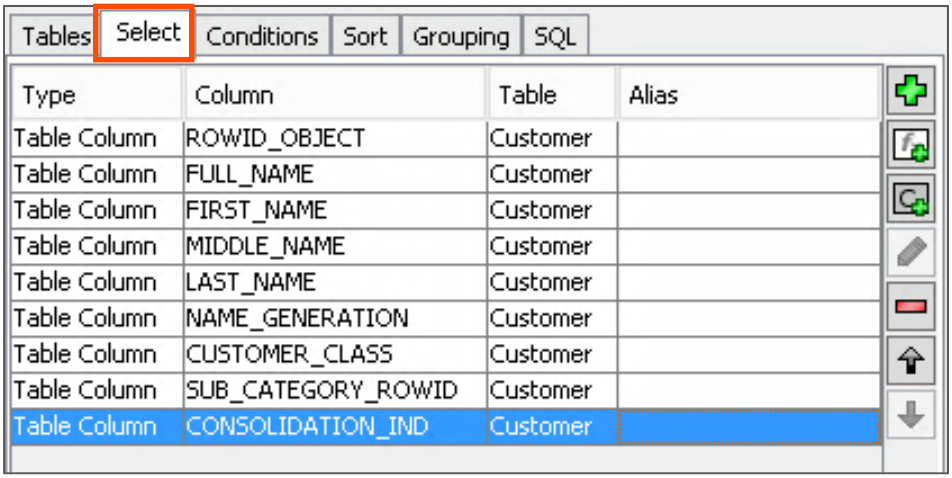


13. Click **Finish**.



Note: The Tables tab diagram shows all the columns present in the table.

14. To verify that the correct columns for the query are selected, open the **Select** tab. If necessary, use the **Add** and **Remove** buttons to correct the column selection.



15. Select the **Consolidation_Ind** column, and with the help of the Move Down button on the right, move the attribute to the end of the list.
16. **Save** your work.

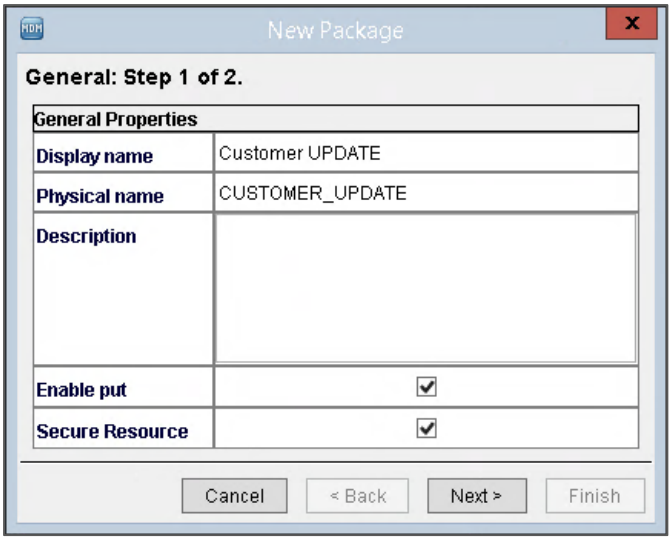
Create a PUT-enabled Package to Support Data Edits and Merges

You must use a PUT-enabled package either when you execute an SIF put request, that inserts or updates records, or when you use Merge Manager and Data Manager tools in the MDM Hub Console. As you will use these tools in the upcoming sections to perform manual merge and unmerge, you must define them here.

17. From the Model workbench, select the **Packages** tool.
18. Right-click anywhere in the Packages pane and select **New Package**.
19. Click **Next** in the wizard.
20. Enter **Customer UPDATE** as the display name for the package.

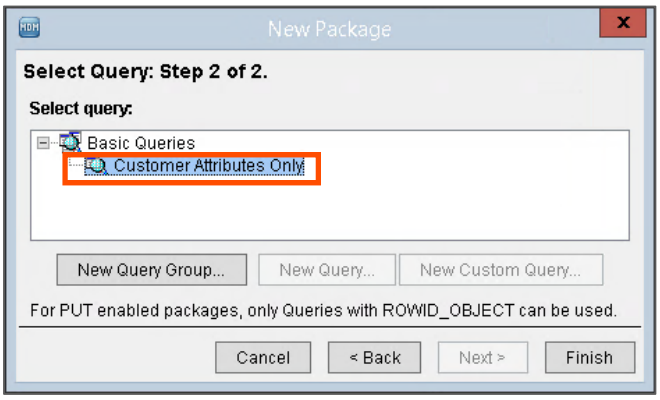
    The **New Package Wizard** automatically suggests a physical name when you click the field or press the <Tab> key.
21. Click the **Enable PUT** option to enable inserting or updating of records in the base object table.

    Note: PUT-enabled packages cannot include joins to other tables, it cannot be based on system tables or other packages, and on the queries that have constant columns, aggregate functions, or group by settings.
22. Ensure that you check the **Secure Resource** flag.



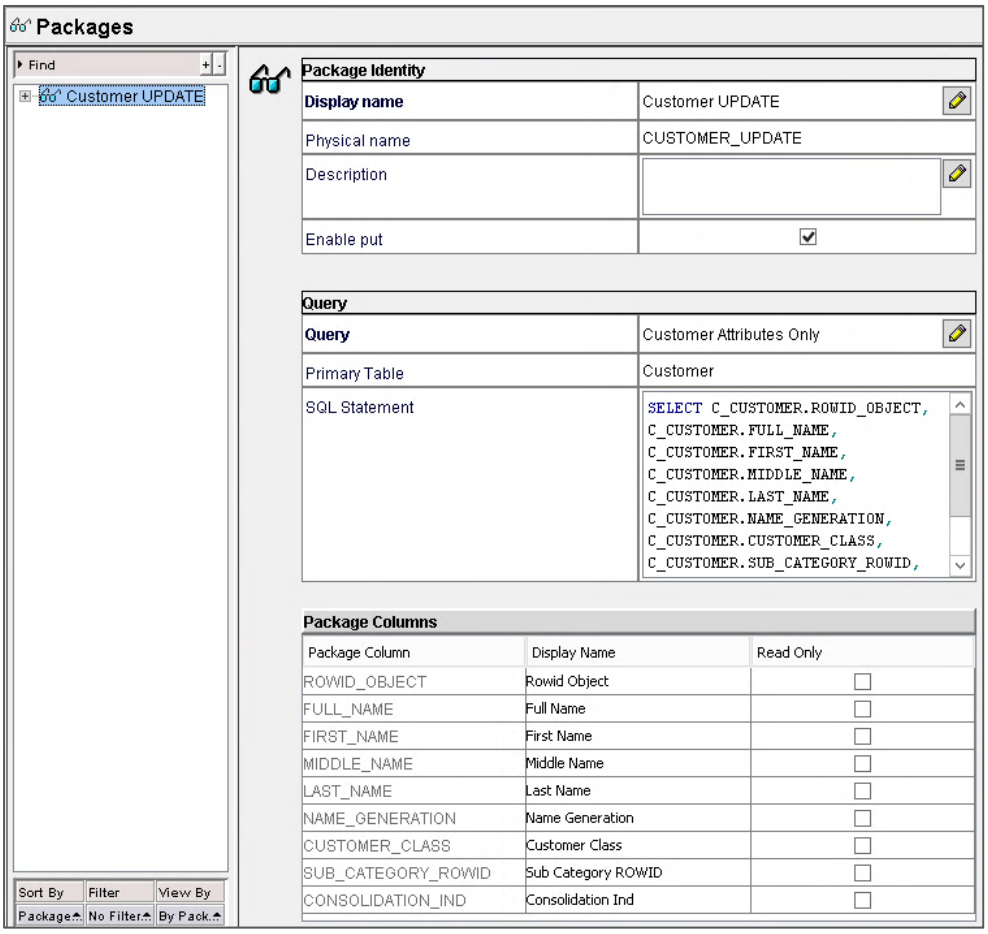
23. Click **Next**.

24. Expand the **Basic Queries** query group that you created previously and select the **Customer Attributes Only** query as the query for the new package.



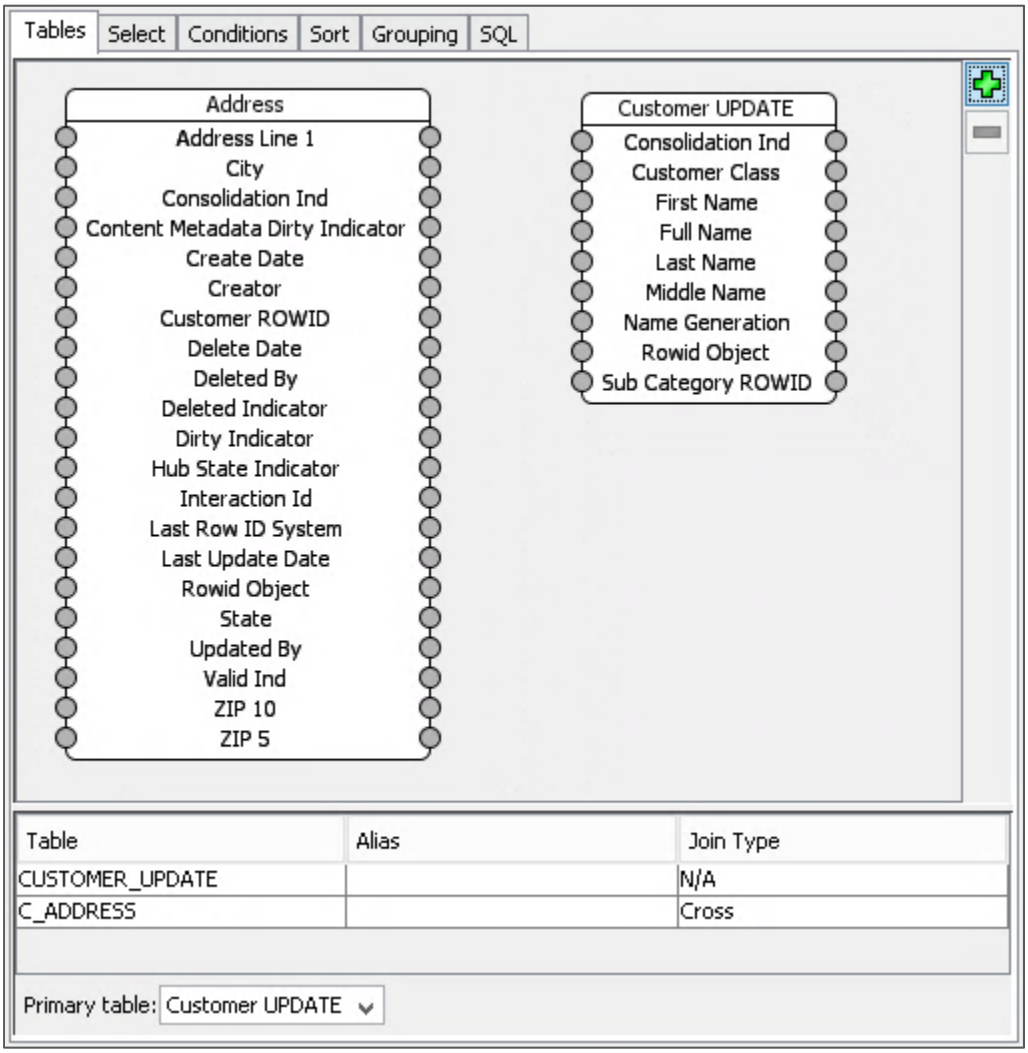
Note: You can also create a new query directly from the New Package Wizard.

25. Click **Finish**.

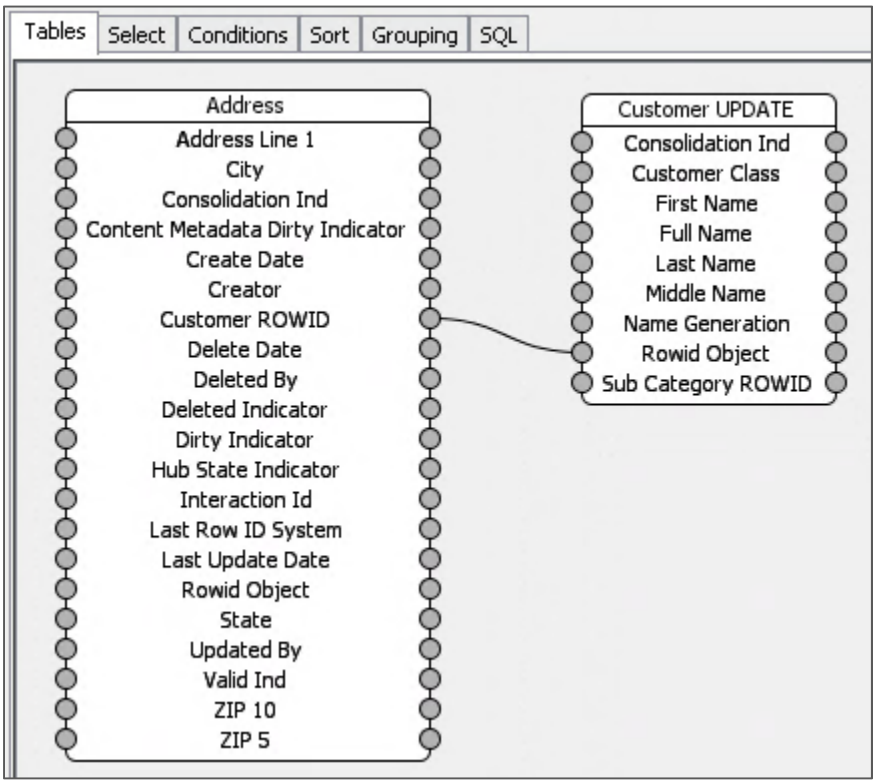


Create a More Complex Query

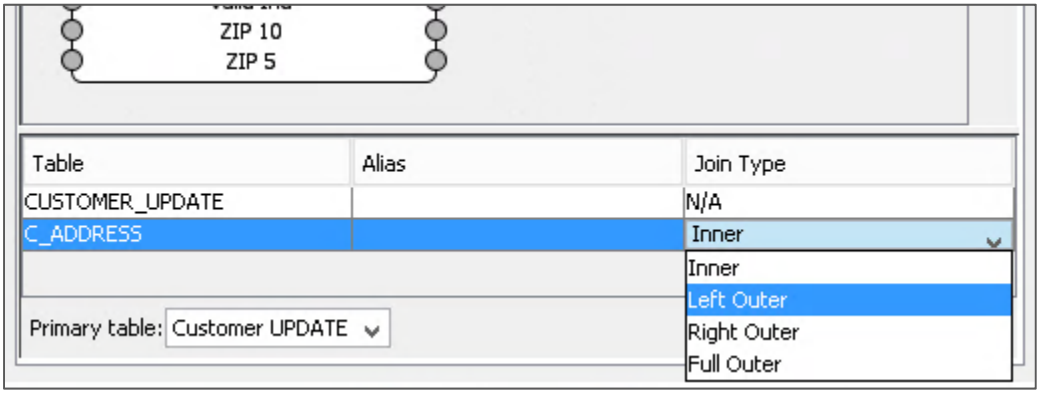
26. Return to the Queries tool.
27. Right-click on the Basic Queries group and choose **New Query**.
28. Click **Next**.
29. Enter **Customer Address** as the query name.
30. Choose **Basic Queries** as the **Query Group**.
31. Select **Customer UPDATE** package as the **Primary Table**.
32. Click **Next**.
33. Select all columns from the column list.
34. Click **Finish**.
35. In the **Tables** tab, click on the **Add Table** button (+) and add the **Address** table to the query.
36. The **Address** table is shown on the **Tables** tab. The added table's icon is always dropped on the left side of the canvas, so you will have to drag the table icon to reveal both.



37. Manually join the Address table's **Customer_ROWID** column to the **Rowid Object** column of the Customer UPDATE package.



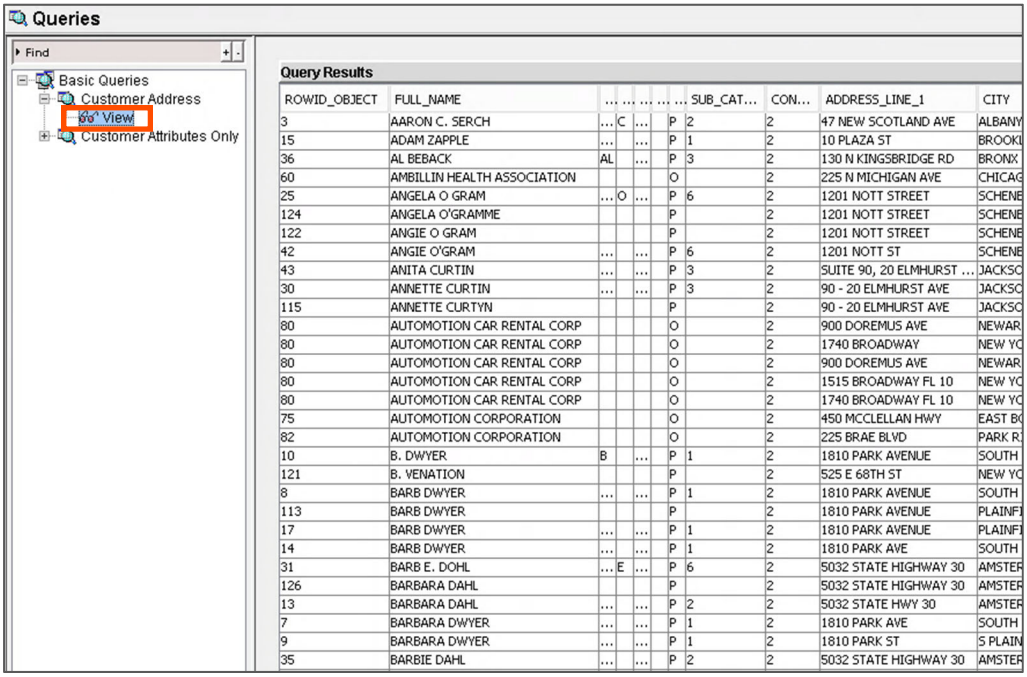
38. To make sure that the query returns all **Customer** records, even ones that do not have an **Address** record, click the **Join Type** column for **C_ADDRESS** and select **LEFT OUTER** as the join type.



39. Click the **Select** tab.
40. Click the **Add Table Column** button (**+** button) to add more columns to the query.
41. In the **Add Table Column** dialog, expand **Address** and select the following columns: **Address Line 1**, **City**, **State**, **ZIP 10**, and **Valid Ind**.
42. Click **OK**.

    For reference: Alias values for the columns may be defined if that will help improve the user interface experience. Double-click the attribute to enter the edit mode.
43. Go to the **Sort** tab.

44. Click the **+** button to add a sort to the query results.
45. Select **Full Name** from **Customer UPDATE** as the table column to sort on.
46. **Save** the query.
47. Click the **View** node under **Customer Address** in the Queries tree. This allows you to review the result of the query.



ROWID_OBJECT	FULL_NAME	...	...	...	...	...	SUB_CAT...	CON...	ADDRESS_LINE_1	CITY
3	AARON C. SERCH	...	C	...		P	2	2	47 NEW SCOTLAND AVE	ALBANY
15	ADAM ZAPPLE	...		...		P	1	2	10 PLAZA ST	BROOKL
36	AL BEBACK	AL		...		P	3	2	130 N KINGSBRIDGE RD	BRONX
60	AMBILLIN HEALTH ASSOCIATION					O		2	225 N MICHIGAN AVE	CHICAG
25	ANGELA O GRAM	...	O	...		P	6	2	1201 NOTT STREET	SCHENE
124	ANGELA O'GRAMME					P		2	1201 NOTT STREET	SCHENE
122	ANGIE O GRAM					P		2	1201 NOTT STREET	SCHENE
42	ANGIE O'GRAM	...		...		P	6	2	1201 NOTT ST	SCHENE
43	ANITA CURTIN	...		...		P	3	2	SUITE 90, 20 ELMHURST ...	JACKSO
30	ANNETTE CURTIN	...		...		P	3	2	90 - 20 ELMHURST AVE	JACKSO
115	ANNETTE CURTYN					P		2	90 - 20 ELMHURST AVE	JACKSO
80	AUTOMOTION CAR RENTAL CORP					O		2	900 DOREMUS AVE	NEWAR
80	AUTOMOTION CAR RENTAL CORP					O		2	1740 BROADWAY	NEW YO
80	AUTOMOTION CAR RENTAL CORP					O		2	900 DOREMUS AVE	NEWAR
80	AUTOMOTION CAR RENTAL CORP					O		2	1515 BROADWAY FL 10	NEW YO
80	AUTOMOTION CAR RENTAL CORP					O		2	1740 BROADWAY FL 10	NEW YO
75	AUTOMOTION CORPORATION					O		2	450 MCCLELLAN HWY	EAST BO
82	AUTOMOTION CORPORATION					O		2	225 BRAE BLVD	PARK R
10	B. DWYER	B		...		P	1	2	1810 PARK AVENUE	SOUTH
121	B. VENATION					P		2	525 E 68TH ST	NEW YO
8	BARB DWYER	...		...		P	1	2	1810 PARK AVENUE	SOUTH
113	BARB DWYER					P		2	1810 PARK AVENUE	PLAINFI
17	BARB DWYER	...		...		P	1	2	1810 PARK AVENUE	PLAINFI
14	BARB DWYER	...		...		P	1	2	1810 PARK AVE	SOUTH
31	BARB E. DOHL	...	E	...		P	6	2	5032 STATE HIGHWAY 30	AMSTER
126	BARBARA DAHL					P		2	5032 STATE HIGHWAY 30	AMSTER
13	BARBARA DAHL	...		...		P	2	2	5032 STATE HWY 30	AMSTER
7	BARBARA DWYER	...		...		P	1	2	1810 PARK AVE	SOUTH
9	BARBARA DWYER	...		...		P	1	2	1810 PARK ST	S PLAIN
35	BARBIE DAHL	...		...		P	2	2	5032 STATE HIGHWAY 30	AMSTER

In the walkthrough of this lab, you:

- Created a simple query
- Used the query to create a PUT-enabled package to support data edits and merges
- Created a more complex query

Create Four Packages Based on the Four Queries

In this section of the lab, use what you've learned in the preceding walkthroughs to complete the Skills Application exercise.

48. Create a query called **Customers with Notes**. The primary table for the query is **Customer UPDATE**. Add the **Customer Note** base object (in the Tables tab) to the query and manually create the join from **Customer Note** (Customer_ROWID) to **Customer UPDATE** (Rowid_Object).

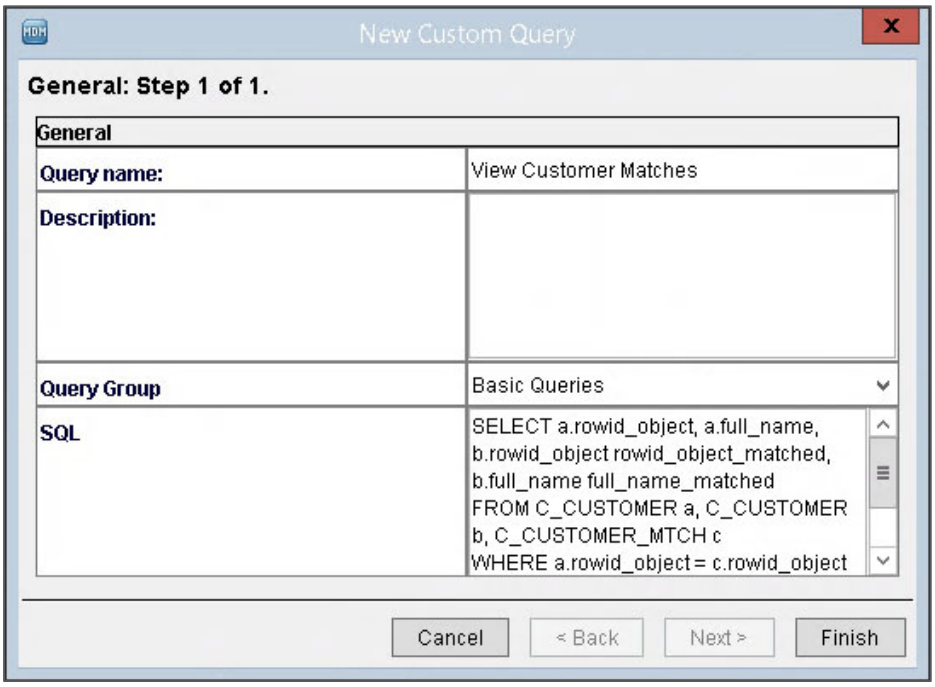
Leave the join as an INNER join, as we only want to show customers with entries in the **Customer Notes** table.

Ensure that the query has Rowid Object, Full Name, and Customer Class columns from **Customer UPDATE** as well as the Note Text column from **Customer Note** (you will have to add this if it is not there).

You can use the **View** option to review the data that the query returns. If the view does not show you the columns you expect, then click **Queries** > **Refresh** from the main menu.

49. Create a package called **Customer Address READ**. This package is based on the **Customer Address** query you created in the previous walkthrough. This package is not PUT-enabled.
50. Create a package that uses the **Customers with Notes** query – name the package as **Customers with Notes READ**.
51. Create a new custom query called **View Customer Matches**. The SQL for this query is:

```
SELECT a.rowid_object, a.full_name, b.rowid_object
rowid_object_matched, b.full_name full_name_matched
FROM C_CUSTOMER a, C_CUSTOMER b, C_CUSTOMER_MTCH c
WHERE a.rowid_object = c.rowid_object
AND b.rowid_object = c.rowid_object_matched
AND a.rowid_object <> b.rowid_object
ORDER BY a.rowid_object, b.rowid_object
```



52. Use the **View** option to review the results of the query.

53. Create a package called **Customer Matches READ** that uses the previous query.

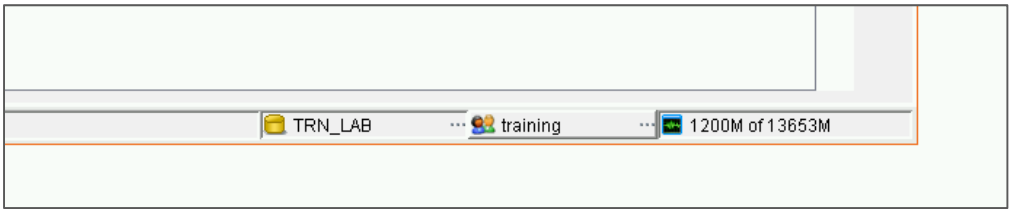
In the walkthrough for this lab, you:

- Created four packages all based on four queries that you built. One of the packages you created was a PUT-Enabled package.

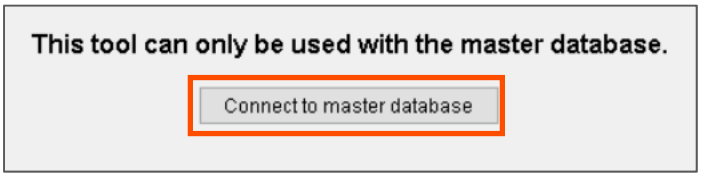
Option 2 Tasks – Import Queries and Packages from Training_Complete ORS

Log in as the administrator and connect to the **Training_Complete** ORS.

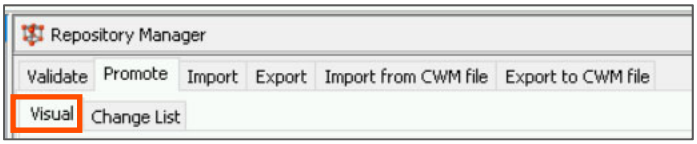
1. At the right-hand bottom of the Hub console, click **training** and enter **admin/admin**.
2. Select the repository as **Training_Complete**.



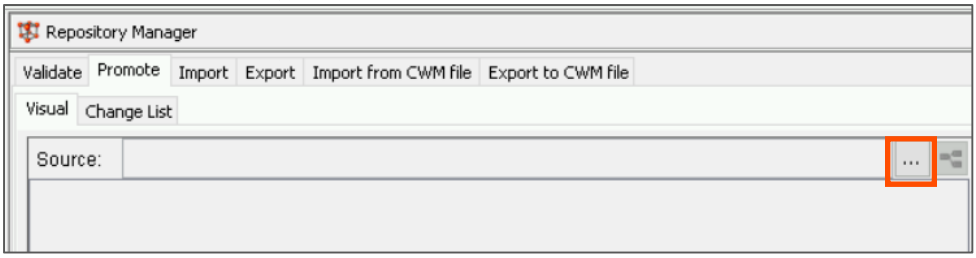
3. Click **Configuration** > **Repository Manager**.
4. Click **Connect to master database**.
   Note: Repository Manager works only with **MDM Hub Master Database**.



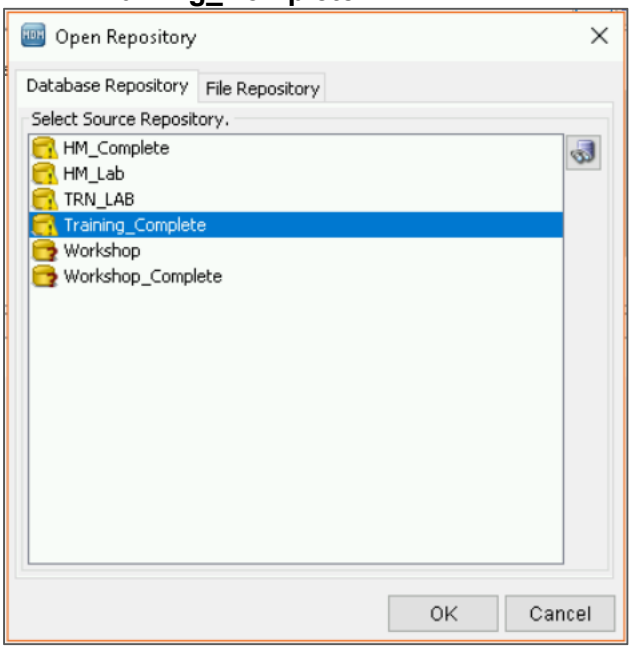
5. Click the **Promote** > **Visual** tab.



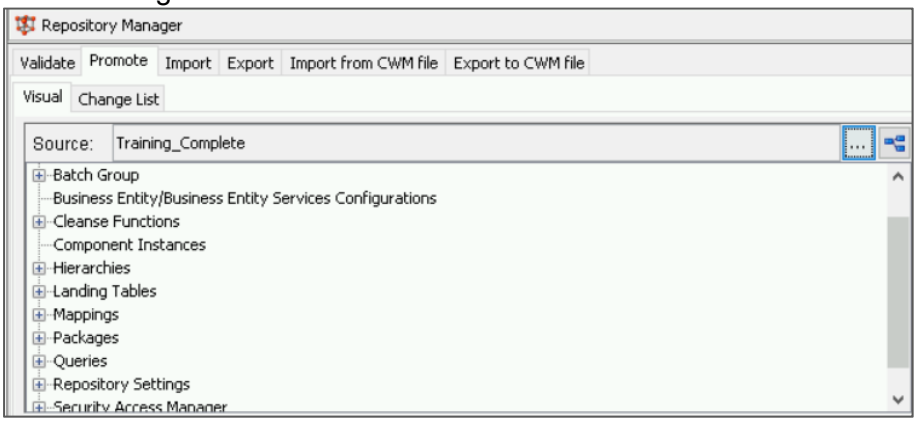
6. To select the **Source**, click the 3 dots button.



7. Select **Training_Complete** as the source and click **OK**.



The source gets loaded.

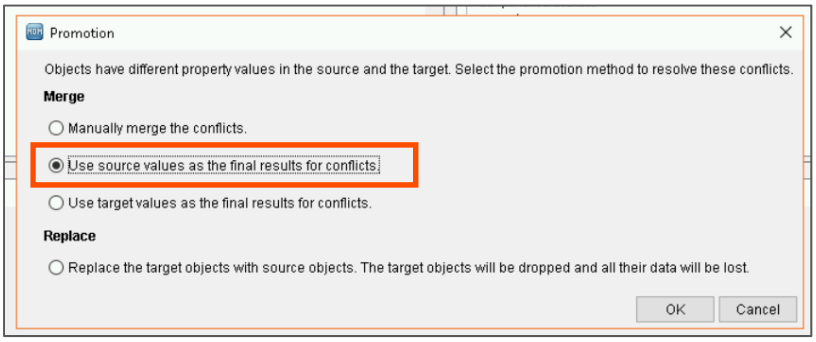


8. Click the **Target** drop down, and select **TRN_LAB**.

9. Select **Queries** and **Packages** from the Source.

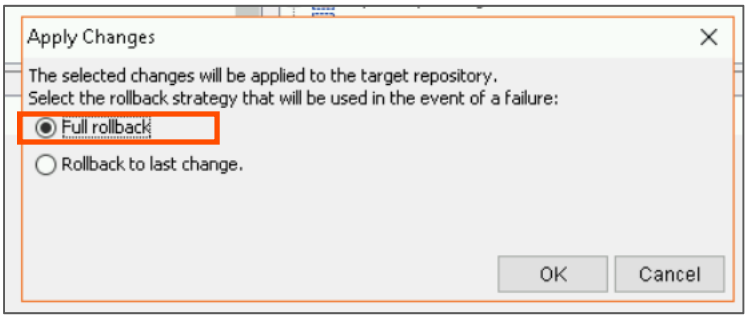
10. Click the **Promote** icon.

11. Click **Use source values as the final results for conflicts**.



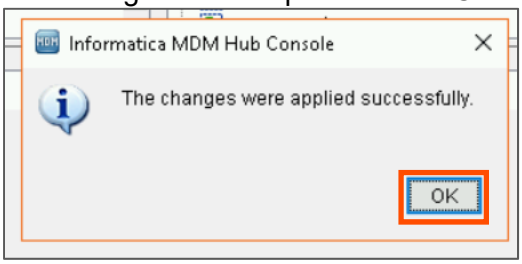
12. Click **OK** in the Impact Analyzer window.
13. Click the **Apply Changes** button.

14. Select **Full rollback** and click **OK**.

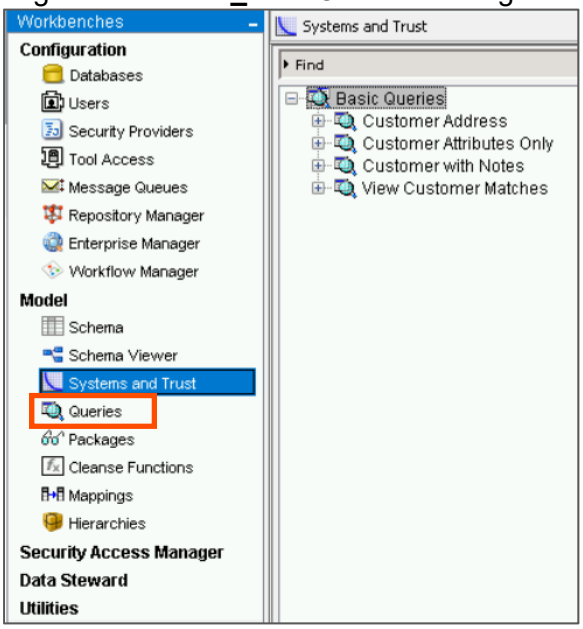


15. Click **NO** for the **Validate integrity check**.

16. The changes will be updated now. Click **OK**.



17. Log in to the **TRN_LAB** ORS and navigate to **Model** > **Queries**.



18. Notice that the **Queries** and **Packages** are imported successfully from the **Training_Complete** ORS.

This concludes the lab.

Module 11: Configure Batch Processes and CLI Support

Lab 11-1: Configure Batch Groups

Overview:

Configure Batch Jobs to execute the stage, load, match, and merge processes from within Stored Procedures.

In this lab, you will create two batch groups to execute STAGE and LOAD jobs from the Sales and CRM systems.

Objective:

- Create a batch group
- Add levels to a batch group
- Add jobs to a level

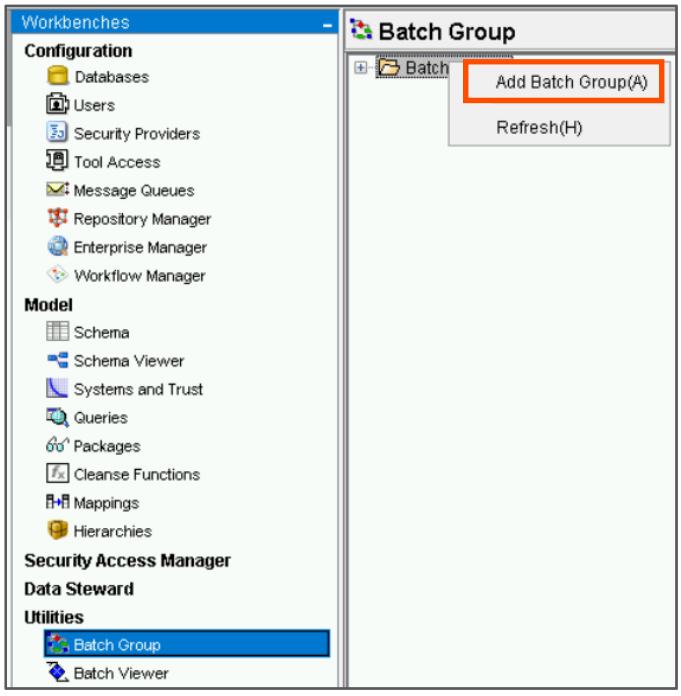
Duration:

15 minutes

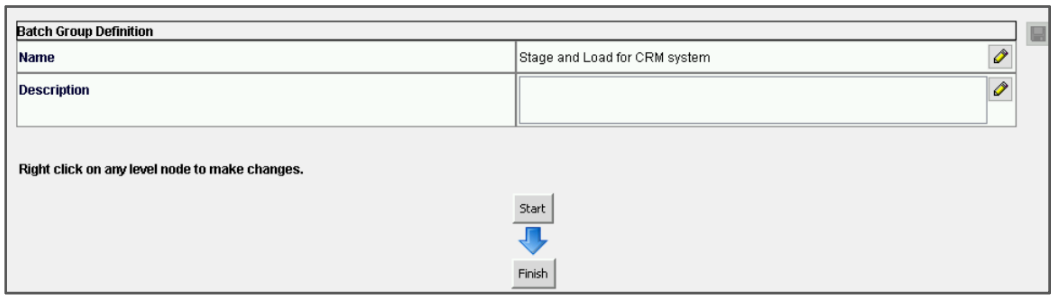
Tasks:

Create a Batch Group and add Levels and Jobs to the Batch Group

1. Select the **Batch Group** tool from the **Utilities** workbench and verify that you still have a **Write Lock**.
2. Right-click on the **Batch Groups** list and click **Add Batch Group**.



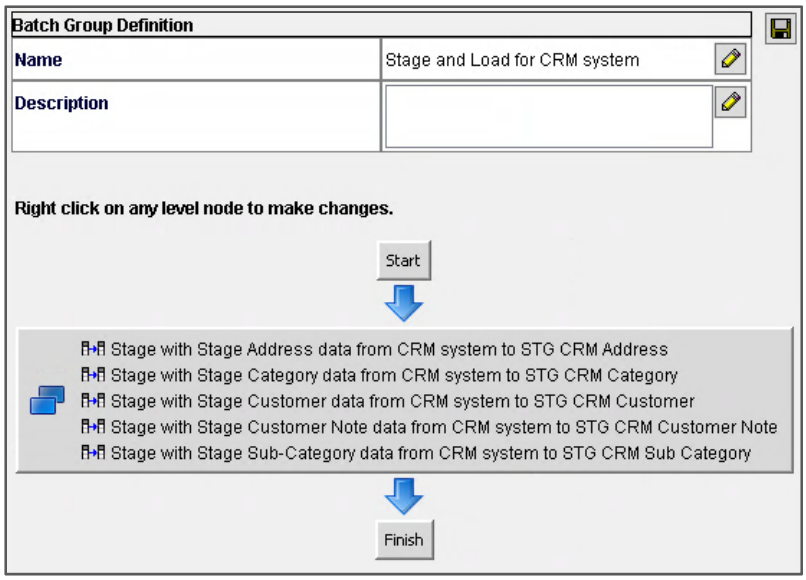
3. Name the new batch group as **Stage and Load for CRM system**.



4. Right-click on the **Start** button and choose **Add level below**.

The **Choose jobs to add to Batch Group** dialog appears.

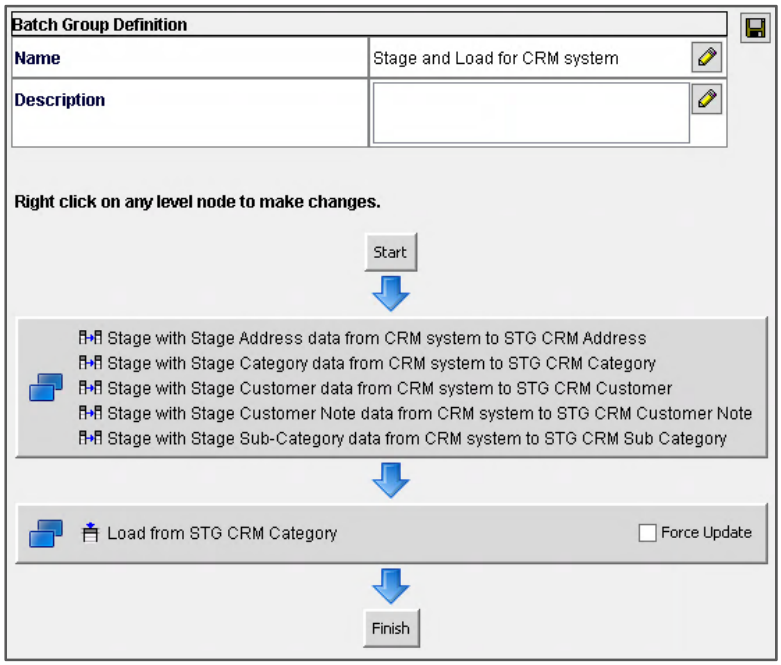
5. Select the **Stage jobs** for the **CRM system** for the following base objects:
   Address, **Category**, **Customer**, **Customer Note**, and **Sub Category**.
   To select the multiple jobs, you can Ctrl+Click the jobs in the selection window.
   Note: Do not select the Shipping and Billing staging jobs.
6. Click **OK**.
   The selected stage jobs are added to the same level in the batch group. This means that you can execute those in parallel. The order of the jobs within a batch group level makes no difference.



7. Right-click on the level you have just added and select **Add level below**.
8. Select the **load job** that loads the **Category** base object from the CRM system.

   You can see the load job that you just added in the same level. This means it will execute once all the jobs from the previous level are complete.

   Also, observe that when an additional setting is available for a job, such as **Force Update** for a Load job, you can set it in the batch group as well, if required.



In the walkthrough of this lab, you:

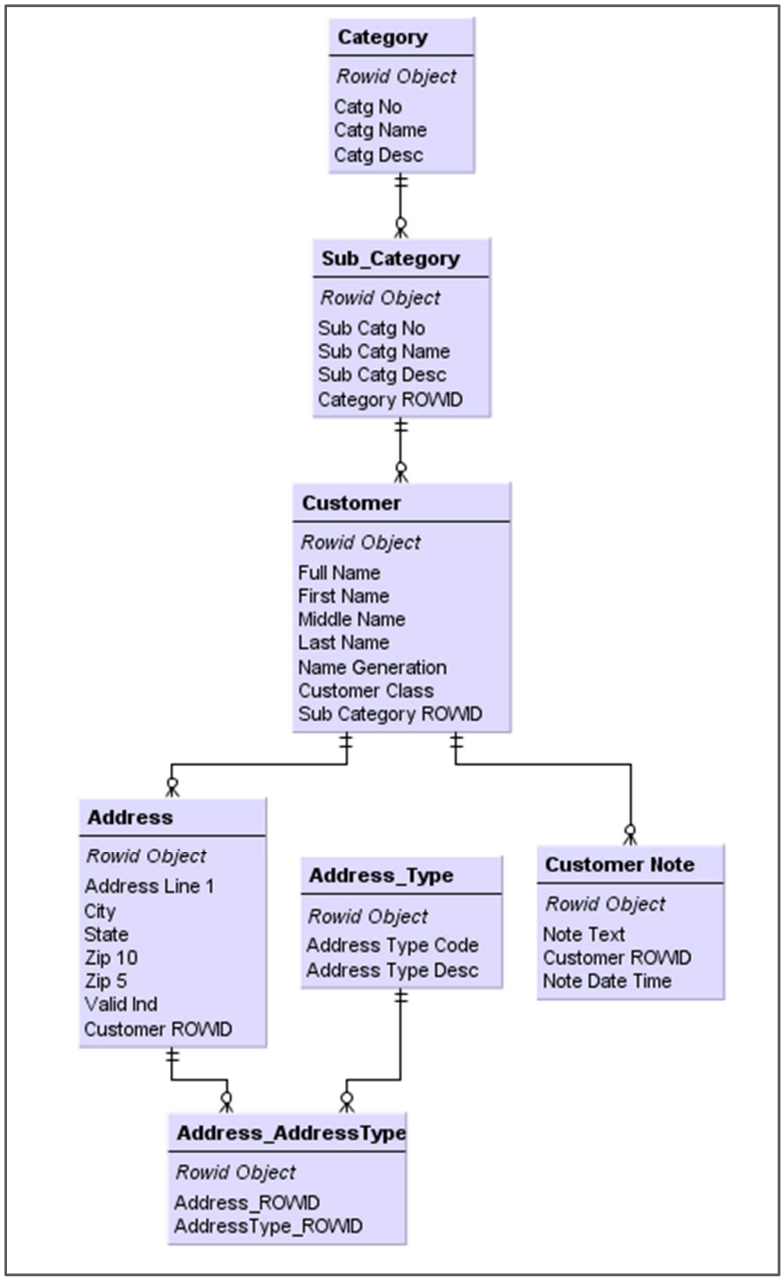
- Created a batch group
- Added levels to a batch group
- Added jobs to a level

Create Two Batch Groups to Execute STAGE and LOAD Jobs

In this section of the lab, use what you have learned in the preceding walkthroughs to complete the Skills Application exercise. You will continue the same batch group and add levels in this section.

Hint: A solution screenshot is provided for your reference.

Complete the batch group from the preceding walkthrough so that it executes the **Load jobs** for the **Address**, **Customer**, **Customer Note**, and **Sub Category** base objects. Consider the dependencies between these objects when you choose the order to execute the Load jobs.

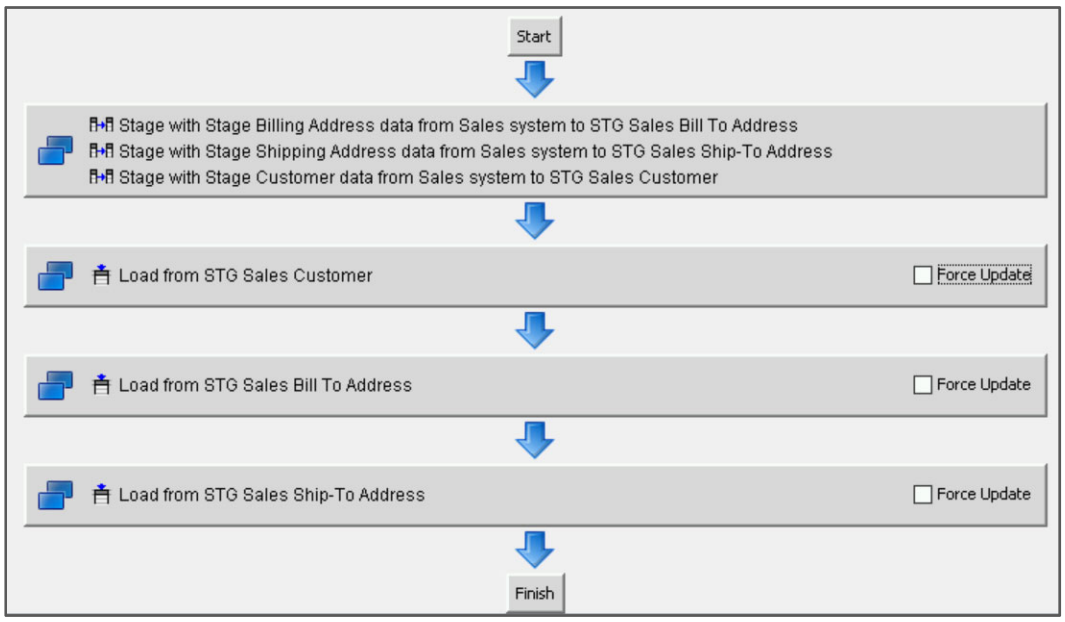


9. Continue the batch group and **add the levels** you see in the screenshot.

   Did you observe that you can load the Address records and the Customer Note records in parallel and at the same level? That is because those two are dependent on Customer but have no dependency on each other.

   Here is the **solution**:

10. **Save** your work.
11. Create another batch group named **Stage and Load for Sales system** that will execute the Load and Stage jobs for records from the Sales system.
    The end batch group should result in the following:



Important: **Do not run the batch groups!** You have already staged and loaded the data. However, in an actual implementation when you configure and test the individual jobs batch groups like the ones above, it can make your day-to-day operations much more efficient.

This concludes the lab.

Module 12: Utilize Data Management Tools

Lab 12-1: Merging Records Using Merge Manager

Overview:

You can use the Merge Manager data management tool in the Informatica Hub to manually search, edit, flag, and merge records queued for manual merging in the base object.

In this lab, you will manually edit and merge records using Merge Manager.

Objective:

- Assign records to user assignment list
- Flag records as unique
- Merge records and apply a trust override
- Manually search for matches
- Flag records for automerge
- Determine which match rule caused a match

Duration:

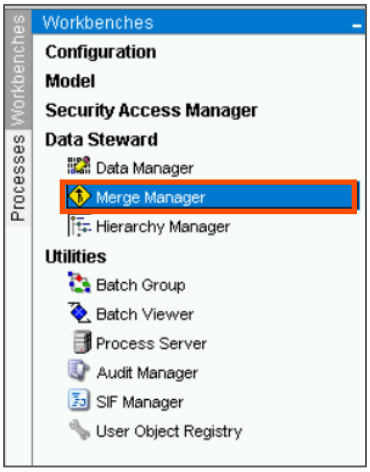
40 minutes

Note
Throughout this and the next exercise, it is possible that your Rowid_Object numbers, attribute values, and column ordering may vary from those presented here, due to possible variations in your mappings, cleansing functions, query, and package definitions. For the purpose of instruction, it is perfectly fine for you to pick records that are similar to those called out in the lab exercises and perform the required actions.

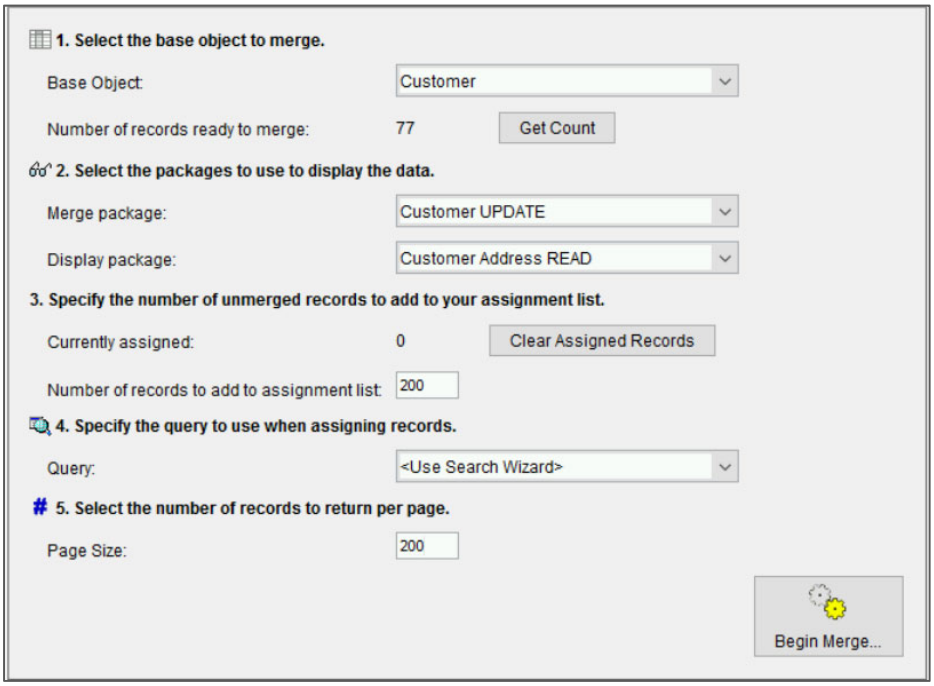
Tasks

Assign Records to User Assignment List

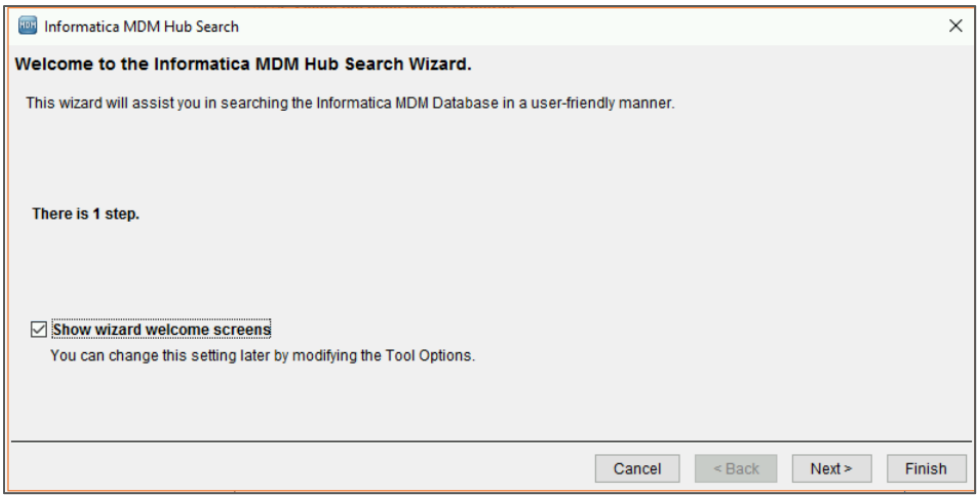
1. Select the **Merge Manager** tool in the Data Steward workbench.



2. In the **Merge Manager** setup screen, select **Customer** as the base object.
3. Click **Get Count** to review the number of records ready to merge.
4. Select **Customer UPDATE** as the merge package (also known as a PUT-enable package) that you can use to merge or update the records.
5. Select **Customer Address READ** as the display package that you can use to display the records.
6. Click on Clear Assigned Records if the number for Currently Assigned is not 0.
7. Change the Number of records to add to assignment list value and Page Size to 200.
8. Select **<Use Search Wizard>** from the **Query** drop-down list.



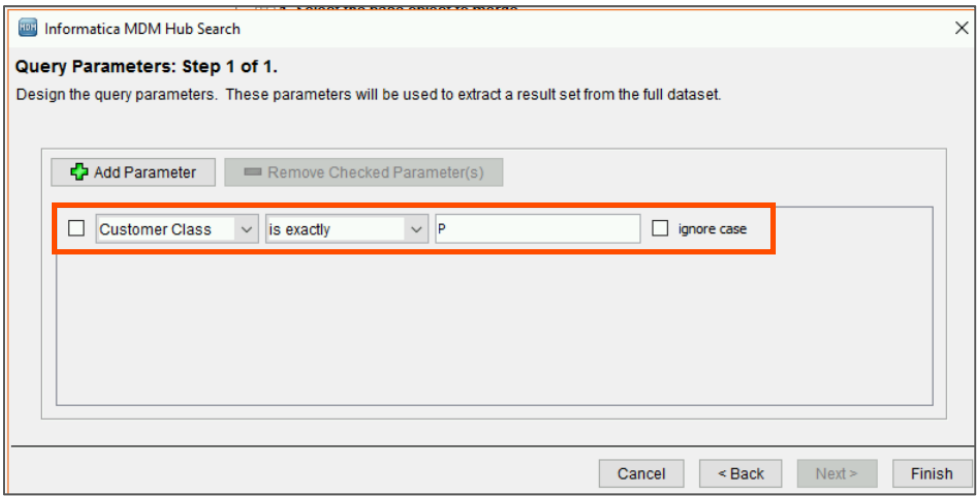
9. Click Begin Merge….
10. The **Search Wizard** is displayed.



11. Click **Next**.
12. Use the **+ Add Parameter** button to add a new parameter.

13. Set up the parameter value to read **Customer Class is exactly P**.

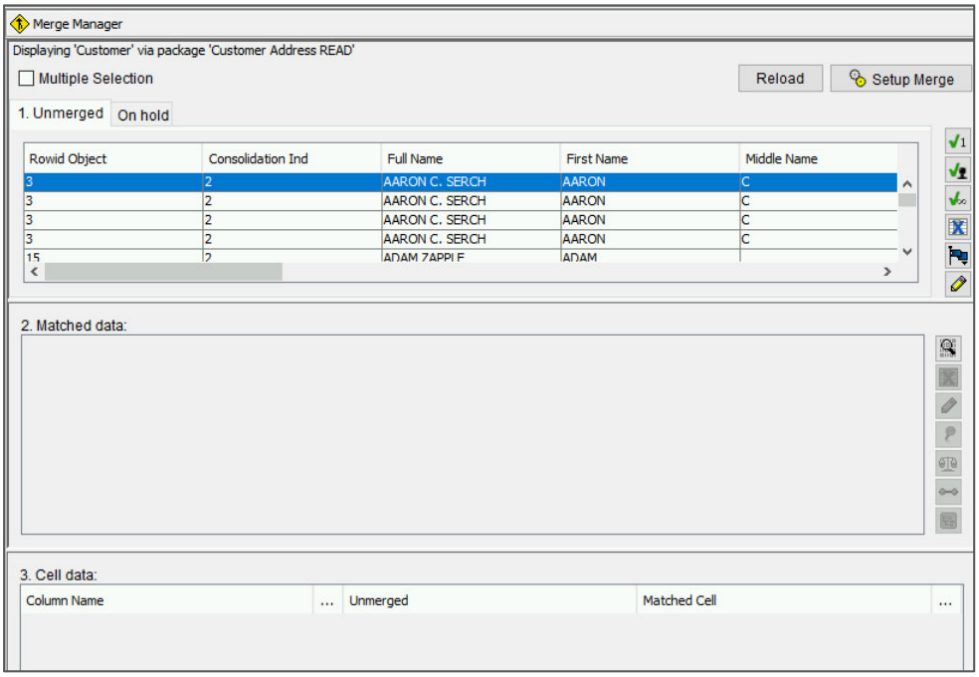
    This will restrict the records added to your assignment list to Persons only, that is, no records for Organizations or Regions.



    Note: If you add multiple parameters, you can use AND or OR conjunctions in Search Wizard queries, but do not mix them in the same query – stick to all AND or all OR, because the Search Wizard does not give you a way to specify precedence of the conditions.

14. Click **Finish** and the **Merge Manager** screen displays.

    Note: The rows and the results may vary for you. The Rowid Object field may also be different for you. Choose a convenient record to perform merge and proceed this lab.



Flag Record as Unique

15. In the **Unmerged** tab, at the top of the **Merge Manager** screen, select the record for ADAM ZAPPLE (Rowid_Object = 15).

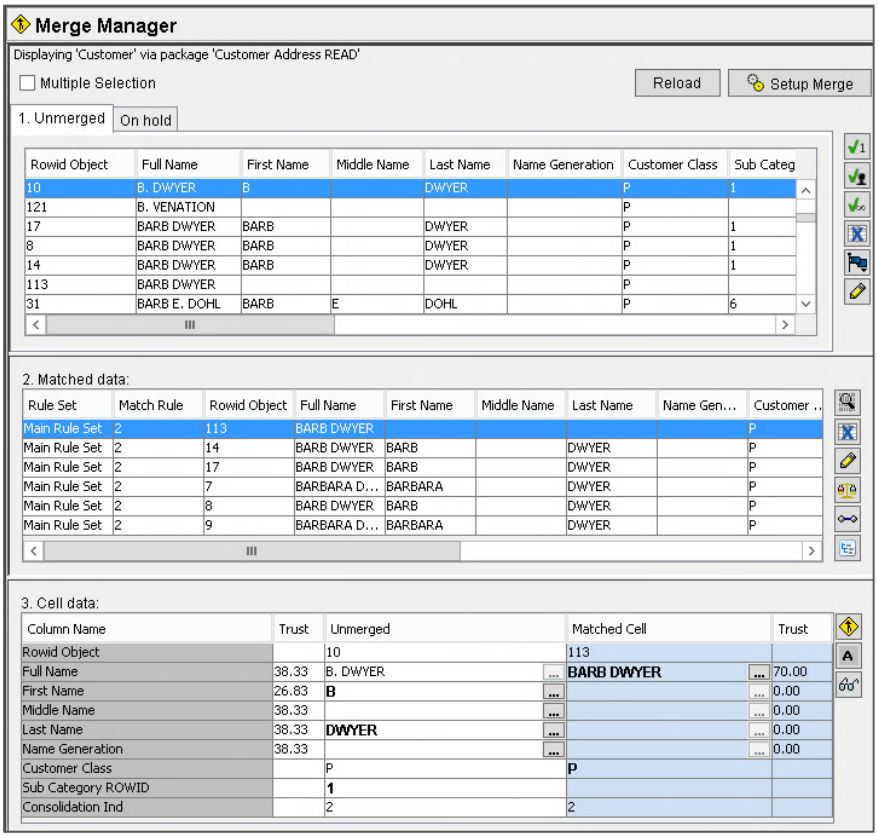
    Note: If ADAM ZAPPLE does not appear as unique in the data that is returned, choose a different unique record. Try STAN BAK or any other record that can appear to have duplicates, but duplicates don't show up in the Matched data section.

    Observe that no matches are found for this record in the **Matched data** section. Thus, you will mark it as unique to take it out of the merge queue.

16. Click the **Accept the unmerged record as unique** button.
17. Observe that ADAM ZAPPLE (or the earlier chosen record) is removed from the **Unmerged** list. Recall that the Merge manager only shows records with a consolidation Indicator of 2 and now ADAM ZAPPLE now has a Consolidation Indicator of 1.

Merge Records with a Trust Override

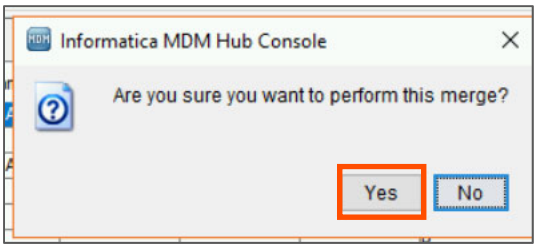
18. In the Unmerged list, scroll to the record for **B .DWYER** (Rowid Object 10)
19. Observe that several matches are found for this record that appear in the Matched Data pane. The matches returned may vary in your implementation.
20. Select one of the matches with the name **BARB DWYER** (Rowid Object 113) from the **Matched data** pane.



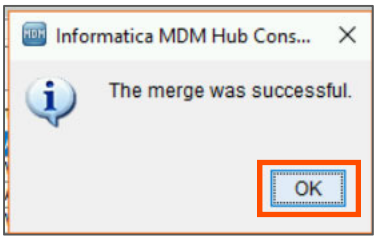
Note: The matches listed may vary.

The bold text in the **Cell Data** pane indicates which values can survive the merge. If the records are merged, the record that appears in the **Unmerged** pane will merge into the record in the **Matched Cell** pane. In the screenshot above, record 10 will merge into record 113, and 113 will be the surviving record.

21. Click the  button in the Cell Data pane to merge the two records.
22. Click **Yes** to perform the merge.



23. Click **OK**.



All the values that were bold, are now a part of the surviving record with Rowid_Object 113.

Observation

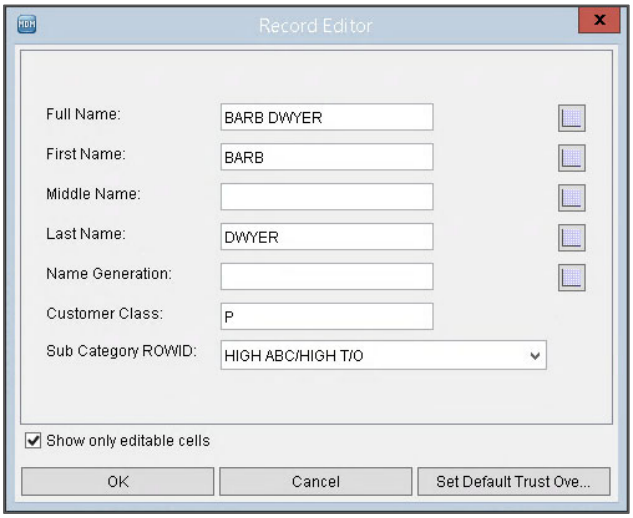
Notice that there are two records with Rowid_Object **113** in the **Unmerged** pane. This is just an artifact of the display package. Recall that the display package **Customer Address READ** is defined to display the child address records.

MDM preserves relationships when you merge the records. So, now that record 10 merged into record 113, and each of them have a child address record, 113 is now linked to both child address records. Thus, it appears twice in the pane, once for each address. If you scroll to the right, you can see the two slightly different addresses.

24. Merge all the matched records for BARB DWYER in the **Unmerged** section and always ensure that the surviving record (bolded) has the following values:
    - Full Name = **BARB DWYER**
    - First Name = **BARB**
    - Middle Name = <blank>
    - Last Name = **DWYER**
25. At the end of all the merges, the consolidated record should have **BARB DWYER** in the Full Name column, **BARB** in the First Name column and **DWYER** in the Last Name column.

26. How many BARB DWYER records show in the **Unmerged** list at the top of Merge Manager? Can you explain why?

    Note: If you end up with the wrong surviving values in any of the fields, click the **Edit** button  to open the **Record Editor**.



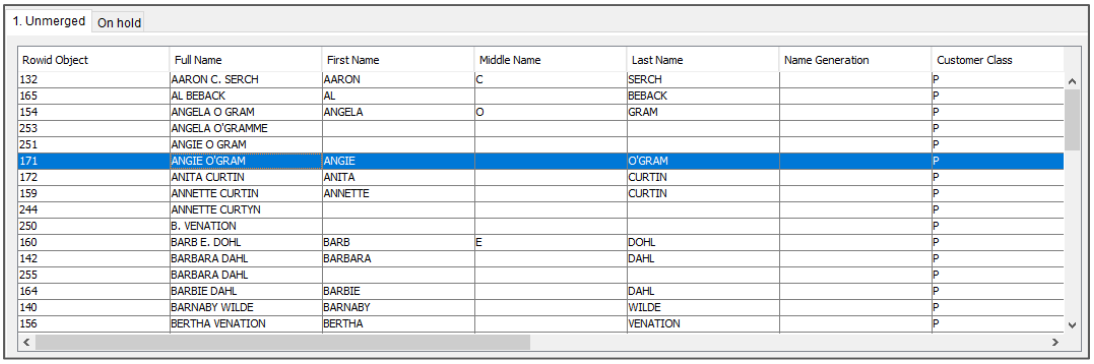
27. When there are no more matches for BARB DWYER, click the  button to flag the merged record as consolidated.

Manually Search for Matches

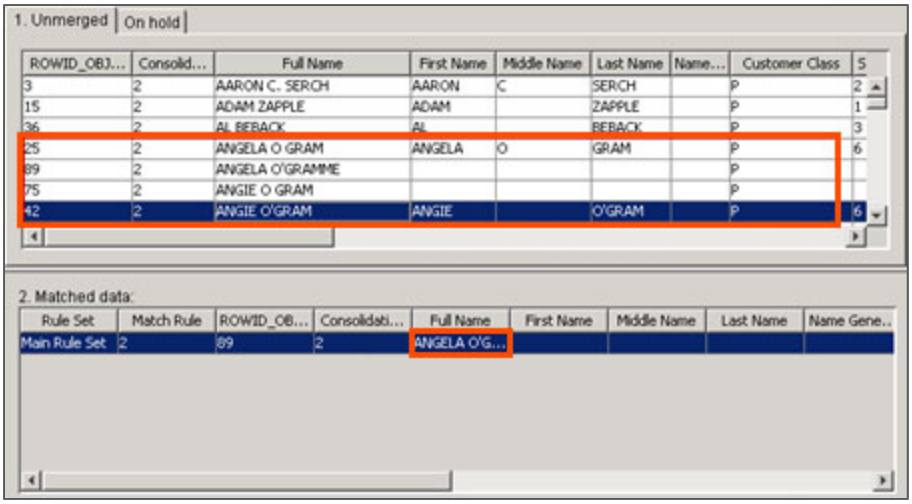
Occasionally, a data steward may find two (2) records that should merge together, but that the match rules have not identified as matches. If there are a large number of records of that nature, the match rules should be tuned to pick up more matches. But if there are just a few records of that nature, then the data steward can manually match them.

28. In the **Unmerged** list, find the record for **ANGIE O'GRAM** (be particular about the spelling because there are a few similar records).

    Even though there are multiple potential matches, only one (1) appears in the **Matched data** section.

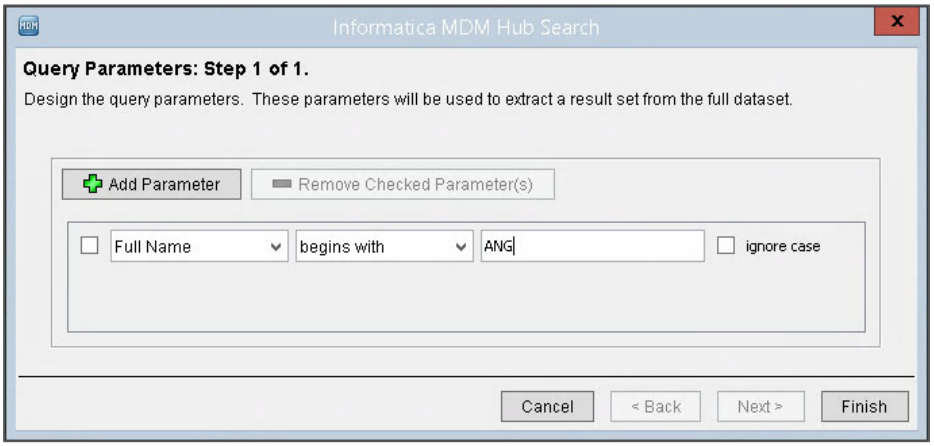


Rowid Object	Full Name	First Name	Middle Name	Last Name	Name Generation	Customer Class
132	AARON C. SERCH	AARON	C	SERCH		P
165	AL BEBACK	AL		BEBACK		P
154	ANGELA O GRAM	ANGELA	O	GRAM		P
253	ANGELA O'GRAMME					P
251	ANGIE O GRAM					P
171	ANGIE O'GRAM	ANGIE		O'GRAM		P
172	ANITA CURTIN	ANITA		CURTIN		P
159	ANNETTE CURTIN	ANNETTE		CURTIN		P
244	ANNETTE CURTYN					P
250	B. VENATION					P
160	BARB E. DOHL	BARB	E	DOHL		P
142	BARBARA DAHL	BARBARA		DAHL		P
255	BARBARA DAHL					P
164	BARBIE DAHL	BARBIE		DAHL		P
140	BARNABY WILDE	BARNABY		WILDE		P
156	BERTHA VENATION	BERTHA		VENATION		P

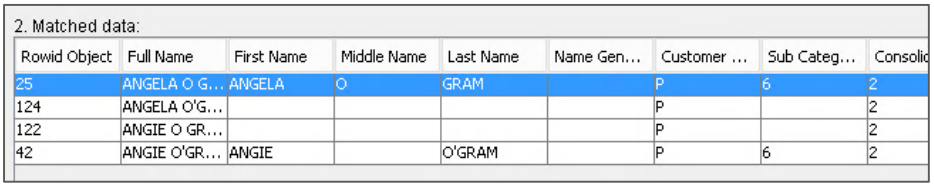


29. Click the Search for potential matches button.
30. In the Search Wizard, click **Next**.
31. Specify the following search criteria:

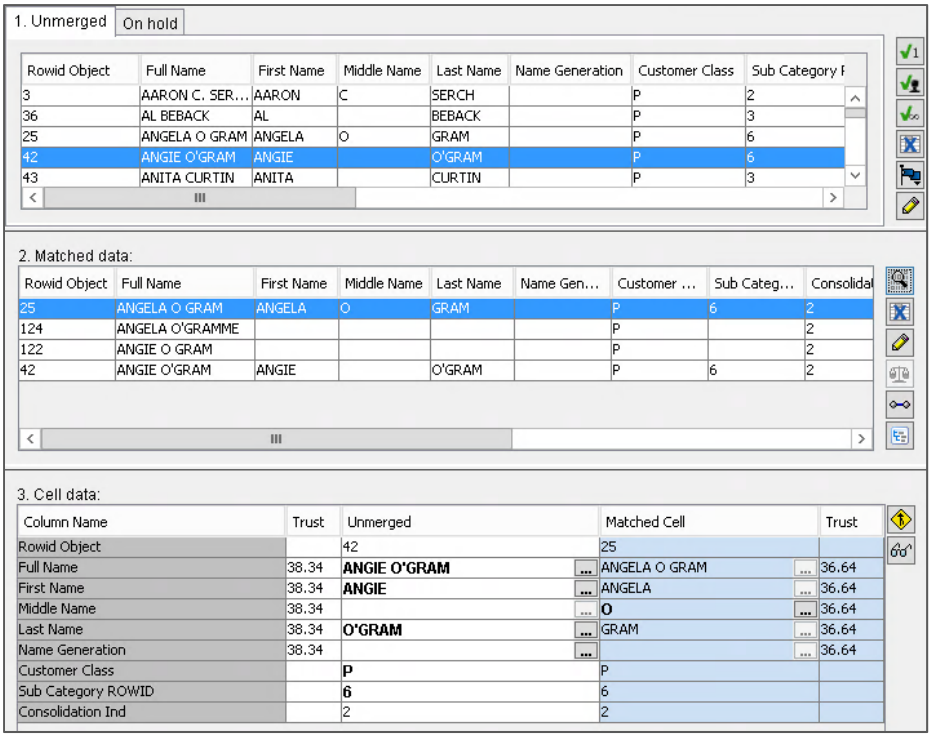
    Full Name begins with ANG



32. Click **Finish**.
33. Multiple records are added to the **Matched Data** section.



Note: The selected **Unmerged** record (ANGIE O'GRAM) is also shown in the **Matched data** section (Look for the record with the same Rowid_Object value as the record selected in the **Unmerged** section. For example, in the image below, the Rowid Object number is 42). This is an artifact of the search routine.

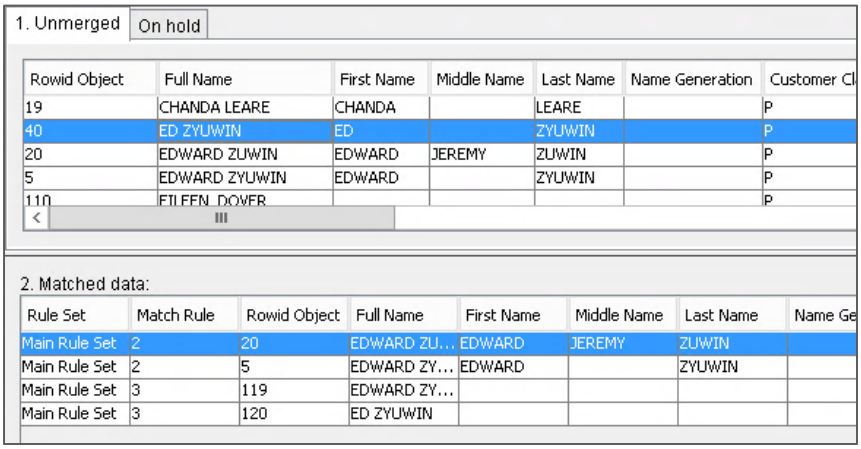


34. Click the **Merge** icon to merge ANGIE O'GRAM (42) into ANGELA O'GRAM (25) and make sure that the values from ANGIE O'GRAM wins.
35. Select ANGIE O'GRAM in the **Unmerged** section.

    Note: It now has matched records – this is because the consolidated record has inherited the matches from the merged ANGELA O GRAM record.
36. Now merge the rest of the ANGIE and ANGELA records.
37. Set the remaining record as unique (consolidated) if necessary.

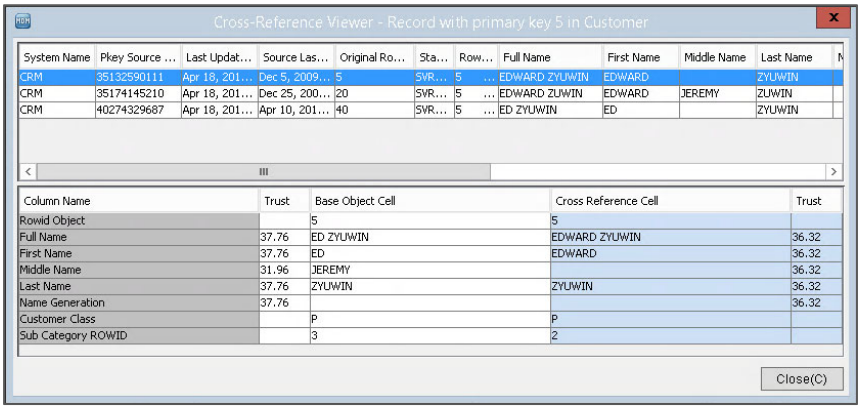
View Cross-references

38. Now merge a few of the various versions of ED ZYUWIN together: 40 into 20, and 20 into 5.



39. In the **Unmerged** section, find and click on the merged ED ZYUWIN record.

40. Click the **View Cross-Reference Records** button.



Note: The Cross-Reference Viewer opens.

41. Review the cross-reference records that contribute to the consolidated base object as a result of the merges you have just completed.
42. Observe that, as you click on the records in the upper pane, if the cross-reference contributes values in the merged record, it displays the value in bold characters.
43. Click **Close**.
44. Click the button to flag the merged **ED ZYUWIN** record as consolidated.

Flag Records for Automerge

It is not necessary for a data steward to try to complete all manual merges in real-time. It is possible to flag matched records for automerge and let the system merge those records the next time when the automerge batch job runs.

45. In the **Unmerged** section, find the record for **EILEEN DOVER-ANFELL**.

    There are multiple matched records: named EILEEN DOVER and E. DOVER.
46. Select the matched record **E. DOVER** and click the **Flag for Automerge** button.
47. Find one of the records called EILEEN DOVER in the **Unmerged** section. In the **Matched Data** section, select the record for EILEEN DOVER-ANFELL, and click the button again.
48. Find the E. DOVER record (or any other remaining variations on EILEEN DOVER's name), and flag it for automerge with the EILEEN DOVER-ANFELL record.

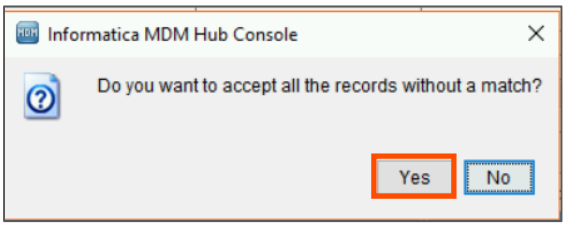
    Note: Those match pairs will be merged the next time the automerge batch job is run. If the data steward is satisfied with the default that survives the values, flagging the records for automerge is much more time efficient.

Flag all Records without a Match as Unique

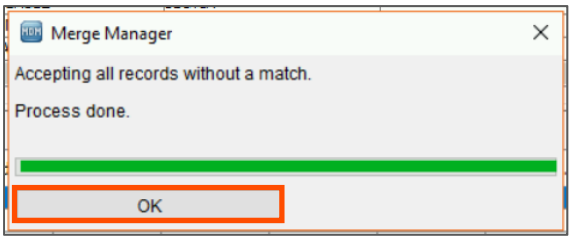
49. Click the records in the **Unmerged** tab at the top of the **Merge Manager** screen and notice that several records do not have any matched records.

    For example: AARON C. SERCH, AL BEBACK, BARNABY WILDE, BERTHA VENATION, STAN BACK, RUSSEL SPROUT, and TAMARA KNIGHT.

50. Click the **Accept all records without a match** button.
51. Click **Yes**.



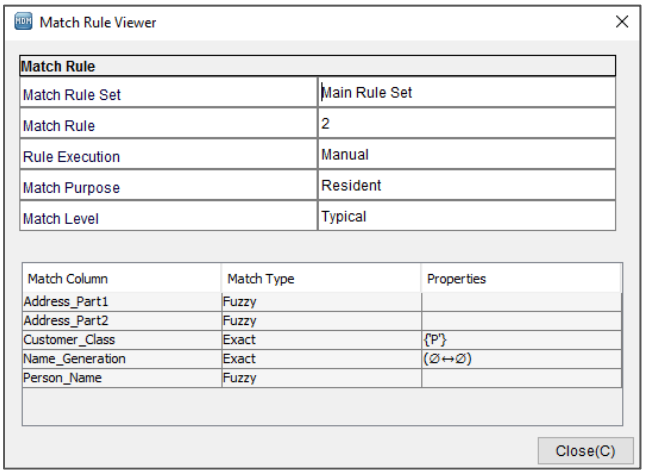
52. Click **OK**.



53. Once the process is complete, notice that the records without a match are no longer shown in the **Unmerged** section. This is because their Consolidation_Ind values are changed to 1.

Determine which Match Rule Caused a Match

54. In the **Unmerged** section, find the record for **ANITA CURTIN** (43). It is matched with a few similar records, but with records where the first names are not ANITA. Sometimes, it is helpful to know which rule can cause the records to be identified as a match pair.
55. Click the **Show Match Rule** button to see which match rules results in these matches.



56. Close the window when you are done.
57. In the walkthrough of this lab, you:
    - assigned records to user assignment list
    - flagged records as unique
    - merged records
    - manually searched for matches
    - flagged records for automerge
    - determined which match rule caused a match

This concludes the lab.

Additional Information

Search Wizard

With the use of Search Wizard, you can define the assignment based on any search criteria that use the columns available to you in the previously selected display package.

You can also leave Query as None and click **Begin Merge** in which case Merge Manager would simply assign the first 50 queued records to you.

Alternately, you can also choose to use a pre-defined query from the list that the Query drop-down displays, if you define such queries in your schema. Be careful to understand what the filter criteria are for that query so that you do not specify an additional criterion that may produce a null set that returns no records.

Module 12: Utilize Data Management Tools

Lab 12-2: Unmerge Records using Data Manager

Overview:

In this lab, you will perform a standard and tree unmerge using Data Manager.

Objective:

- Select records to view in Data Manager
- View cross-references, history, and BVT contributors
- Promote a cell value
- Create a new Customer record
- Change the consolidation flag for a record
- Use standard unmerge to unmerge a consolidated record
- Use tree unmerge to unmerge a consolidated record

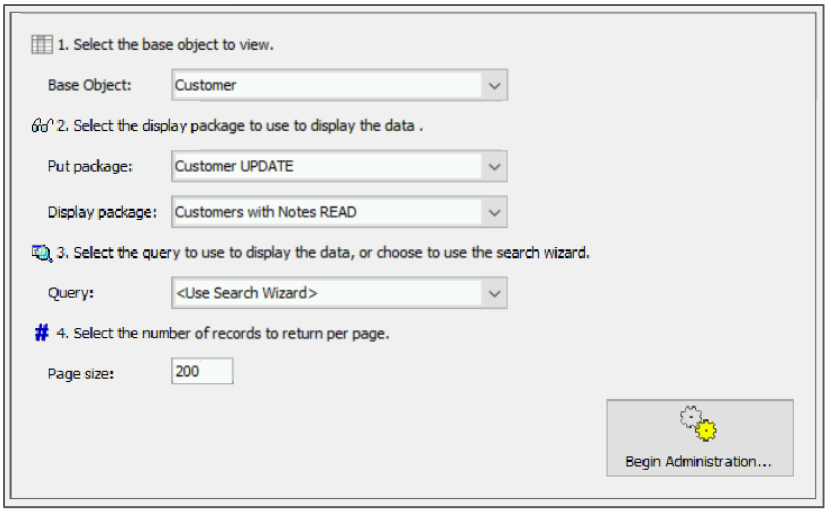
Duration:

25 minutes

Tasks

Select Records to View in Data Manager

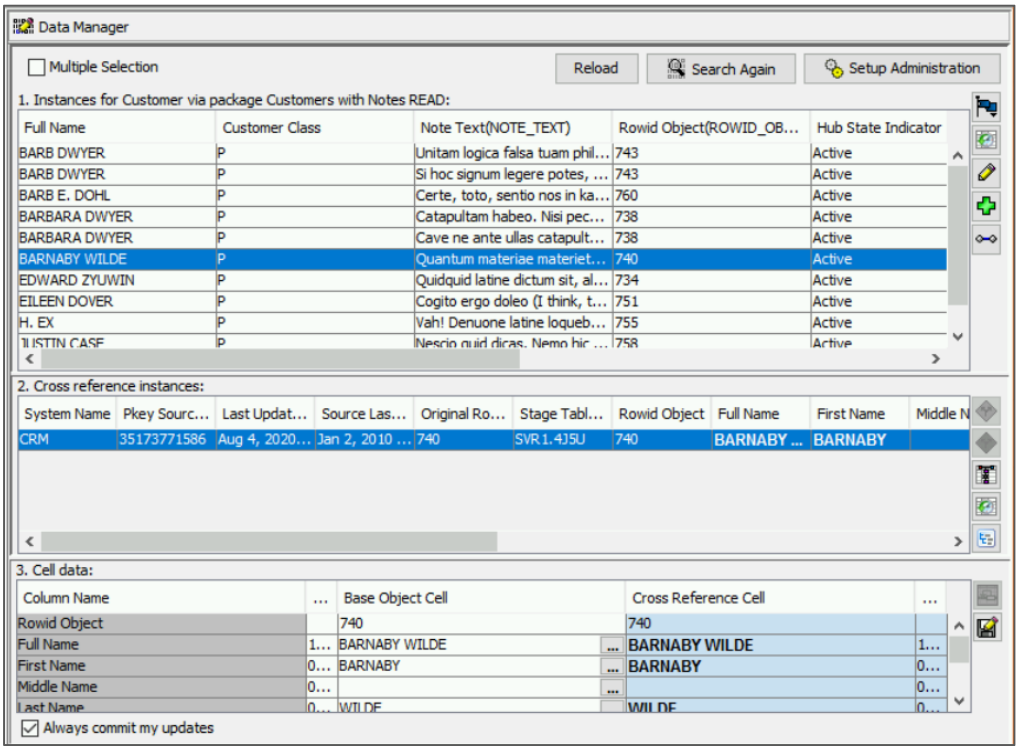
1. Select the **Data Manager** tool in the **Data Steward** workbench.
2. Select the **Customer** base object as the base object you want to review.
3. Select **Customer UPDATE** as the **Put Package** that you will use for any updates or cell overrides in the **Data Manager**.
4. Select **Customers with Notes READ** as the display package that you will use to view the data.
5. Ensure the **Query** is set to <Use Search Wizard>.
6. Set **Page size** to 200.
7. Click **Begin Administration**.



The **Search Wizard** opens.

If you do not specify any query parameters, all records selected by the display package you choose will be returned.

9. For this exercise, do not specify any query parameters, and click **Finish**.
10. Resize the columns so that you can read all the attributes.

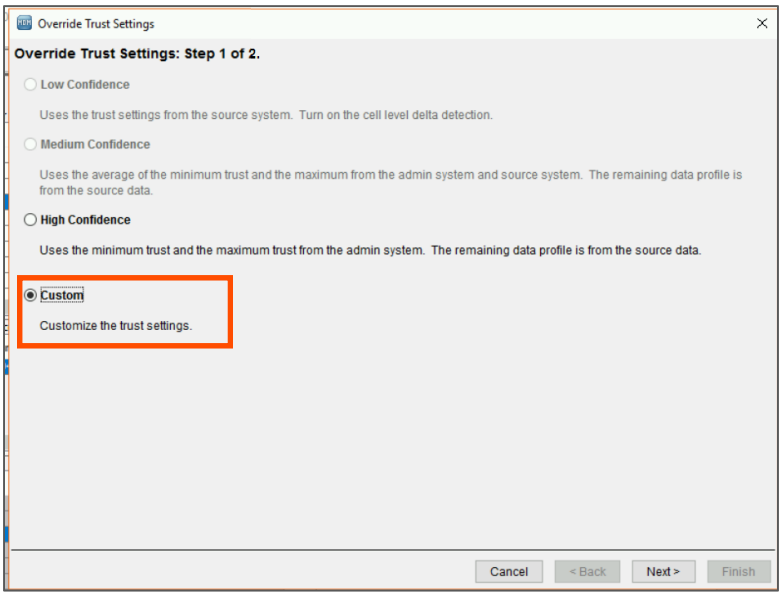


11. Sort the results in the top pane of the **Data Manager** screen by **Full Name**.
12. Select one of the records for **Barb Dwyer** that shows Cross reference instances.

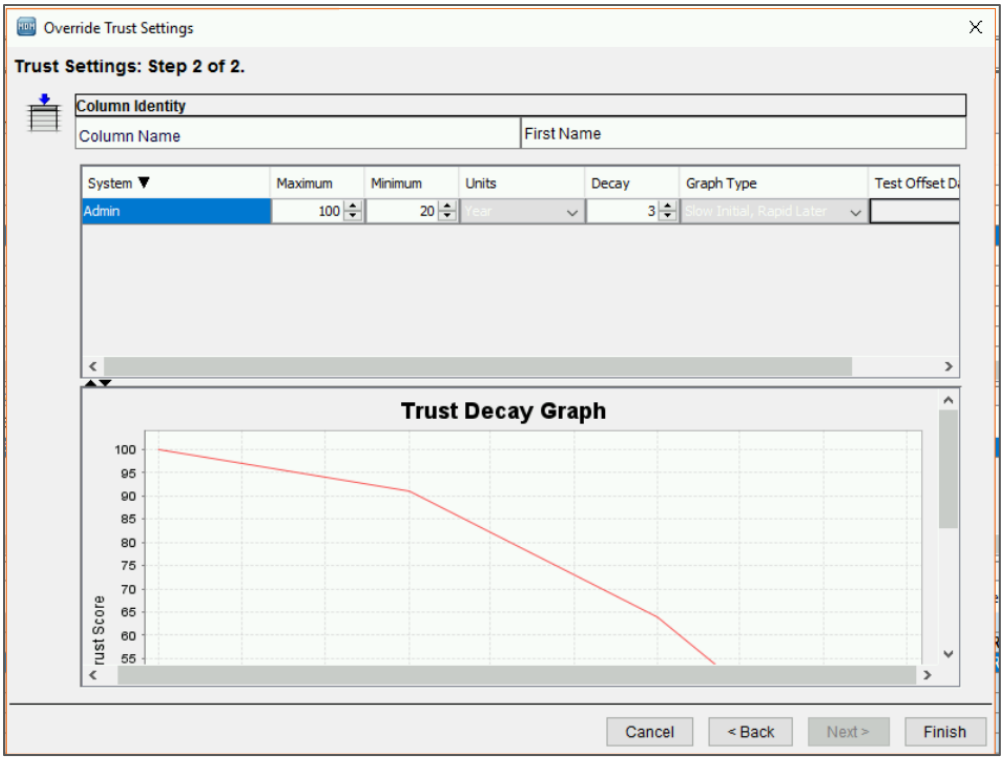
Promote a Cell Value

12. When you work with the consolidated record for BARB DWYER, select a cross reference record that has BARBARA in the **First Name** column.
13. In the **Cell Data** section at the bottom of the screen, click the row for **First Name**.
14. Click the **Increase Trust Score**  button to increase the trust score of the cross-reference record that you select for the **First Name**.

    This promotes the same cross-reference. The **Override Trust Settings** dialog opens.
15. Select **Custom** and click **Next**.



16. Enter the following:
    Maximum trust = 100
    Minimum = 20
    Units = Year
    Decay = 3
    Graph Type = Slow Initial, Rapid Later



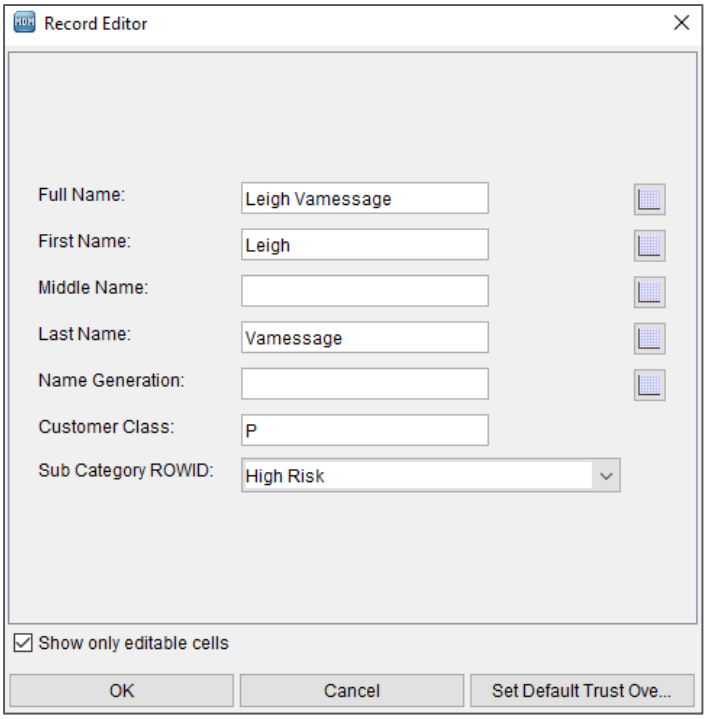
17. Click **Finish**. The winning value for the **First Name** is now Barbara instead of Barb.

2. Cross reference instances:

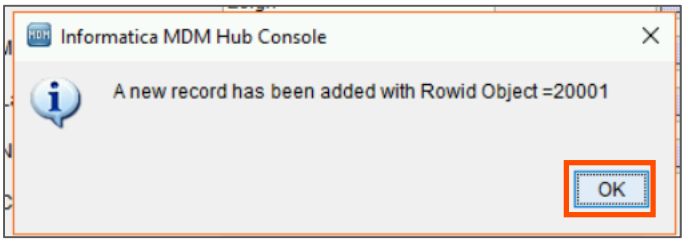
System Name	Pkey Sourc...	Last Updat...	Source Las...	Original Ro...	Stage Tabl...	Rowid Object	Full Name	First Name
CRM	35161064578	Apr 18, 201...	Apr 18, 201...	7		7	BARBARA DWYER	**BARBARA**
CRM	35164306999	Apr 18, 201...	Jan 11, 201...	8	SVR1.25PC	7	BARB DWYER	BARB
CRM	35167549420	Apr 18, 201...	Jul 2, 2009 ...	9	SVR1.25PC	7	BARBARA DWYER	BARBARA
CRM	35170791852	Apr 18, 201...	Jun 1, 2010 ...	10	SVR1.25PC	7	B. DWYER	B
CRM	35173980831	Apr 18, 201...	Jul 3, 2009 ...	14	SVR1.25PC	7	BARB DWYER	BARB
CRM	35174034284	Apr 18, 201...	Jan 2, 2010 ...	17	SVR1.25PC	7	BARB DWYER	BARB
Sales	A420275782	Apr 18, 201...	Apr 8, 2016 ...	113	SVR1.25JN	7	**BARB DWYER**	

Create a New Customer Record

18. Click the **Add Record** button.
19. Enter the Customer data, as shown in the following image, and click **OK**.



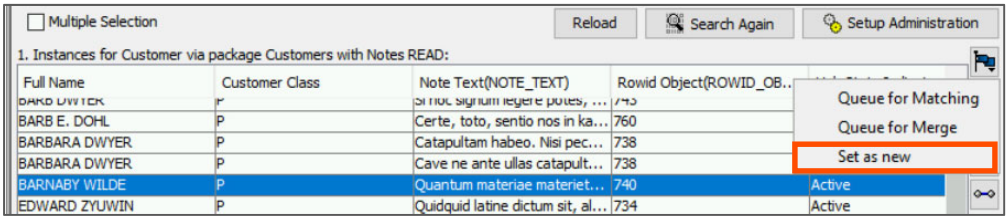
20. Click **OK**.



21. Click **OK** (your record number may be different).

Change the Consolidation Flag for a Record

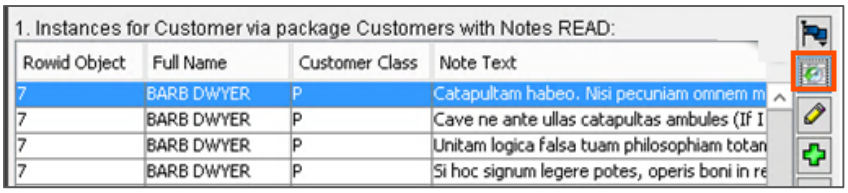
22. Find the record for **BARNABY WILDE**.
23. Click the **Reset** button and choose **Set as New**.



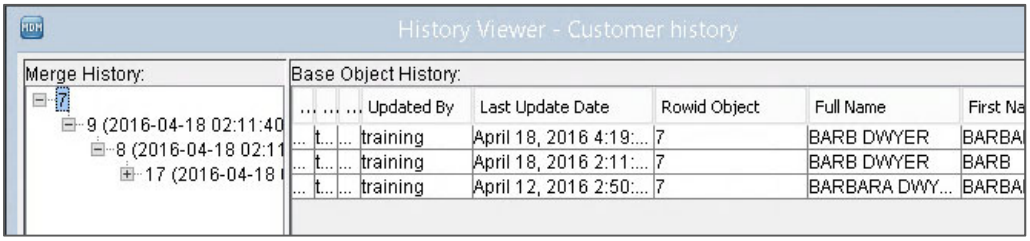
This will change the consolidation indicator value back to 4.

Unmerge a Consolidated Record Using Standard Unmerge

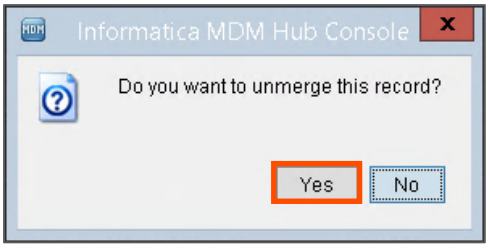
24. Find the record for **BARB DWYER**.
25. View the merge history tree for that record.



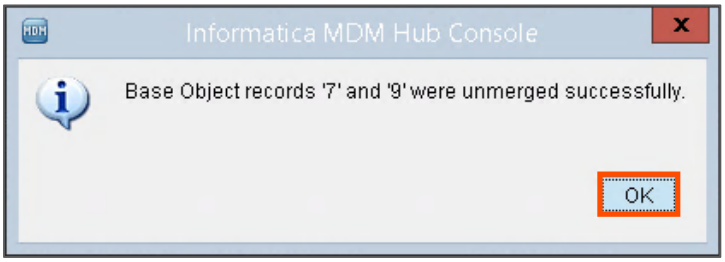
26. Here you can view the **Merge History** of the record. Several source records are merged together, and the tree should look similar to the following (it may differ slightly, depending on the order in which you merged records and the number of records you merged).



27. Close the history window.
28. Select a cross-reference instance that you want to unmerge from the consolidated record.
29. Click the **Unmerge** button.
30. Click **Yes**.



31. You will receive an acknowledgment that the two records that you selected are unmerged.



The unmerged record no longer contributes to the consolidated record.

Unmerge a Consolidated Record Using Tree Unmerge

32. Return to the record for **Barb Dwyer**.
33. View the merge history tree for that record.
34. Choose another cross-reference – and click the **Tree Unmerge**  button.

    You can notice that several cross-references are removed from the consolidated Barb Dwyer record, instead of just one cross-reference as in the previous section of this lab.
35. Click the **Reload** button to refresh the display data.

    Notice that there are now two consolidated records for Dwyer. Each has one or more cross-references as a result of the tree unmerge.

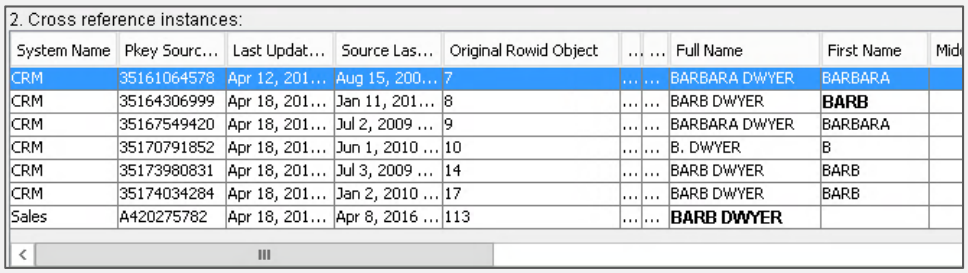
This concludes the lab.

Additional Observation

View Cross-references, History, and BVT Contributors

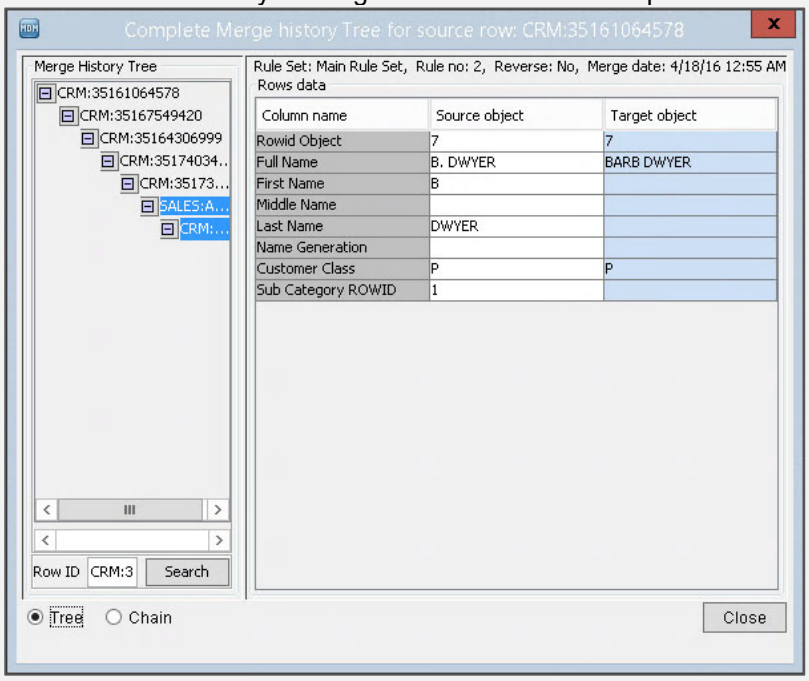
Use the various buttons and information on the Data Manager screen to answer the following questions about a consolidated record:

- How many source records are merged into this consolidated record?
- How many cross-references are from CRM system, and how many are from the Sales system?
- Which cross-reference record contributes the surviving Full Name to the consolidated record?



Open the Merge History Tree by clicking the button called **Display the complete Match history**  . Click on the nodes in the left-hand pane to expand them and see the entire history.

- Which 2 records are first merged together?
  Note: Your merge tree cannot look exactly the same as the one shown below, as it depends on the order in which you merged these records in the previous lab.



Click **Close** to close the Merge history Tree dialog.

Module 13: Additional MDM Product Features

Lab 13-1: Export an ORS to a Changelist

Overview:

In this lab, you will learn how to export a schema, which can be used to either document the configuration of an ORS or be used to move schema to another ORS.

Objective:

- Export an ORS to a change list using Repository Manager

Duration:

10 minutes

Review:

A change list is an XML file that documents the objects within an ORS and how they are configured. A change list can also be a subset of the objects in an ORS, but to create it is beyond the scope of this course. However, it is often necessary for an MDM designer to document the status of an ORS and export the ORS to a change list. It is a very convenient way to accomplish that. The export also provides a back-up of the ORS design.

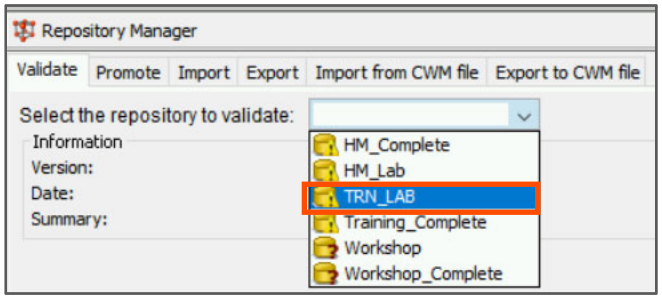
Here, you will export the ORS that was created for this course.

Tasks

Validate the ORS

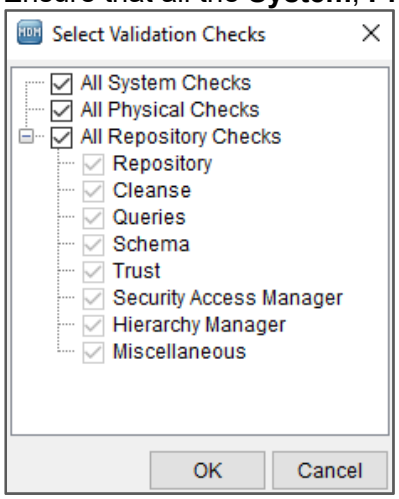
1. Navigate to the **Configuration** > **Repository Manager** tool from the Workbenches pane.
2. Click **Connect to master database**.
3. Click the **Validate** tab in the right-pane.

   Note: It is a good practice to always validate an ORS before you export it.
4. Select the **TRN_LAB** as the ORS to validate.



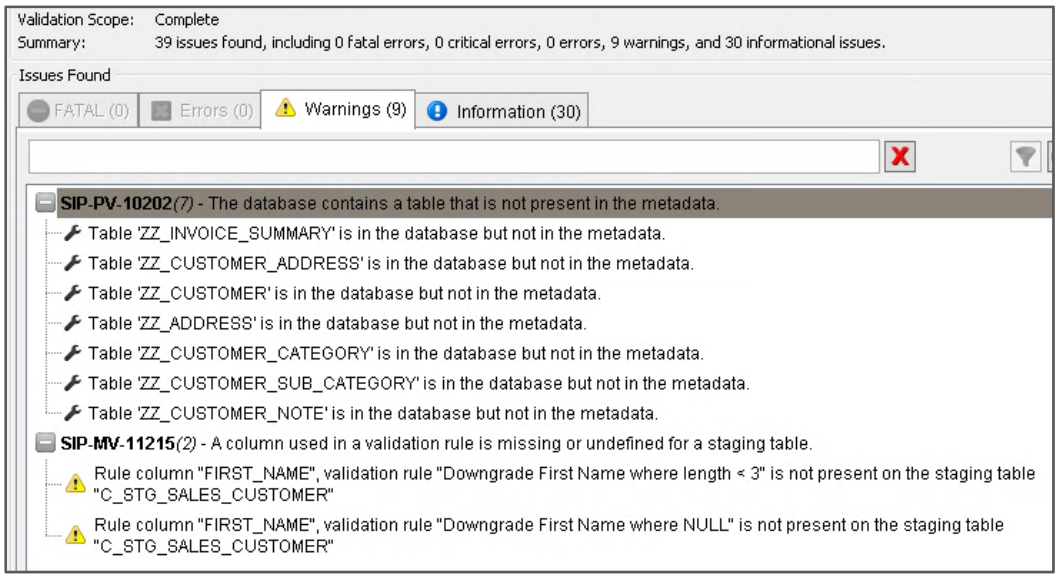
5. Click the **Validate** button.

6. Ensure that all the **System**, **Physical**, and **Repository** checks are selected.



7. To continue, click **OK**.

8. Some warnings and informational messages appear. Look at them, and you can see that MDM recognizes tables that exist in the ORS, but those are not there in the metadata. Normally that should be a concern, however, those tables are the ones that you use to simulate the CRM and Sales source systems, and they should be there for the purposes of this course.



Note

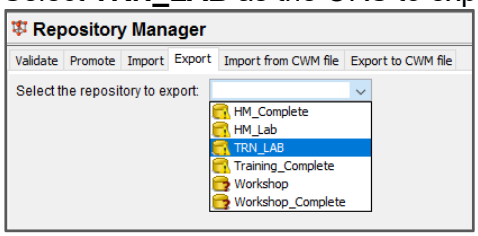
You will also see some warning about columns for validation rules missing for some staging tables. Both tables get data from the Sales source system that does not provide the First Name attribute, so these warnings are appropriate, but of no concern.

The warning messages have to do with Roles and Resources. Those topics are beyond the scope of this course and are covered in the MDM Administration course. In addition, those issues are resolved after the fundamental structure, configuration, and functioning of the ORS is complete and tested.

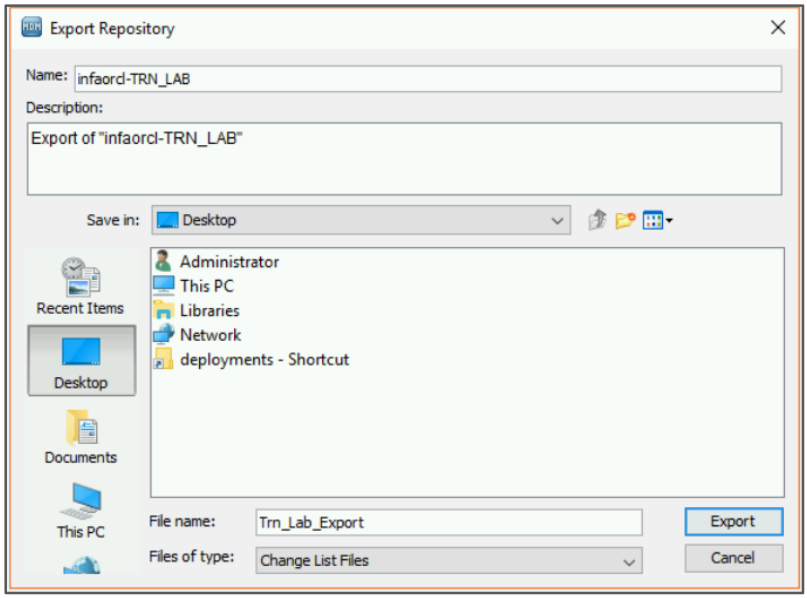
So, you do not need to correct anything before you proceed to export the ORS.

Export an ORS into a Change List

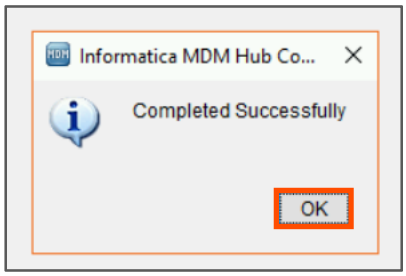
9. Click the **Export** tab.
10. Select **TRN_LAB** as the ORS to export.



11. Click the **Export** button. Navigate to the Desktop location to save the file.
12. Specify an appropriate **File name**. The **Description** is provided by MDM, but you may change it or add to it.
13. Click **Export**.



14. After the export is complete, acknowledge the message.



15. Open the export file with **Notepad++** just to look at the type of information it contains.
    Note: This is an XML file, and the structure of the file is beyond the scope of this course. However if you scroll past the Population file information, to the mid-portion, you should begin to recognize some of the items you created in the ORS.

    As mentioned in the beginning of this lab, the Export file is a convenient way to document and back up the configuration of an ORS.

This concludes the lab.

Module 15: Enabling Hierarchy Manager

Lab 15-1: Connecting to the ORS and Creating the System Repository Base Objects

Overview:

An MDM hierarchy shows the relationships between records in the MDM Hub. Hierarchy Manger allows you to manage hierarchy data that is associated with the records managed in MDM. To use the Hierarchy Manager and create relationships between records, you must enable it. After you enable it, you can connect to the ORS and view the existing schema objects.

To use the Hierarchies modeling tool with an ORS, you must first create the Repository Base Objects (RBO tables) for the ORS. RBO tables are essentially system base objects. RBOs are required base objects that must contain specific columns, which the system maintains – no user intervention is required.

Objective:

- Enable Hierarchy Manager for an MDM Operational Reference Store
- Create system repository base objects (RBO)

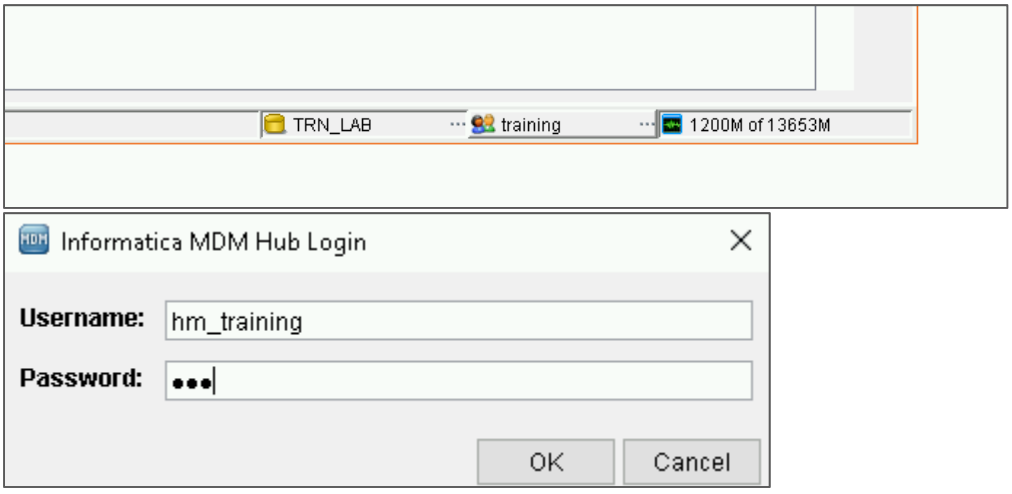
Duration:

10 minutes

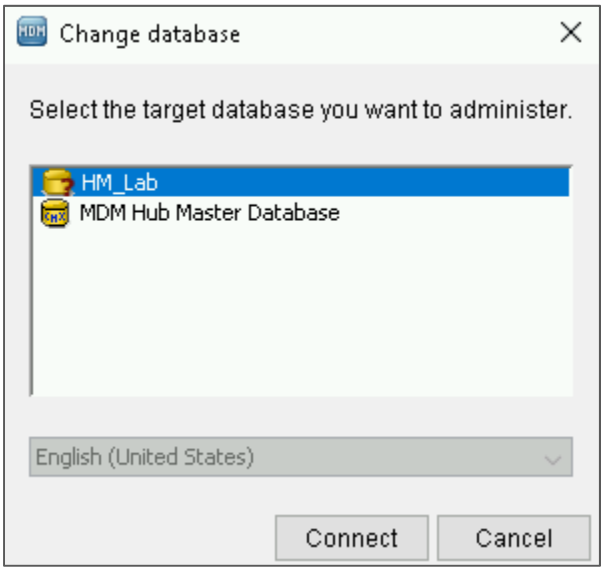
Tasks

First, you need to log in as another user and connect to a different database.

1. At the right-hand bottom of the Hub console, click **training** and enter **hm_training/mdm**.

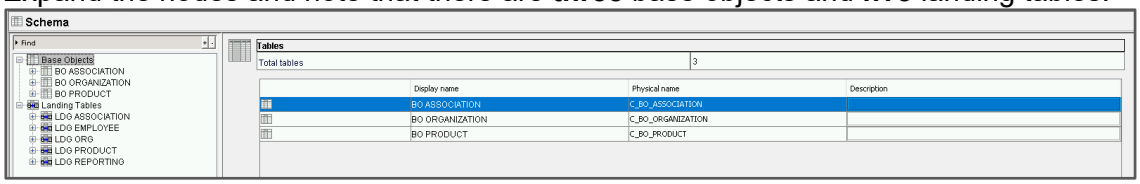


2. Select the repository as **HM_Lab** and click **Connect**.



View the Existing Schema Objects

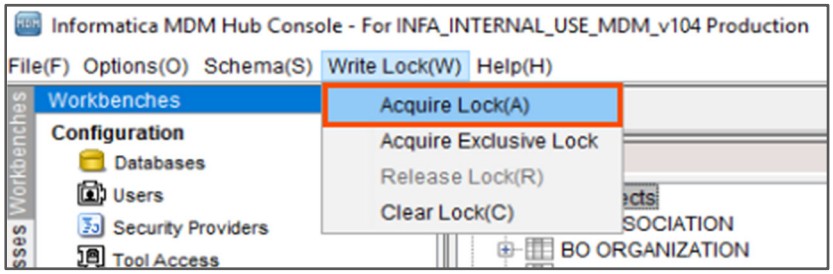
1. To view the current schema objects, from the **Model** workbench, select the **Schema** tool.
2. Expand the nodes and note that there are **three** base objects and **five** landing tables.



Note: The base objects and the landing tables have been created to support the training course. The landing tables have already been loaded with records. After your schema is flushed out and the hierarchies are fully configured, you will run the staging and loading batch jobs and look at the data in the Hierarchy Manager.

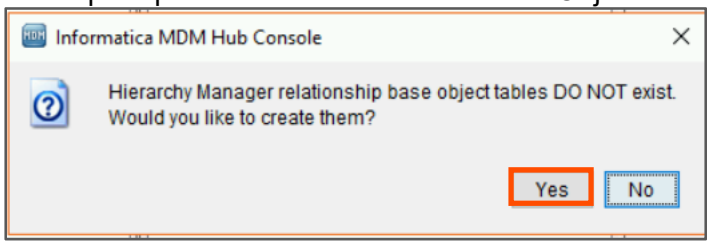
Create System Repository Base Objects

3. While in the Schema workbench, from the Write Lock menu, select **Acquire Lock**.

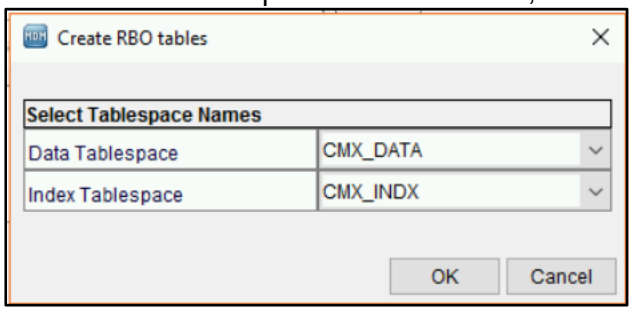


4. From the Model workbench, select the **Hierarchies** tool.

5. When prompted to create Relational Base Object tables, click **Yes**.

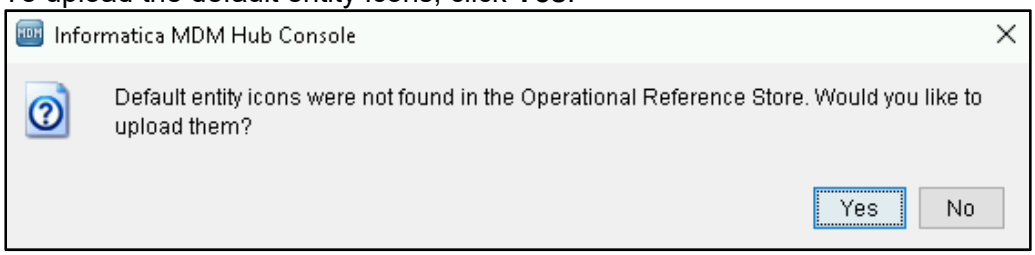


6. In the Select Tablespace Names section, retain the default values and click **OK**.



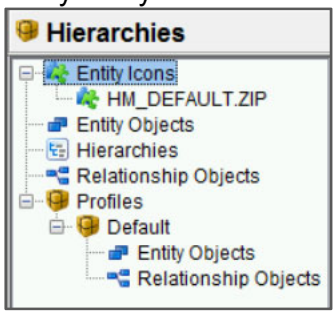
The system will create the RBOs.

7. To upload the default entity icons, click **Yes**.

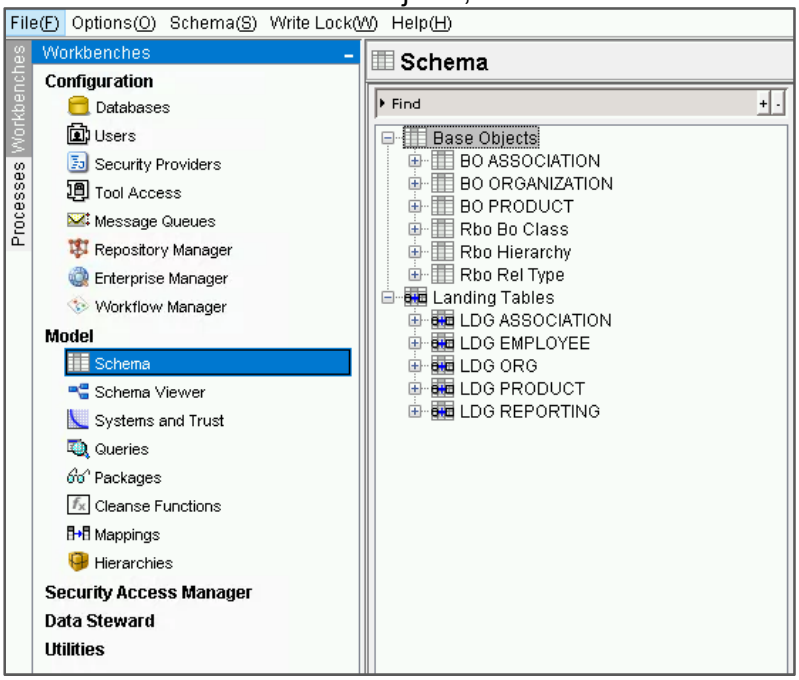


Note: You can upload customized icons for use in HM.

8. After the process completes, the hub opens the Hierarchies tool. Expand the nodes and verify that you have the following items in the Hierarchies pane:



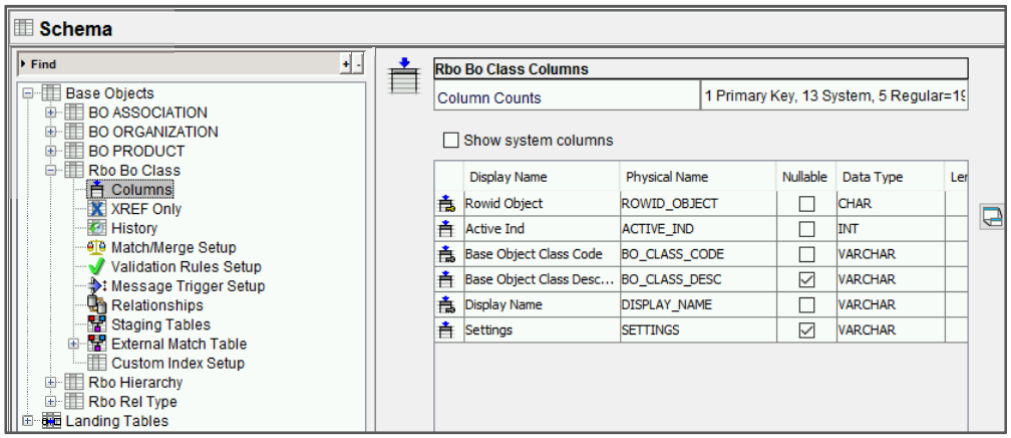
To view the current schema objects, from the Model workbench, select the **Schema** tool.



Note: There are now three additional base objects (Rbo Bo Class, Rbo Hierarchy, and Rbo Rel Type). As mentioned earlier, these are system-level objects that MDM creates and maintains.

9. Expand the repository base object nodes and observe their columns:
    i. The **Rbo Bo Class** table stores the various entity types and their descriptions. The columns of interest here are:

Names
Base Object Class Code (BO_CLASS_CODE)
Display Name (DISPLAY_NAME)
Base Object Class Description (BO_CLASS_DESC)



ii. The **Rbo Rel Type** table stores the various relationship types and their descriptions. The columns of interest here are:

Names
Relationship Type Code (REL_TYPE_CODE)
Display Name (DISPLAY_NAME)
Relationship Type Description (REL_TYPE_DESC)
Direction Indicator (DIRECTION_IND)
Rowid Base Object Class 1(ROWID_BO_CLASS1)
Rowid Base Object Class 2 (ROWID_BO_CLASS2)

iii. The **Rbo Hierarchy** table stores the hierarchies that have been defined and their respective descriptions. The columns of interest here are:

Names
Hierarchy Code (HIERARCHY_CODE)
Display Name (DISPLAY_NAME)
Hierarchy Description (HIERARCHY_DESC)
Row.ID Relationship Table (REL_ROWID_TABLE)

This concludes the lab.

Module 16: Entities and Entity Types

Lab 16-1: Converting Base Objects to Entity Base Object

Overview:

Entity base objects are base objects that are compatible with the hierarchy functionality and allow Hierarchy Manager to display and operate on the entities (records). There are two ways to create an entity base object (EBO) – either convert existing base objects to entity base objects or create them from the scratch. We will demonstrate both techniques.

To make the existing base objects compatible with the hierarchy functionality, you will convert them to entity base objects.

Objective:

- Create Entity Base Objects from existing Base Objects

Duration:

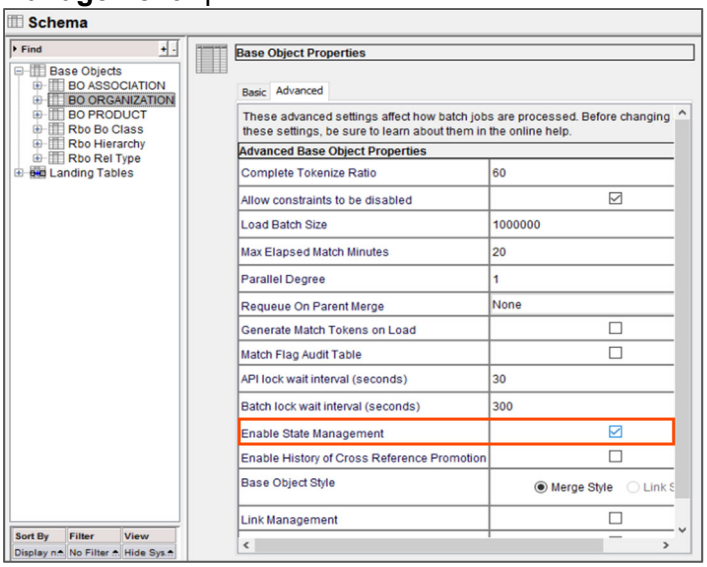
10 minutes

Tasks

Convert existing Base Objects to Entity Base Objects

Before you convert a base object to an entity base object, you must verify whether State Management is enabled for that base object.

1. Under **Model** workbench, open the **Schema** tool.
2. Acquire a **Write Lock** (if not acquired already).
3. For the **BO_ORGANIZATION** object, click on the **Advanced** tab.
4. In the **Advanced Base Object Properties** section, select the **Enable State Management** option.



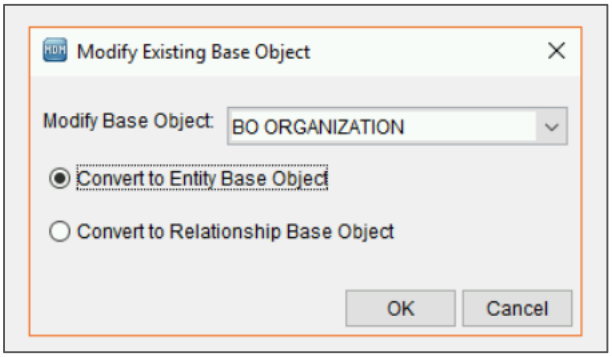
5. **Save** your work.

   Note: If state management is not enabled for any base object, then while converting it to an entity base object you will get an error message, 'Please enable state management on base object and try again.'

6. From the **Model** workbench, select the **Hierarchies** tool.
7. In the **Hierarchies** pane, right-click **Hierarchies**, and select **Convert BO To Entity/Relationship Object**.

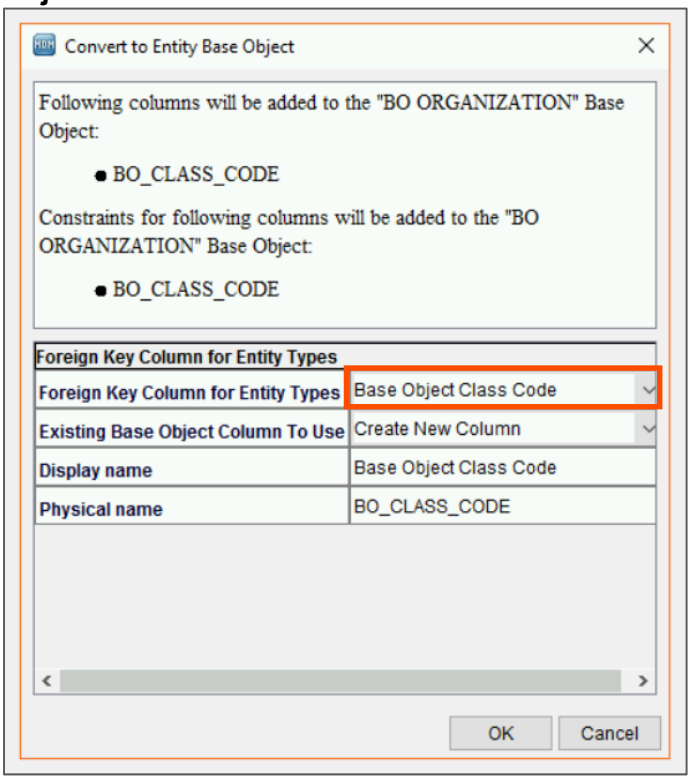
   Note: You do not need to select **Entity Objects** before executing this command. The command appears independent of any item selected in the pane.

8. From the drop-down, select **BO ORGANIZATION** and select the **Convert to Entity Base Object** option.
9. Click **OK**.



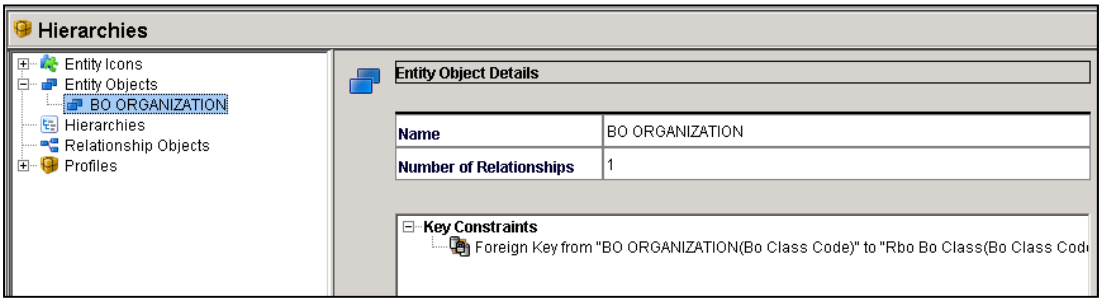
   Note: The Convert to Entity Base Object dialog box appears on the screen. Recall that various entity types are stored in the **Rbo Bo Class** base object. The dialog asks whether you want to use "Base Object Class Code" or "Rowid Object" to designate the entity type in the **Rbo Bo Class** base object.

10. In the window, from the Foreign Key Column for Entity Types drop-down, select **Base Object Class Code** and click **OK**.



11. Upon completion, expand the **Entity Objects** node.

    The new entity object **BO ORGANIZATION** will be visible, and the Entity Object Details pane will show a foreign key constraint between **BO ORGANIZATION** and **Rbo Bo Class**. The base object is now compatible with the Hierarchy Manager functionality.



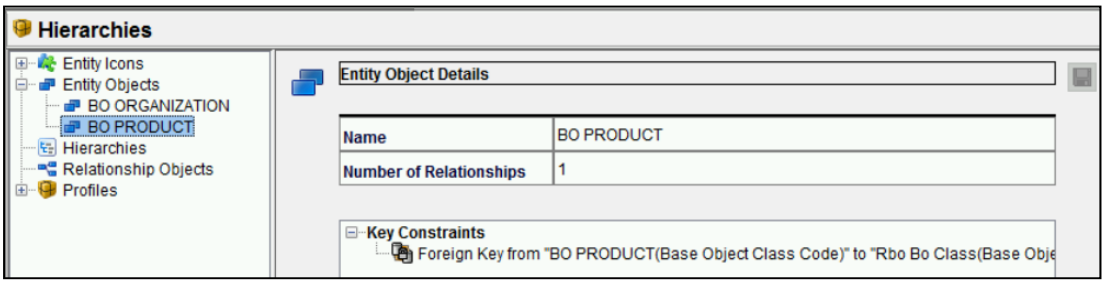
    Note: In case of an error, you can save your work and re-log into the Console.

12. Similarly, repeat the steps to convert the **BO PRODUCT** base object into an entity base object.

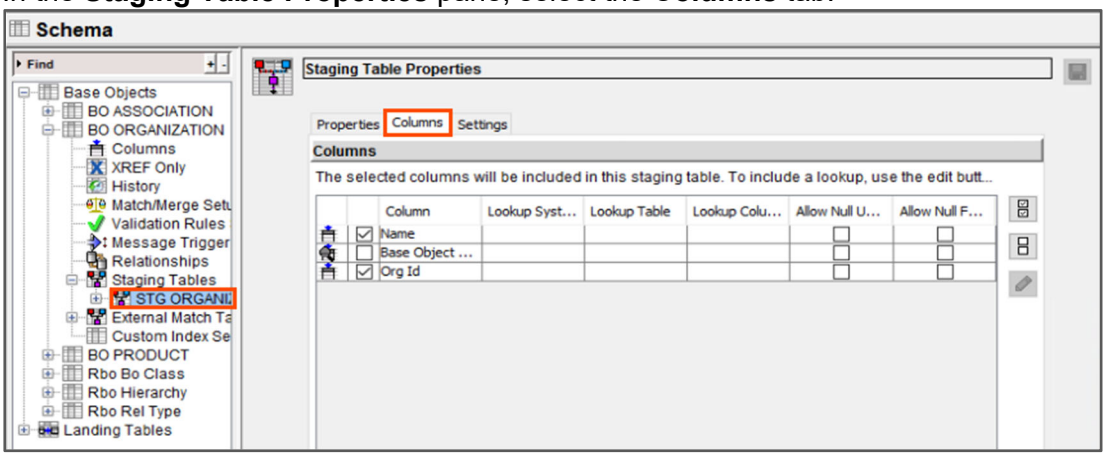
    Tip: Right-click **Hierarchies**, select **Convert BO To Entity/Relationship Object**.

    Recall: Do not forget to check "**Enable State Management**" option for the BO Product.

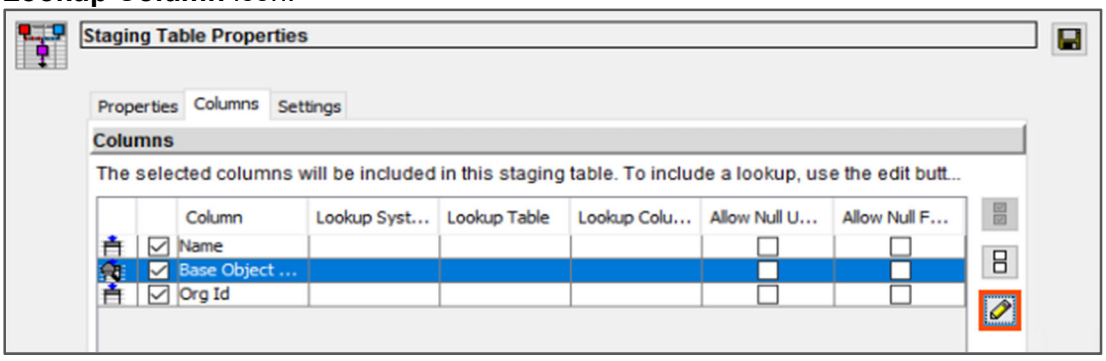
13. When completed, the **Entity Objects** node will contain both entity objects as shown below:



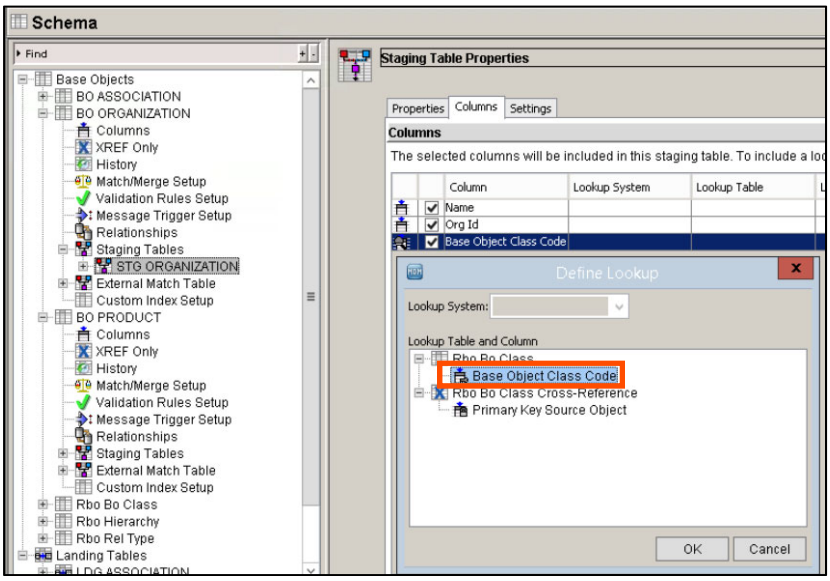
14. In the **Model** workbench, using the **Schema** tool, expand the base object **BO ORGANIZATION** and select its staging table **STG ORGANIZATION**.
15. In the **Staging Table Properties** pane, select the **Columns** tab.



16. Edit the lookup properties for the **Base Object Class Code** column by selecting the **Edit Lookup Column** icon.

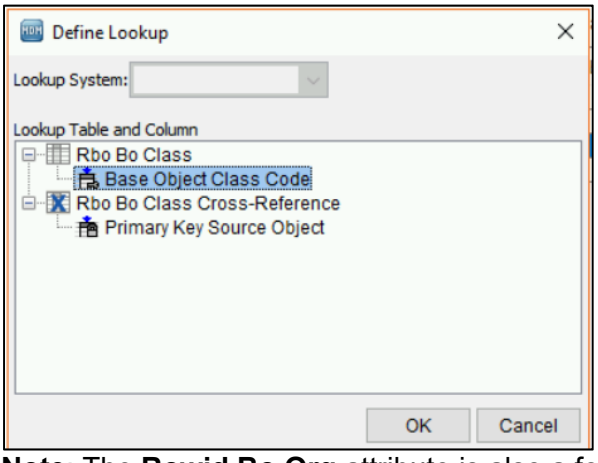


17. In the **Define Lookup** window, from the **Rbo Bo Class** table, select the **Base Object Class Code** column as the lookup.



Note: This causes the records to be associated with the proper entity type when they are loaded in the base object. It is a reuse of the lookup functionality to associate the entities with the appropriate entity type.

18. Click **OK**.
19. **Save** your work.
20. Similarly, enable lookup for the **STG PRODUCT** staging table for the **BO PRODUCT** base object.



Note: The **Rowid Bo Org** attribute is also a foreign key and you should take care of its lookup later. A lookup cannot be defined until a relationship to the primary key is put in place.

21. **Save** your work.

This concludes the lab.

Module 16: Entities and Entity Types

Lab 16-2: Creating New Entity Base Objects

Overview:
You can also create new entity base objects as per your business requirements. In this lab, you will create new entity base objects from scratch.

Objective:

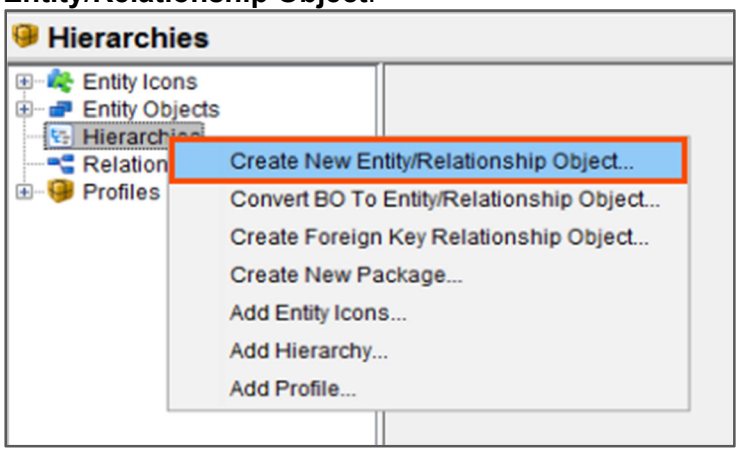
- Create new Entity Base Objects

Duration:
10 minutes

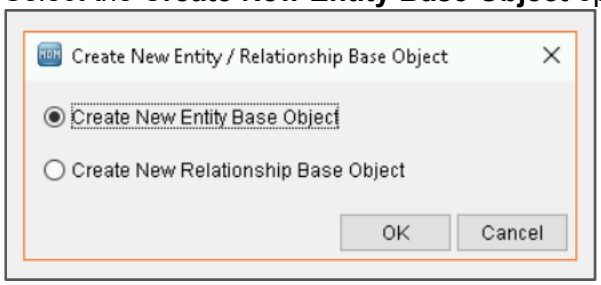
Tasks

Create new Entity Base Objects

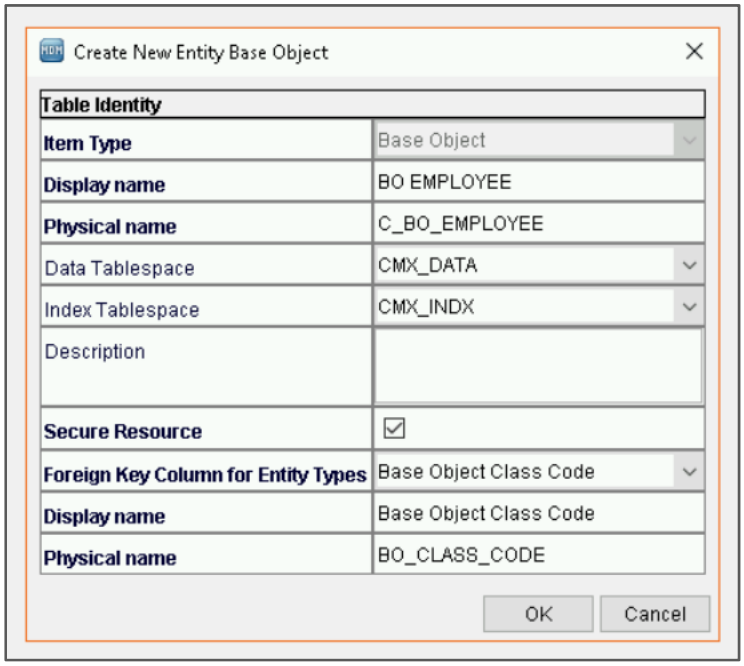
1. From the **Model** workbench, select the **Hierarchies** tool.
2. Acquire a **Write Lock**, if necessary.
3. In the Hierarchies pane, right-click **Hierarchies**, and select **Create New Entity/Relationship Object**.



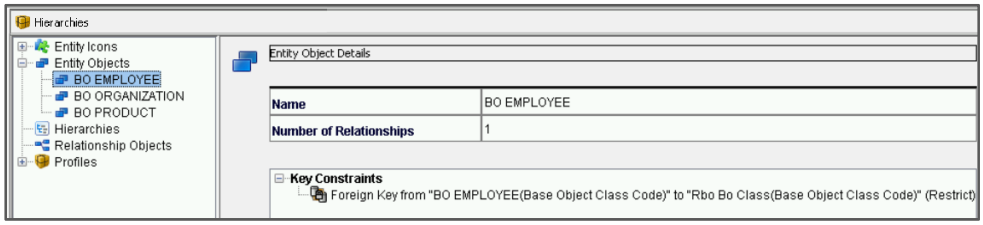
4. Select the **Create New Entity Base Object** option and click **OK**.



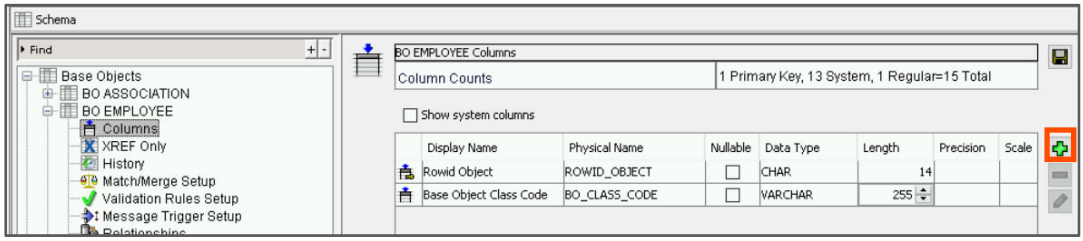
6. Enter the Display Name as **BO EMPLOYEE**.
   The Physical Name will get populated as **C_BO_EMPLOYEE**.
7. From the “Foreign Key Column for Entity Types” drop-down, select **Base Object Class Code** and retain the other default values.
8. Click **OK**.



9. After the process completes, expand the **Entity Objects** node.
   You can see the new entity object **BO EMPLOYEE**. When you select the BO EMPLOYEE entity object, the **Entity Object Details** pane shows the foreign key constraint to **Rbo Bo Class**.

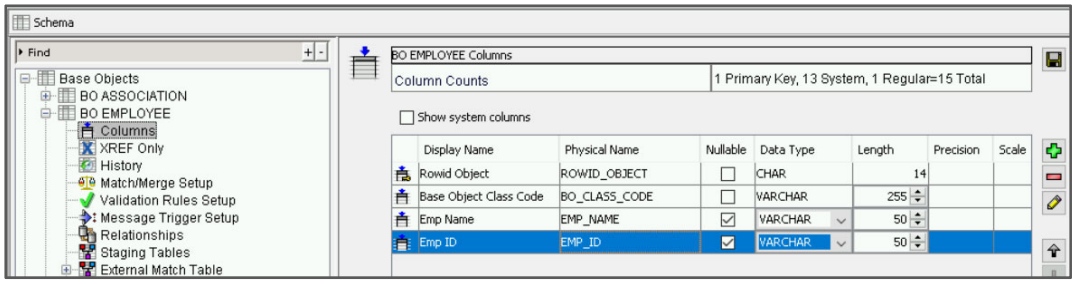


10. In the **Model** workbench, navigate to the **Schema** tool and note that under Base Objects, the new base object **BO EMPLOYEE** has been created.
11. Expand the **BO EMPLOYEE** base object, select the **Columns** option.
12. Click the **+** icon to add new columns.



13. Add the following two columns:

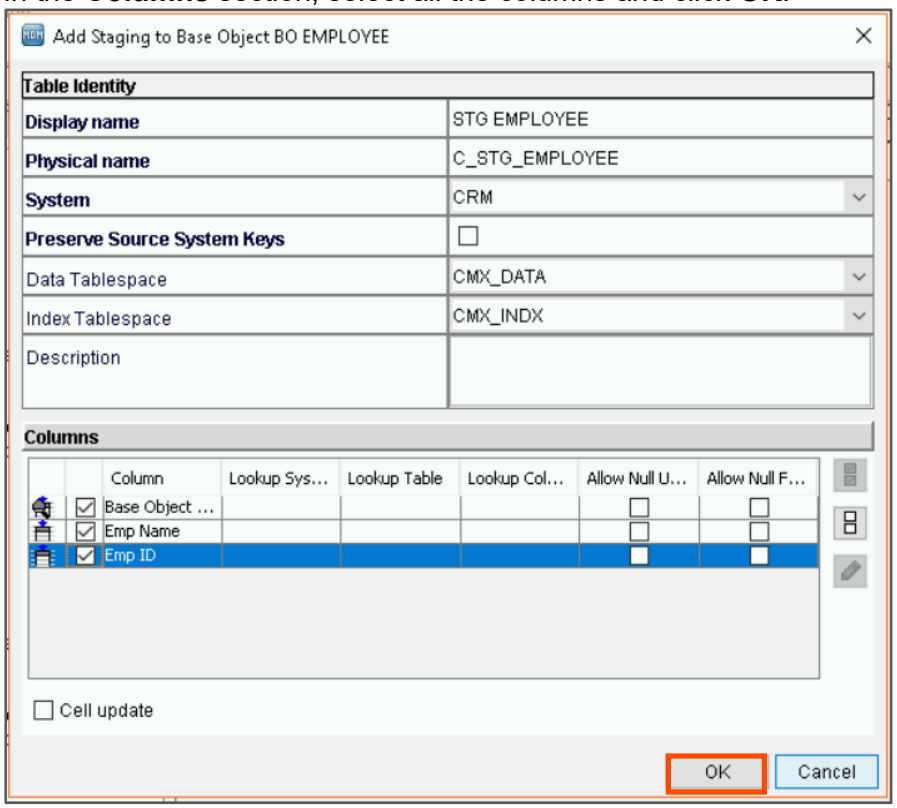
Display Name	Physical Name	Data Type	Length
Emp Name	EMP_NAME	VARCHAR	50
Emp ID	EMP_ID	VARCHAR	50



14. **Save** your work.

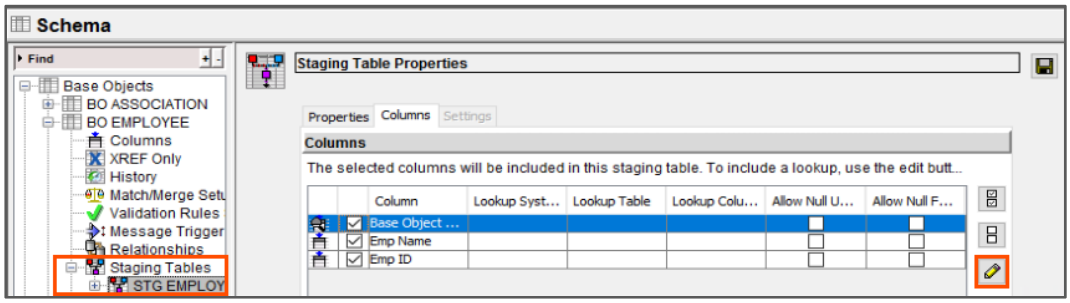
Now, create a staging table **STG EMPLOYEE** for the **BO EMPLOYEE** base object.

15. Under BO EMPLOYEE, right-click **Staging Tables** and select **Add Staging Table…(S)**.
16. Enter the Display Name as **STG EMPLOYEE**.
17. In the **Columns** section, select all the columns and click **OK**.

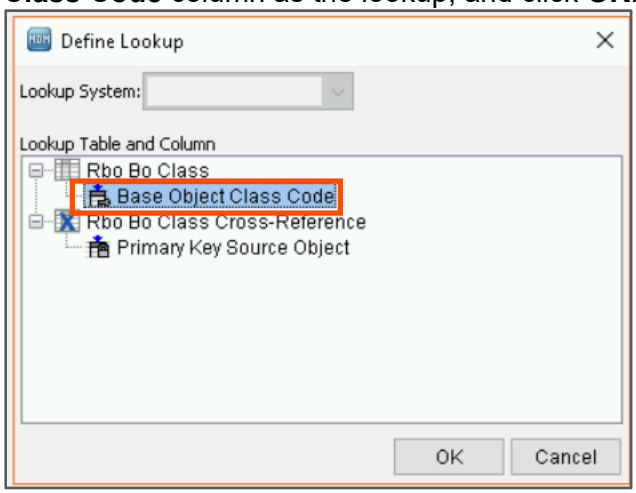


18. Select the **STG EMPLOYEE** table.

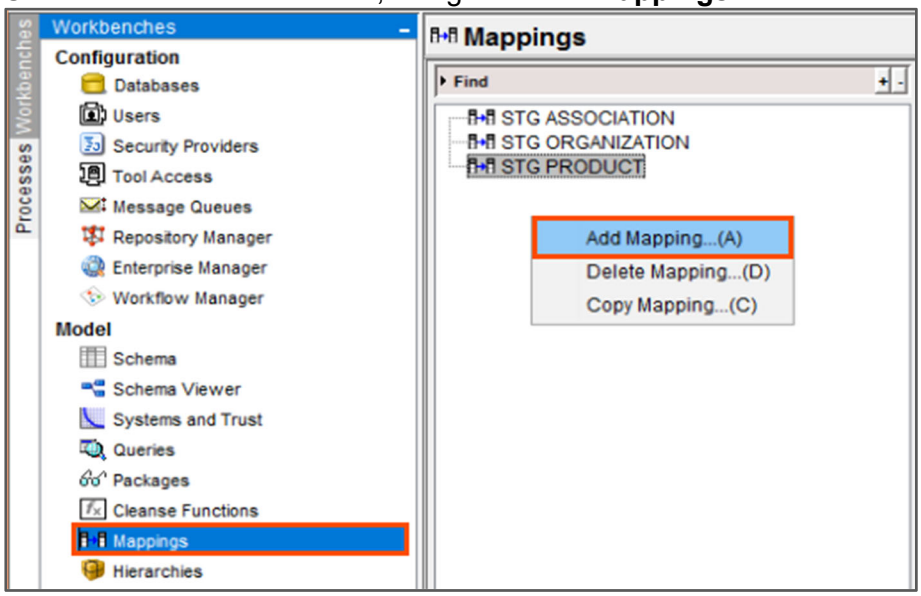
19. From the **Columns** tab, select the **Base Object Class Code**, and click the **Edit lookup column** icon.



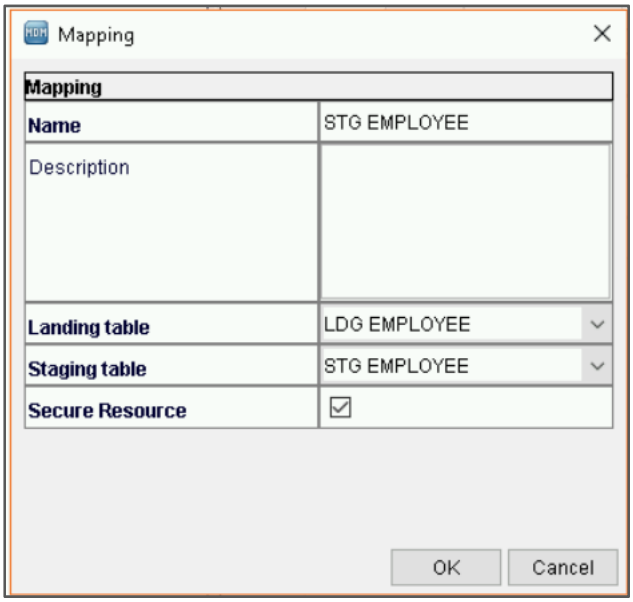
20. In the **Define Lookup** window, from the **Rbo Bo Class** table, select the **Base Object Class Code** column as the lookup, and click **OK**.



21. **Save** your work.
22. Under the Model Workbench, navigate to the **Mappings** tool and create a new mapping.

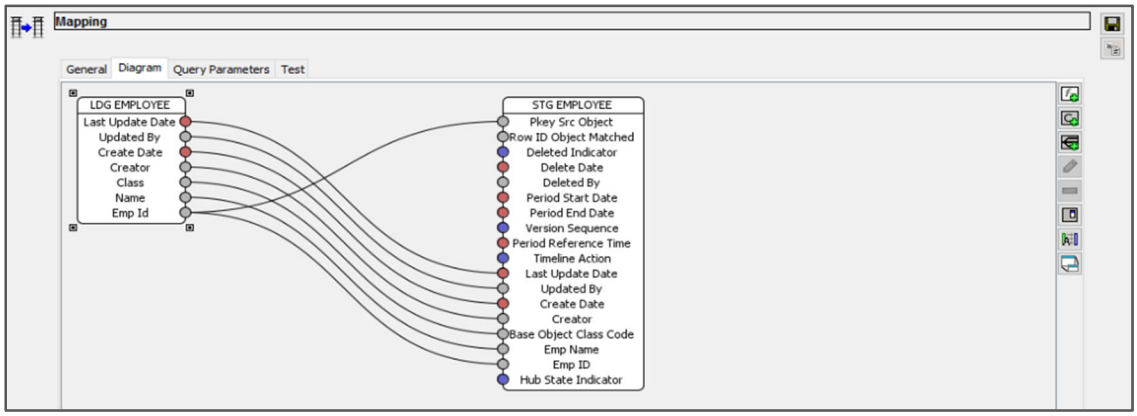


23. Enter the name as **STG EMPLOYEE**.
24. From the Landing table drop-down, select **LDG EMPLOYEE**.
25. From the Staging table drop-down, select **STG EMPLOYEE**.



Note: In this implementation the mappings are named after the staging tables.

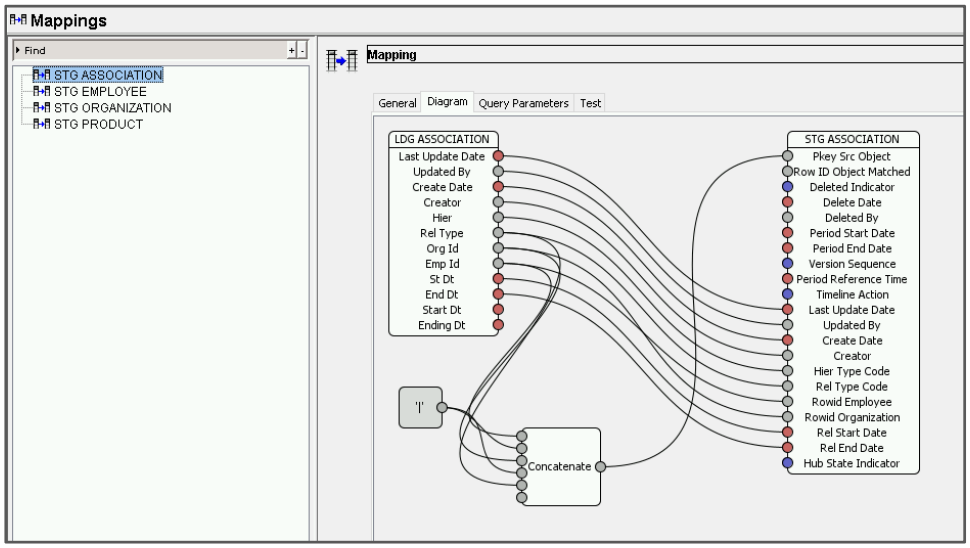
26. Click **OK**.
27. For the **STG EMPLOYEE**, in the Diagram tab, map the data from landing to staging as shown below and save the mapping:



Note: As many columns in the two tables have the same names, you can make use of the **Auto mapping** function. Right click anywhere on the mapping screen and from the pop-up window options, choose **Auto mapping**.

28. Similarly, for the other mappings that are present already, ensure that the fields are mapped.

29. Also, check that the **Class** field is mapped to the **Base Object Class Field** in all the mappings, except the STG ASSOCIATION mapping.



Note: STG ASSOCIATION mapping does not have the Class field.

This concludes the lab.

Module 16: Entities and Entity Types

Lab 16-3: Creating Entity Types

Overview:

The Entity Base Objects (EBO) contains records that represent the various entities. However, these records represent different entity types. For example, a Health Care Practitioner EBO may have entities of type Doctor, Nurse Practitioner, Physical Therapist, and Dentist. Each record associates with one entity type (It is possible to configure HM to allow association with more than one type, but that is beyond the scope of this course. Contact Product Support or Informatica Professional Services for more details on this option).

In order to keep our model simple, we will create one entity type for each Entity Base Object.

Objective:

- Create Entity Types for Entity Base Objects

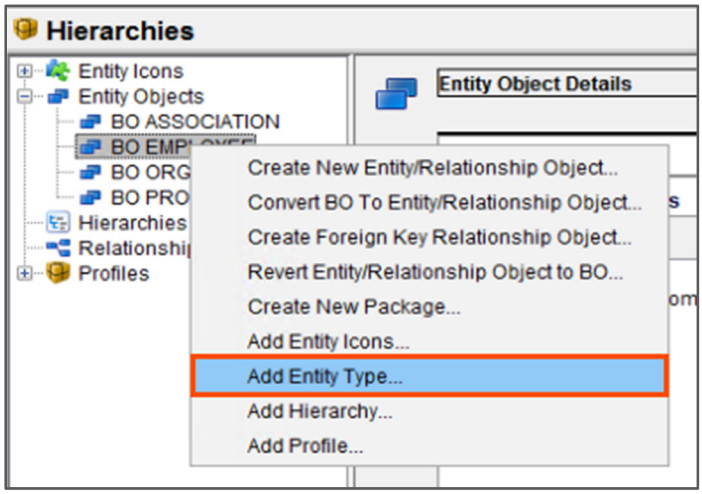
Duration:

10 minutes

Tasks

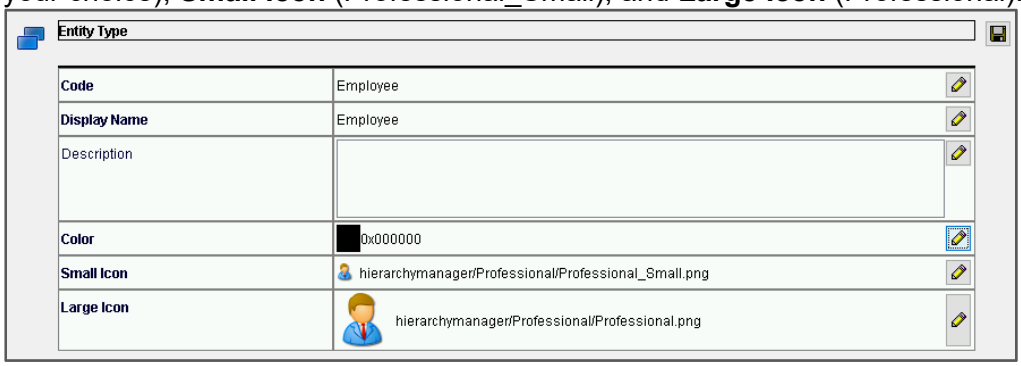
Create Entity Types for Entity Base Objects

1. In the **Model** Workbench, click the **Hierarchies** tool.
2. Under the Entity Objects node, right-click on the **BO EMPLOYEE** entity and select **Add Entity Type…**.



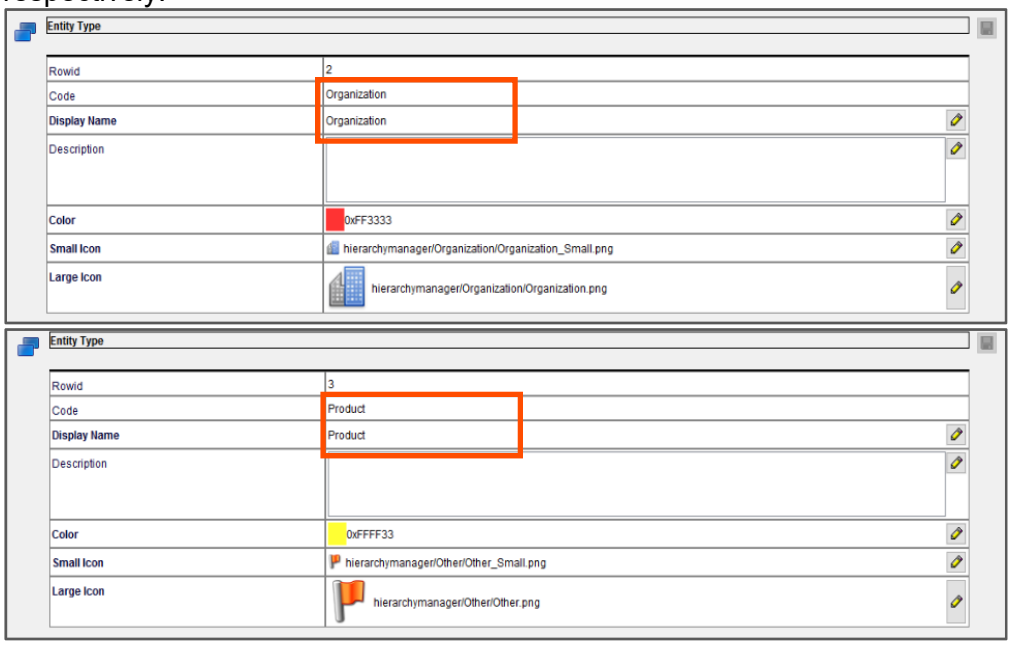
3. Enter the Code and Display Name as **Employee**.

4. Use the **Edit** buttons to set the **New Entity Type** properties. Select a **Color** (any color of your choice), **Small Icon** (Professional_Small), and **Large Icon** (Professional).



5. **Save** your changes.
6. A new entity type **Employee** now appears under the **BO EMPLOYEE** entity object.

   Note: A new value called Rowid has been added to the Entity Type.
7. Similarly, for the entities **BO ORGANIZATION** and **BO PRODUCT**, create the entity types **Organization** (use the “Organization” icon) and **Product** (use the “Other” icon) respectively.



8. **Save** your work.

This concludes the lab.

Module 16: Entities and Entity Types

Lab 16-4: Using a Query to View the Entity Types

Overview:

Entity Types are stored in the **Rbo Bo Class** base object. You use the Queries tool under the Model workbench to create a query for this RBO to view the data. There should now be three records in this base object; one each for Employee, Organization, and Product.

You might need to update the properties of the Entity Types that you created. In this lab, you will explore how to edit an entity type and delete an existing entity type.

Objective:

- Write a query to view the Entity Types
- Edit and delete existing Entity Types

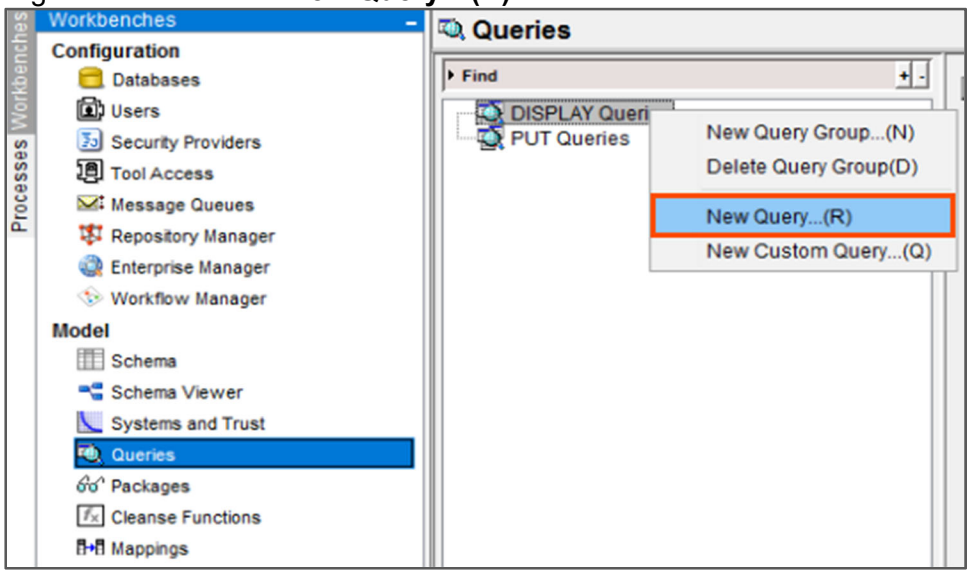
Duration:

10 minutes

Tasks

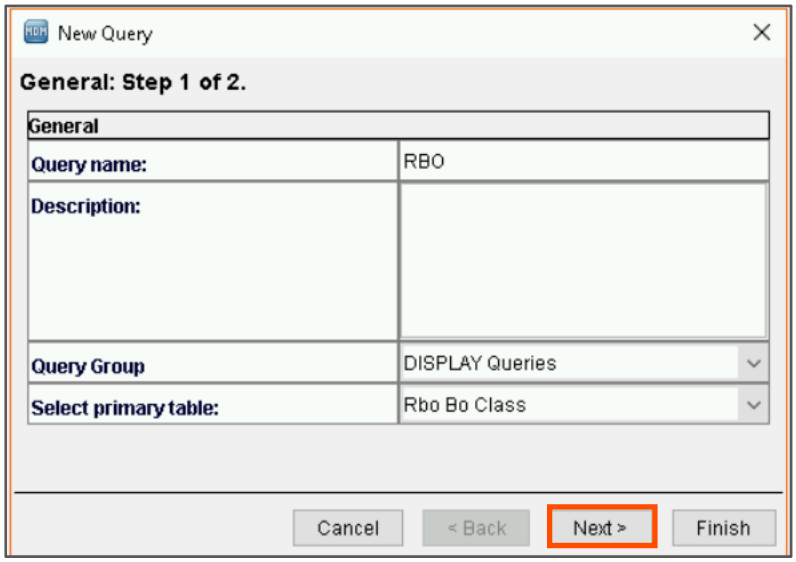
Use a query to View the Entity Types

1. To create a query, in the **Model** workbench, open the **Queries** tool.
2. In the Queries pane, select the **DISPLAY Queries** query group.
3. Right-click and select **New Query…(R)**.

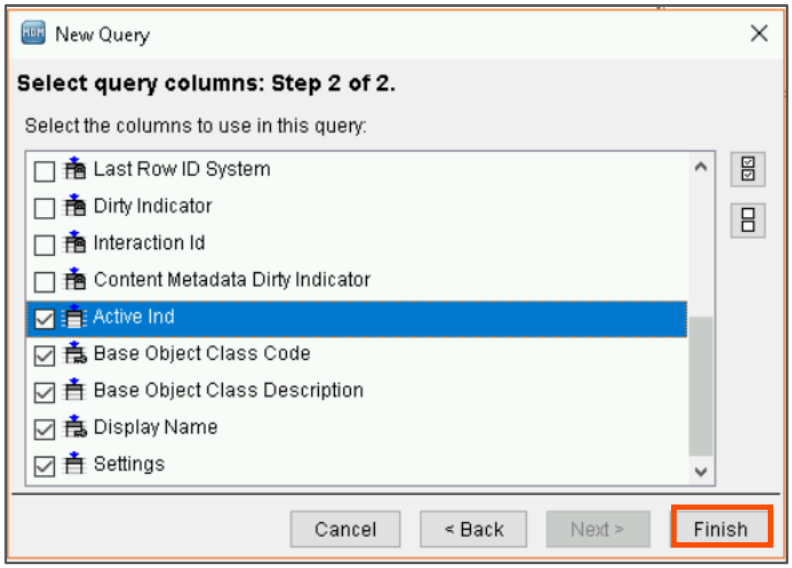


Note: The **New Query Wizard** appears.

4. Name the query as **RBO** and from the **Select Primary Table** drop-down, select **Rbo Bo Class**.
5. Click **Next**.



6. Select the columns: **Rowid Object**, **Base Object Class Code**, **Display Name**, **Base Object Class Description**, **Settings**, and **Active Ind**.
   Recommended: Click the **Check None** icon on the right to clear all the columns and then select the desired columns.
7. Click **Finish**.



8. Expand the new query and view the results.

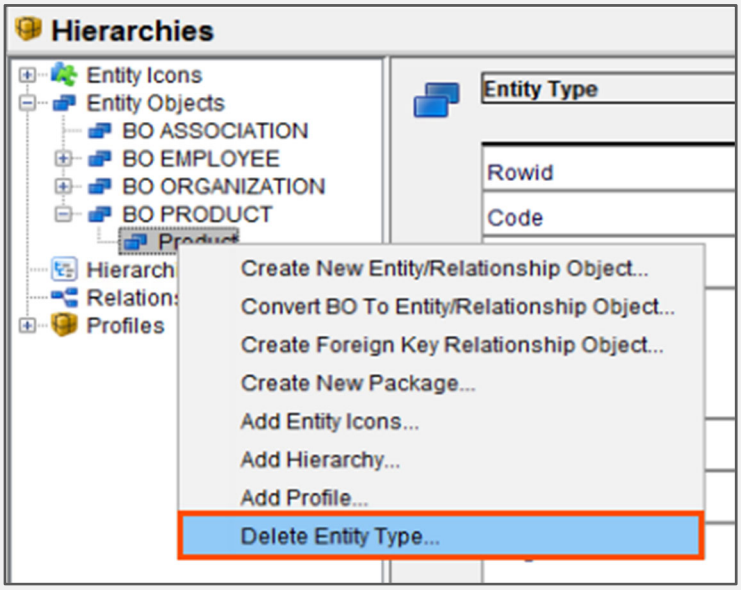
   Note: It is worth emphasizing here that these are the Entity Types that we created for each EBO and that they are stored in the **RBO Bo Class** repository base object.

This concludes the lab.

Additional Information

Edit and Delete Entity Types

1. To edit an Entity Type, select the entity type and modify the properties.
   Note: You cannot change the Entity Type Code attribute after you save it, and in case you need to change it, you must delete the Entity Type and recreate it again with the new Code value.
2. To delete Entity Types, in the Hierarchies tool, right-click on the entity type, and select **Delete Entity Type…**.



Note: If you already have data loaded and try to delete an Entity Type, Hierarchy Manager will not allow you due to referential integrity constraints. For this training, don't delete any of the entity types you have created.

Module 17: Hierarchies, Relationships, and Relationship Types

Lab 17-1: Editing Hierarchies and Using a Query to View the Hierarchy Objects

Overview:
An MDM hierarchy shows the relationships between records in the MDM Hub. The relationships can be between records in the same entity base object or between records in different entity base objects. You must configure the hierarchy before you can populate the hierarchy with data.

Objectives:

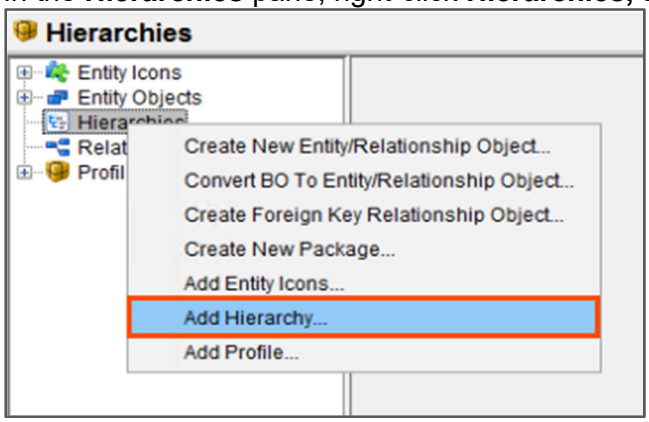
- Create a new Hierarchy
- Edit an existing Hierarchy
- Delete an existing Hierarchy
- Use a Query to View the Hierarchy Objects

Duration:
5 minutes

Tasks

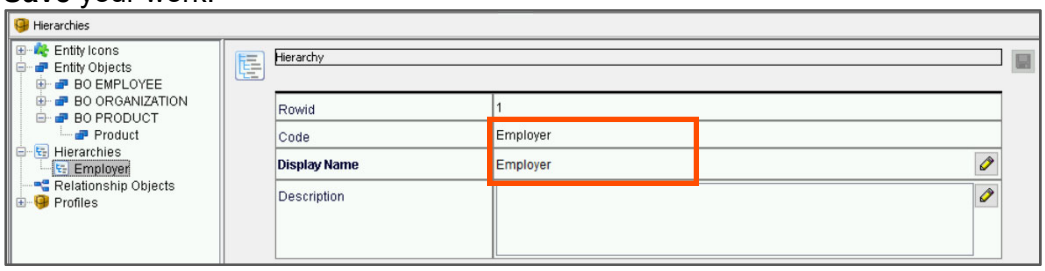
Create a New Hierarchy

1. From the **Model** workbench, select the **Hierarchies** tool.
2. Acquire a **Write Lock**, if necessary.
3. In the **Hierarchies** pane, right-click **Hierarchies,** and select **Add Hierarchy…**.

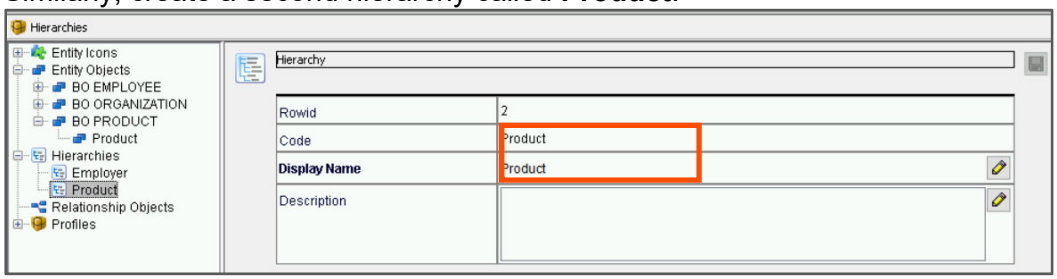


4. Edit the **New Hierarchy** object's properties and set the **Code** and **Display Name** to **Employer**.

5. **Save** your work.

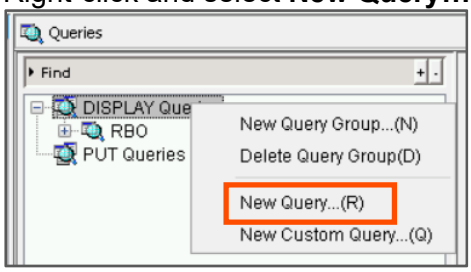


6. Similarly, create a second hierarchy called **Product**.



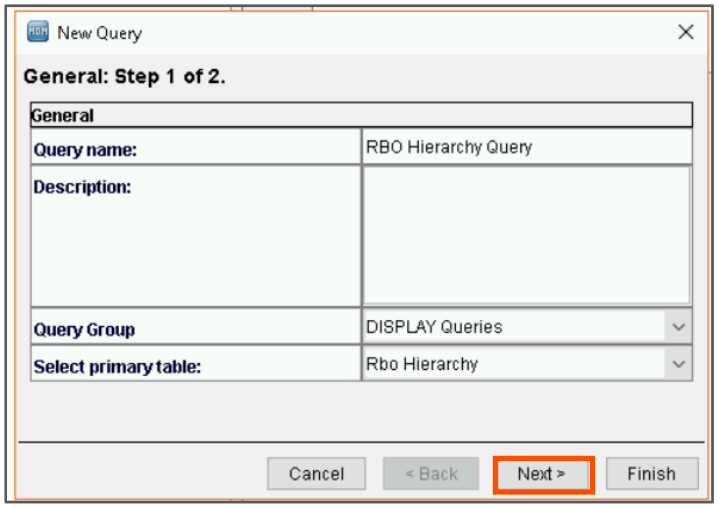
Use a Query to View the Hierarchy Objects

7. To create a query, in the **Model** workbench, open the **Queries** tool.
8. In the **Queries** pane, select the **DISPLAY Queries** group.
9. Right-click and select **New Query…**

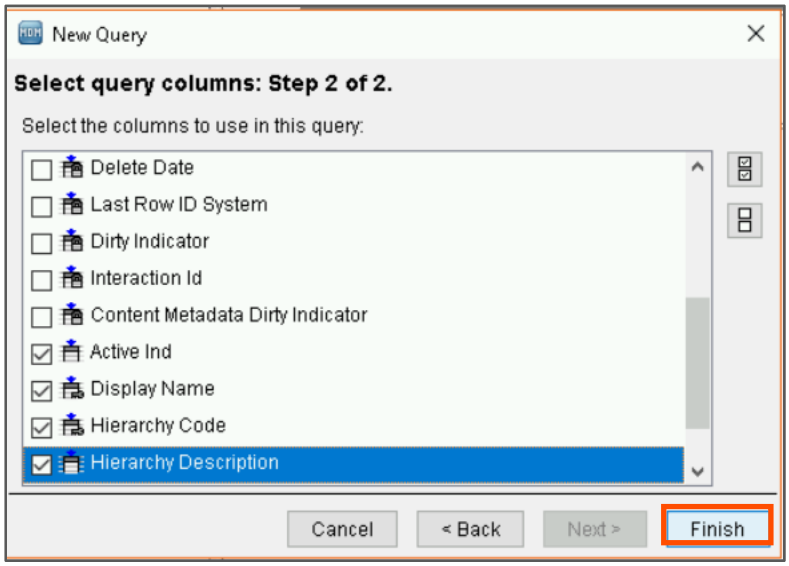


The **New Query Wizard** appears.

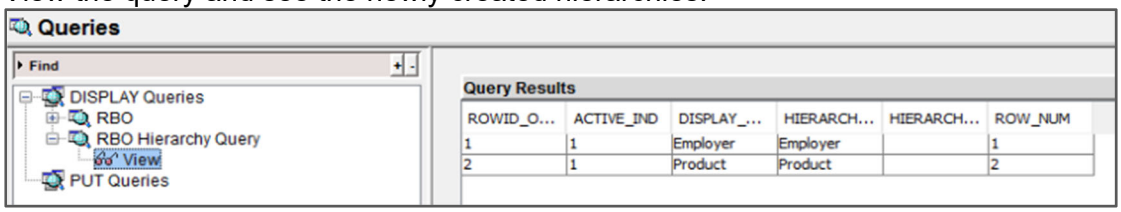
10. Name the query as **RBO Hierarchy Query**, select the Primary Table as **Rbo Hierarchy**, and click **Next**.



11. Select the columns: **Rowid Object**, **Hierarchy Code**, **Display Name**, **Hierarchy Description**, and **Active Ind**.
12. Click **Finish**.



13. View the query and see the newly created hierarchies.



This concludes the lab.

Additional Information

Edit an existing Hierarchy

To edit a Hierarchy, select the hierarchy and modify its properties.

Note: After you save the hierarchy, you cannot edit the Code attribute. If you need to change it, delete and re-create the hierarchy.

Delete an existing hierarchy

To delete a Hierarchy, right-click on the hierarchy and select **Delete Hierarchy**.

Module 17: Hierarchies, Relationships, and Relationship Types

Lab 17-2: Converting Base Objects to Relationship Base Objects

Overview:
Relationship types define the relationships between entity types and the appearance of the relationships when you view the hierarchy. In this lab, you will practice how to convert base objects to relationship base objects.

Objectives:

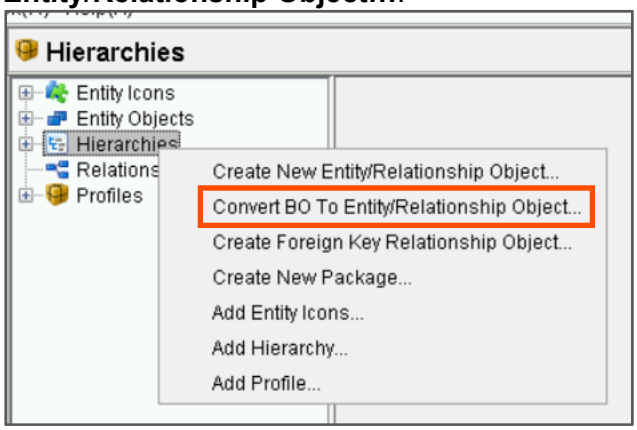
- Convert Base Objects to Relationship Base Objects

Duration:
10 minutes

Tasks

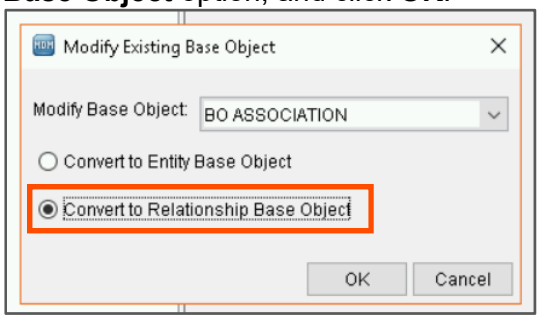
Convert Base Objects to Relationship Base Objects

1. From the **Model** workbench, select the **Hierarchies** tool.
2. Acquire a **Write Lock**, if necessary.
3. In the Hierarchies pane, right-click **Hierarchies**, and select **Convert BO To Entity/Relationship Object…**.



4. Ensure that you have **enabled** the **state management** for the **BO ASSOCIATION** object.

5. From the drop-down, select **BO ASSOCIATION**, select the **Convert to Relationship Base Object** option, and click **OK**.

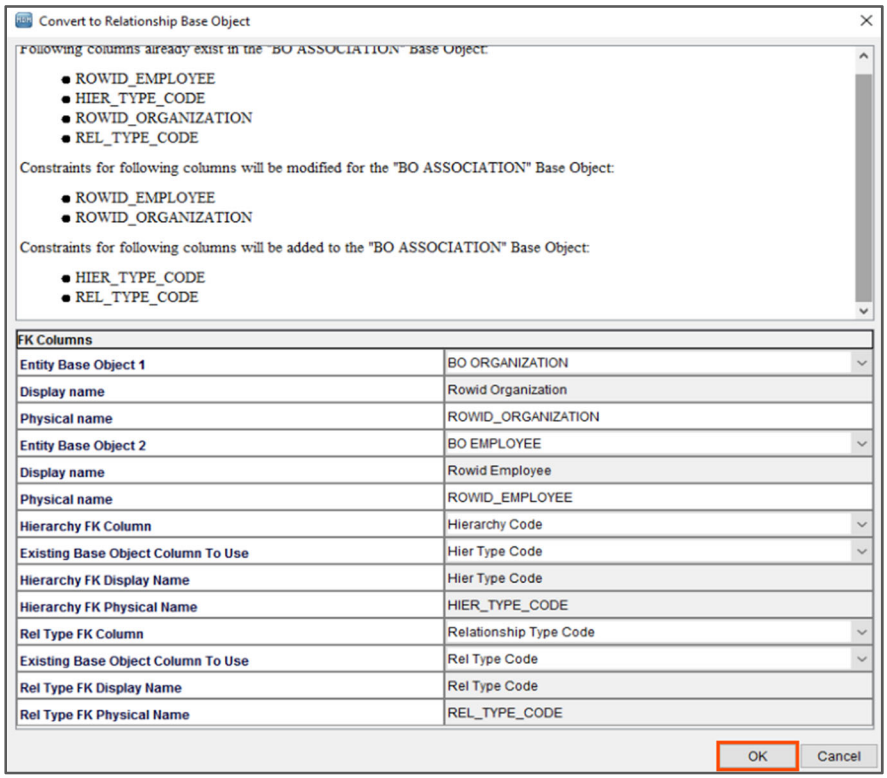


6. The **Convert to Relationship Base Object** window appears. Set the fields as mentioned below:

Entity Base Object 1:	BO ORGANIZATION
Display Name:	Rowid Organization
Physical Name:	ROWID_ORGANIZATION
Entity Base Object 2:	BO EMPLOYEE
Display Name:	Rowid Employee
Physical Name:	ROWID_EMPLOYEE
Hierarchy FK Column:	Hierarchy Code
Existing BO Column To Use:	Hier Type Code
Hierarchy FK Display Name:	Hier Type Code
Hierarchy FK Physical Name:	HIER_TYPE_CODE
Rel Type FK Column:	Relationship Type Code
Existing BO Column To Use:	Rel Type Code
Rel Type FK Display Name:	Rel Type Code
Rel Type FK Physical Name:	REL_TYPE_CODE

Recommended: If the dialog box is too narrow, the drop-down arrows of the fields will not be visible. In that case, drag it from the right to expand it.

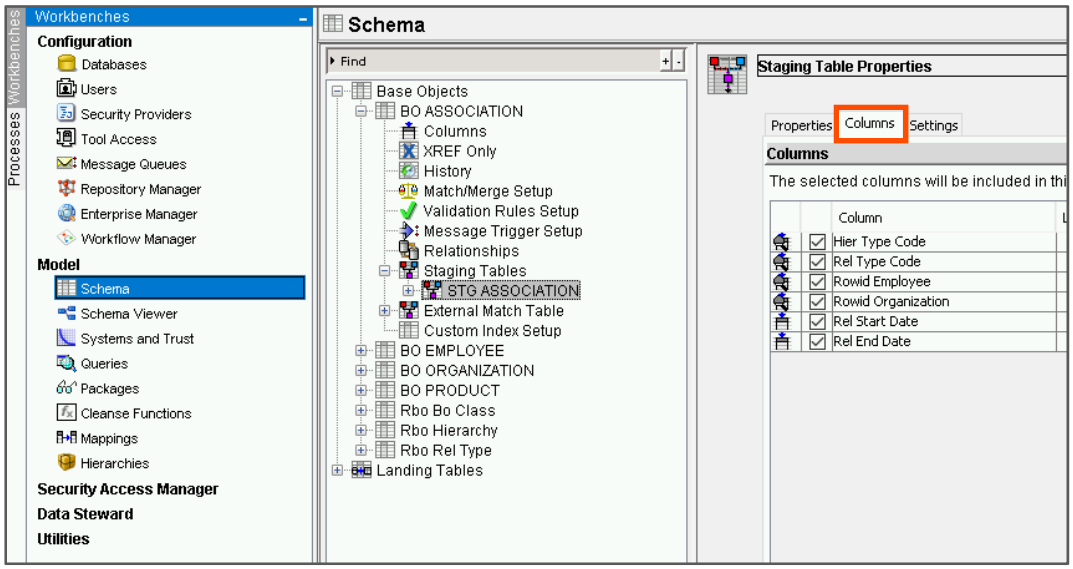
7. Click **OK**.



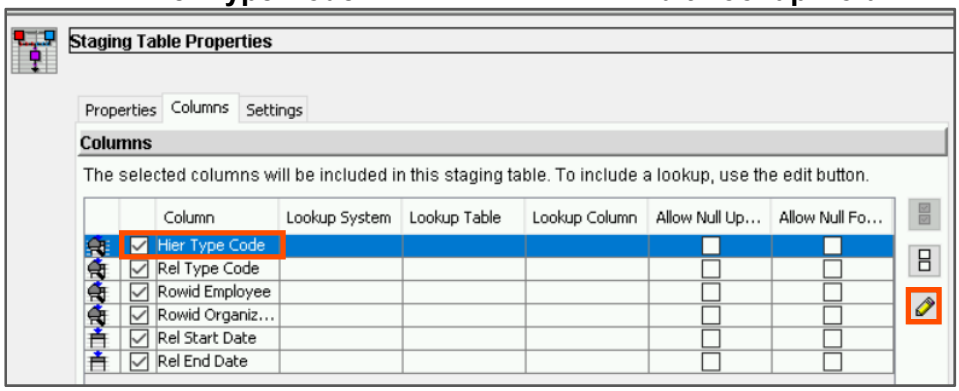
8. When the process completes, **BO ASSOCIATION** is visible under the **Relationship Objects** node.

9. Using the **Schema** tool, in the **Model** workbench, select the staging table **STG ASSOCIATION** from the **BO ASSOCIATION** base object.

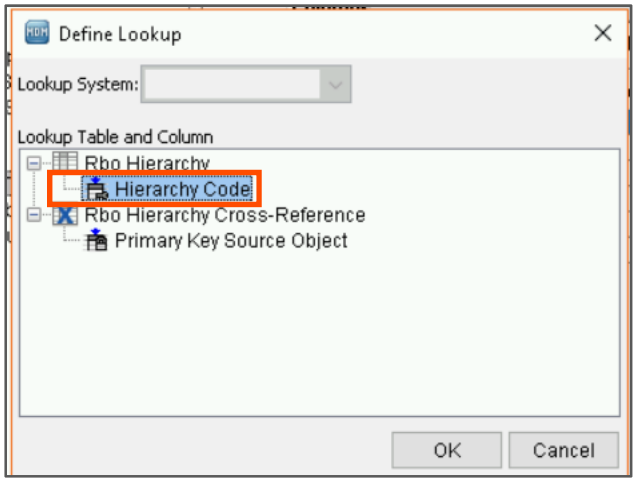
10. In the **Staging Table Properties** pane, select the **Columns** tab.



11. Select the **Hier Type Code** column and click the **Edit Lookup Column** icon.

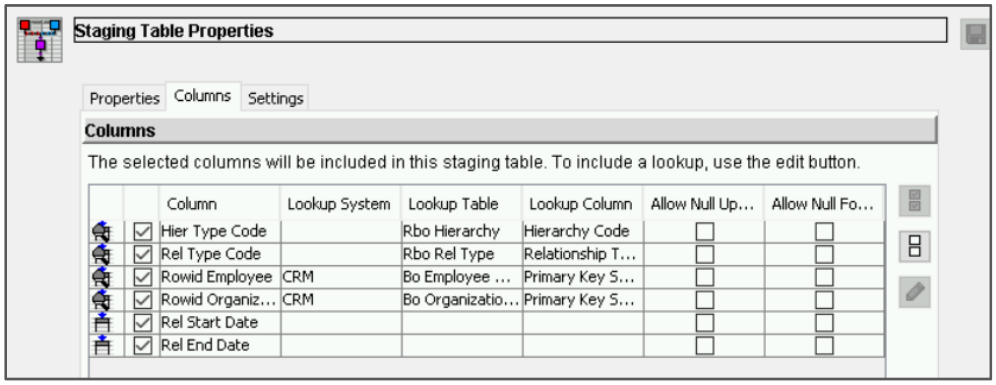


12. From the **Rbo Hierarchy** table, set the lookup to the **Hierarchy Code**.
13. Click **OK**.

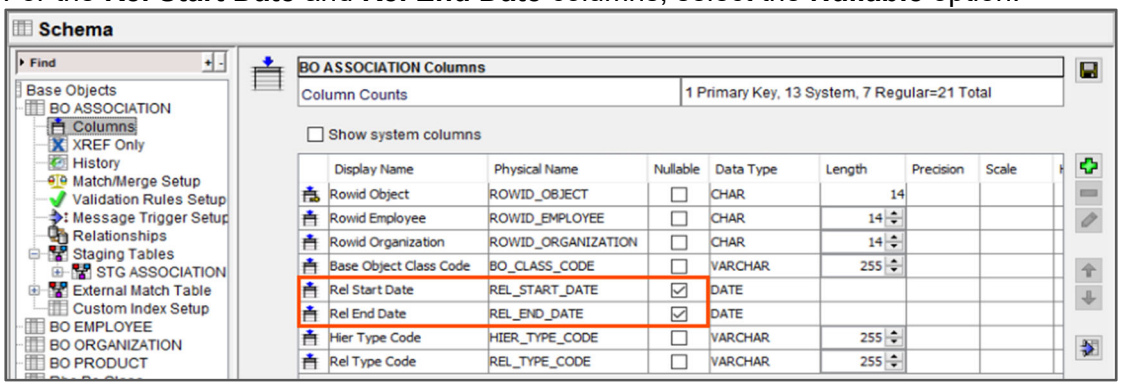


14. Similarly, for the **Rel Type Code** column, from the **Rbo Rel Type** lookup table, set the lookup to the **Relationship Type Code**.

15. For the **Rowid Employee** column, from the **BO Employee Cross-reference** table, set the lookup to the **Primary Key Source Object**.
16. For the **Rowid ORGANIZATION** column, from the **BO Organization Cross-reference** table, set the lookup to the **Primary Key Source Object**.
17. **Save** your work.



18. Click **BO_ASSOCIATION** > **Columns**.
19. For the **Rel Start Date** and **Rel End Date** columns, select the **Nullable** option.

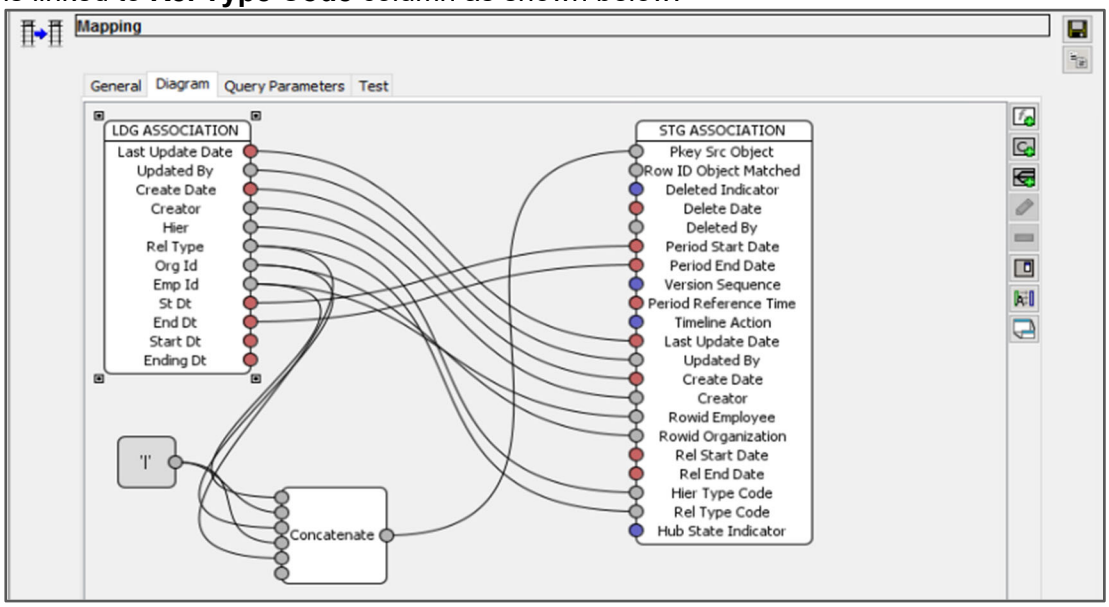


20. **Save** your work.
21. Under the Model workbench, select **Mappings**.
22. Select **STG ASSOCIATION** and select **Diagram**.

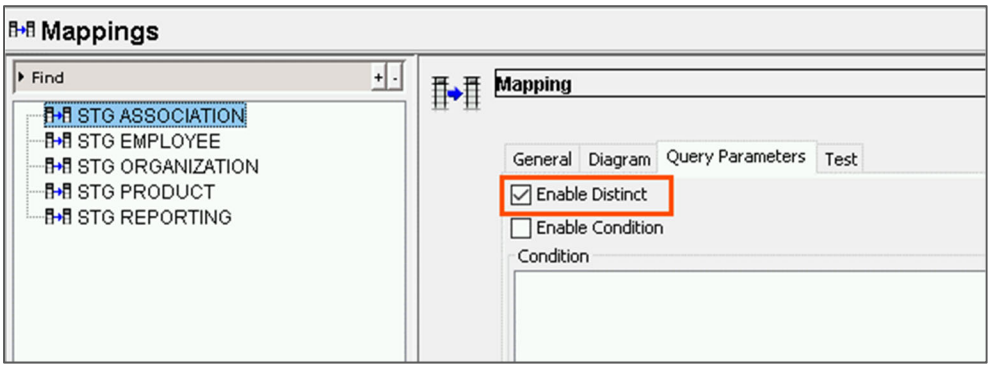
Now, adjust the mapping **STG ASSOCIATION** so that the landing table **St Dt** and **End Dt** columns map to the **Period Start Date** and **Period End Date** columns of the staging table respectively.

23. Delete the existing links for the **St Dt** and **End Dt** columns.
24. Add new links between **St Dt** and **Period Start Date** and **End Dt** and **Period End Date**.

25. Also, ensure that the **Hier** column is linked to the **Hier Type Code** column and **Rel Type** is linked to **Rel Type Code** column as shown below.



26. In the **Query Parameters** tab, select the **Enable Distinct** checkbox.



27. Similarly, enable this option for all the other mappings.
28. **Save** your work.

This concludes the lab.

Module 17: Hierarchies, Relationships, and Relationship Types

Lab 17-3: Creating Foreign Key Relationship Objects

Overview:
Relationship types define the relationships between entity types and the appearance of the relationships when you view the hierarchy. In this lab, you will create a different type of relationship known as Foreign Key Relationship Objects.

Objective:

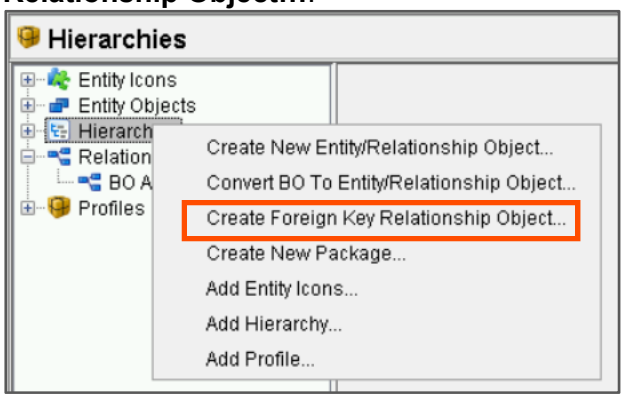
- Create Foreign Key Relationship Objects

Duration:
5 minutes

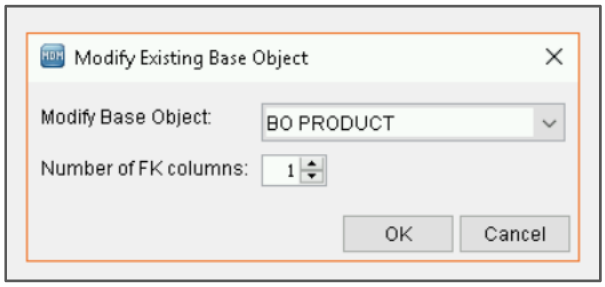
Tasks

Create Foreign Key Relationship Objects

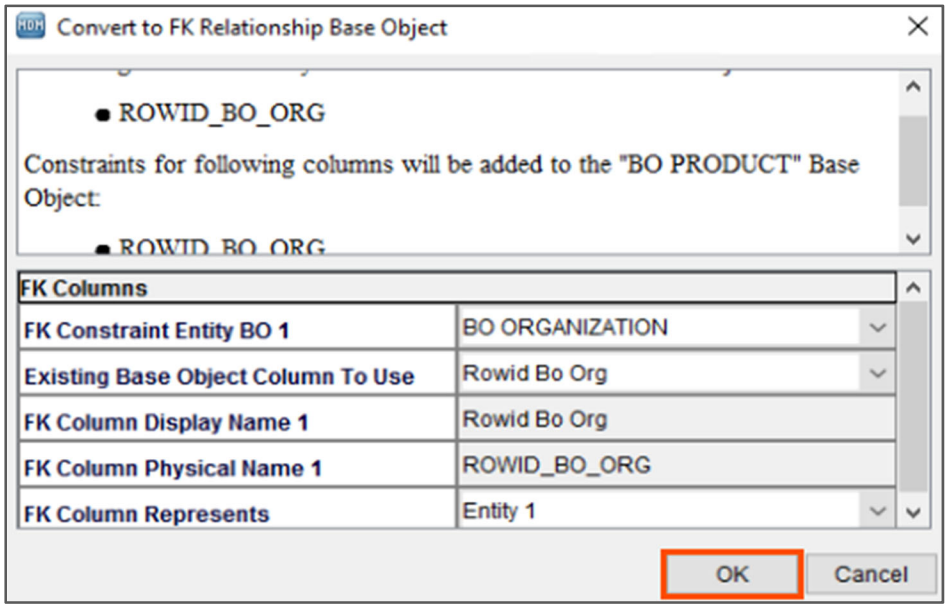
1. From the **Model** workbench, select the **Hierarchies** tool.
2. From the console main menu, acquire a **Write Lock**, if necessary.
3. In the **Hierarchies** pane, right-click **Hierarchies**, and select **Create Foreign Key Relationship Object…**.



4. From the drop-down, select **BO PRODUCT**, retain the **Number of FK columns** as **1**, and click **OK**.



5. From the **FK Constraint Entity BO 1** drop-down, select **BO ORGANIZATION** and from the **Existing Base Object Column To Use** drop-down, select **Rowid Bo Org**.
6. From the **FK Column Represents** drop-down, select **Entity 1**, as the relationship will be from the organization to the product (implying that the product is **Entity 2**).
7. Click **OK**.

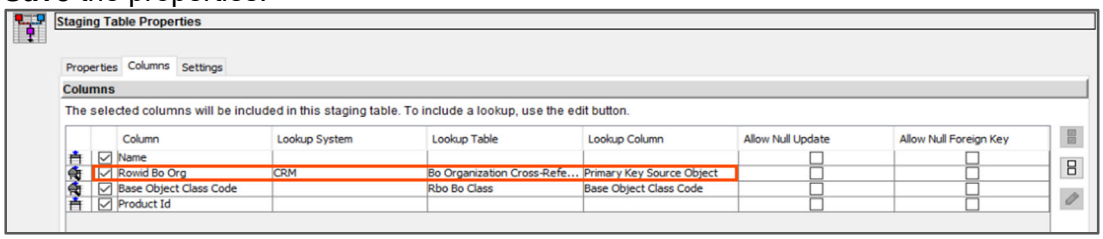


Upon completion, **BO PRODUCT (Rowid Bo Org)** will be visible under the **Relationship Objects** node. Foreign key relationship objects are followed by the foreign key column display name.

Important: Hierarchy Manager has a constraint that a foreign key relationship can only belong to one hierarchy.

8. From the Model Workbench, open **Schema** and select the staging table for **BO PRODUCT**.
9. Select the **Columns** tab.

10. For the **Rowid Bo Org**, add the lookup as **Primary Key Source Object**.
11. **Save** the properties.



This concludes the lab.

Module 17: Hierarchies, Relationships, and Relationship Types

Lab 17-4: Creating New Relationship Objects

Overview:
Relationship types define the relationships between entity types and the appearance of the relationships when you view the hierarchy. In this lab, you will explore how to create new relationship objects.

Objective:

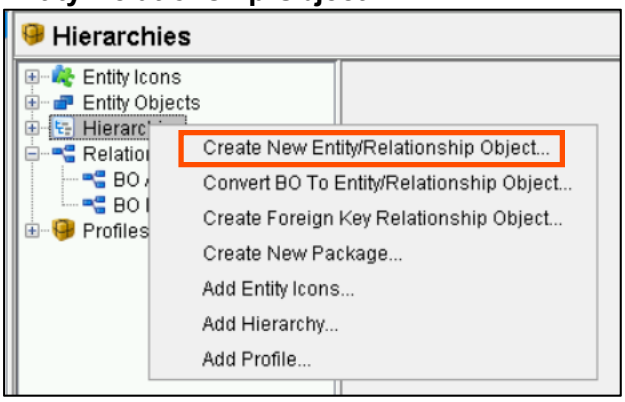
- Creating New Relationship Objects

Duration:
15 minutes

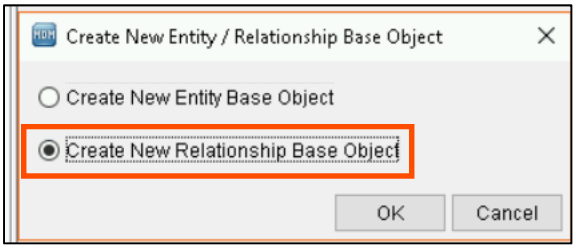
Tasks

Create New Relationship Objects

1. From the **Model** workbench, select the **Hierarchies** tool.
2. From the console main menu, acquire the **Write Lock**, if necessary.
3. In the **Hierarchies** pane, right-click **Hierarchies,** and select **Create New Entity/Relationship Object…**.



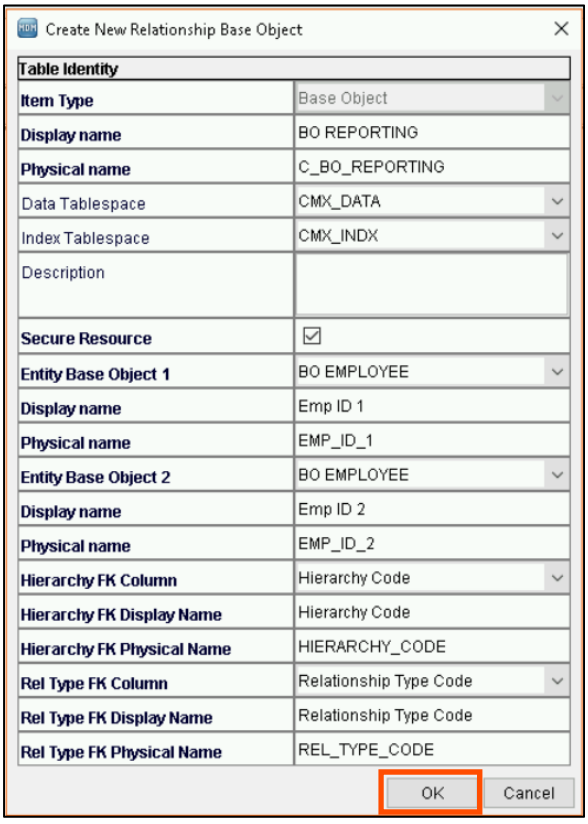
4. Select **Create New Relationship Base Object** and click **OK**.



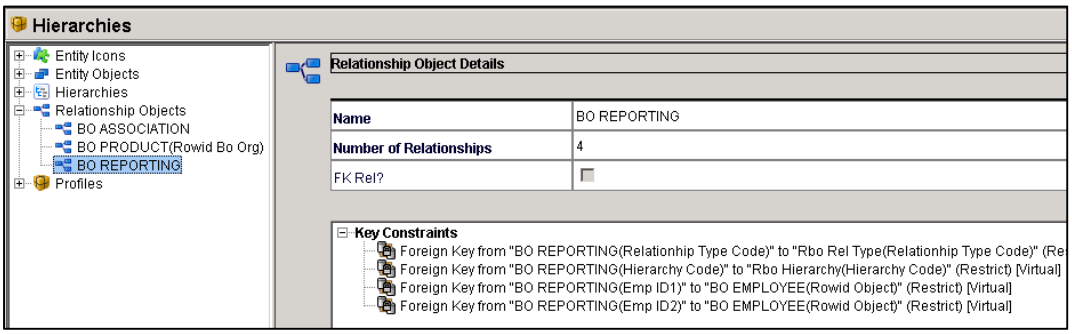
5. In the **Create New Relationship Base Object** window, populate the fields as mentioned below:

Display Name:	BO REPORTING
Physical Name:	C_ BO_REPORTING
Data Tablespace:	CMX_DATA
Index Tablespace:	CMX_INDX
Entity Base Object 1:	BO EMPLOYEE
Display Name:	Emp ID 1
Physical Name:	EMP_ID_1
Entity Base Object 2:	BO EMPLOYEE
Display Name:	Emp ID 2
Physical Name:	EMP_ID_2
Hierarchy FK Column:	Hierarchy Code
Hierarchy FK Display Name:	Hierarchy Code
Hierarchy FK Physical Name:	HIERARCHY_CODE
Rel Type FK Column:	Relationship Type Code
Rel Type FK Display Name:	Relationship Type Code
Rel Type FK Physical Name:	REL_TYPE_CODE

6. Click **OK**.



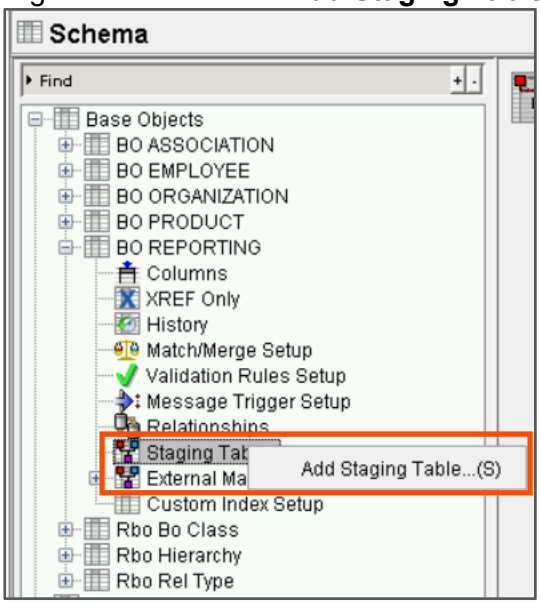
7. Upon completion, the new relationship object **BO REPORTING** will be visible under the **Relationship Objects** node.



Note: In the **Schema** tool also, you will see that the base object **BO REPORTING** has been created.

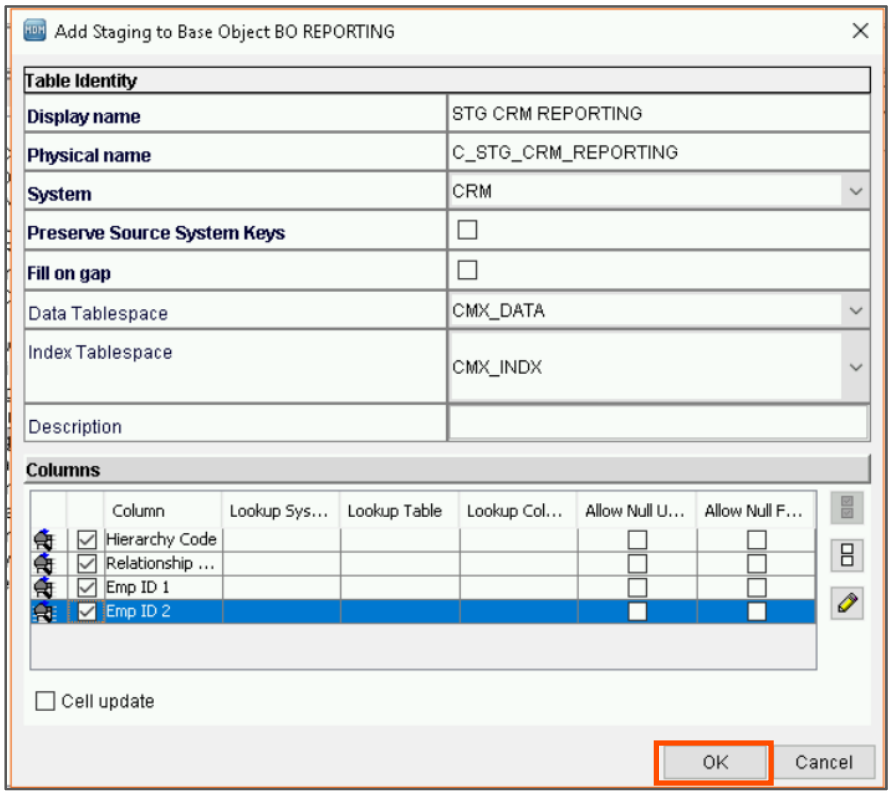
Create a Staging Table STG CRM REPORTING

8. Navigate to **Schema** > **Base Objects** > **BO REPORTING** > **Staging Tables**.
9. Right-click and select **Add Staging Table…(S)**.

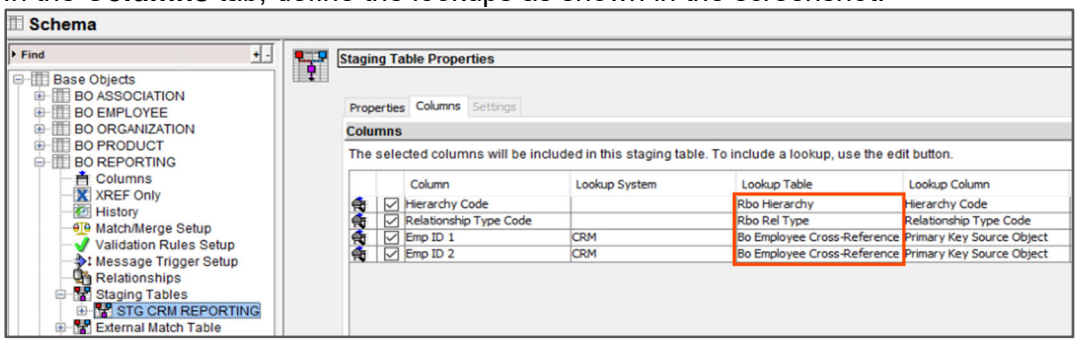


10. Enter the values as shown in the screenshot.

11. Select all the columns below and click **OK**.



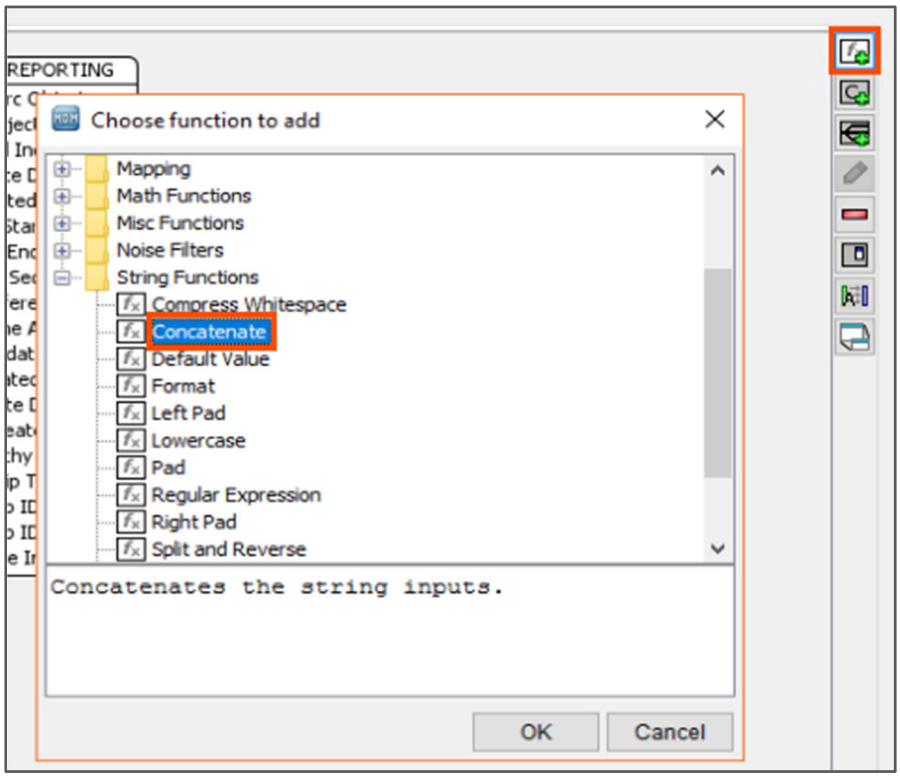
12. Expand **Staging Table** and select **STG CRM REPORTING**.
13. In the **Columns** tab, define the lookups as shown in the screenshot.



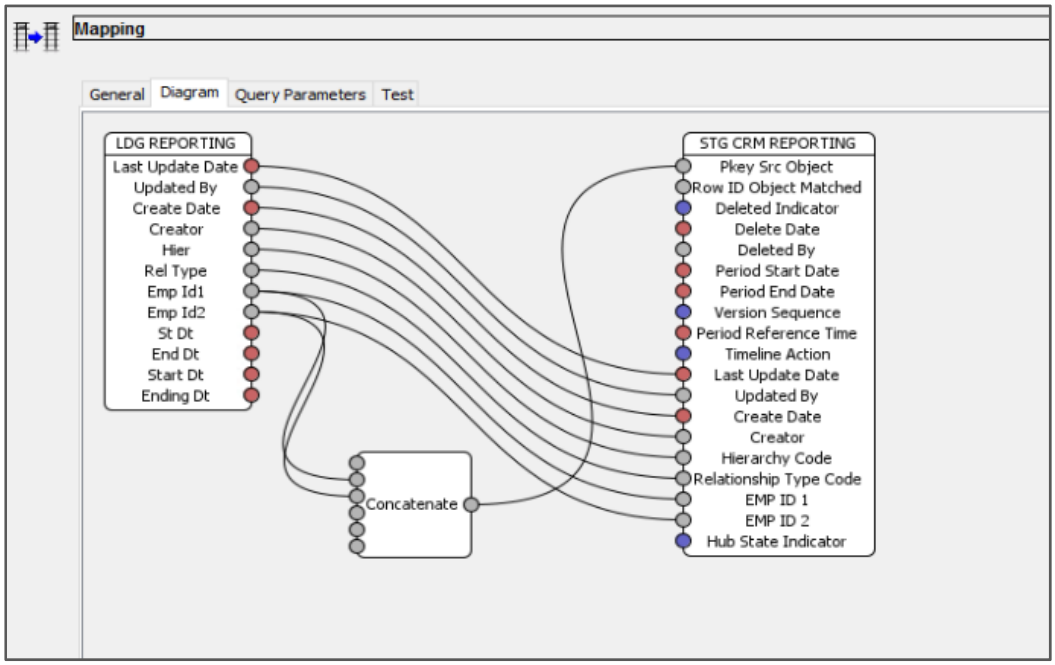
14. **Save** your work.
15. In the **Mappings** tool, add a new mapping, **STG REPORTING**, with the landing and staging tables as **LDG REPORTING** and **STG CRM REPORTING** respectively.
16. Click **OK**.
    Note: If you do not see the new mapping in the list, release and acquire the lock again.

17. **Automap** the columns.

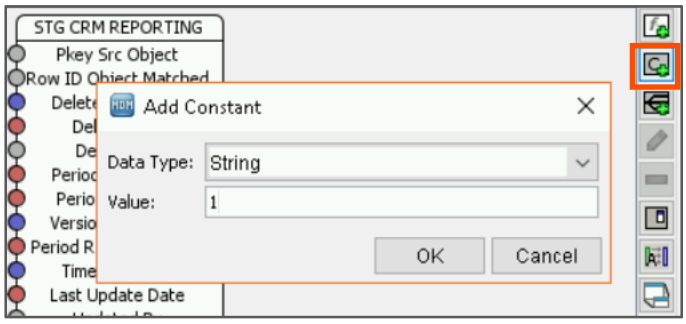
18. Select the **Add Function** option, and from the **String** section, select **Concatenate** and click **OK**.



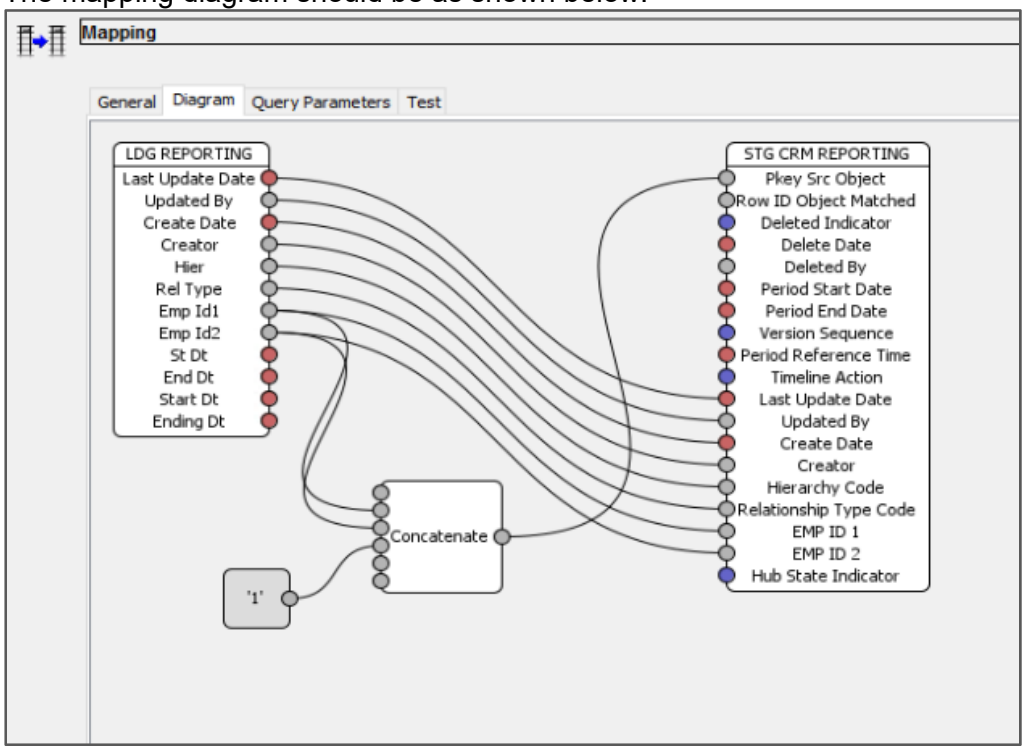
19. Link the **Emp Id1**, **Emp Id2** from the landing table and **Pkey Src Object** column from the staging table, to the **Concatenate** function. The mapping should appear as shown below.



20. Select the **Add Constant** option and enter **1** in the Value field.
21. Click **OK**.



The mapping diagram should be as shown below:



22. **Save** your work.

This concludes the lab.

Module 17: Hierarchies, Relationships, and Relationship Types

Lab 17-5: Creating, Editing, and Deleting Relationship Types

Overview:
Relationship types define the relationships between entity types and the appearance of the relationships when you view the hierarchy. In this lab, you will explore how to create, edit, and delete relationship types.

Objective:

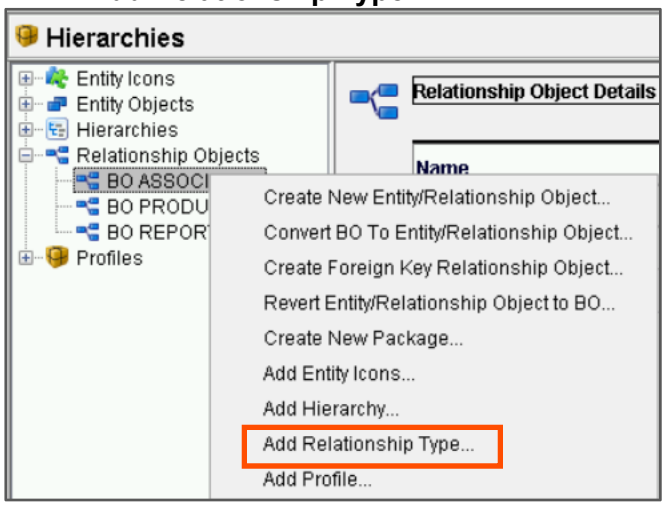
- Create, edit, and delete Relationship types

Duration:
5 minutes

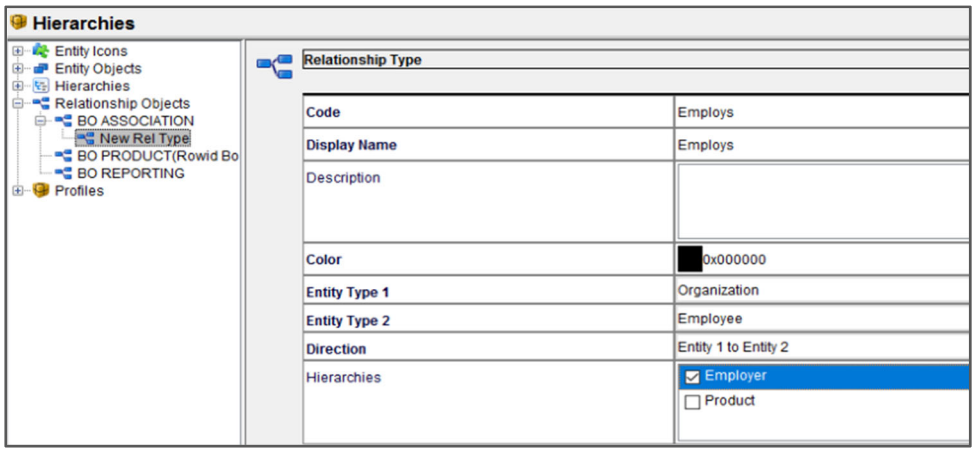
Tasks

Create, Edit and Delete Relationship types

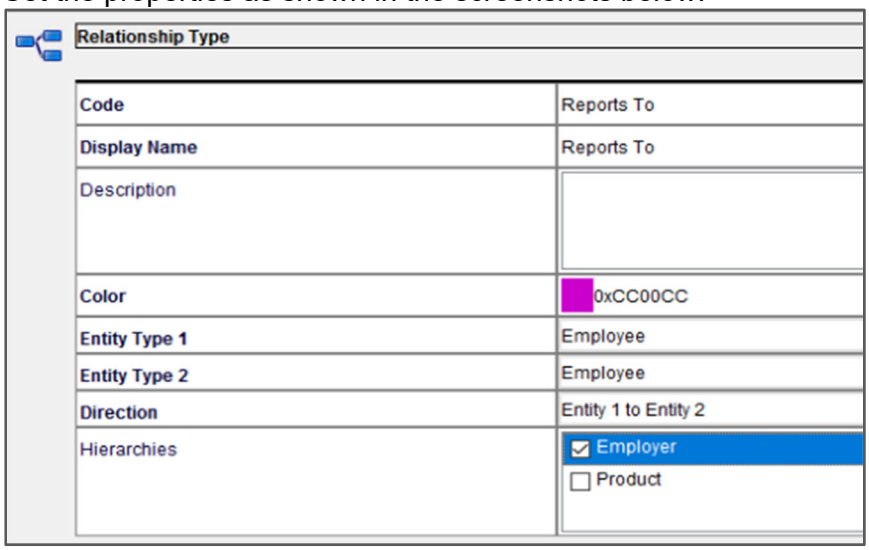
1. From the **Model** workbench, select **Hierarchies**.
2. Acquire a **Write Lock**, if necessary.
3. In the **Hierarchies** pane, right-click on the relationship object **BO ASSOCIATION** and select **Add Relationship Type…**.



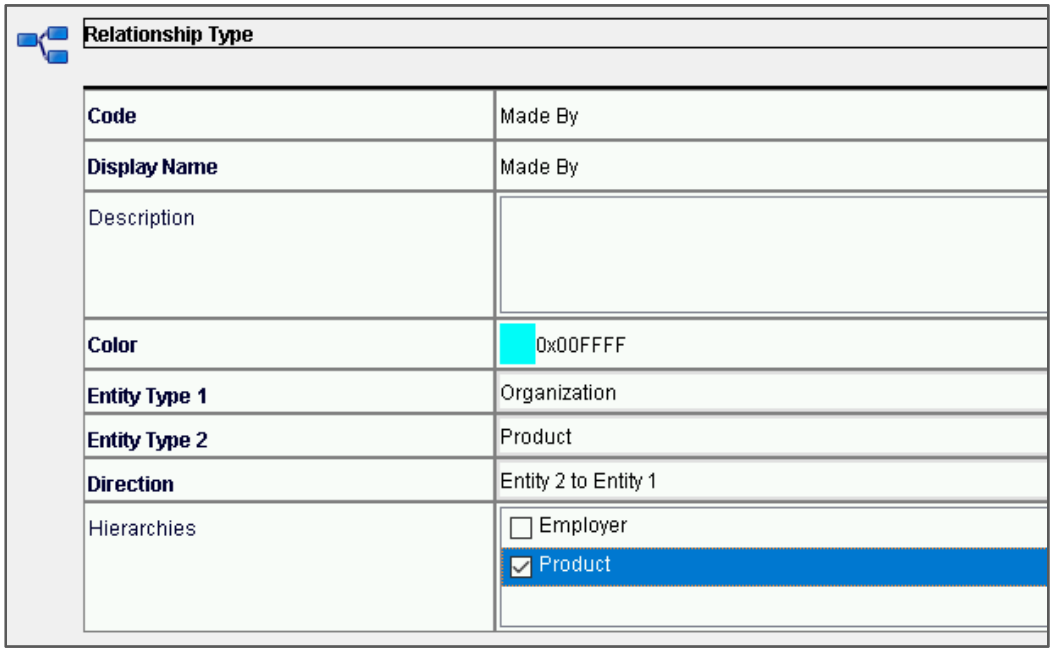
4. Enter the following details:
   Code and Display Name: **Employs**
   Color: Set **any** color
   Entity Type 1: **Organization**
   Entity Type 2: **Employee**
   Direction: **Entity 1 to Entity 2**
   Hierarchies**: Employer**



5. **Save** your work.
6. Add the relationship type **Reports To** to the BO REPORTING relationship object.
   Tip: Right-click **BO REPORTING** relationship object and select **Add Relationship Type**.
7. Choose a direction type that is consistent with the language used for the relationship name. The **Reports To** relationship relates two instances of the entity type **Employee**; **Employee 1** reports to **Employee 2** so the option should be **Entity 1 to Entity 2**.
   Set the properties as shown in the screenshots below:



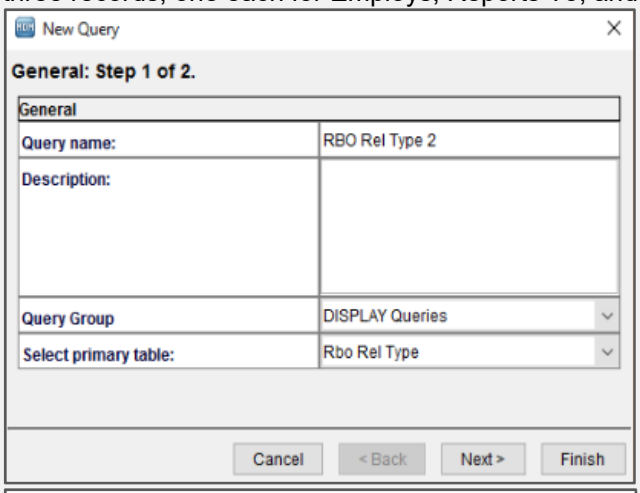
8. Add the relationship type **Made By** to the BO PRODUCT relationship object.
   Tip: Right-click **BO PRODUCT** relationship object and select **Add Relationship Type**.
9. For the **Made By** relationship, **Entity Type 2** (Product) is Made By **Entity Type 1** (Organization), so the direction is **Entity 2 to Entity 1**, and select **Product** for Hierarchies.

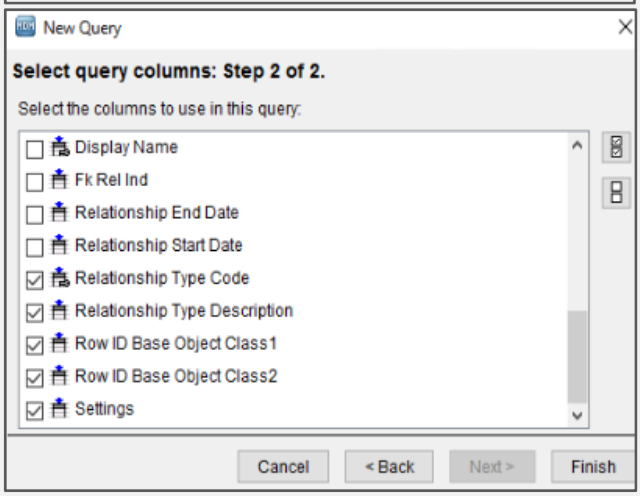


This concludes the lab.

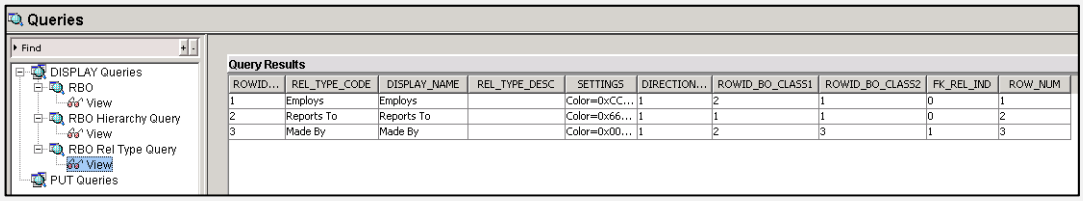
Optional

1. Relationship Types are stored in the Rbo Rel Type base object. Use the Queries tool in the Model workbench to create a query **RBO Rel Type 2** to view the data. There should now be three records, one each for Employs, Reports To, and Made By.





Note: Ensure that the Rowid Object column is also selected.



2. To edit Relationship Types, select the relationship type and modify the properties. However, the Code attribute cannot be edited once the Relationship Type is saved. To make any changes, the Relationship Type must be deleted and re-created with the new code.
3. To delete Relationship Types, right-click on the relationship type and select **Delete Relationship Type…**.

Module 18: Packages

Lab 18-1: Creating Relationship Object Packages

Overview:
In MDM, “packages” define the data that you can access in the Hierarchy Manager. You use the Hierarchies tool to create the hierarchy packages and associated queries. When you create the package, you select which columns are available to configure access for. You cannot configure a column to be available in the Hierarchy Manager if the column is not part of the package.

In this lab, you will create packages for previously created relationship objects.

Objective:

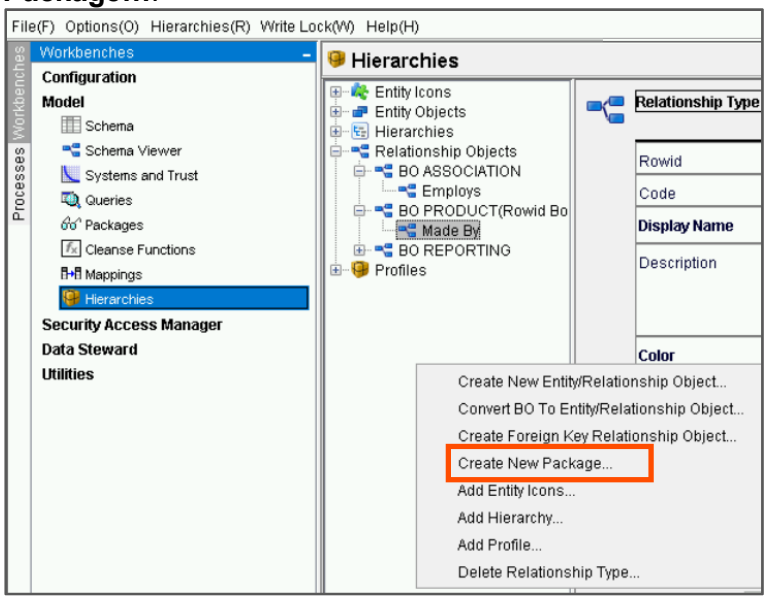
- Create a package for a relationship object

Duration:
15 minutes

Tasks

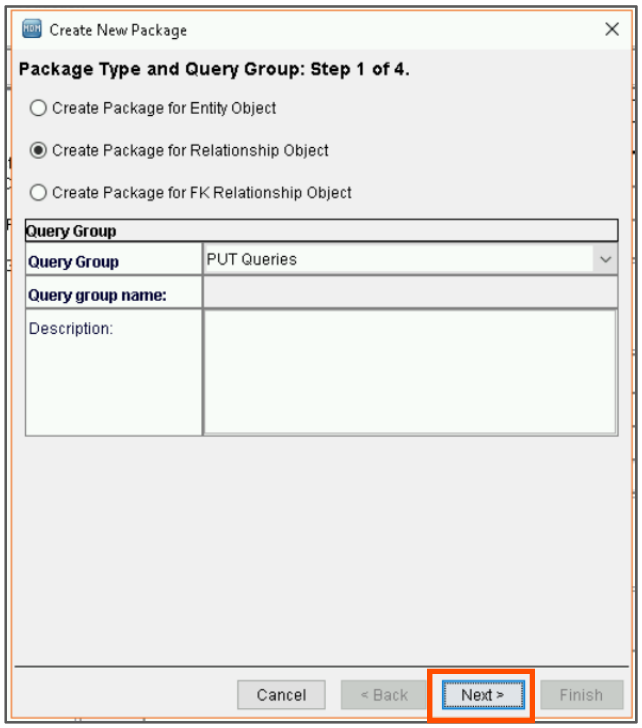
Create a Package for a Relationship Object

1. From the **Model** workbench, select the **Hierarchies** tool.
2. Acquire a **Write Lock**, if necessary.
3. In the **Hierarchies** pane, right-click anywhere in the pane and select **Create New Package…**.



4. Select the **Create Package for Relationship Object** option and select the **PUT Queries** Query Group.

5. Click **Next**.

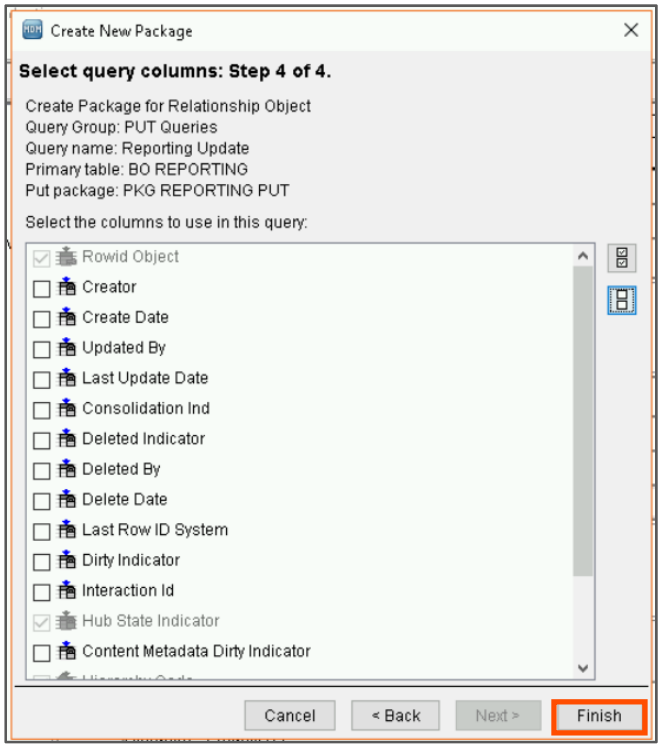


Note: The “PUT Queries” and “DISPLAY Queries” query groups have been defined in the preparation of this course. If no query groups exist or if you don’t see an appropriate query group, then you would select the “**New Query Group**” option and create the desired query group.

6. Set the Query Name to **Reporting Update** and select the primary table as **BO REPORTING**.
7. Click **Next**.

8. Set the Display Name to **PKG REPORTING PUT** and select the **Enable put** option.
9. Click **Next**.

10. Clear all the columns, retaining the default grayed-out selected columns.
11. Click **Finish**.



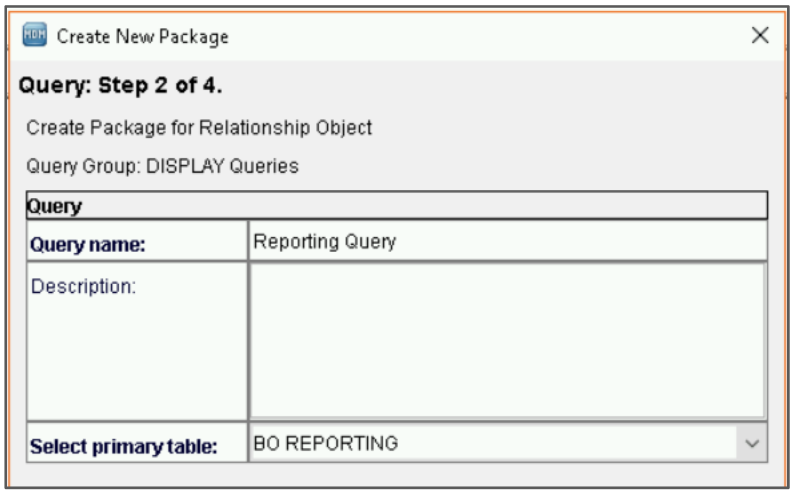
The Put reporting package is created. Now, you will create the corresponding Display package.

12. In the **Hierarchies** pane, right-click anywhere on the pane and select **Create New Package…**.

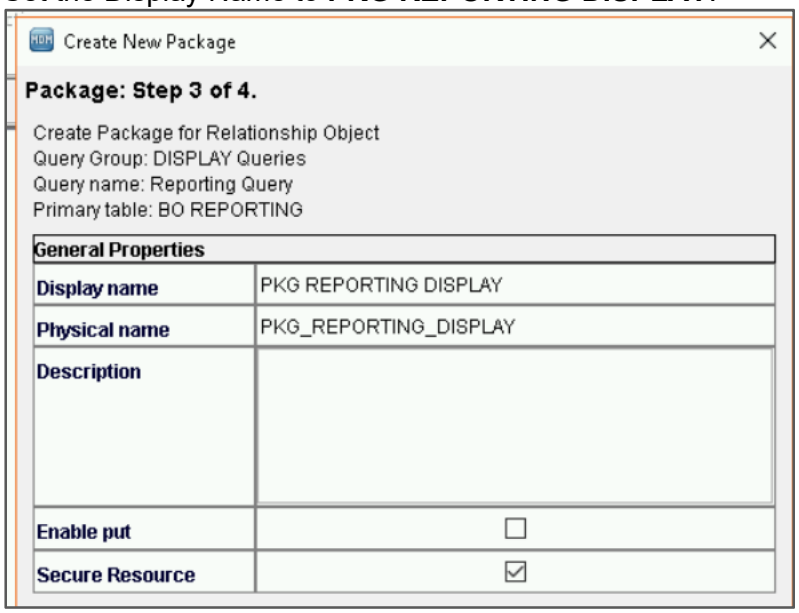
13. Select the **Create Package for Relationship Object** option and select the **DISPLAY Queries** Query Group.
14. Click **Next**.

15. Set the Query Name to **Reporting Query** and select the primary table as **BO REPORTING**.

16. Click **Next**.

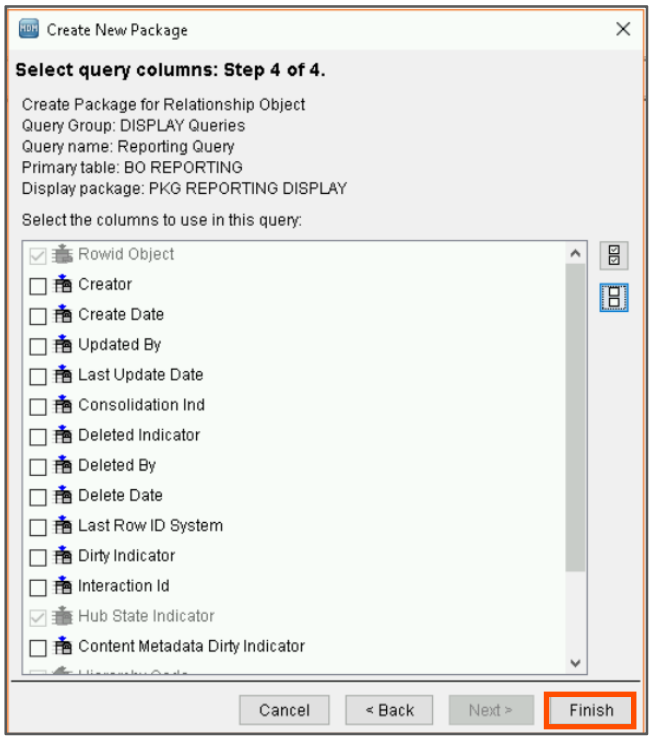


17. Set the Display Name to **PKG REPORTING DISPLAY**.

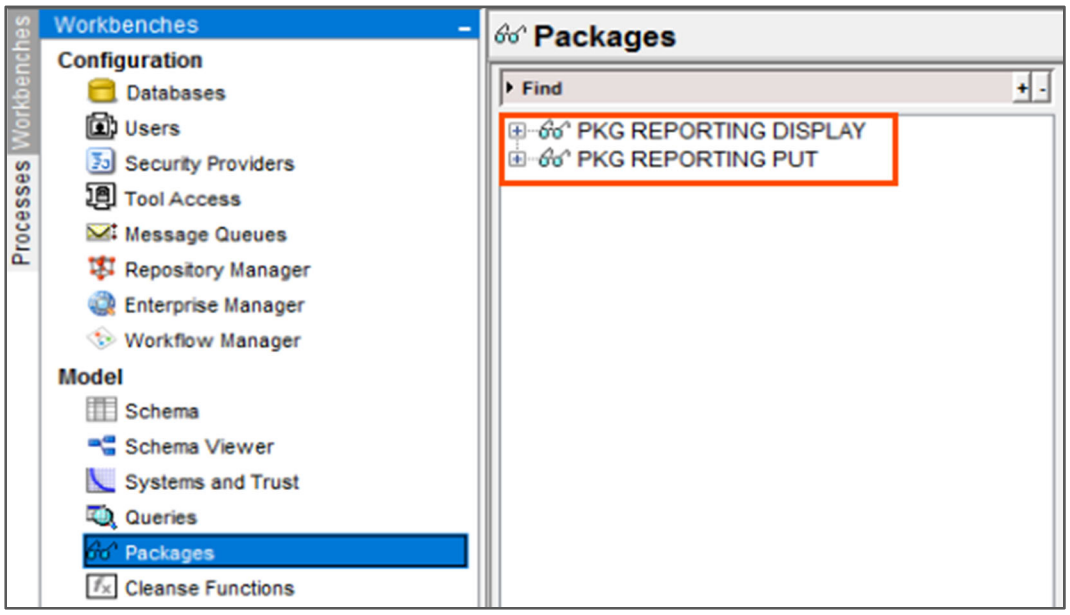


18. Click **Next**.
19. Clear all the columns, retaining the default grayed-out selected columns.

20. Click **Finish**.



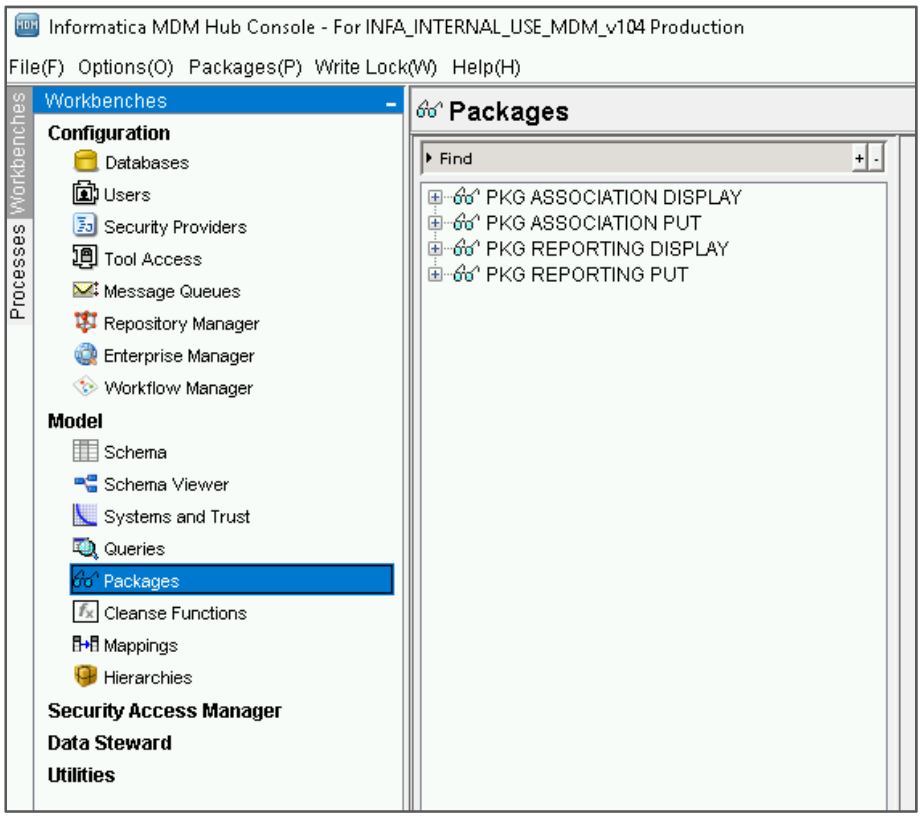
21. Navigate to the **Model** workbench and select **Packages**. The packages **PKG REPORTING PUT** and **PKG REPORTING DISPLAY** are now visible.



23. Similarly, create the relationship object PUT and DISPLAY packages for the **BO ASSOCIATION** relationship object:

Query	Package
Association Update	PKG ASSOCIATION PUT
Association Query	PKG ASSOCIATION DISPLAY

24. Confirm that the packages and queries have been created in the **Packages** tool (Remember that if you expand the package node you will see the associated query).



This concludes the lab.

Module 18: Packages

Lab 18-2: Creating FK Relationship Object Packages

Overview:
In this lab, you will create packages for Foreign Key Relationship Objects.

Objective:

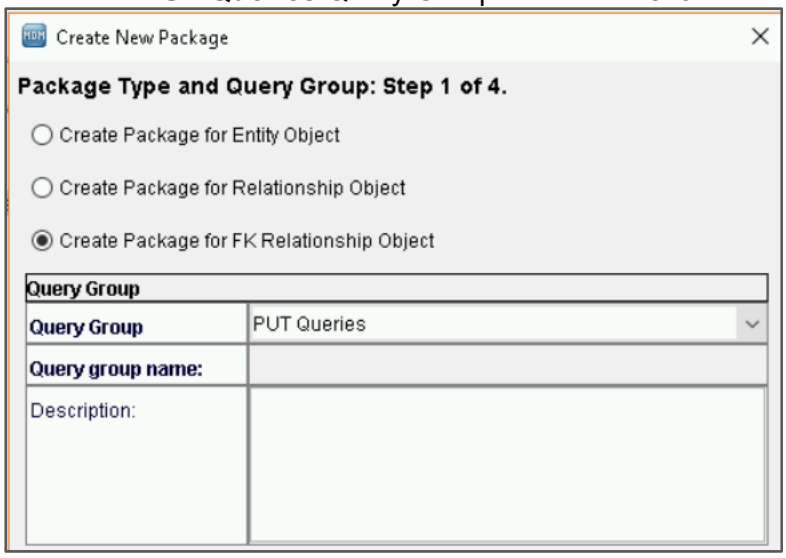
- Create a package for a foreign key relationship object

Duration:
15 minutes

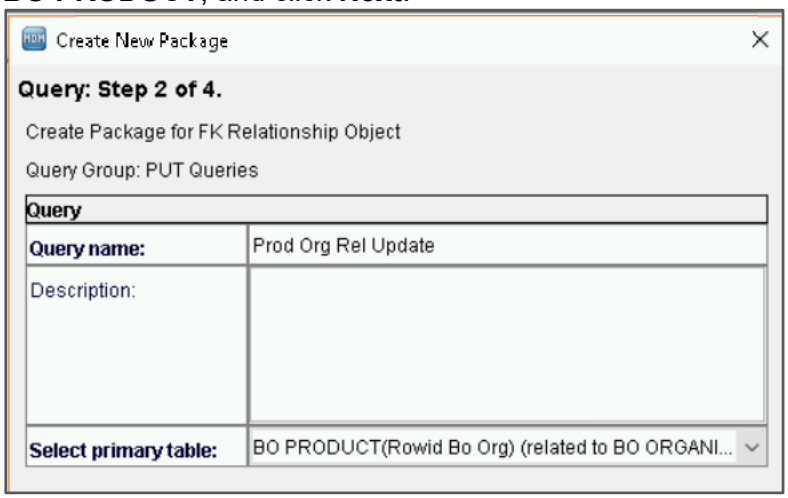
Tasks

Create a Package for a Foreign Key Relationship Object

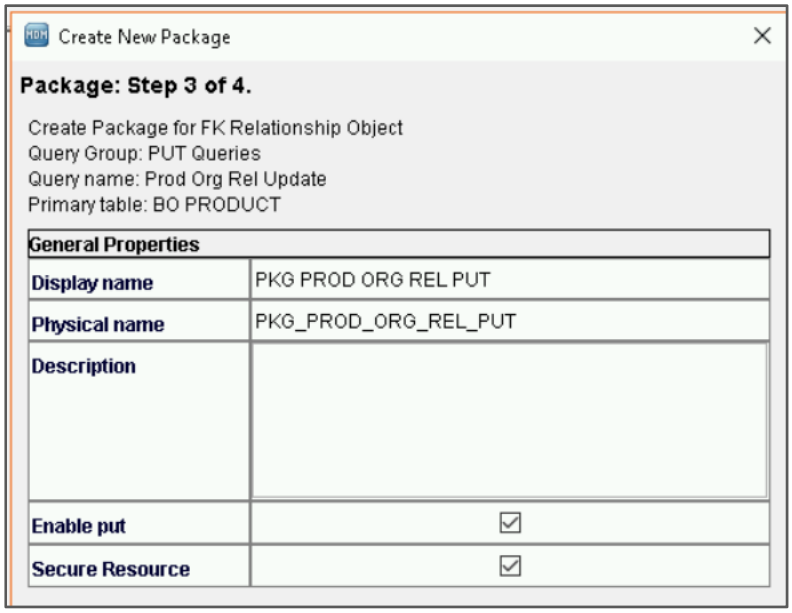
1. From the **Model** workbench, select the **Hierarchies** tool.
2. Acquire a **Write Lock**, if necessary.
3. In the **Hierarchies** pane, right-click anywhere in the pane and select **Create New Package…**.
4. Select the **Create Package for FK Relationship Object** option.
5. Select the **PUT Queries** Query Group and click **Next**.



6. Enter the Query name as **Prod Org Rel Update**, select the primary table **BO PRODUCT**, and click **Next**.

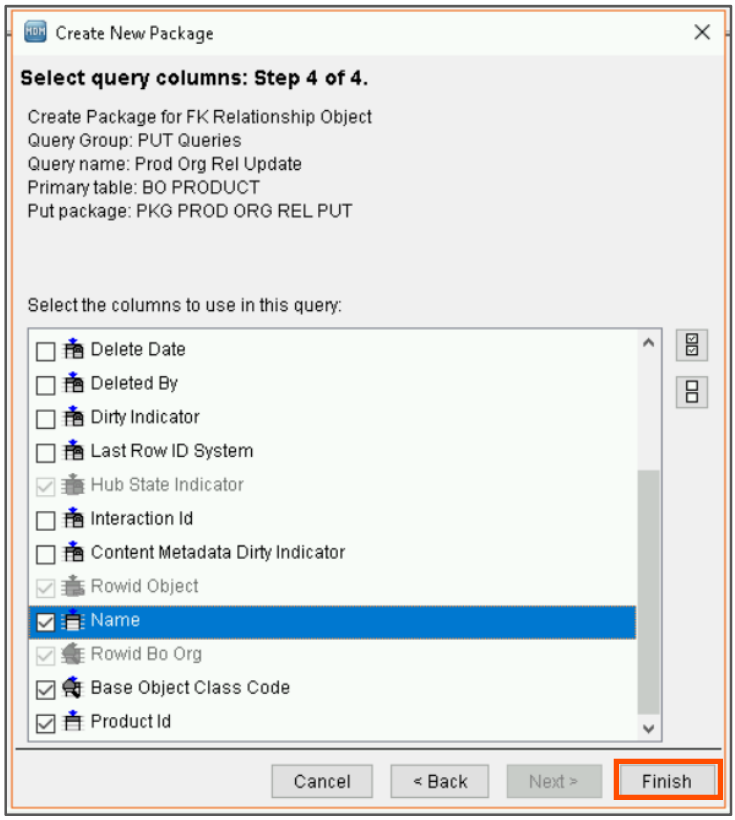


7. Enter the Display Name as **PKG PROD ORG REL PUT**, select **Enable put**, and click **Next**.



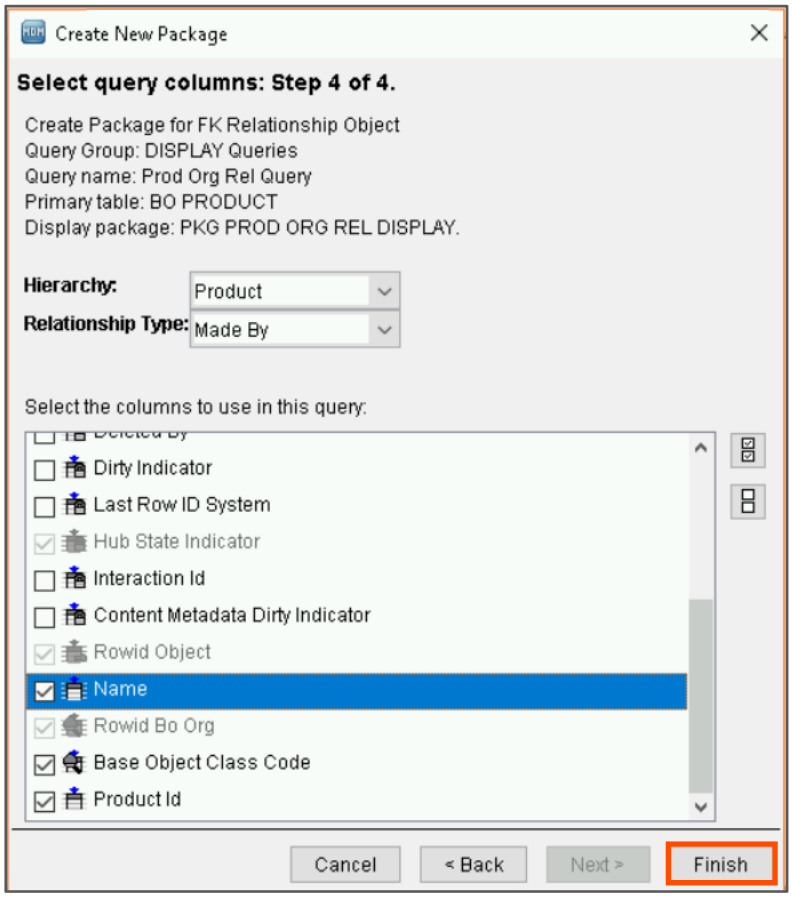
8. Clear all the columns retaining only the selected defaults.

9. Select additional columns **Base Object Class Code**, **Name**, and **Product Id** and click **Finish**.



10. Now, create the corresponding **DISPLAY** package for the FK relationship object **BO PRODUCT**.
    Note: The Step 4 in the wizard to define display packages for FK relationship objects is slightly different.
11. Enter the query name as **Prod Org Rel Query** and the package name as **PKG PROD ORG REL DISPLAY**.
12. In the next step, clear all the columns, and select the **Base Object Class Code**, **Name**, and **Product Id** columns for the query.

13. Also, set the **Hierarchy** to **Product** and the **Relationship Type** to **Made By** and click **Finish**.



Note: If there were multiple hierarchies and relationship types that were relevant to this FK relationship object, then you would have selected the ones appropriate for this package.

This concludes the lab.

Module 18: Packages

Lab 18-3: Creating Entity Object Packages

Overview:
In this lab, you will create packages for Entity Objects.

Objective:

- Create Entity Object packages

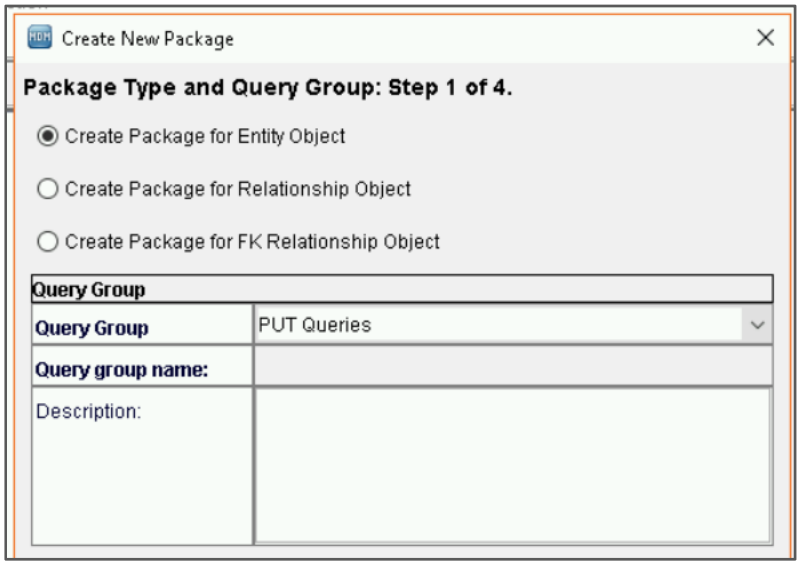
Duration:
20 minutes

Tasks

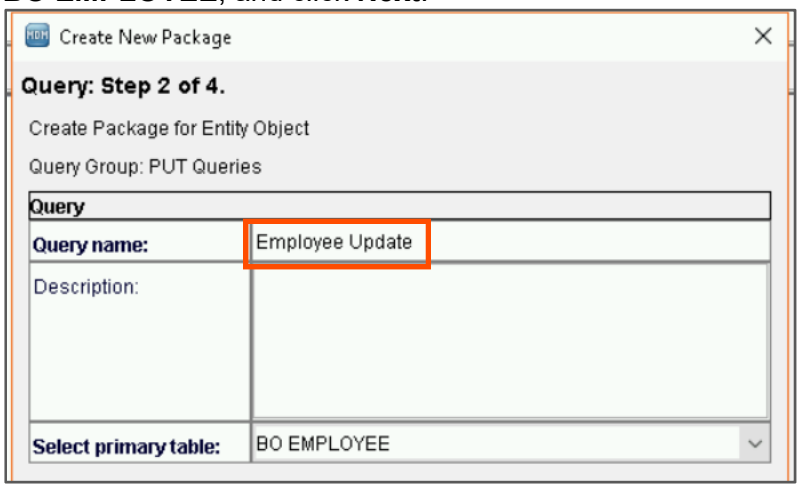
Create Entity Object Packages

You have created the display and put packages for the relationship objects, now you will create the display and put packages for the entity objects.

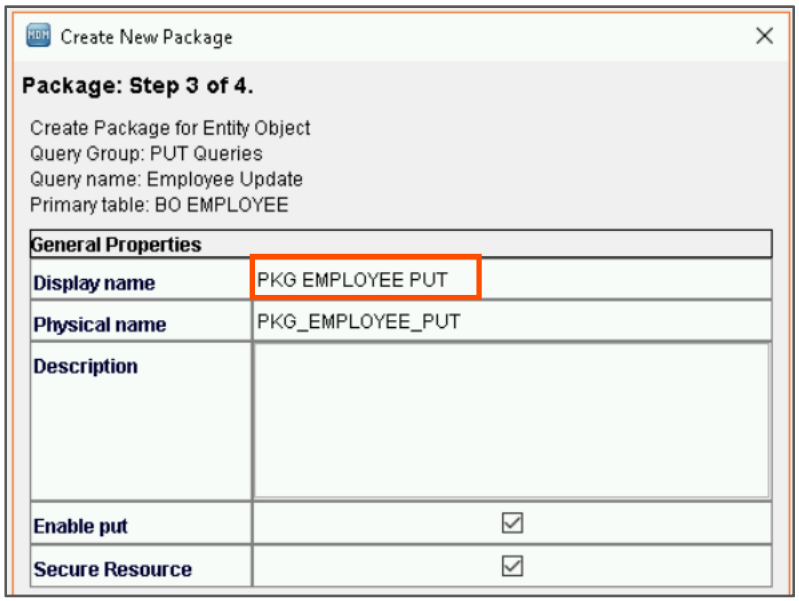
1. From the **Model** workbench, select the **Hierarchies** tool.
2. Acquire a **Write Lock**, if necessary.
3. In the **Hierarchies** pane, right-click anywhere on the pane and select **Create New Package…**.
4. Select **Create Package for Entity Object** and the **PUT Queries** Query Group.
5. Click **Next**.



6. Enter the Query name as **Employee Update**, select the primary table **BO EMPLOYEE**, and click **Next**.

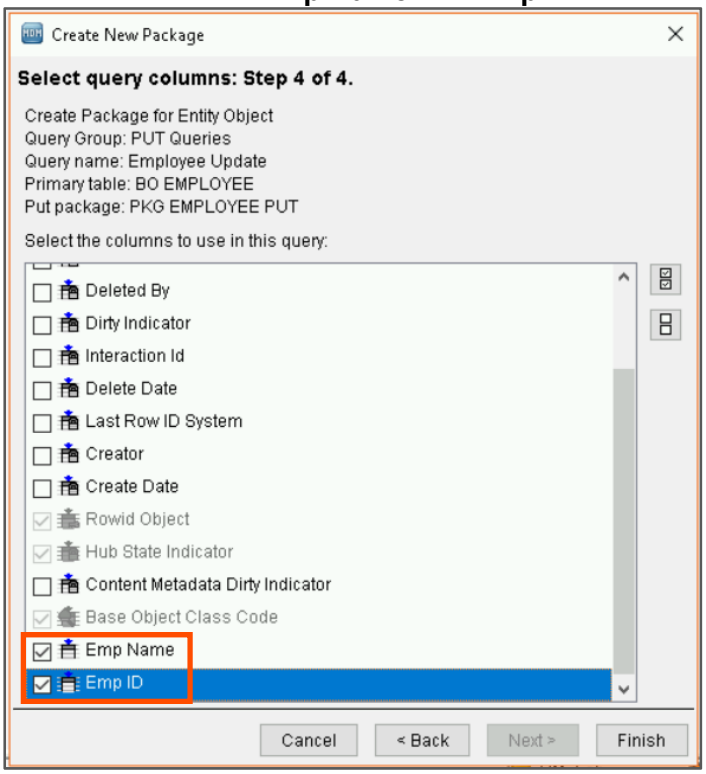


7. Enter the Display Name as **PKG EMPLOYEE PUT**, and select the **Enable put** checkbox.

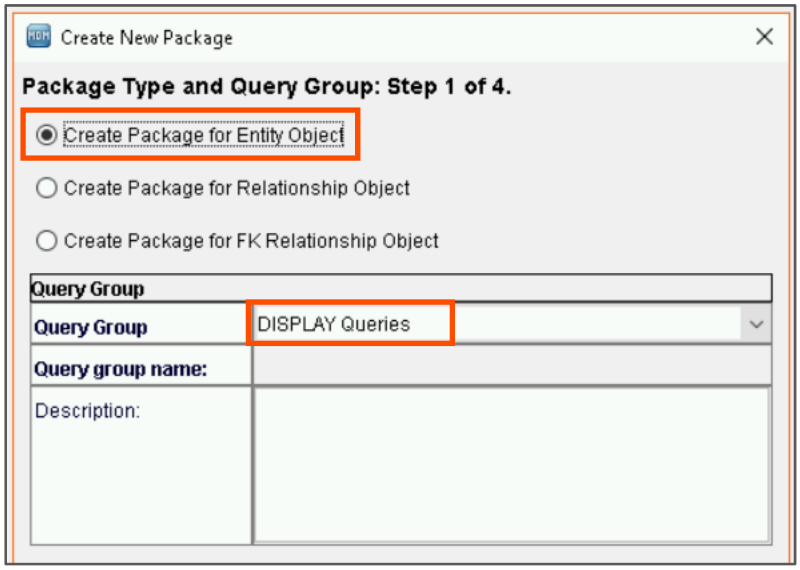


8. Click **Next**.
9. Clear all the columns, retaining the selected default **Rowid Object**, **Hub State Indicator**, and **Bo Class Code** columns.

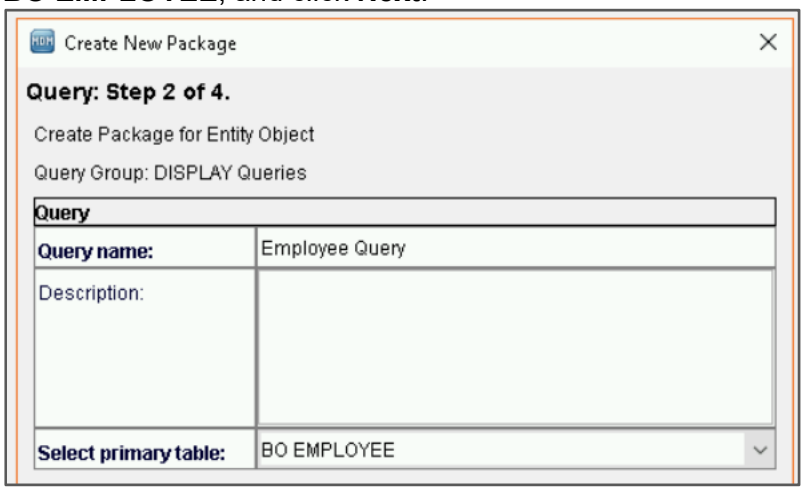
10. Select the columns **Emp Name** and **Emp ID** and click **Finish**.



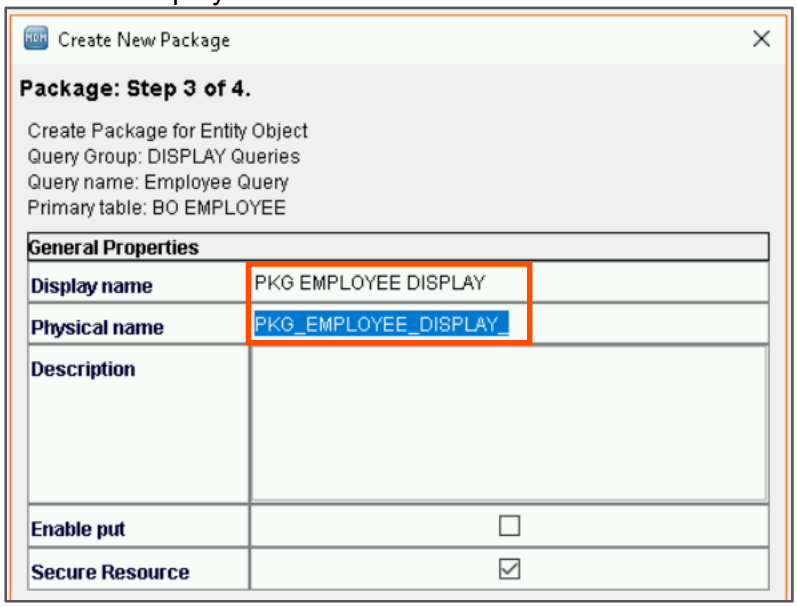
11. In the **Hierarchies** pane, right-click anywhere on the pane and select **Create New Package…**.
12. Select **Create Package for Entity Object** and the **DISPLAY Queries** Query Group and click **Next**.



13. Enter the Query name as **Employee Query**, select the primary table **BO EMPLOYEE**, and click **Next**.

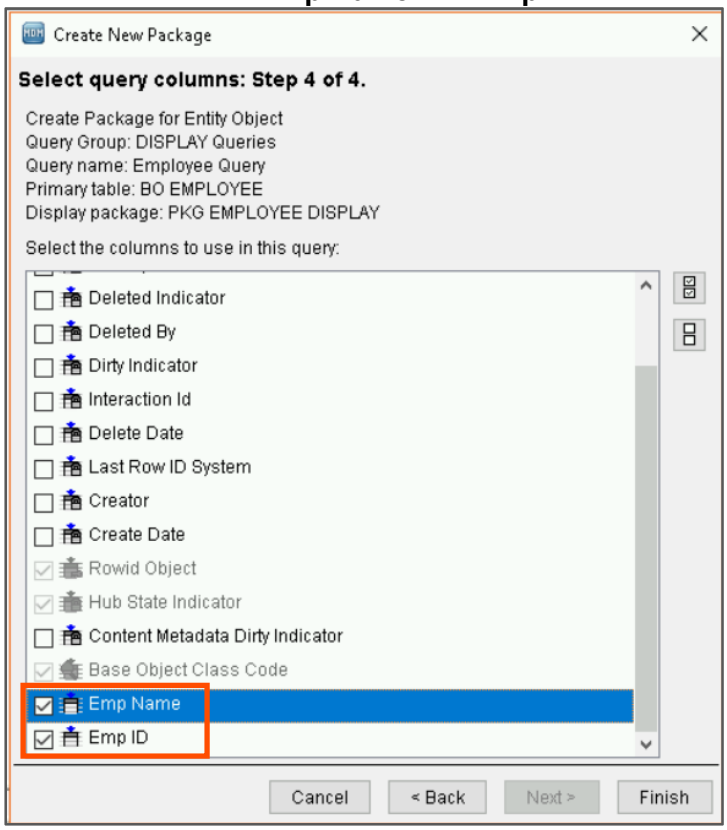


14. Enter the Display Name as **PKG EMPLOYEE DISPLAY** and click **Next**.



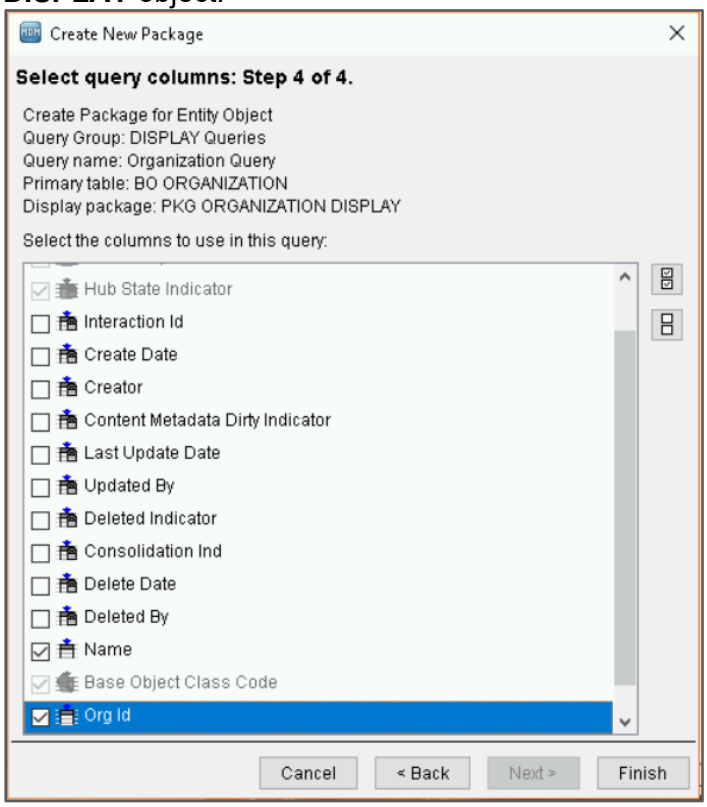
15. Clear all the columns, retaining the default **Rowid Object**, **Hub State Indicator**, and **Base Object Class Code** columns.

16. Select the columns **Emp Name** and **Emp ID** and click **Finish**.

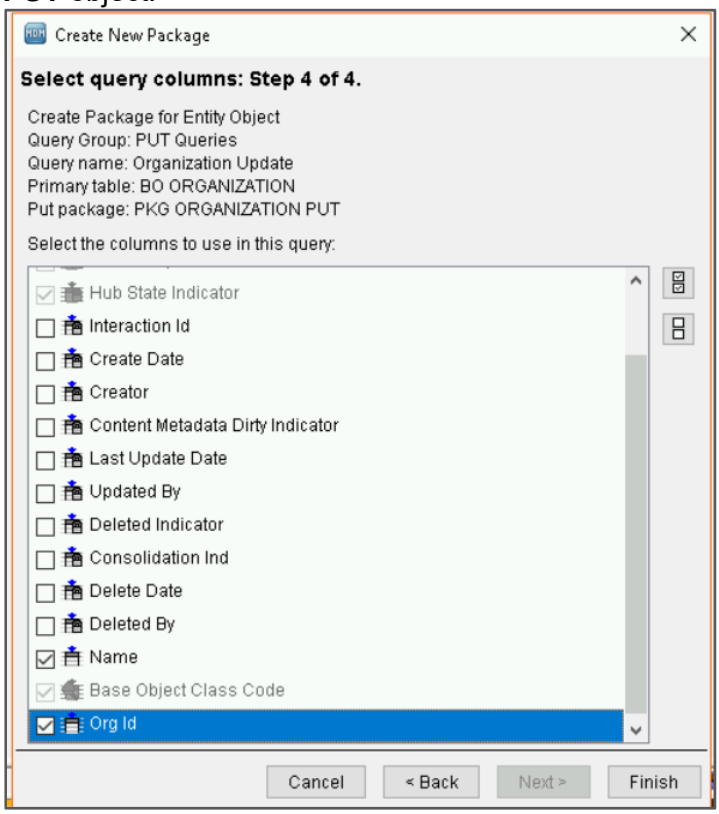


17. Repeat the steps above to create the DISPLAY and PUT packages for the **BO ORGANIZATION** entity object.
18. Ensure that the query name, package name, and selected columns are as shown in the screenshots below:

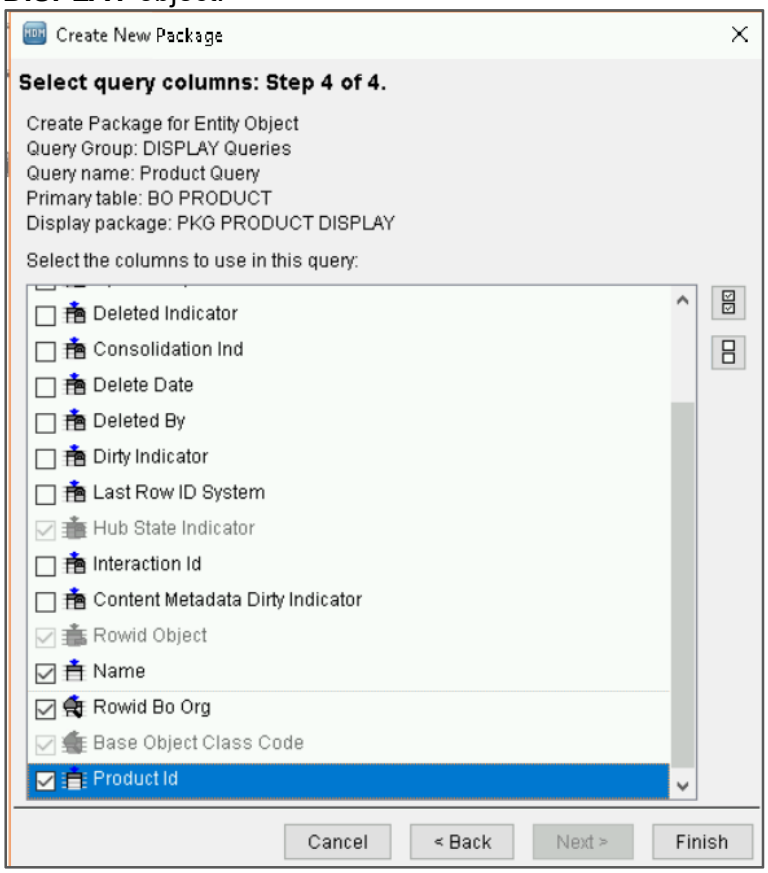
DISPLAY object:



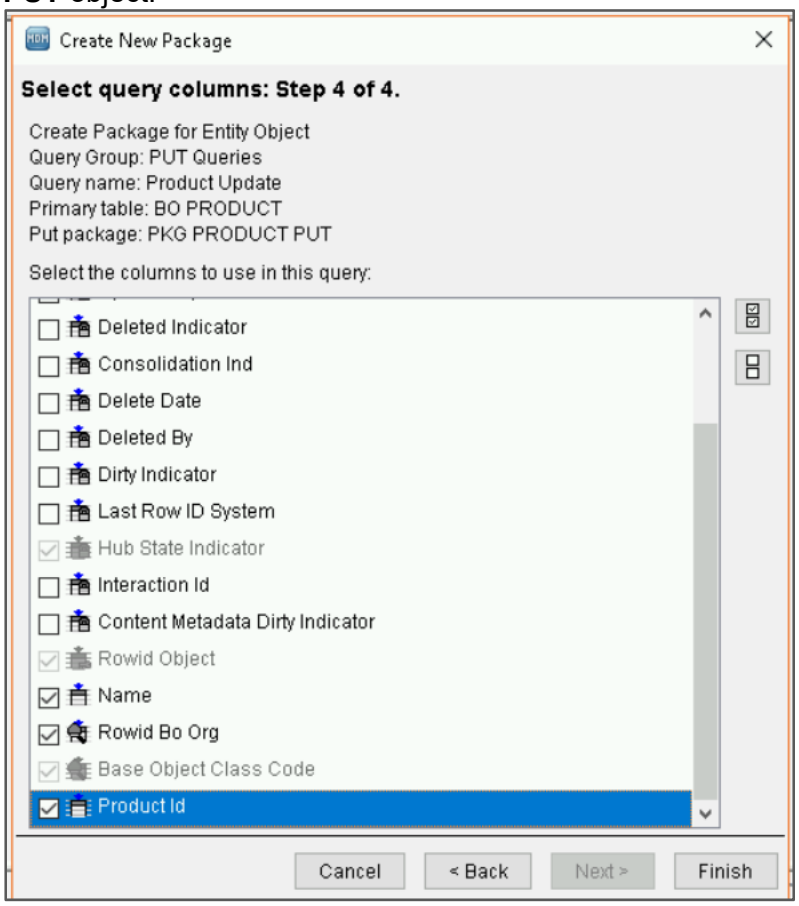
PUT object:



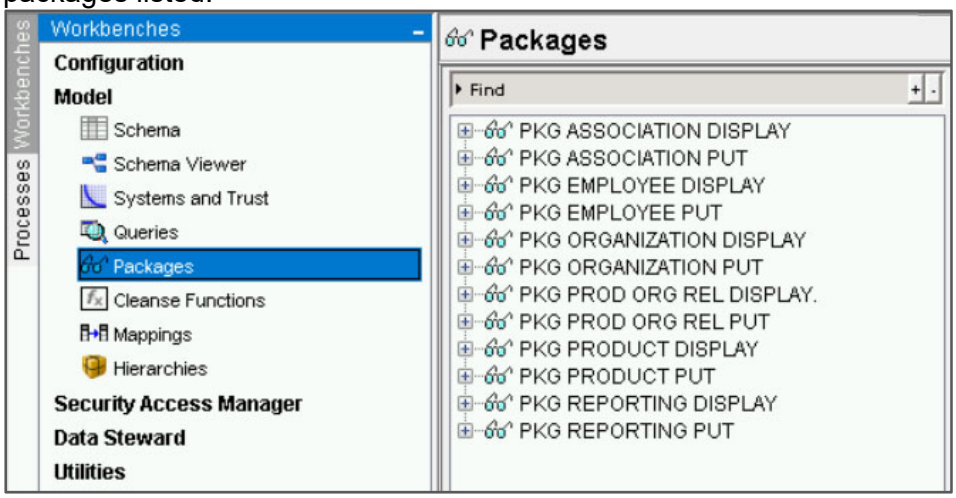
19. Repeat the steps above to create the DISPLAY and PUT packages for the **BO PRODUCT** entity object.
    Ensure that the query name, package name, and selected columns are as shown in the figures below.
    DISPLAY object:



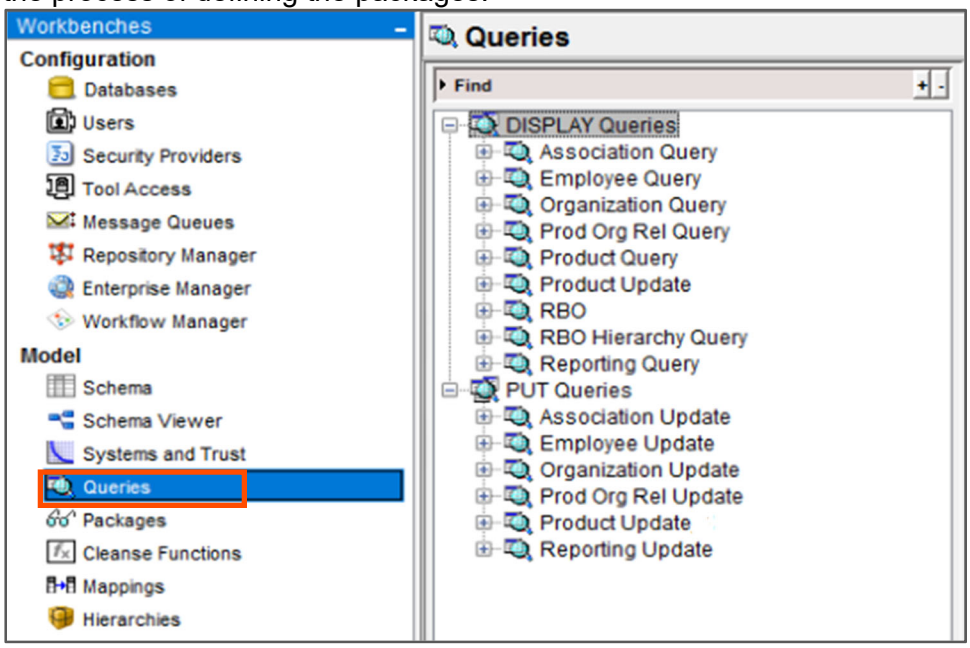
PUT object:



20. From the Model workbench, select the **Packages** tool. You should now see a total of 12 packages listed.



21. Select the **Queries** tool, and you will see the corresponding queries that were created in the process of defining the packages.



This concludes the lab.

Module 19: Profiles

Lab 19-1: Creating New Profiles and Assigning Packages to Profile Objects

Overview:
In this lab, you will create a new profile called **Admin** Profile and enable it for all the configured Entity Types and Relationship Types. You will then assign PUT and DISPLAY packages for all the Entity and Relationship Types and configure the properties that drive the look and feel of the HM user interface. You will then assign packages to the profile objects created. Finally, you will validate the profiles created.

Objective:

- Create a new HM Profile
- Assign Packages to Profile Objects
- Validate Profiles

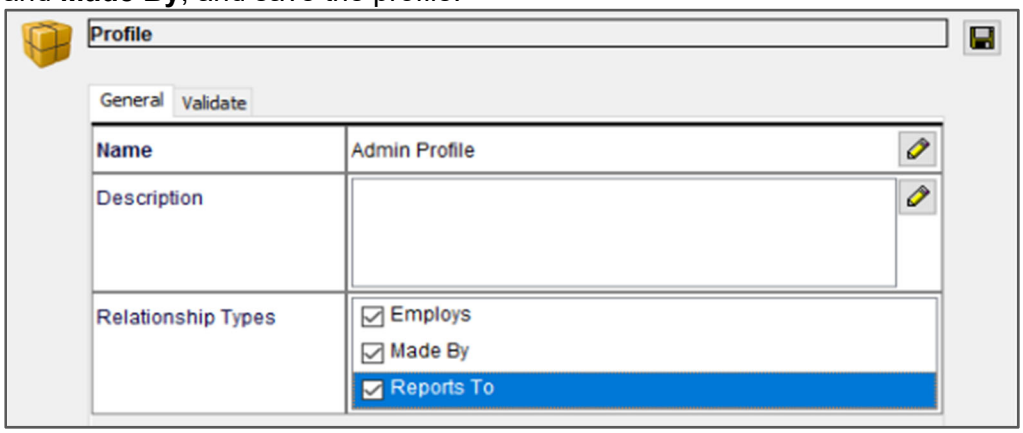
Duration:
25 minutes

Tasks

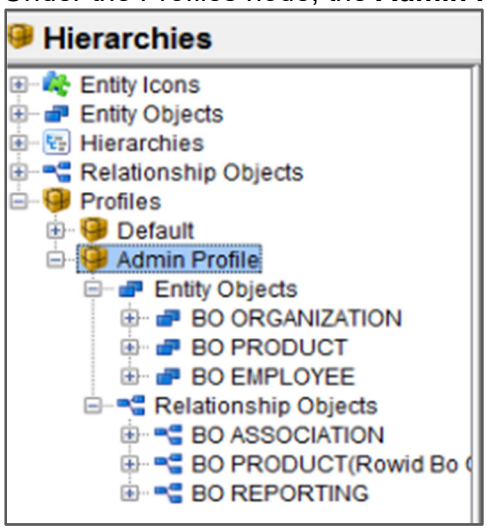
Creating a new HM Profile

1. From the **Model** workbench, select the **Hierarchies** tool.
2. Acquire a **Write Lock**, if necessary.
3. In the **Hierarchies** pane, right-click anywhere on the pane and select **Add Profile…**.

4. Enter the Name as **Admin Profile**, select Relationship Types **Employs**, **Reports To**, and **Made By**, and save the profile.



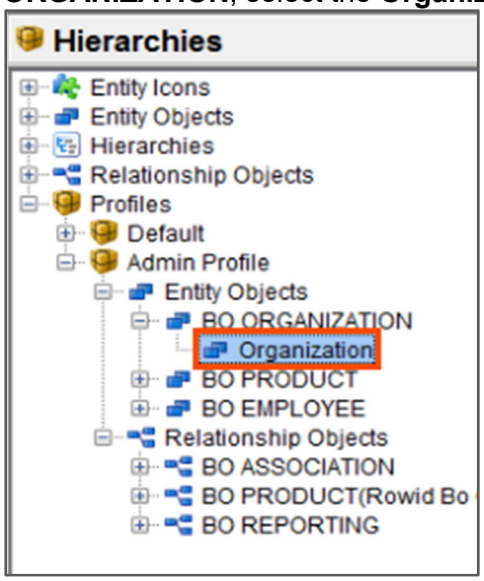
Under the Profiles node, the **Admin Profile** is visible.



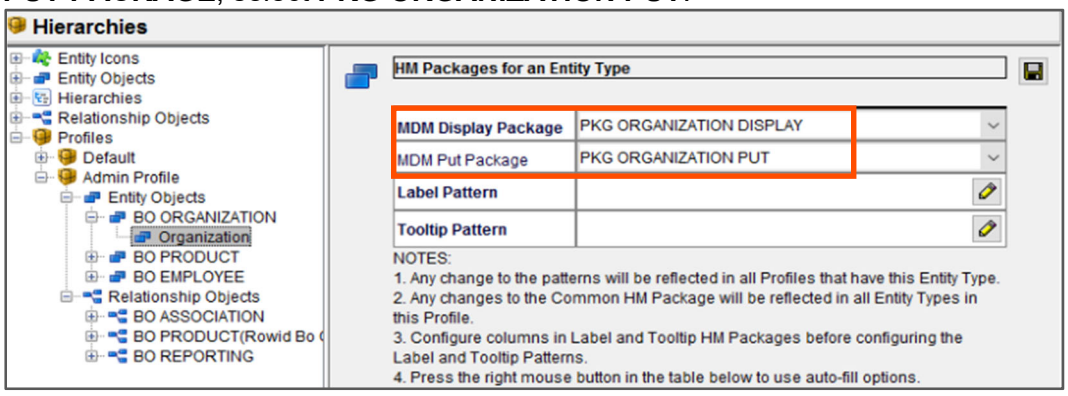
Note: The Default profile is available for your use. However, you cannot rename it. If you do not want the default profile to appear, you can delete it after you create the desired profile.

Assign Packages to Profile Objects

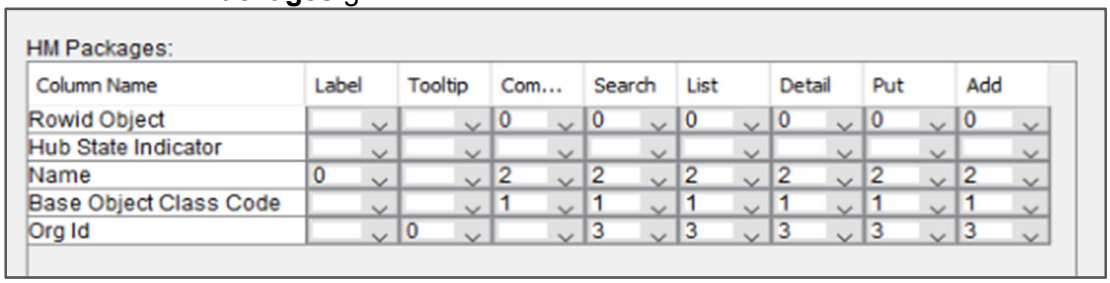
5. In the **Profiles** group, under the **Admin Profile**, under the entity object **BO ORGANIZATION**, select the **Organization** entity type.



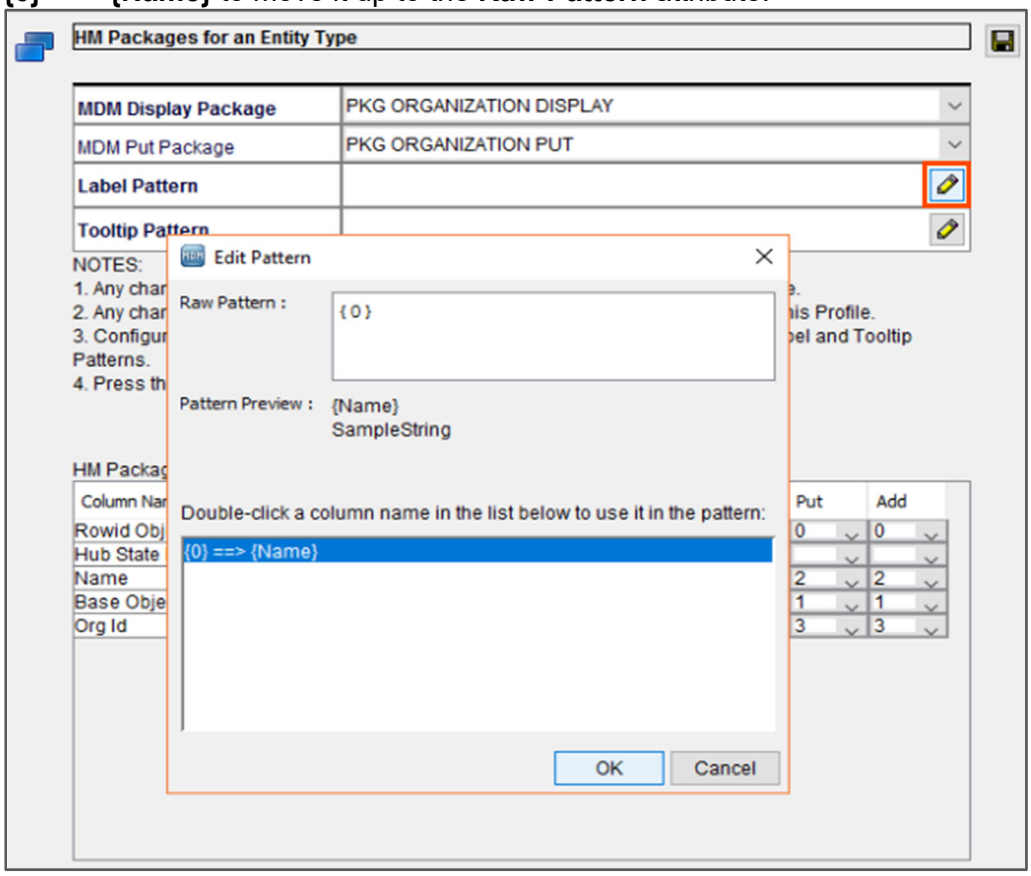
6. For **MDM DISPLAY PACKAGE**, select **PKG ORGANIZATION DISPLAY** and for **MDM PUT PACKAGE**, select **PKG ORGANIZATION PUT**.



7. Fill out the **HM Packages** grid as shown in the screenshot.

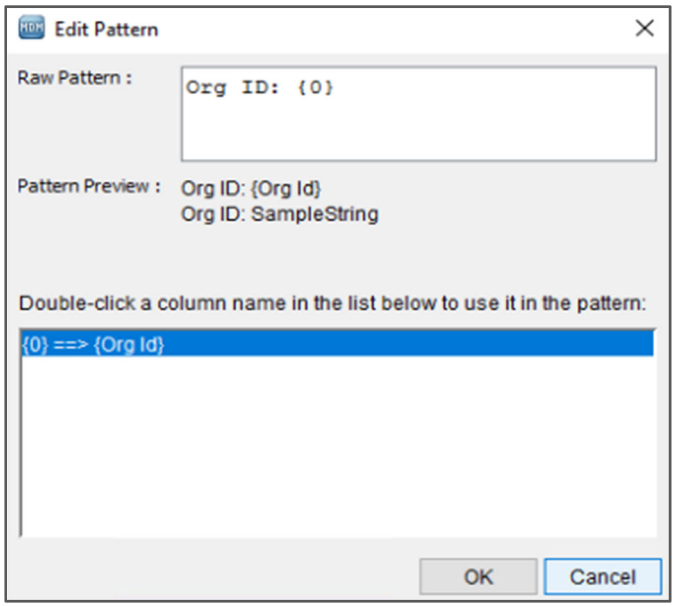


8. After the grid is filled out, for **Label Pattern**, click on the **Edit** icon and double-click on **{0} ==> {Name}** to move it up to the **Raw Pattern** attribute.

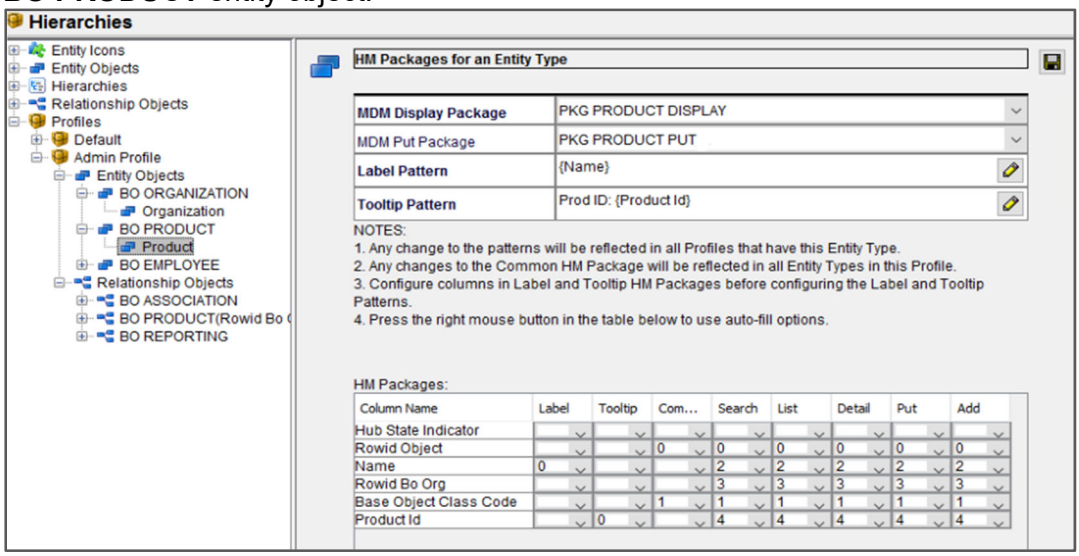


9. Click **OK**.
10. Similarly, for **Tooltip Pattern**, click on the **Edit** icon.

11. Add the text "**Org ID:** " to the **Raw Pattern** window (note the space after the colon), then double-click on **{0} ==> {Org Id}** to move it up to the **Raw Pattern** attribute.
Note: The Pattern Preview shows you how the test will be displayed in the HM user interface.

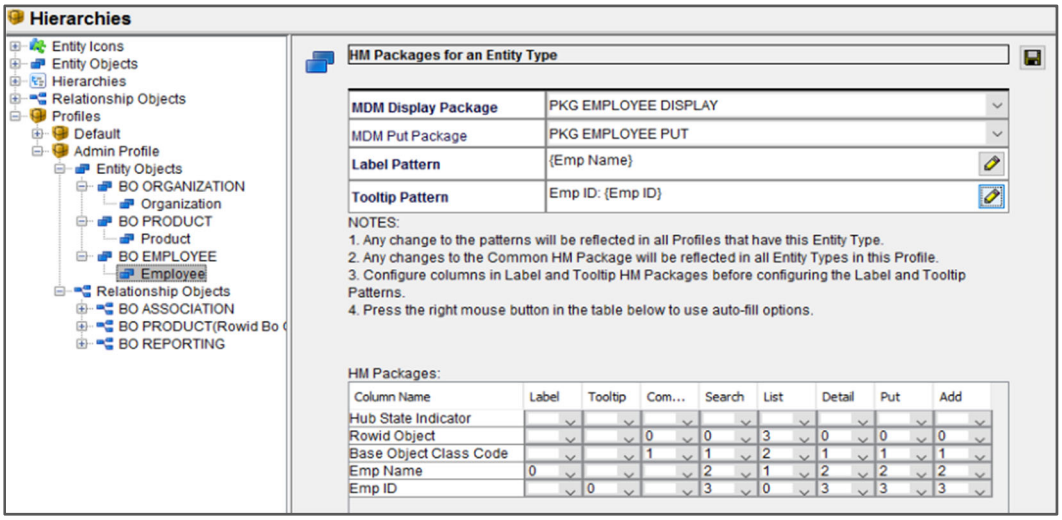


12. Click **OK** and save your work.
13. Again, repeat the steps above for the **PRODUCT** entity type in the **BO PRODUCT** entity object.



14. **Save** your work.

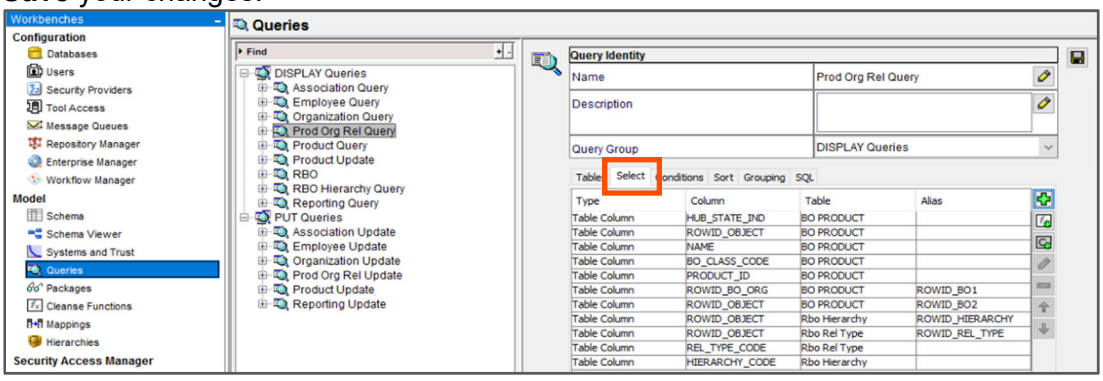
15. Repeat steps above for the **EMPLOYEE** entity type under the **BO EMPLOYEE** entity object. Note the reverse order for the HM List package. You will see the effect of this difference later.



16. **Save** your work.

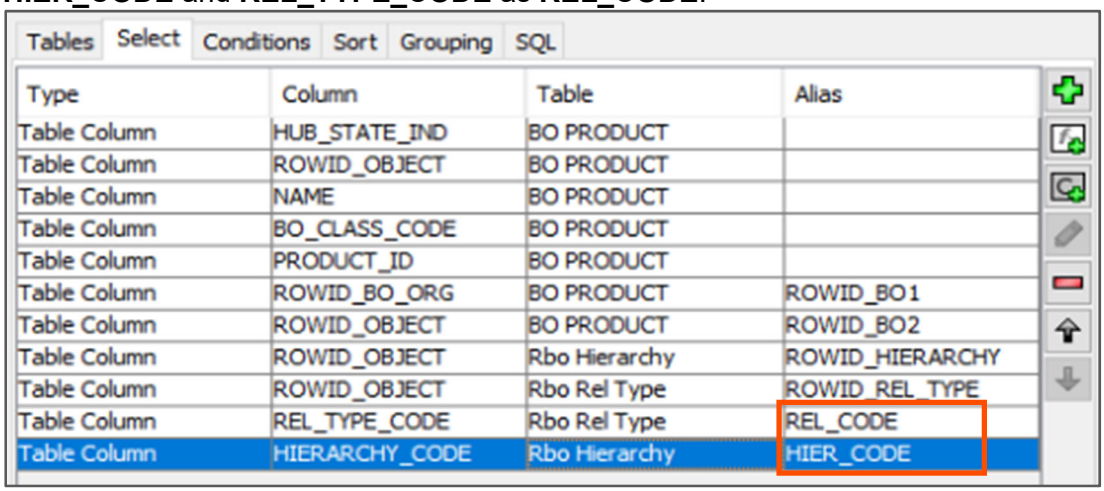
Now the HM packages for the entity types are created. Next, you can define the packages for the relationship types. But before that, you must make an adjustment to the product-organization relationship display query so that the Relationship Type and Hierarchy codes are included in the query and it is available to the package for display.

17. From the **Queries** tool, select **Prod Org Rel Query**.
18. Click the **Select** tab.
19. From **Rbo Rel Type**, add the column **Relationship Type Code** and from **Rbo Hierarchy**, add the column **Hierarchy Code**.
    Note: Use the **+** sign to add the columns.
20. **Save** your changes.



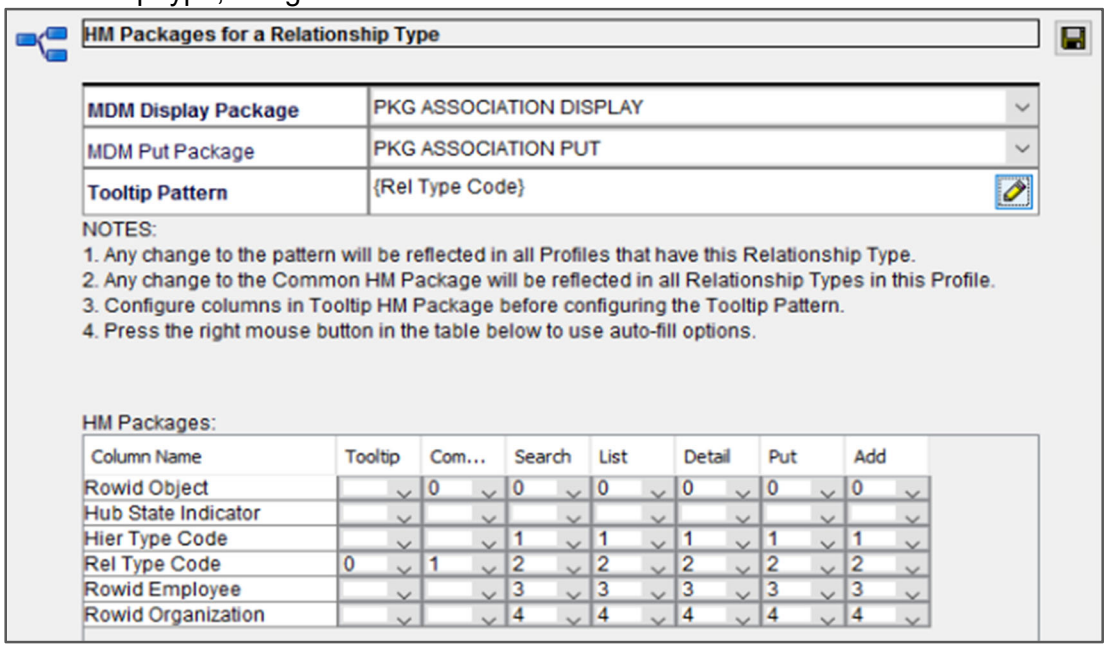
Note: The hub inserts the columns, and for each column, a generic alias is created (C1 and C2).

21. Double-click on the alias column to add the alias for the **Hierarchy Code** column as **HIER_CODE** and **REL_TYPE_CODE** as **REL_CODE**.



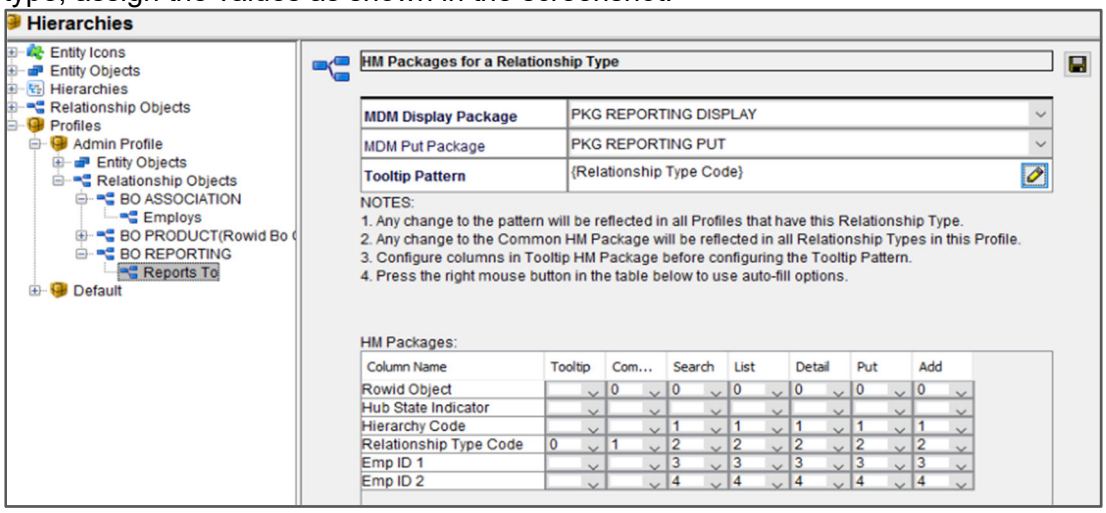
Type	Column	Table	Alias
Table Column	HUB_STATE_IND	BO PRODUCT	
Table Column	ROWID_OBJECT	BO PRODUCT	
Table Column	NAME	BO PRODUCT	
Table Column	BO_CLASS_CODE	BO PRODUCT	
Table Column	PRODUCT_ID	BO PRODUCT	
Table Column	ROWID_BO_ORG	BO PRODUCT	ROWID_BO1
Table Column	ROWID_OBJECT	BO PRODUCT	ROWID_BO2
Table Column	ROWID_OBJECT	Rbo Hierarchy	ROWID_HIERARCHY
Table Column	ROWID_OBJECT	Rbo Rel Type	ROWID_REL_TYPE
Table Column	REL_TYPE_CODE	Rbo Rel Type	REL_CODE
Table Column	HIERARCHY_CODE	Rbo Hierarchy	HIER_CODE

22. **Save** your work.
23. Navigate to the **Hierarchies** tool.
24. Under the **BO ASSOCIATION** Relationship Objects section, for the **Employs** relationship type, assign the values as shown in the screenshot.

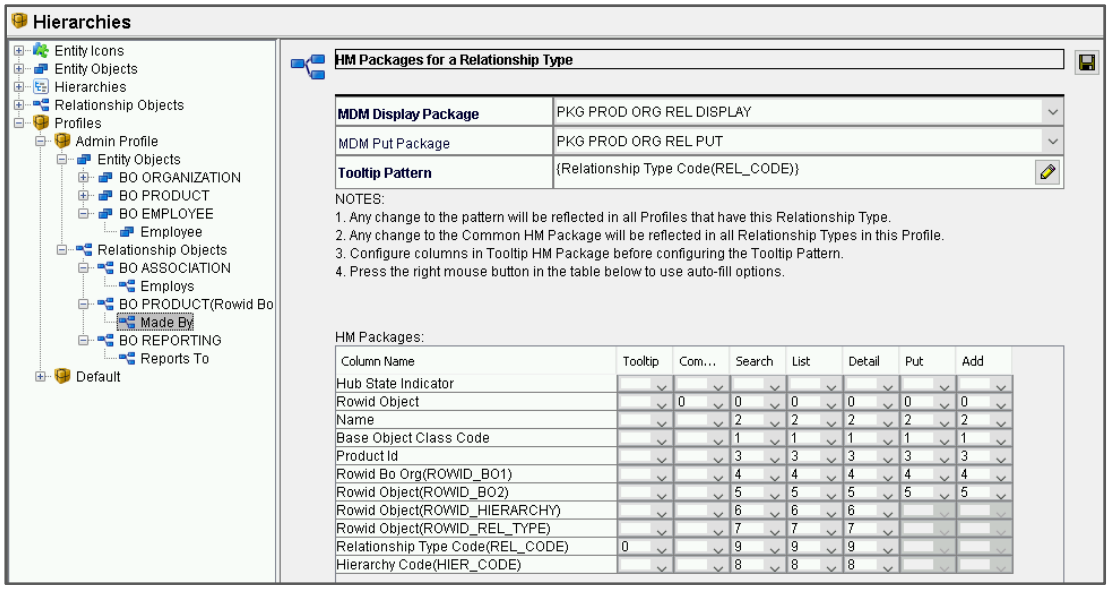


Column Name	Tooltip	Com...	Search	List	Detail	Put	Add
Rowid Object		0	0	0	0	0	0
Hub State Indicator							
Hier Type Code			1	1	1	1	1
Rel Type Code	0	1	2	2	2	2	2
Rowid Employee			3	3	3	3	3
Rowid Organization			4	4	4	4	4

25. Under the **BO REPORTING** Relationship object section, for the **Reports To** relationship type, assign the values as shown in the screenshot.



26. Under the **BO PRODUCT** Relationship object section, for the **Made By** relationship type, assign values as shown in the screenshot.

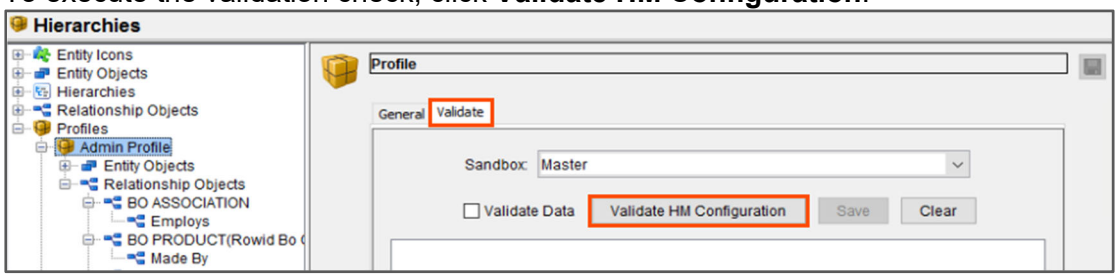


Note: Some of the columns for the Put and Add HM Packages are greyed out. See if you can figure out why. The key lies in the Display and Put packages and how they are defined.

Validate Profiles

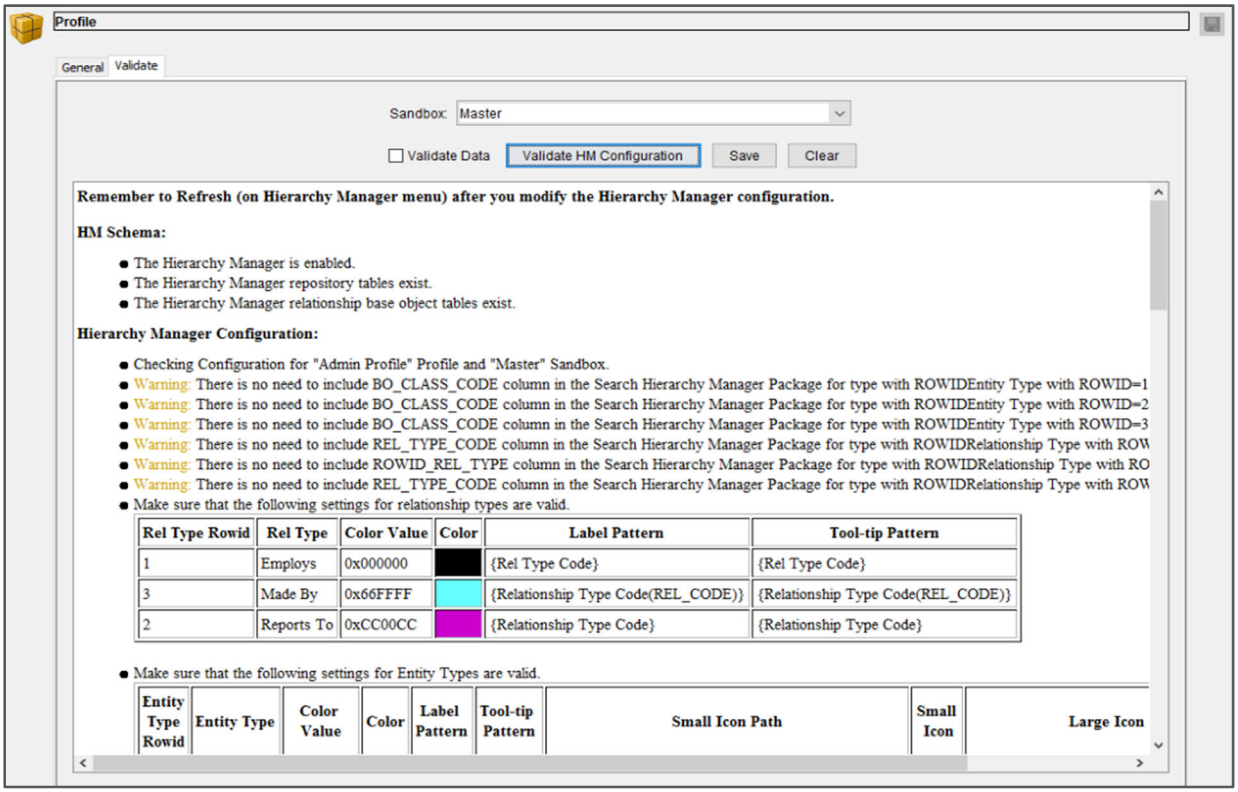
Now that the entity objects, relationship objects, hierarchies, and associated profiles are defined in the entire configuration (as they relate to the profile), they can be validated.

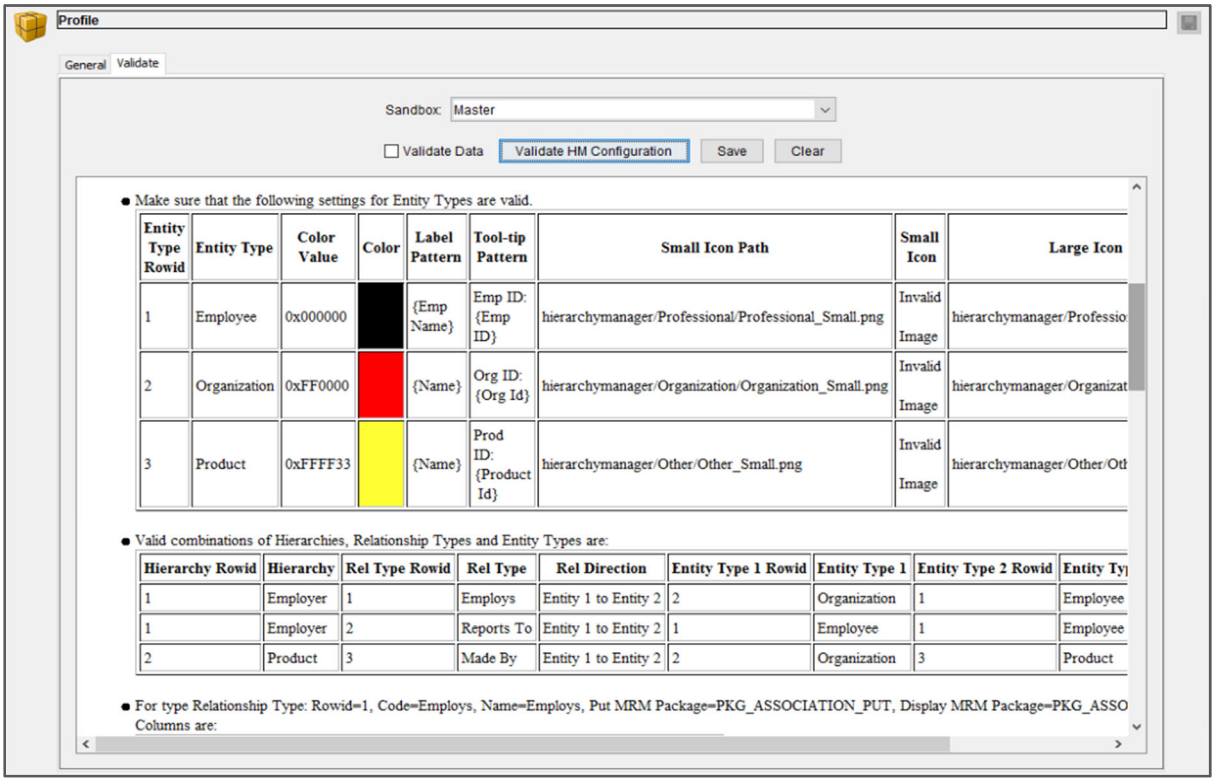
27. Select the **Admin Profile** and then select the **Validate** tab in the Profile pane.
28. To execute the validation check, click **Validate HM Configuration**.



Note: This is similar in concept to the validation function in the Metadata Manager, but it only checks those aspects of the ORS that are relevant to hierarchies.

Your results may differ in minor ways (such as color and icon choices). Go through your report and look at the types of data that the validation check produces. If there is a serious error, a warning message pop-up window appears that describes the problem. Note the warnings for the HM Search packages – these are not the errors and correcting them is optional.





This concludes the lab.

ADDITIONAL INFORMATION

Copy, Edit, and Delete Profiles:

Editing a profile involves adding and/or removing the entity/relationship objects in the profile or modifying the packages and package display attributes assigned to the objects in the profile.

1. To Copy a profile, select the profile name and right-click to select **Copy Profile**.
2. To **Add** or **Remove** entity/relationship types to or from a profile, select the profile name under **Profiles** and check or uncheck the appropriate relationship types under the **General** tab. Note that the entity types used in the profile are a function of the relationship types selected.
3. To **Delete** a profile, select the profile name and right-click to select **Delete Profile**.

Module 20: Loading Data and Testing the HM Configuration

Lab 20-1: Loading the HM Data

Overview:
In this lab, you will migrate the HM data from landing tables to the Base Objects. The data for this course is pre-loaded into the landing tables. Note that as stage jobs do not have parent-child dependencies, the order in which the jobs are run is not usually significant. It is possible that depending on how a specific MDM implementation is designed, there may be other factors such as, external processes that would require an order in which the jobs are executed.

Objective:

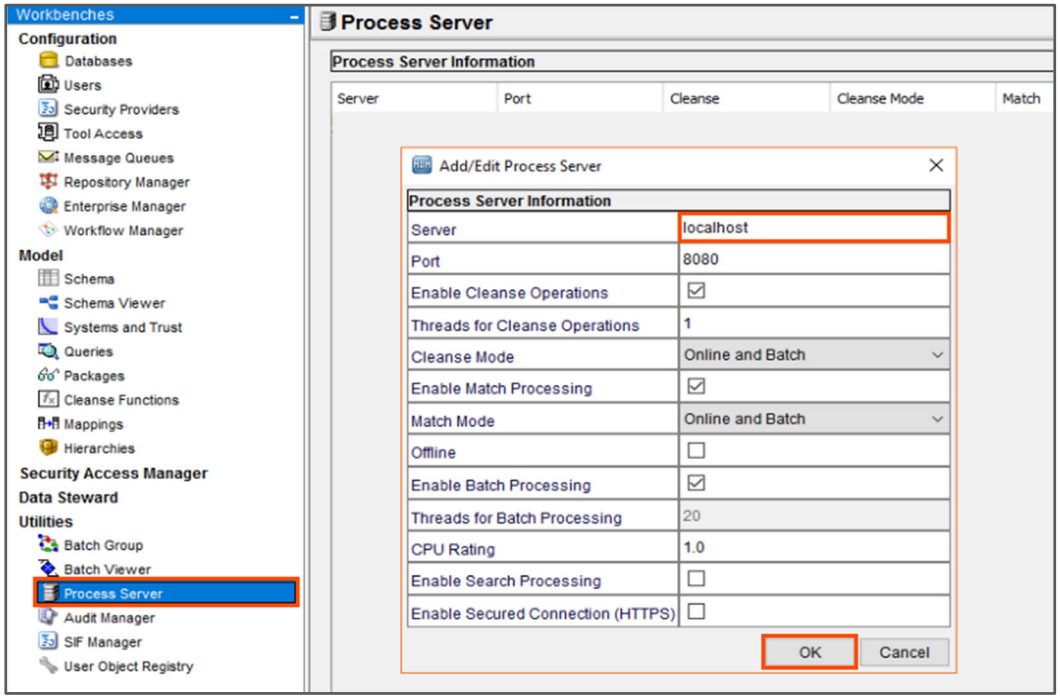
- Load the HM Data from the landing tables to base objects

Duration:
20 minutes [This may vary according to the time required to complete execution of the batch jobs]

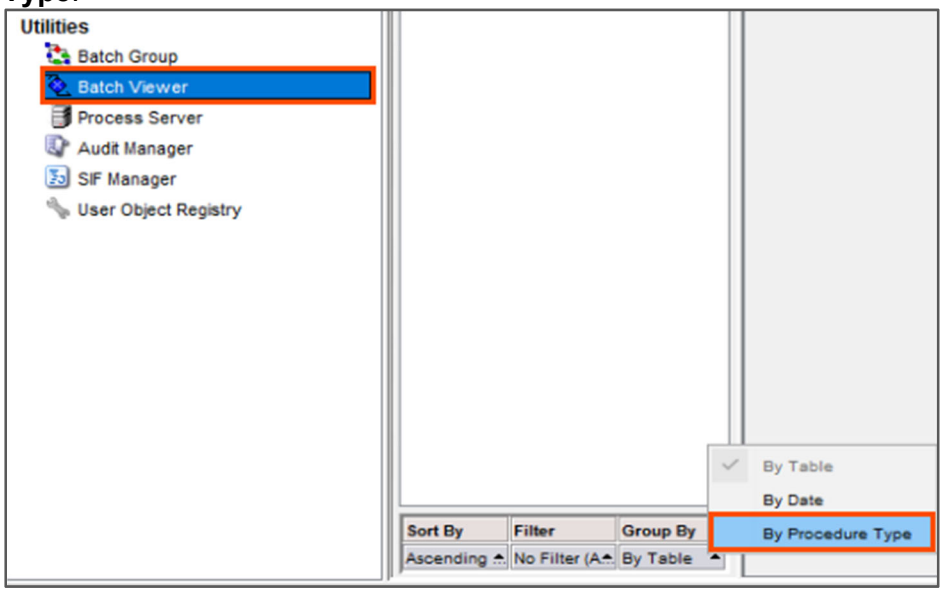
Tasks

Load HM Data from Landing Tables to Base Objects

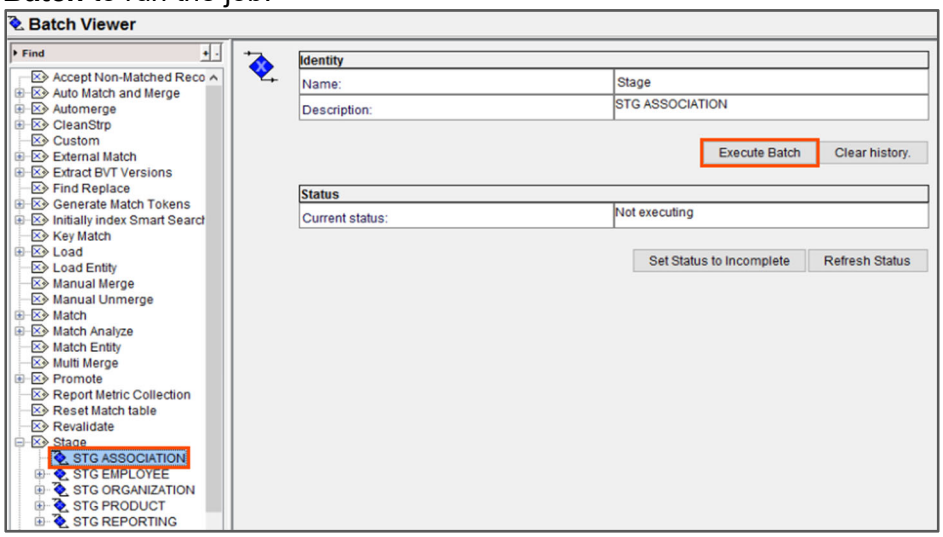
1. To add a Process Server, in the Process Server tool, click the **Add Process Server (+)** icon, and name it as **localhost**.



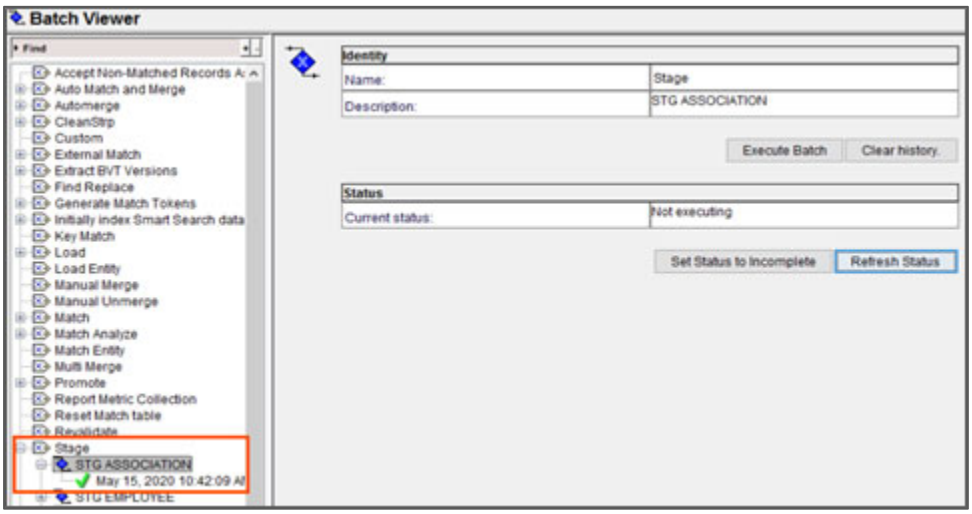
2. Click **OK** and save the changes.
3. From the batch viewer tool, in the Group By pane, change the option to **By Procedure Type**.



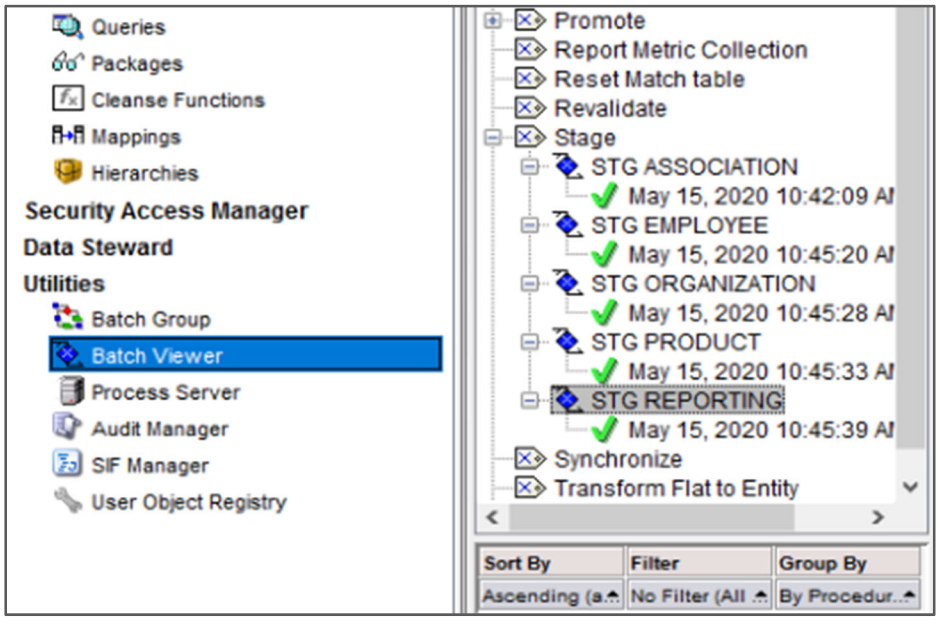
4. Expand the **Stage** option, select the **STG ASSOCIATION** stage job, and click **Execute Batch** to run the job.



5. Observe that the status is reflected after the successful execution.

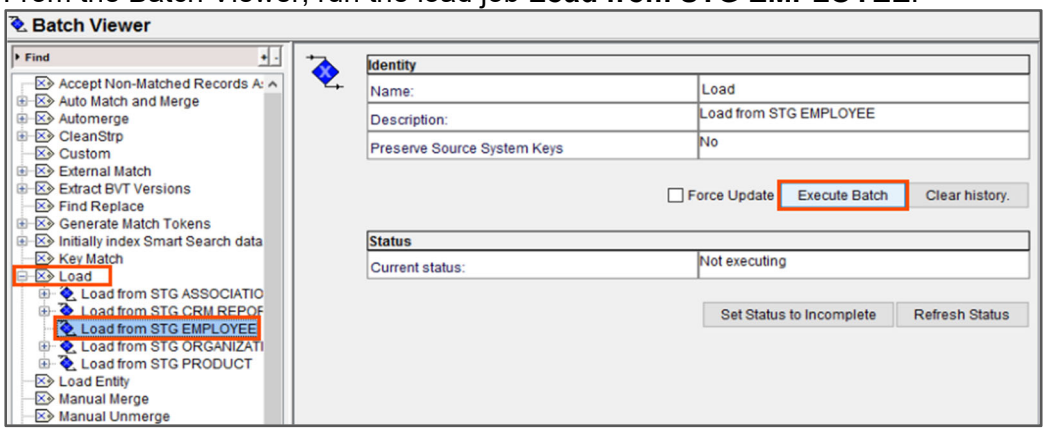


6. Similarly, run the following stage jobs:
   STG EMPLOYEE
   STG ORGANIZATION
   STG PRODUCT
   STG REPORTING



Note: If you encounter warnings in any of the stage jobs, ignore them.

7. From the Batch Viewer, run the load job **Load from STG EMPLOYEE**.



8. Similarly, run the other load jobs in the following order:

   Load from STG ORGANIZATION

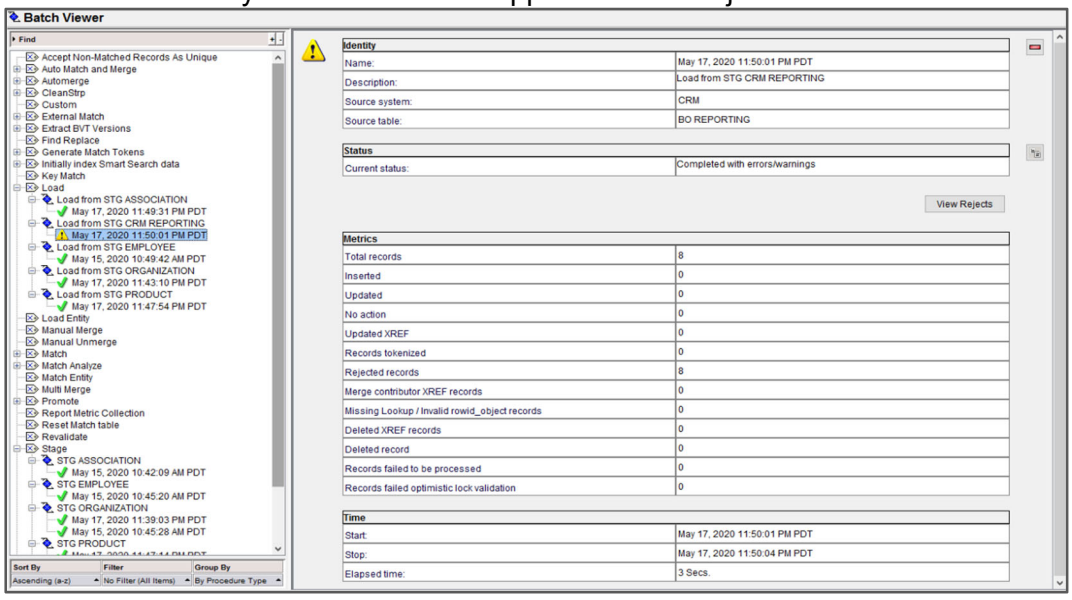
   Load from STG PRODUCT

   Load from STG ASSOCIATION

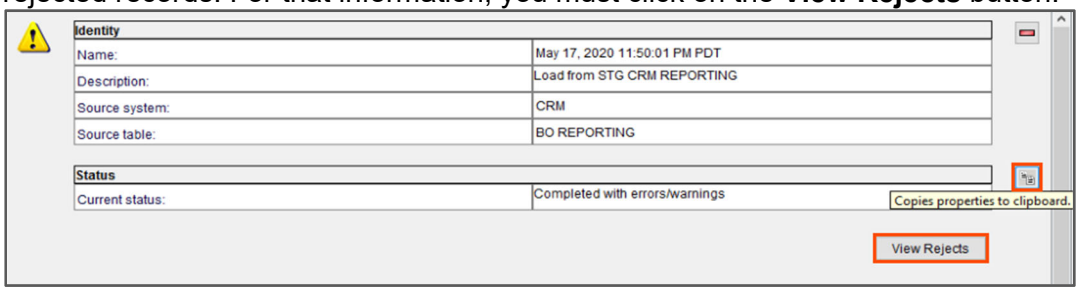
   Load from STG CRM REPORTING

   Note: Can you determine what factors/dependencies prescribe this order? As a clue, there is one FK base object relationship and two HM relationship types involved.

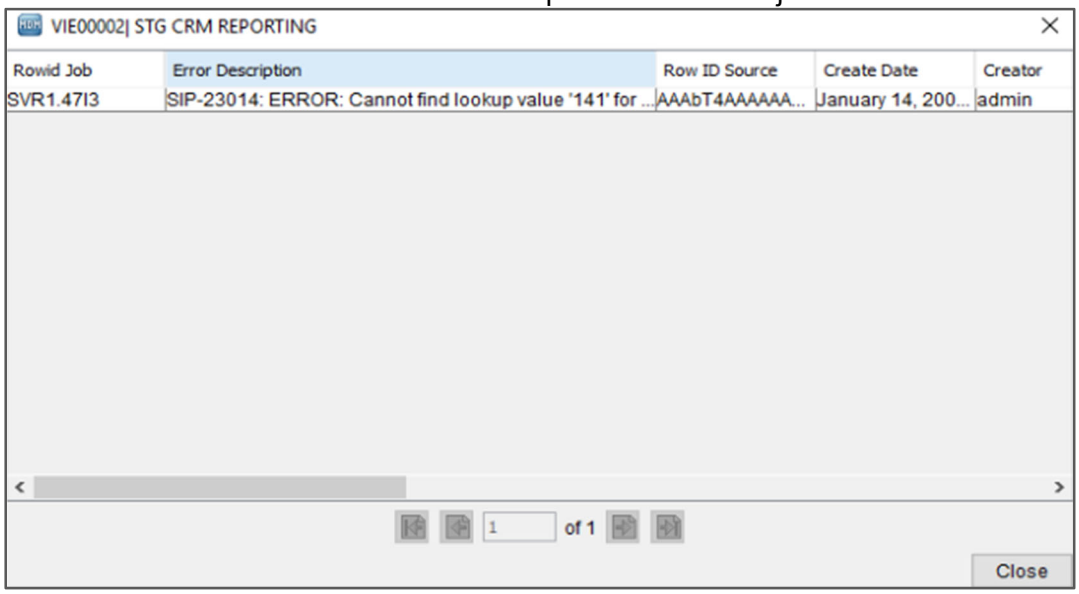
9. All jobs should execute cleanly except for the Load job **STG CRM REPORTING**. This is intentional so that you can see what happens when a reject occurs.



10. If you want to keep a handy record of the results of the execution of a job you can click on the **Copies properties to Clipboard** button and paste the results of this status into the application of your choice. Note that the copy function does not include details of any rejected records. For that information, you must click on the **View Rejects** button.



11. Click on **View Rejects**, expand the window, and adjust the fields so you can clearly read the fields and determine the nature of the problem for the rejected record.



Note: Here we see that there was a problem with the lookup process and Employee ID number 141 could not be found. Examination of the **BO EMPLOYEE** base object records show that, sure enough, there is no record with an ID number of 141.

This concludes the lab.

Module 20: Loading Data and Testing the HM Configuration

Lab 20-2: Test the HM Implementation

Overview:
The scope of this course does not cover the Hub console Hierarchy Manager tool user interface; however, we will show you enough so that you can see the effect of the implementation that you have defined in this course.

Objective:

- Test the HM Implementation

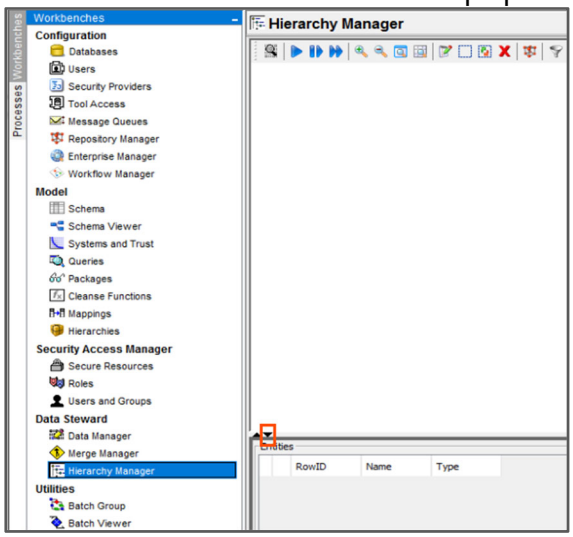
Duration:
20 minutes

Tasks

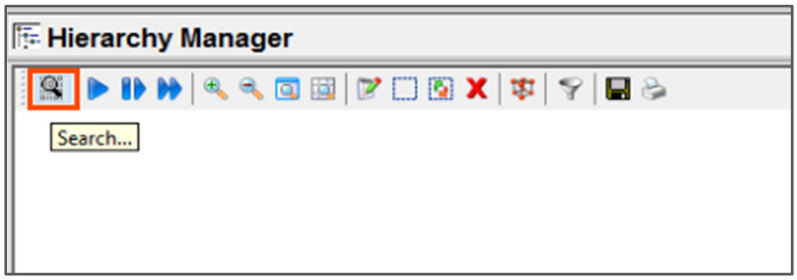
Testing the HM Implementation

1. From the **Data Steward** workbench, select the **Hierarchy Manager** tool.
2. To maximize the **Hub Console** window, click the **right** arrow.
   This will collapse the **Workbenches** pane and free up more room for the hierarchy manager.

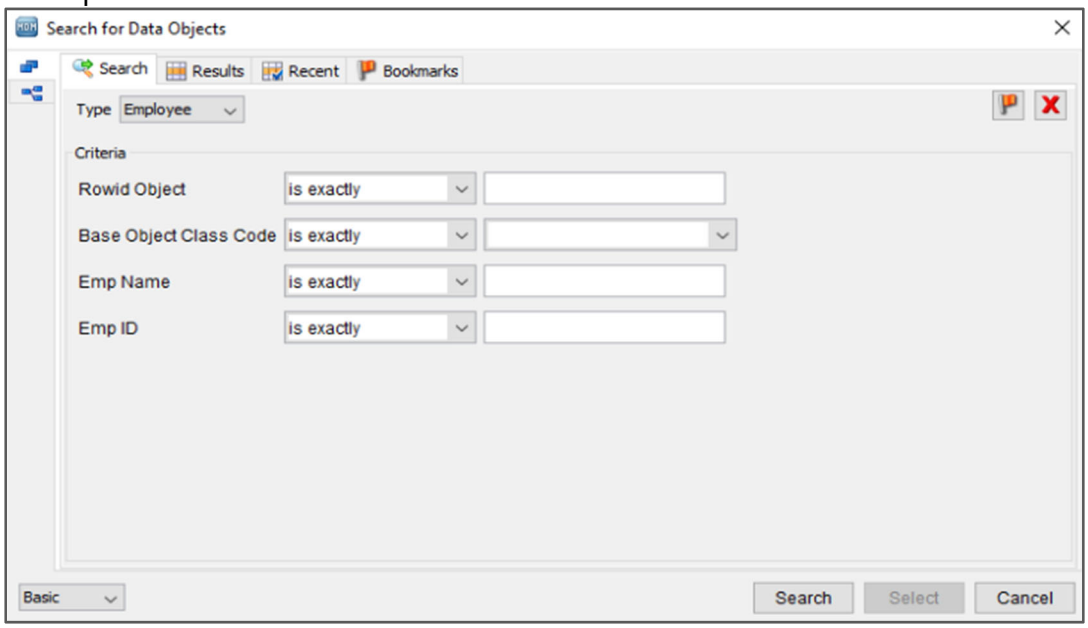
3. To hide the Entities and Relationships panes, click the **downward** arrow.



4. Select the **Search** function.

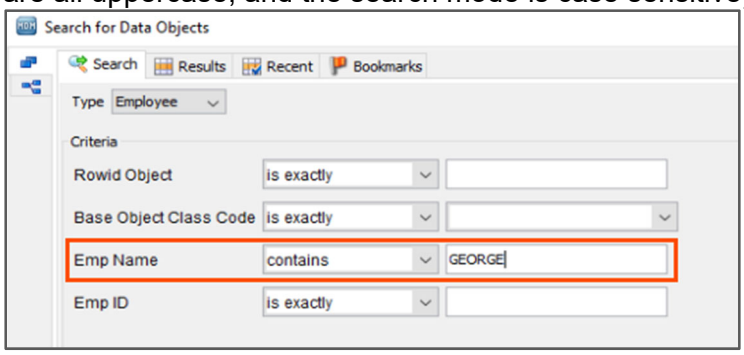


This opens the search window.

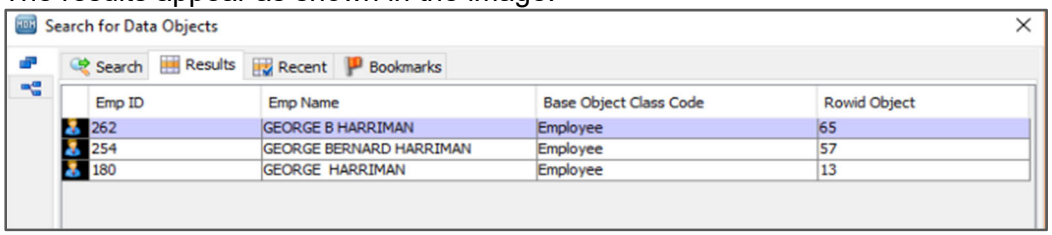


In this section, you will search for employees whose name contains “George”.

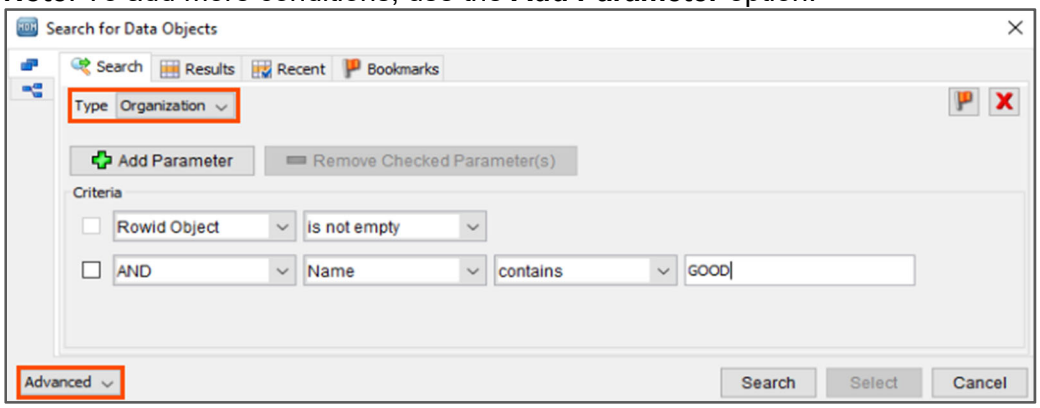
5. Change the **Emp Name** operator to **contains** and enter “GEORGE” (the source records are all uppercase, and the search mode is case sensitive).



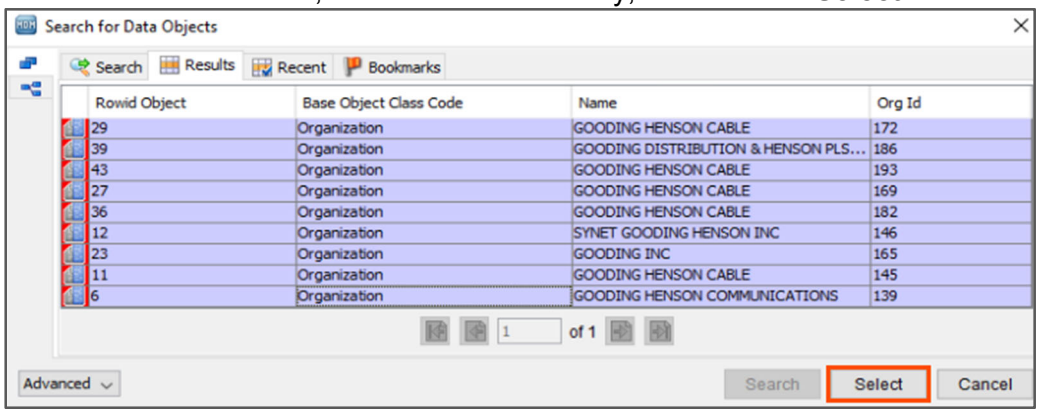
6. Click **Search**.
   The results appear as shown in the image.



7. Use an **Advanced** search to look for organizations that have **GOOD** in their name.
   Note: To add more conditions, use the **Add Parameter** option.

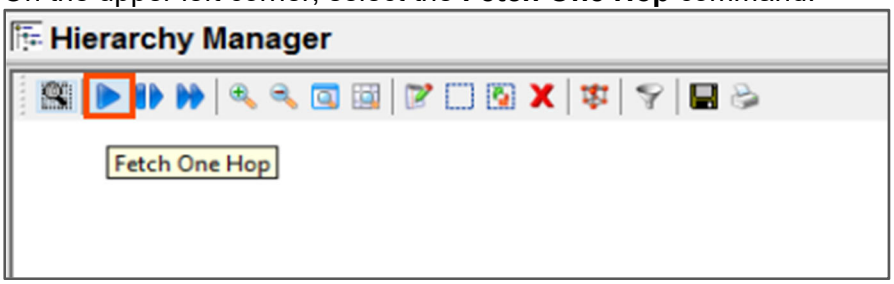


8. To select all the records, shift-click the last entry, and click the **Select** button.

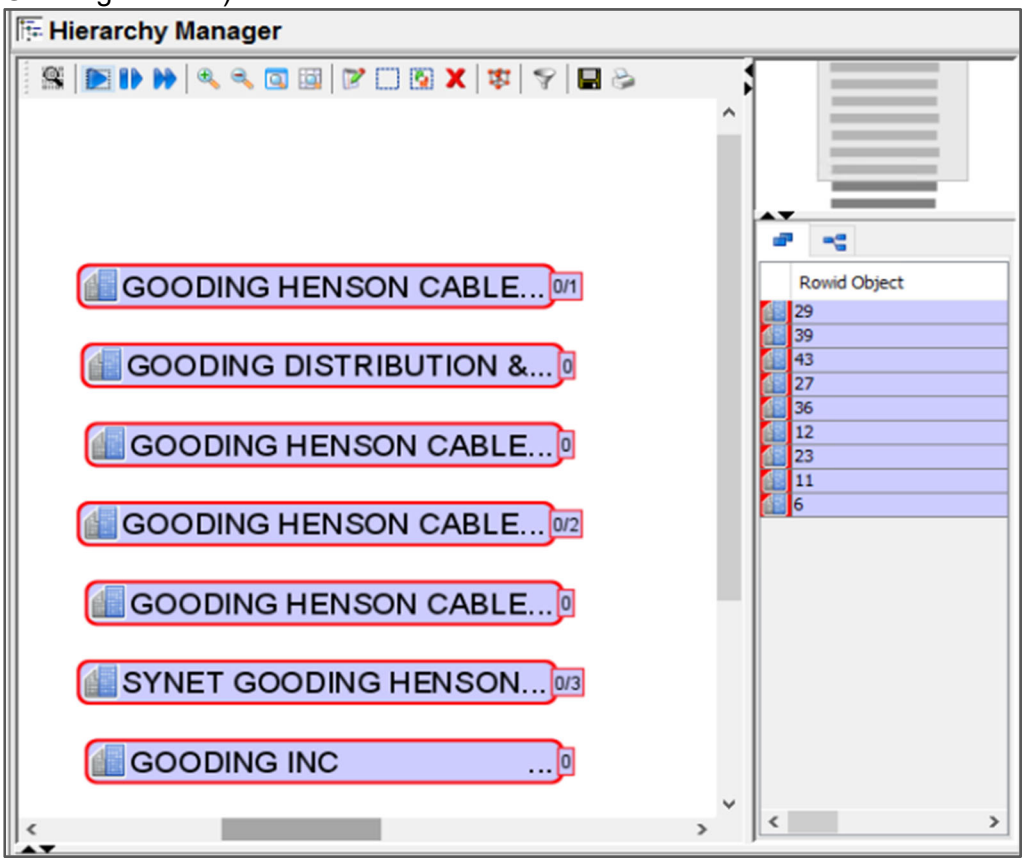


In the right-hand pane and on the canvas, all the selected entities appear.

9. On the upper left corner, select the **Fetch One Hop** command.



Nothing appears to happen other than the number of relationships displayed versus the number of relationships in the database for that record (for example, 0/3 for Synet Gooding Henson).

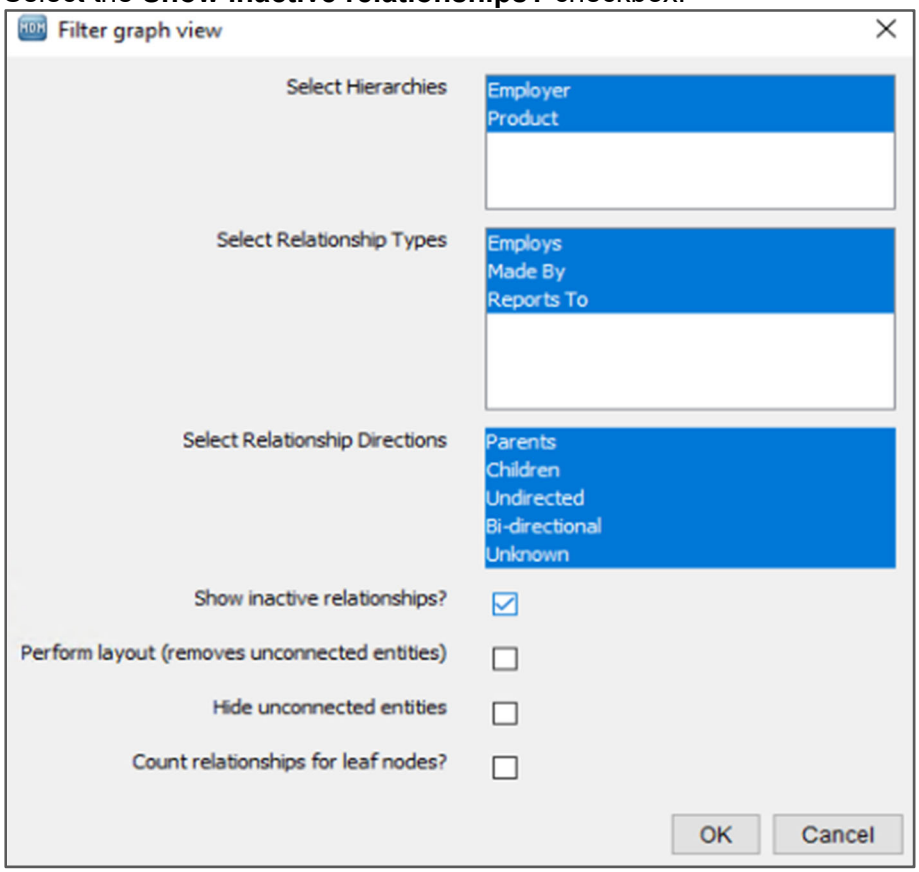


By default, Hierarchy Manager only displays active relationships, and it turns out that all the relationships reflected in the records have an end-date that is older than today's date for this class. However, we can configure HM to also display inactive relationships.

10. Click the **Filter** icon to select the **Filter Graph View** function.

    Here you can select which hierarchies, relationship types, and relationship directions are displayed as well as some other features.

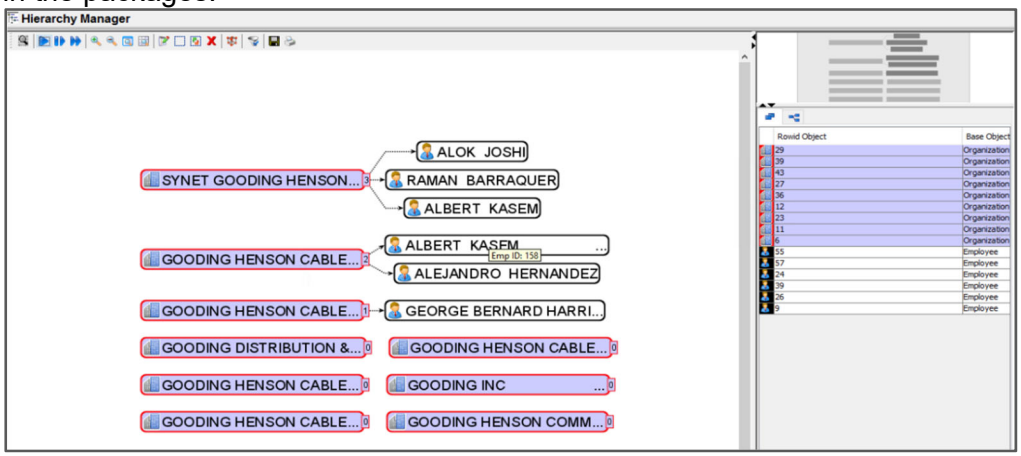
11. Select the **Show inactive relationships?** checkbox.



12. Click **OK**.
13. Select the **Fetch One Hop** command again, and the inactive relationships are now displayed.

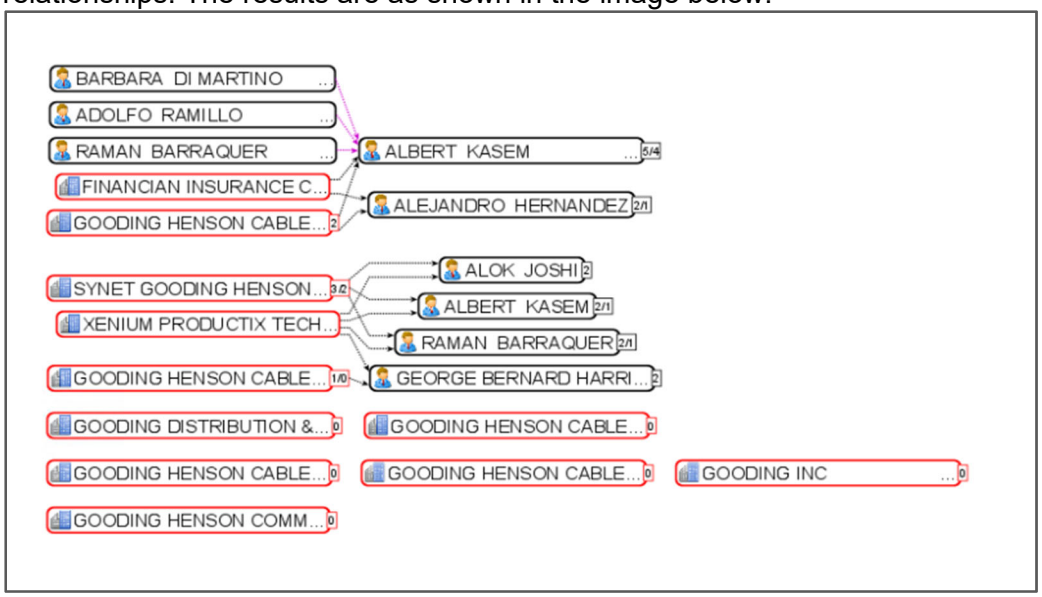
    Notice that the inactive relationships are indicated by dotted lines. Also, the contents of the lower right-hand pane display all the entities and relationships that appear on the canvas and that the columns displayed correspond to the Common HM Package definitions for entities and relationships.

14. In addition, hover over the relationships and entities to see the tooltips that you defined in the packages.



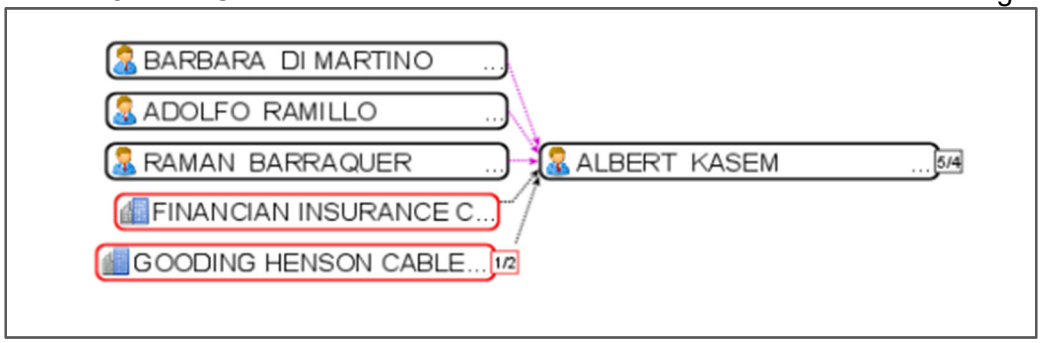
Looking at this graphical view of entities and relationships, consolidating these records, which are probably duplicates, it becomes quite apparent that the data stewards have their work cut out for them.

15. Select all the employees and again use the **Fetch One Hop** command to expand their relationships. The results are as shown in the image below:



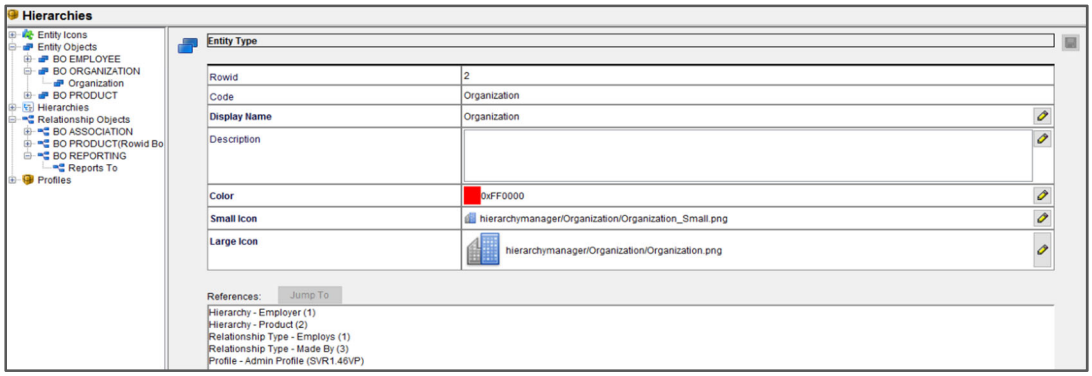
Here again, we can see that we have our work cut out for us. We have multiple records that appear to refer to the same entity; we have employees that are apparently employed by multiple companies; and relationships that are inactive. All of this will have to be examined by data stewards, merged wherever appropriate and relationships updated as necessary.

16. Now, select the **GOODING HENSON CABLE…** employs **GEORGE BERNARD HARRI…** records and all the records below them.
17. Using the **Clear Selected** command (**X**), remove them from the canvas. Also, remove the **ALEJANDRO HERNANDEZ** record. The results are as shown in the image:



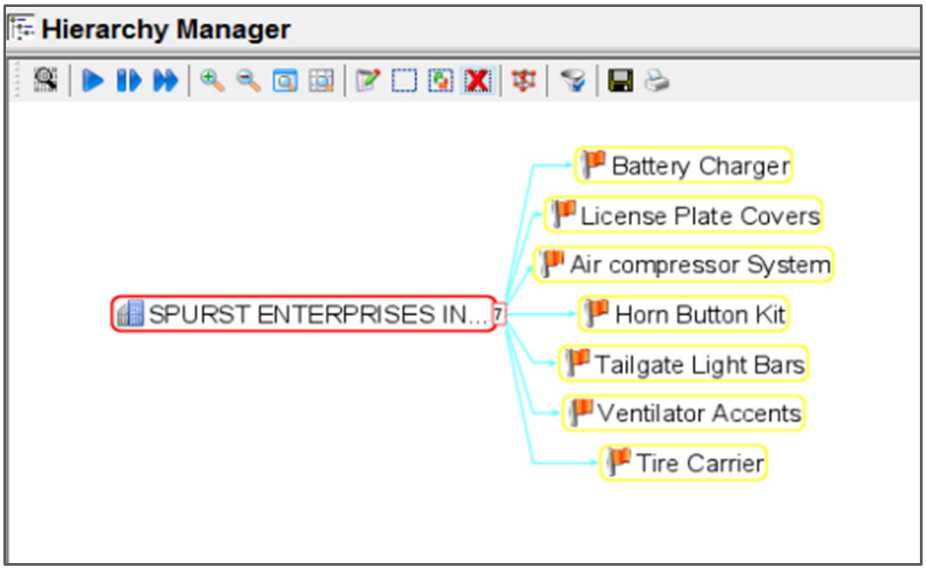
What we have left is an organizational hierarchy (although an ambiguous one), which was the original goal of this course. It is clear that there are three employees who report to Albert Kasem, but Albert appears to be employed by two organizations. It is unlikely that all four employees work for two different companies, but that will be up to the data steward to investigate and figure out what the reality is and then correct the records if necessary.

Notice that the relationships are correct – the employees report to Albert and the organizations employ Albert, but it doesn't look like a standard hierarchal organization chart. That is because of the way we defined the relationship base object **Reports To**. The **Employee** (entity 1) **reports to** (direction 1 to 2) another **Employee** (entity 2), so the employees are drawn to the left of entity 2. If the direction of the relationship had been reversed and we constructed the relationship as **Employee** (entity 2) supervises (direction 2 to 1) **Employee** (entity 1) the employees would have been drawn to the right of the supervisor, and the diagram would then resemble a traditional organization chart.



18. Do a basic search for organizations containing **SPURST ENTERPRISES** in their name.
19. Select the **organizations** and expand them by one hop.

Here is an example of the second hierarchy we set out to construct, the product manufacturer hierarchy along with potential duplicate records for the Spurst Enterprises organization.



This is the end of the formal lab exercises.

For more information on the functionality of the Hub HM console, please see the **MDM Data Steward Guide** for a detailed description of the user interface. In addition, the **Using IDD** and **IDD Configuration** courses cover the hierarchy manager functionality in the Informatica Data Director application.

This concludes the lab.