Oracle Data Watch and Repair for Master Data Management

User's Guide

April, 2008

ORACLE

Table of Contents

1. Introduction

1.1 What is Data Watch and Repair for Master Data Management?

Data Watch and Repair for Master Data Management (DWR) is a profiling and correction solution created to assist data governance processes in Oracle's Master Data Management (MDM) solutions. MDM applications must successfully consolidate and clean up a system's master data, sharing it with multiple connected entities to achieve a single view of the data. However, MDM systems face a never-ending challenge: the constant state of flux of master data. Not only does master data quickly become out of date as new events happen and need to be captured in the system, but also any incoming data can potentially be inaccurate, either from entry mistakes or purposely misrepresented data.

Therefore, in order to leverage this achieved single view, it is crucial to implement a data governance plan. Being able to look at the data constantly, thoroughly and easily ensures that any data decay can quickly be noticed and the necessary correction or cleansing steps can be taken. Consequently, it will be easier to keep up reliable and useful data to make asserted and timely business decisions. Data Watch and Repair is a tool created for these specific tasks. The product, therefore, includes the following,

- ***Profiling function:*** to discover the structure of the data and understand it
- ***Application of data rules:*** to evaluate compliance and quality of the data
- ***Correction maps:*** to specify the necessary cleansing strategy to be used with data not compliant to a given data rule

In addition to these capabilities, the product also comes with a set of pre-written data rules that are common and relevant in an MDM context. Sample data rules for both customer and product hubs are created for out-of-the-box usage to perform basic data governance tasks. Moreover, they serve as example rules that illustrate how to define data rules, facilitating the learning of how to implement new data rules in the rule syntax.

Being in charge of data lifecycle activities, data stewards are most likely to be the primary users of this product. A typical day as a data steward might involve running initial data profiling tasks to inspect the data and to run data fixing routines if necessary. As part of these activities, a data steward could also customize the data rules to gain more insight on suspect data or as a consequence of a new data quality bar imposed or changing business requirements.

OWB currently only runs with Oracle database. If an MDM application is running on a third-party database, OWB will still need an Oracle database license, and a gateway needs to be setup with the third-party database.

1.2 Overview of Product Functionality

As mentioned above, the tool allows data profiling and analysis on the MDM solutions' databases. Additionally, data rules can be applied to the profiled data to assess data compliance, and when necessary, correction mappings can be created to populate corrected data back into the system. This process is illustrated in Figure 1 below, where the basic flow of the functionality is shown.

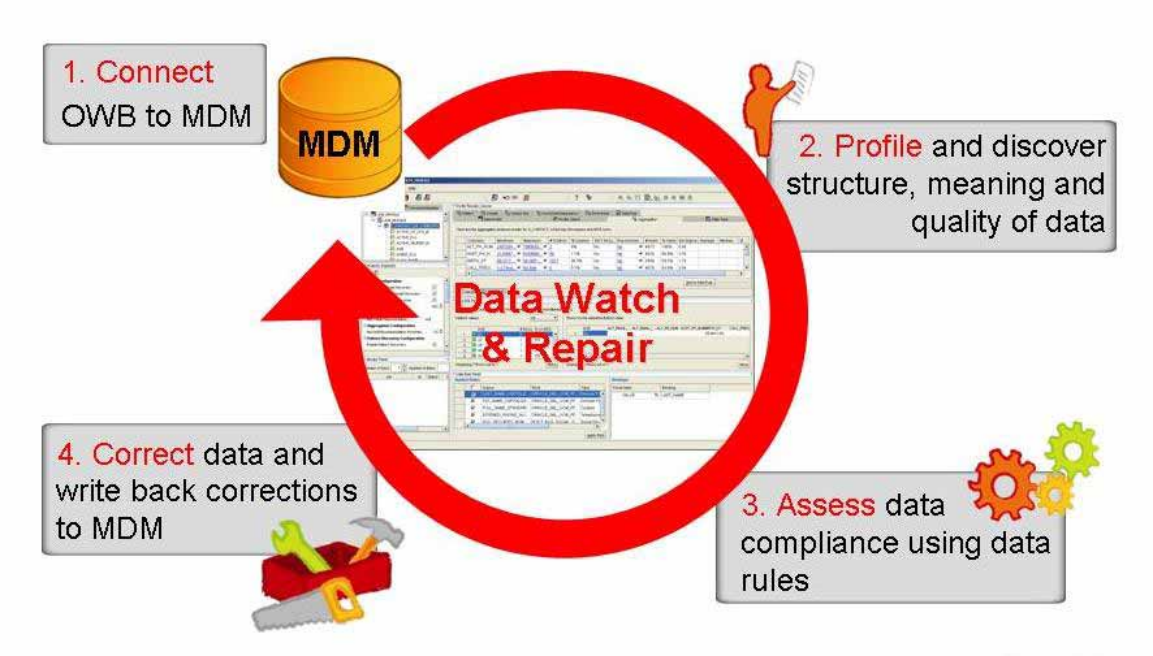


Figure 1. Diagram of Data Watch and Repair process

This functionality is powered by Oracle Warehouse Builder (OWB) and has been integrated to work with Oracle's MDM product family. The solution includes,

- **Warehouse Builder connector for Data Watch and Repair for Master data Management:** to import metadata from a specific MDM application (Customer Data Hub, Universal Customer Master, Product Information Management Hub) into OWB for profiling and correction purposes or for general data integration purposes for MDM

- **Profiling:** the ability to profile up to 165 columns on any table selected from those imported by the connector

- **Data rules:** these help ensure data quality by determining the conditions for legal data within a table or legal relationships between tables. Data rules can be created in the following ways:

    - ***Manual creation***: used to define exactly what data is considered compliant or non-compliant within a table. To illustrate how these rules are defined, a set of rules specific to

common MDM tasks is included as well. These rules are described with more detail later, but examples of these are contact completeness and name capitalizations.

- ***Derived:*** when viewing profiling results, one can also create data rules. This allows the user to quickly and seamlessly move between data profiling and data correction. For example, if an attribute is found to have a domain with a set of recurrent values, a domain data rule with these discovered values can be created automatically.

- **Data correction:** correction schemas can be created to clean up or report data that is legal according to the data rules. Non-compliant data can be deleted, reported or corrected. After determining this, the corrected data can be written back to the MDM database.

A user can iterate through these steps regularly whenever necessary or during scheduled data quality routines. The advantage of this tool is that it gives the user a very thorough and accessible view of the system's data, discovering data features on many different dimensions. Crucial business rules can be measured regularly, while new data rules can be applied as new data inconsistencies are exposed.

Furthermore, there is added value in having all these product components integrated within the OWB Design Center. This makes the transition between the different steps in the process (profiling, then creating data rules and correcting inconsistencies) very fluid and allows an ongoing, iterative process for maintaining and monitoring data quality.

1.3 Solution Architecture

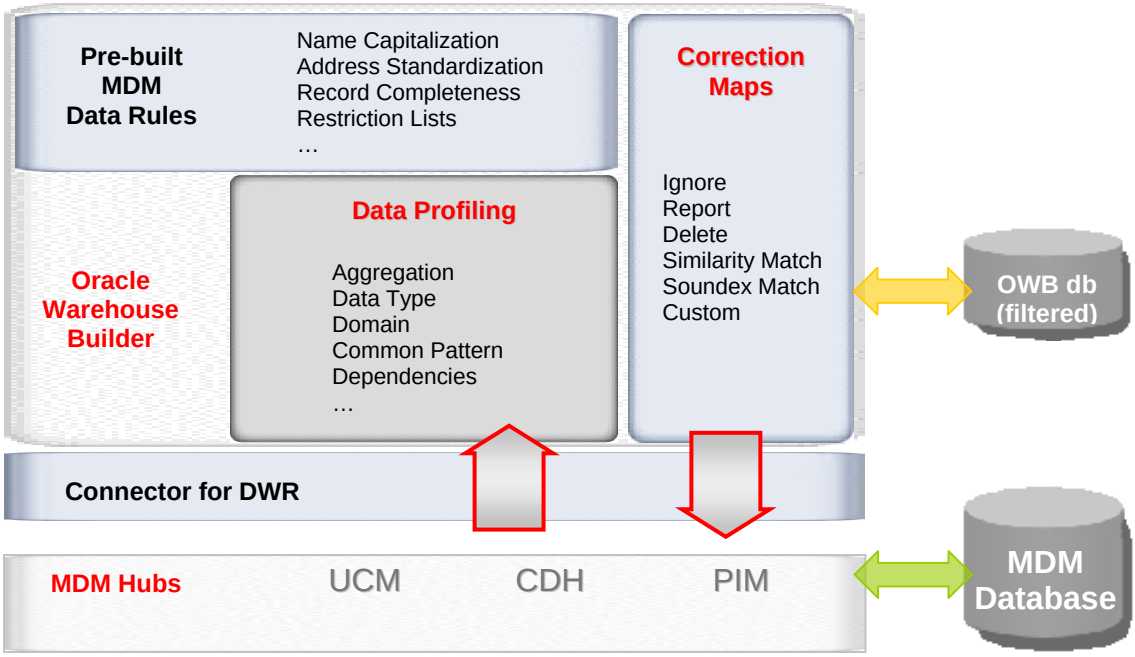


Figure 2. Diagram of solution architecture

1.4 Product Requirements and Options

To use this product, you need the following software installed in your system:

- Oracle Database 11g Release 1
- Oracle Warehouse Builder (OWB) 11g Release 1 installed with the database
- OWB Data Quality Option, installed with Oracle Warehouse Builder
- OWB Connector for Oracle Master Data Management, installed[1] with Oracle Warehouse Builder
- One or more of the following MDM applications: Customer Data Hub, Product Information Management or Universal Customer Master

The product itself is offered as an additional connector type for OWB. There are 3 different options to support each of the current MDM applications. The connectors enable access to data and metadata in a set of relevant tables for each different application.

1.4.1 Universal Customer Master (UCM)

This connector includes the relevant base tables for UCM, those containing important customer data. Additionally, the related Source Data History (SDH) tables are included to allow profiling during initial loads, as well as the EIM interface tables to enable writing back the corrected customer data. Table 1 below shows the tables included in the UCM connector. In addition to the tables, data rules that check for basic data quality metrics in customer data are included as well.

Tables imported by UCM connector into OWB	
EIM_ACCOUNT	S_DEDUP_RESULT
EIM_ACCOUNT3	S_DYN_HRCHY
EIM_ADDR_PER	S_DYN_HRCHY_REL
EIM_ASSET	S_DYNHR_RPT_REL
EIM_BU	S_EMP_PER
EIM_CONTACT	S_ORG_EXT
EIM_CONTACT3	S_ORG_GROUP
EIM_DYN_HRCHY	S_ORG_PRTNR
EIM_EMPLOYEE	S_PARTY
EIM_FN_ASSET1	S_PARTY_GROUP
EIM_GROUP	S_PARTY_PER
EIM_POSITION	S_PARTY_REL
EIM_UCM_ADRPER	S_PARTY_RPT_REL
EIM_UCM_ASSET	S_POSTN
EIM_UCM_ASTCON	S_PRIVACY
EIM_UCM_CON	S_UCM_ADDR_PER

[1] In the initial version of OWB 11g Release 1 you will need to install the connector from a separate directory on the install media.

EIM_UCM_CONCHLD	S_UCM_ASSET
EIM_UCM_OGPCHD	S_UCM_ASSET_CON
EIM_UCM_ORG	S_UCM_ATGP
EIM_UCM_ORGCHLD	S_UCM_ATGP_FLD
EIM_UCM_ORGGRP	S_UCM_ATGP_RULE
EIM_UCM_PRIVCY	S_UCM_CON_ATGP
EIM_USER	S_UCM_CON_CHILD
EIM_USERLIST	S_UCM_CON_MERGE
S_ADDR_PER	S_UCM_CONF_LVL
S_ASSET	S_UCM_CONTACT
S_ASSET_CON	S_UCM_DEDUP
S_ASSETCON_ADDR	S_UCM_OGP_ATGP
S_BU	S_UCM_OGP_CHILD
S_CIF_AST_MAP	S_UCM_ORG_ATGP
S_CIF_CON_MAP	S_UCM_ORG_CHILD
S_CIF_EXT_SYST	S_UCM_ORG_EXT
S_CIF_ORG_MAP	S_UCM_ORG_MERGE
S_CIF_ORGRP_MAP	S_UCM_ORGGRP
S_CIF_SYS_DTL	S_UCM_PRIVACY
S_CON_ADDR	S_UCM_RULE_SET
S_CONTACT	S_USER S_USERLIST

Table 1: UCM tables imported into Data Watch and Repair connector

1.4.2 Customer Data Hub (CDH*)*

This connector includes the relevant core tables for CDH, those containing important customer data. Additionally, the related bulk import tables are included to allow profiling during initial loads. Due to the EBS architecture, write-back are only supported through the application's APIs. Currently a write-back solution is not provided for CDH, but the next release of the product will allow the user to write back corrected data through the APIs for the most commonly used attributes. Table 2 below shows the tables included in the CDH connector. In addition to the tables, data rules that check for basic data quality metrics in customer data are included as well.

Tables imported by CDH connector into OWB	
HZ_CONTACT_POINTS	HZ_IMP_FINNUMBERS_INT
HZ_IMP_ADDRESSES_INT	HZ_IMP_FINREPORTS_INT
HZ_IMP_ADDRESSUSES_INT	HZ_IMP_PARTIES_INT
HZ_IMP_CLASSIFICS_INT	HZ_IMP_RELSHIPS_INT
HZ_IMP_CONTACTPTS_INT	HZ_ORGANIZATION_PROFILES
HZ_IMP_CONTACTROLES_INT	HZ_PARTIES
HZ_IMP_CONTACTS_INT	HZ_PERSON_PROFILES
HZ_IMP_CREDITRTNGS_INT	HZ_ORGANIZATION_PROFILES (view)

Table 2: CDH tables imported into Data Watch and Repair connector

1.4.3 Product Information Management (PIM)

This connector includes the relevant core tables for PIM, those containing important product data. Additionally, the related bulk import tables are included to allow profiling during initial loads. Due to the EBS architecture, write-back are only supported through the application's APIs. Currently a write-back solution is not provided for PIM, but the next release of the product will allow the user to write back corrected data through the APIs for the most commonly used attributes. Table 3 below shows the tables included in the PIM connector. In addition to the tables, data rules that check for basic data quality metrics in product data are included as well.

Tables imported by PIM connector into OWB	
BOM_BILL_OF_MTLS_INTERFACE	EGO_MTL_SY_ITEMS_EXT_B
BOM_COMPONENTS_B	ENG_CHANGES_EXT_B
BOM_INVENTORY_COMPS_INTERFACE	ENG_CHANGES_LINES_EXT_B
BOM_REF_DESGS_INTERFACE	MTL_CATEGORY_SETS_B
BOM_STRUCTURES_B	MTL_ITEM_CATALOG_GROUPS_VL (view)
BOM_SUB_COMPS_INTERFACE	MTL_ITEM_CATEGORIES
EGO_AML_INTF	MTL_ITEM_CATEGORIES_INTERFACE
EGO_ATT_GROUPS_V (view)	MTL_ITEM_REVISIONS_INTERFACE
EGO_ATTRS_V (view)	MTL_SYSTEM_ITEMS_B
EGO_ITEM_PEOPLE_INTF	MTL_SYSTEM_ITEMS_INTERFACE
EGO_ITM_USR_ATTR_INTRFC	

Table 3: PIM tables imported into Data Watch and Repair connector

1.4.4 Additional Features

In addition to the connectors, MDM-specific data quality metrics are addressed by each option. Some are met with the built-in functionality in the profiling step, while others have been addressed with custom data rules. Data rules addressing additional requirements for Customer Hubs have been included. CDH and UCM share the same rules as they both master customer data. The data quality metrics addressed for CDH and UCM are shown in Table 4 below. A more detailed description of the built-in functionality and the custom content can be found in the Examining and Understanding Profile Results and the Custom MDM Rules: Customer Data Rules section, respectively.

DQ Metric	Customer Rule	Description or Example
Completeness	Attribute completeness	Discovers # and % of null values
	Contact Completeness	Requires all name, address, SSN, … are not null
Conformity	Data Type	Data type, length, & precision documented & found
	Data Domain	All values are within specified domain
	Restricted values	Values are not in a list of restricted values: e.g. "66666…"…

Uniqueness	Unique Key discovery	Discovers # and % of unique values
Pattern and Common Format	Name Standardization	First, Middle and Last name not null & properly capitalized
	Common Pattern	Common pattern required (e.g. phone, email), % conformity
	Name Capitalization	"Aaaa", "Aaa Bbbb", "Aa-Bbb", …
	Extended Phone Numbers	More extensive definition of allowable phone number formats. Can extend to specific country formats, etc.
	International Phone Numbers	
No Access Lists	by Name Only	Restriction lists for specific records, filtered by name, SSN and/or emails
	by Name or SSN	
	by Email List	

Table 4: Data Quality metrics addressed in Customer Hubs (UCM and CDH)

2. Setting up Oracle Data Watch and Repair for Master Data Management

2.1 Standalone OWB client, remote server

2.1.1 Oracle Warehouse Builder installation

Install the Oracle Warehouse Builder (OWB) client on your computer if your OWB repository will reside in a remote server. If you are using the OWB client where the database resides, then you do not need to install anything, it should already be included in your Oracle database environment. You will need to unlock and set a password for the OWBSYS user in the database.

2.1.2 Data Watch and Repair installation

If this is the first time you are using OWB, you will need to create a workspace for OWB, with the OWB user. To do so, follow these steps:

- Go to: **Programs > *ORACLE_HOME* > Warehouse Builder > Administration > Repository Assistant**
- Click next
- Review Database Information, click next
- Select **Manage Warehouse Builder workspaces**
- Select **Create a new Warehouse Builder workspace**
- Select **Create a workspace with a new user as workspace owner**
- Enter DBA password for user 'SYSTEM'
    - Create workspace owner
    - Assign any username and password
- Assign any name to the workspace (e.g. 'ws1')
- (Optional) Register other database users as workspace users
    - Review summary of chosen options and click Finish.

After doing this, you will need to import the relevant MDM connector(s) into OWB. Your 11.1.0.6 install medium has a DWR directory with all the scripts. If not download the appropriate files from Metalink referring to patch ID: 6996453. The **DWR** folder should have two types of files, .tcl scripts and .xml documents. The TCL scripts can be run either in OMB Plus or directly from the OWB Design Center. The XML documents are the actual connector definitions that specify which metadata to import from the applications. To import the connector(s), follow these steps:

- Locate the "create_mivs.tcl" script in the <OWB_HOME>/owb/misc/DWR/. This script contains the installation commands for the connectors.

- If import is done from an OMB Plus session, check the connection information for the OMBCONNECT command, to make sure it matches the desired OWB repository. Then run the script within OMB Plus.
- To import the connector(s) within the OWB Design Center,
    - Open this script in a text editor.
    - Ignore the first line, since the Design Center is already connected to the correct repository.
    - Copy and paste the remaining lines into the OMB*Plus window in the Design Center
    - If the import was successful, three new nodes appear in the **Project Explorer** under the **Applications** node. One node for Customer Data Hub, one node for Product Information Management Hub and one node for Universal Customer Master respectively.

2.1.3 Connecting to and importing from MDM application

Once you have imported the connector definition, you need to create a module in which your specific MDM application's metadata will be imported. You will have to define a name as well as the location information for your MDM application database for OWB to connect to it. To create a connection location to an MDM hub instance:

- Go to **Connection Explorer**, typically found on the right side of the Design Center.
- Go to **Locations > Databases > Oracle**
- Right-click on **Oracle** and select **New…**
- Follow the steps to create a new Oracle Database Location. The following details must be provided to connect to the database:
    - User Name
    - Password
    - Host
    - Service Name
    - Port
    - Version
- Like most other metadata in OWB, the location information can also be imported into and exported from different OWB repositories as metadata packages. Do note, however, that although locations will be automatically created when you import them, the password will not be included for security purposes. The password must be manually entered.

Once the location is created, then you can go on and create the new module for your MDM application. To do so:

- Right-click on your new application node under Applications and click on **New…**
- Follow the steps in the Create Module wizard, by naming the module and then defining a location from which the module will import metadata.
- When you are done, a new node will appear under your previous Application node (e.g. ORACLE_DWR_FOR_CDH > CDH_MODULE).
- To import the metadata of the relevant tables, right-click on your new module node and click on **Import** to open the Import Metadata Wizard.

- Follow the steps in the wizard to import the tables, views or sequences you wish to import the metadata for, select the specific objects you want to import and wait until the wizard is done.

Now, the metadata from your MDM application has been imported into OWB. You can expand the module to view the tables, views and sequences imported. Additionally, you can view the data on any of these objects by right-clicking and clicking on **Data…** This brings up the relational data viewer where you can explore your system's data.

2.1.4 Importing MDM data rules

After importing the metadata from the MDM application, you are ready to import the custom data rules created for both Customer Hubs (UCM and CDH). The content for MDM consists mainly of customized data rules that check the compliance to common data MDM data metrics around customer data. These rules are in metadata packages which can be imported into OWB, in the same way location information can be imported.

To do so, in the Design Center, go to the **Design** menu and go to **Import >> Warehouse Builder Metadata**. In File Name, browse to <OWB_HOME>/owb/misc/DWR/ and choose customer_data_rules.mdl. Now import this file into the Warehouse Builder repository. The imported data rules will appear in the Public Data Rules node in the Global Explorer. You now have a folder called MDM_CUSTOMER_RULES, which contains all rules.

3. Creating a Data Profile

Data profiling is the first step to improve the data quality of a system within any organization. It is a robust and thorough data analysis method used to discover and evaluate the defects in the system's data. By doing data profiling, you can discover things about your data, including (but not limited to) a domain of valid zip codes, a range of product discounts, columns that hold the pattern of an e-mail address, a one-to-many relationship between columns and anomalies and outliers within columns.

Data profiling offers three main types of analysis: attribute analysis, functional dependency, and referential analysis. You can also create custom profiling processes using data rules, allowing you to validate custom rules against the actual data and get a score of their accuracy. Figure 3 displays a representation of the types of data profiling and how you can perform each type.

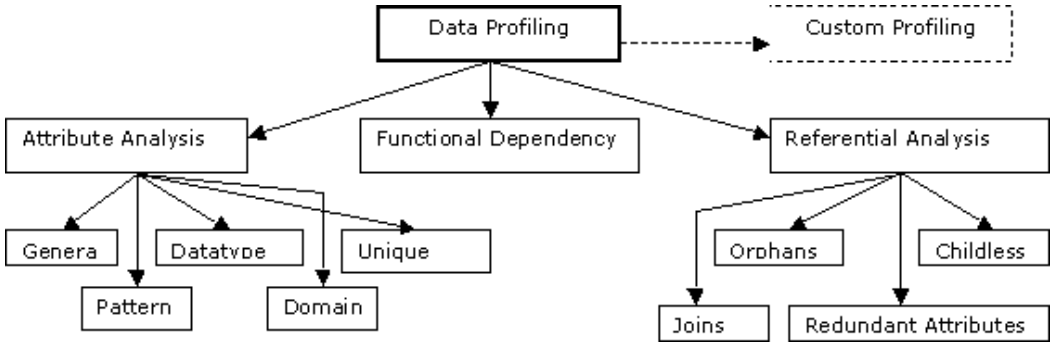


Figure 3: Types of data profiling

Tthe results of a data profile job are presented in the Data Profile Editor in both tabular and graphical format. The user can also drill down into any of the results and look at the actual data related to this result. In addition to doing this, within the data profile itself, the user can then derive data rules, manually or automatically, based on what has been learned from the profiling results. The level of compliance to data rules is immediately populated or updated.

To create a data profile for your MDM application, the following steps should be followed:

- Connecting to your MDM application
- Selecting the tables and attributes to be profiled
- Selecting options for data profile job
- Examine data profiling results

To connect to the MDM application, enter the OWB Design Center and look under the **Applications** in the **Project Explorer** to expand it. If you have imported the metadata from your MDM application with the corresponding connector, there should be a node for your application. For example, in Figure 4 below, the UCM connector is shown in the **ORACLE_DWR_FOR_UCM** node under the Applications node. This means it has already been installed and the tables' metadata has been imported.

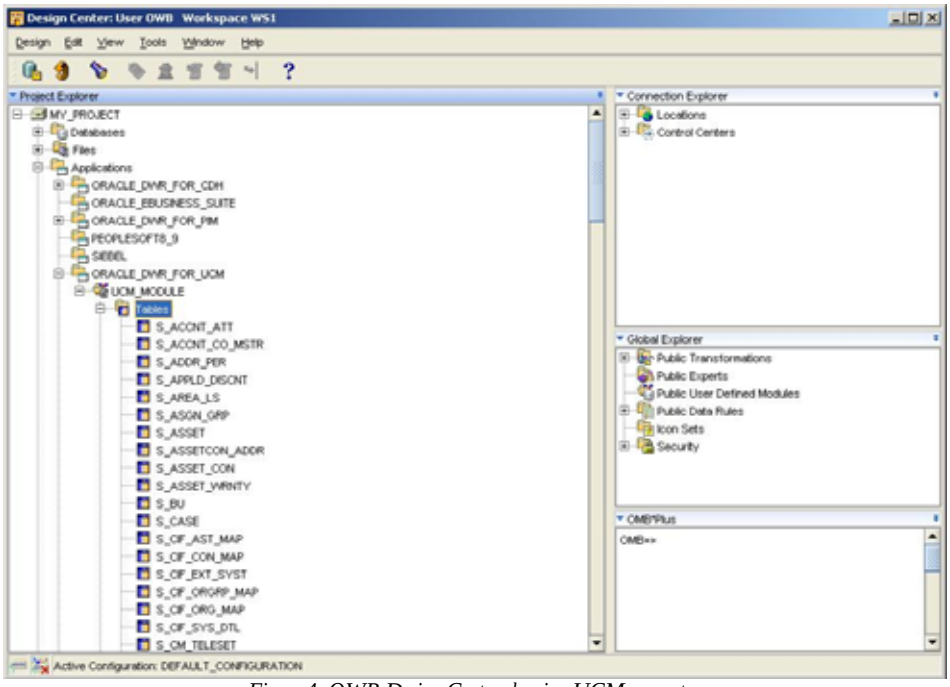


Figure 4: OWB Design Center showing UCM connector

For more information about installing the connectors and checking the connection from OWB to your MDM application, please refer to the Setup section.

3.1 Selecting profile objects

If you see the MDM application and its tables in the OWB Design Center, you are ready to create a data profile. To do so, click on the **Data Profiles** node to expand it. You will see some previously created data profiles there, if any have been created. To invoke the **Create Data Profile Wizard**, right-click on **Data Profiles** and click on **New…**

Then, follow the two steps of the wizard. First, you need to name the data profile, and then select the tables to be profiled, as shown below in Figure 5. OWB has a limitation of being able to profile only up to 165 columns in each table selected. When choosing a table with more than 165 columns, the wizard will give you an option to choose the first 165 columns. Alternatively, you may create an **Attribute Set** for any tables beforehand, to choose a subset of 165 or less columns, making sure you include those most important or relevant to your profile job. This can be done by opening the **Data Object Editor** window for any given table (by double-clicking on the table within the connector's list of tables). Within the editor's **Attribute Sets** tab, new attribute sets can be defined.

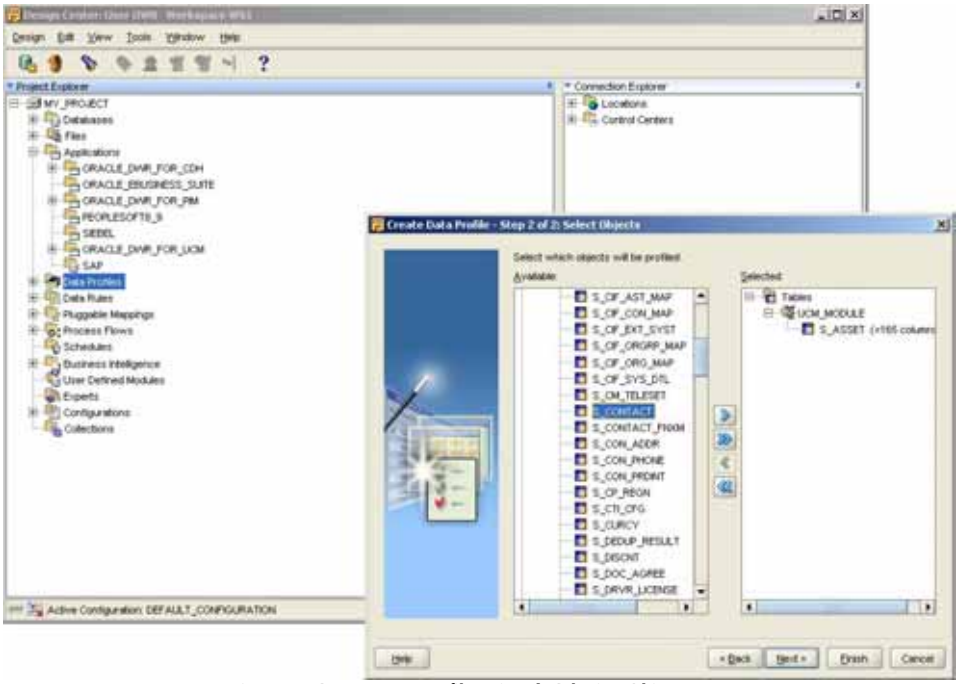


Figure 5: Create Data Profile Wizard: Selecting Objects screen

By finishing the steps of the profile, the data profile is created, although not yet run. To do this, double-click on the newly created data profile (it should be listed in the expanded list under the **Data Profiles** node). The **Data Profile Editor** should now be open with the new job. Once the Data Profile Editor is open, you can configure your profile job before running it. You can choose to run certain profiling options, such as discovery of data type, common format, pattern, and domain. For each of these, there might also be additional parameters to be set (for example, a minimum number of rows counted for a value to be considered within a domain). More detail on the specific configuration options can be found in the documentation for OWB.

To profile the data, within the Data Profile editor, from the **Profile** menu, select **Profile**. After the metadata preparation is complete, the Profiling Initiated dialog box is displayed informing you that the profiling job has started. Click **OK**. The profiling process will continue to run until it is completed. You can view the status of the profiling job in the Monitor Panel of the Data Profile Editor. After the profiling job is complete, the status displays as complete.

3.2 Examining and Understanding Profile Results

After the profile operation is complete, the profiling results can be viewed and analyzed in the Data Profile Editor itself. The profiling results contain a variety of analytical and statistical information about the data profiled. The user can immediately drill down into anomalies and view the data that caused them. You can then determine what data must be corrected, and apply data rules to your data to assess the compliance of your data to your expected business rules.

To view the profile results, select objects in the Profile Objects tab on the top left corner of the object tree to focus the results on a specific object. The results of the selected object are displayed in the Profile

Results Canvas. You can switch between objects. The tab that you had selected for the previous object remains selected. The Profile Results Canvas contains the following tabs that display the results of data profiling:

Data Profile: This is a space for any textual notes regarding the data profile can be recorded and saved.

Profile Object: This tab shows the data in the profile object that has been profiled. Since it queries all the data directly from the MDM database, it can take up to a few minutes to populate. Different queries can be done on the data itself to examine it directly.

Aggregation: This tab shows the following results for each profiled column in the profile object,

- **Minimum:** The minimum value with respect to the inherent database ordering of a specific type.
- **Maximum:** The maximum value with respect to the inherent database ordering of a specific type.
- **# and % Distinct:** The number of distinct values and the percentage they represent in the entire row set.
- **NOT NULL:** Whether or not a column is required.
- **Recommended NOT NULL:** Whether or not a column is recommended to be required.
- **# and % Nulls:** The total number of nulls and the percentage they represent in the entire row set.
- **Six-Sigma:** For each column, the number of null values (defects) to the total number of rows in the table (opportunities).
- **Average:** The arithmetic mean for the entire row set, if these values are numerical.
- **Median:** The median for the entire row set, if these values are numerical.
- **Std Dev:** The standard deviation for the entire row set, if these values are numerical.

The tabular and graphical aggregation tabs are shown below in Figures 6 and 7 respectively.

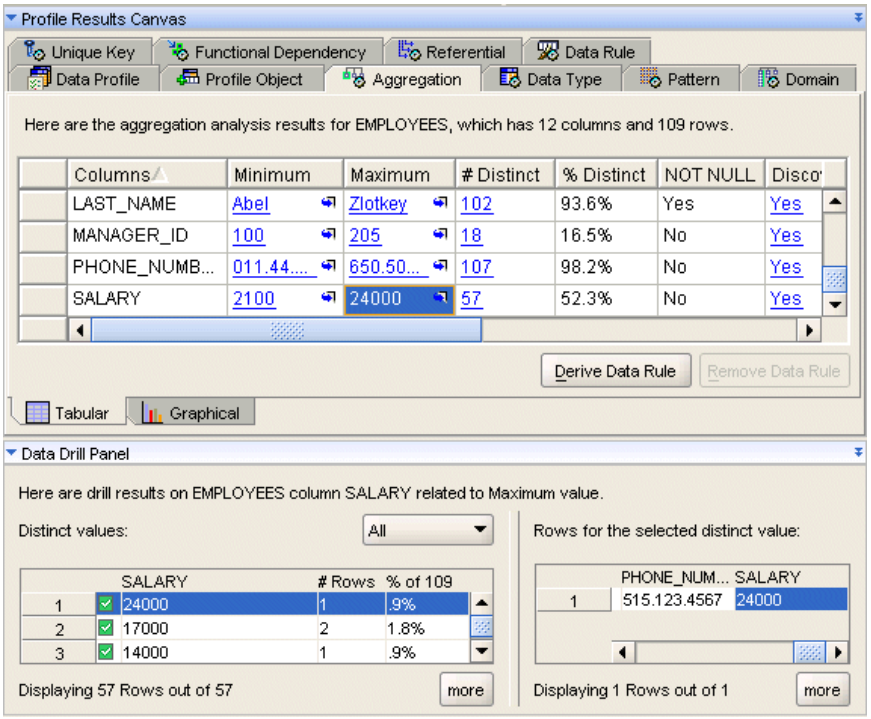


Figure 6: Aggregation tab: Tabular results

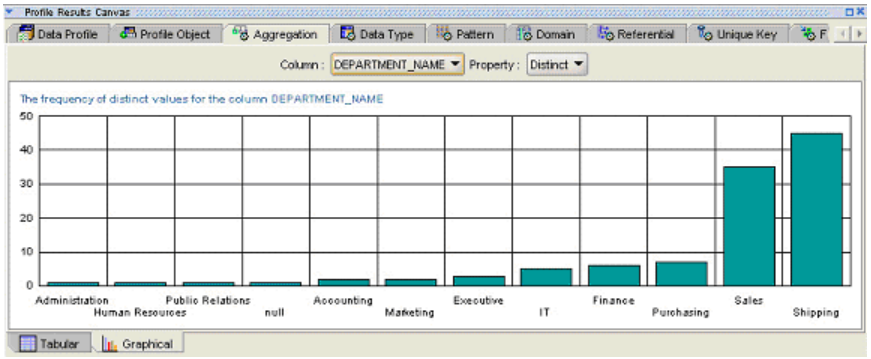


Figure 7: Aggregation tab: Graphical results

Data Type: This tab shows the following results for each profiled column in the profile object,

- **Documented Datatype:** Datatype stated in column's metadata.
- **Dominant Datatype and %:** Actual dominant datatype found and percentage of total records with this datatype.
- **Documented Length:** Length stated in column's metadata.
- **Minimum Length:** Min value of length found in data.
- **Maximum Length:** max value of length found in data.
- **Dominant Length and %:** Actual dominant length found and percentage of all records with this length.

The same 5 columns for length are also given for both precision and scale.

Pattern: This tab shows any patterns discovered, it shows the following results for each column,

- **Dominant Character Pattern and % Compliance:** Shows the most common pattern found, character by character (e.g. it can determine a zip code column has mostly values of 5 digits in a row, or 5 digits, a hyphen and followed by 4 digits). The percentage of compliant rows to this dominant pattern is also shown.
- **Dominant Word Pattern and % Compliance:** Shows the most common word pattern found. The percentage of compliant rows to this dominant pattern is also shown.
- **Common Format and % Compliance:** Based on patterns and other techniques, tries to figure out if an attribute is a Name, Address, Date, Boolean, Social Security Number, E-mail or URL. The percentage of compliant rows to this dominant pattern is also shown.

Domain: This tab displays results about the possible set of values that exist in a certain column. The results be viewed from either the tabular or graphical sub-tabs. Figure 8 shows a screenshot of tabular domain results.

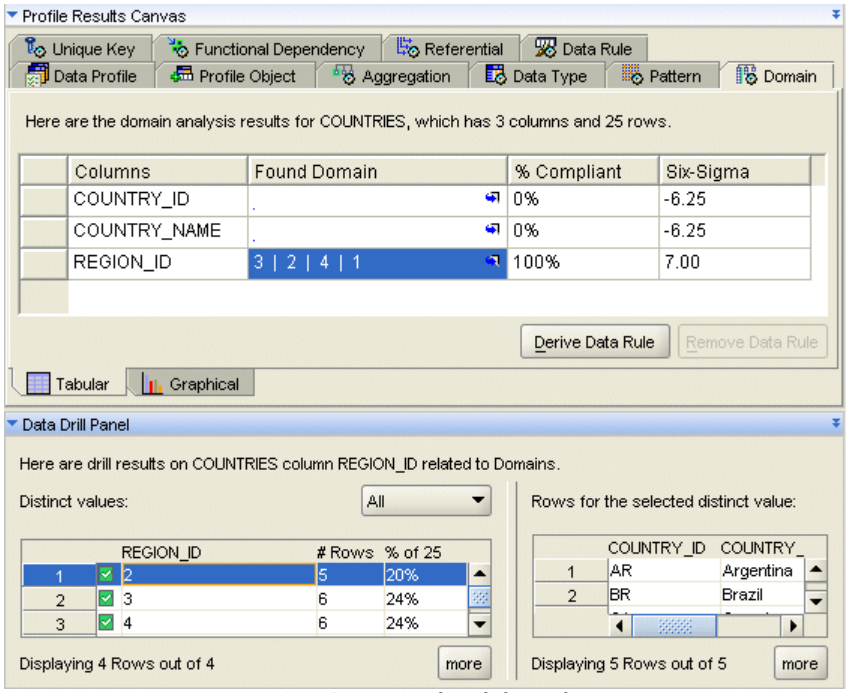


Figure 8: Domain tab: Tabular results

Unique Key: This tab provides information about the existing unique keys that were documented in the data dictionary, and possible unique keys or key combinations that were detected by the data profiling operation. The uniqueness % is shown for each. For each unique key, discovered or documented, the following is given:

- **Documented?** Shows if this was documented as a unique key before.
- **Discovered? and Local Attribute(s):** From analysis, the profiler might determine a unique key should be created. Local Attribute(s) denotes which columns should be used for this unique key.
- **# and % Unique:** The number of rows and percentage of total row set in which this attribute or set of attributes is unique.
- **Six-sigma:** For each unique key, the number of values that do not comply with the documented unique key (defects) to the total number of rows in the table (opportunities).

Functional Dependency: This tab shows any attribute(s) that seem to depend on or determine other attributes.

- **Determinant:** The name of the attribute that is found to determine the attribute listed in the Dependent
- **Dependent:** The name of the attribute found to be determined by value of another attribute
- **# Defects and % Compliance:** These columns show the number of values that did not show the discovered dependency, and the percentage of values that did meet it.
- **Six Sigma:** The six sigma value.
- **Type:** The suggested action for the discovered dependency.

Referential: This tab shows the foreign keys documented, as well as relationships discovered during profiling. It also shows the level of compliance for each relationship. In addition to the tabular and graphical sub-tabs, two other sub-tabs are available: Joins and Redundant Columns. *Joins* shows a join analysis, with the relative size and counts for joins, orphans, and childless objects; while *Redundant Columns* show the columns in a child table that are present in the primary one.

Data Rule: This tab shows the rules that have been defined. For each rule, it gives the name, the rule type, the origin (if it was derived or custom written), the percentage of compliance (the percentage of rows that comply with the rule) and the # of defects (the number of rows that do not comply). The data rules on this tab reflect the active data rules in the Data Rule panel. You do not directly create data rules on this tab. For more information about how to apply data rules and understand the compliance measures, refer to the Looking at data rule compliance section.

4. Creating a data rule

After viewing and understanding the data profiling results, data rules can be created to attain more insight on specific aspects of the data. A data rule is an expression that determines the set of legal data that can be stored within a data object. Data rules also determine the set of legal relationships that can exist between data objects. Although data rules can be created and applied manually, they can also be easily derived while examining data profiling results.

4.1 Deriving a Data Rule

When reviewing profiling results, you can determine which findings you want derived into data rules. The types of results that warrant data rules vary. Common derived data rules include a detected domain, a functional dependency between two attributes, or a discovered unique key.

To do this, select the tab that displays the results from which you want to derive a data rule, and then select the cell that contains the specific result. Then, click on the **Derive Data Rule** button (which should not be grayed out if the result can be derived into a data rule). The Derive Data Rule Wizard opens. Follow the wizard steps, some fields are already filled in by default. The Name and Type fields are already populated based on the tab from which the data rule is derived. The Type filed cannot be changed if the data rule is derived.

Additional fields in the lower portion of this page define the parameters for the data rule. Some of these fields are populated with values based on the result of data profiling. The number and type of fields depends on the type of data rule. At the end of the wizard, the Summary page is displayed and you can review the options set before completing the data rule creation.

Once you are done, the data rule is created and it appears in the Data Rule panel of the Data Profile Editor. The derived data rule is also added to the Derived_Data_Rules node under Data Rules node in the Project Explorer. You can reuse this data rule by attaching it to other data objects. Figure 9 shows the Derive Data Rule button when selecting a discovered domain for the REGION_ID column.

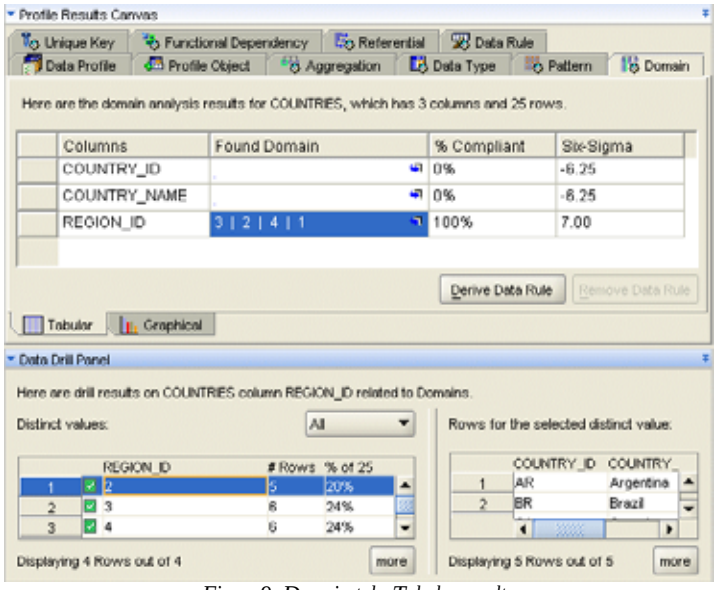


Figure 9: Domain tab: Tabular results

4.2 Custom MDM Rules: Customer Data Rules

As mentioned, a set of data rules of common usage within customer hubs have been pre-seeded in the product. This is not intended to be an extensive list by any means. Instead, they are aimed to address some requirements commonly required in customer hub implementations.

The data rules are written in SQL expressions and the patterns are defined in Oracle regular expression. For more information about the syntax of the data rules, refer to the Understanding Oracle regular expression section. In addition to the built-in data rules that check if data has the standard form of a social security number, a phone number, or has proper date formatting, the following are data rules to test customer data on additional constraints.

4.2.1 Attribute Dependency

Description: This rule enforces that if one column, Attribute1, is filled, then the other one, Attribute2, must be as well (e.g. If a supplier field is used, then a supplier contact information must be present.) There is also a filter that calls only looks for records created after a certain creation date.

Rule Definition:

Attribute	Type
ATTRIBUTE1	VARCHAR2
ATTRIBUTE2	VARCHAR2
CREATIONDATE	DATE

```
"THIS"."CREATIONDATE" < '28-MAR-08' OR
 "THIS"."ATTRIBUTE1" IS NULL OR
( "THIS"."ATTRIBUTE1" IS NOT NULL AND "THIS"."ATTRIBUTE2" IS NOT NULL)
```

4.2.2 Contact Completeness

Description: This rule checks the following attributes and checks that their values are not null. The attributes are associated with a specific column at the time of applying the rule.

Rule Definition:

Attribute	Type
EMAIL	VARCHAR2
SSN	VARCHAR2
CELLPHONE	VARCHAR2
BIRTHDATE	DATE
FIRSTNAME	VARCHAR2
MIDDLENAME	VARCHAR2
LASTNAME	VARCHAR2

"THIS"."EMAIL" is not null And
"THIS"."SSN" is not null And
"THIS"."CELLPHONE" is not null And
"THIS"."BIRTHDAY" is not null And
"THIS"."FIRSTNAME" is not null And
"THIS"."MIDDLENAME" is not null And
"THIS"."LASTNAME" is not null

4.2.3 Extended Phone Numbers

Description: This rule extends the regular built-in rule that recognizes standard phone numbers to include more patterns.

Rule Definition:

Format to use: Telephone Number

Regular expression Values	Example
(^[[:space:]]*[0-9]{3}[[:punct:]]|[:space:]]?[0-9]{4}[[:space:]]*$)	'xxx-xxxx', 'xxx xxxx'
(^[[:space:]]*((\([0-9]{3}\))|([0-9]{3}))[[:punct:]]|[:space:]]?[0-9]{3}[[:punct:]]|[:space:]]?[0-9]{4}[[:space:]]*$)	'xxx-xxx-xxxx', 'xxx xxx xxxx' '(xxx)-xxx-xxxx', '(xxx) xxx xxxx'
(^[[:space:]]*(1|\(1\))[[:punct:]]|[:space:]]?((\([0-9]{3}\))|([0-9]{3}))[[:punct:]]|[:space:]]?[0-9]{3}[[:punct:]]|[:space:]]?[0-9]{4}[[:space:]]*$)	'1-xxx-xxx-xxxx' '1 xxx xxx xxxx' '(1)-(xxx)-xxx-xxxx' '(1) (xxx) xxx xxxx'…
(^[[:space:]]*011[0-9]*[[:space:]]*$)	'011xxx…'
(^[[:space:]]*[+]33[0-9]{9}[[:space:]]*$)	'+33xxxxxxxxx'
(^[[:space:]]*[+]31[0-9]{9}[[:space:]]*$)	'+31xxxxxxxxx'
(^[[:space:]]*[+]49[0-9]{10}[[:space:]]*$)	'+49xxxxxxxxx'

4.2.4 Full Name Standardization

Description: This rule checks that the values for first name, middle name and last name are all NOT NULL, to check that a full name is entered. Moreover, each value should also be capitalized (with the first letter upper case, and the remainder as lower case).

Rule Definition:

Attribute	Type
NAME_F	VARCHAR2
NAME_L	VARCHAR2
NAME_M	VARCHAR2

("THIS"."NAME_F" IS NOT NULL AND REGEXP_LIKE("THIS"."NAME_F",'(^[[:space:]]*[A-Z][a-z]*[[:space:]]*$)')) AND

("THIS"."NAME_L" IS NOT NULL AND REGEXP_LIKE("THIS"."NAME_M",'(^[[:space:]]*[A-Z][a-z]*[[:space:]]*$)')) AND ("THIS"."NAME_M" IS NOT NULL AND REGEXP_LIKE("THIS"."NAME_L",'(^[[:space:]]*[A-Z][a-z]*[[:space:]]*$)'))

4.2.5 Name Capitalization

Description: This rule defines a similar capitalization as the previous one but is applied to any and not more than one column. The rule also allows for more patterns of capitalization, accounting for values with two words separated by a space, hyphen or other forms of punctuation. This rule could be applied to multiple name columns separately.

Rule Definition:

Regular expression Values	Example
(^[[:space:]]*[A-Z][a-z]*[[:space:]]*$)	'Aaaaa…'
(^[[:space:]]*[A-Z][a-z]*-[A-Z][a-z]*[[:space:]]*$)	'Aaa..-Bbbbb…'
(^[[:space:]]*[A-Z]'[A-Z][a-z]*[[:space:]]*$)	'A'Bbbbb…'
(^[[:space:]]*[A-Z][a-z]*[[:space:]][A-Z][a-z]*[[:space:]]*$)	'Aaa… Bbbbb…'

4.2.6 No Access List by Name Only

Description: This rule flags any values that are in a previously defined "No Access List" as such, to ensure that these values will not be accidentally given access. The values that are denied access are specified by the First Name and Last Name attributes. The "No Access List" is a separately defined table.

Rule Definition:

("THIS"."NAME_F" not in (select name_f from no_access_list_by_name)) or
("THIS"."NAME_L" not in (select name_l from no_access_list_by_name))

4.2.7 No Access List by Name or SSN

Description: This rule is similar to the previous one, except it filters out any values with matching First Name or Last name and a matching Social Security Number.

Rule Definition:

(("THIS"."FIRST" NOT IN (SELECT NAME_F FROM NO_ACCESS_LIST_BY_NAME)) Or
("THIS"."LAST" NOT IN (SELECT NAME_L FROM NO_ACCESS_LIST_BY_NAME))) AND ("THIS"."SSN" IS NULL OR "THIS"."SSN" NOT IN (SELECT SSN FROM NO_ACCESS_LIST_BY_SSN))

4.2.8 No Email List

Description: This rule is similar to the previous one, except it filters out any values with an Email Address that appears in the "No Access List".

Rule Definition:

"THIS"."EMAIL" IS NULL OR "THIS"."EMAIL" NOT IN (SELECT EMAIL FROM NO_EMAIL_LIST)

4.2.9 International Phone Numbers

Description: This rule is similar to the Extended Phone Numbers rule but shows how a particular rule can be created for specific patterns, such as phone number patterns for different countries. Namely, this example is set to parse and recognize phone numbers from Spain.

Rule Definition:

Format to use: Telephone Number

Regular expression Values	Example
(^[[:space:]]*[+]34[0-9]{9}[:space:]*$)	'+34' + 'xxxxxxxxx'
(^[[:space:]]*[/(]34[/)][0-9]{9}[:space:]*$)	'(34)xxxxxxxxx'
(^[[:space:]]*[/(]34[/)][0-9]{9}[:space:][0-9]{4}[:space:]*$)	'(34)xxxxxxxxx xxxx'

4.3 Creating a Data Rule Manually

In addition to deriving rules and using the custom rules provided, you can also create a rule manually. An object can contain any number of data rules.

To create a data rule manually, you can go directly to the Project Explorer in OWB's main window. Right-click on the **Data Rules** node and click on **New…**, this opens the Create Data Rule Folder window, where you can make a new folder for data rules. If you want to create a data rule within an existing folder, just right-click directly on the folder's node and click on **New…** to open the Create Data Rule wizard.

Unlike deriving data rules, this wizard will not have any pre-populated fields. As before, creating a data rule within the wizard consists of the following steps: naming the rule, defining the type and parameters associated with the rule and finally reviewing the options. Table 5 shows the types of rules you can create.

Rule Type	Description
Domain List	A domain list rule defines a list of values that an attribute is allowed to have. For example, the Gender attribute can have 'M' or 'F'.
Domain Pattern List	A domain pattern list rule defines a list of patterns that an attribute is allowed to conform to. The patterns are defined in the Oracle Database regular expression syntax. An example pattern for a telephone number is as follows: (^[[:space:]]*[0-9]{ 3 }[[:punct:]|:space:]]?[0-9]{ 4 }[[:space:]]*$)
Domain Range	A domain range rule defines a range of values that an attribute is allowed to have. For example, the value of the salary attribute can be between 100 and 10000.
Common Format	A common format rule defines a known common format that an attribute is allowed to conform to. This rule type has many subtypes: Telephone Number, IP Address, SSN, URL, E-mail Address.
No Nulls	A no nulls rule specifies that the attribute cannot have null values. For example, the department_id attribute for an employee in the Employees table cannot be null.
Functional Dependency	A functional dependency defines that the data in the data object may be normalized.
Unique Key	A unique key data rule defines whether an attribute or group of attributes are unique in the given data object. For example, the name of a department should be unique.
Referential	A referential data rule defines the type of a relationship (1:x) a value must have to another value. For example, the department_id attribute of the Departments table should have a 1:n relationship with the department_id

	attribute of the Employees table.
Name and Address	A name and address data rule uses the Warehouse Builder Name and Address support to evaluate a group of attributes as a name or address.
Custom	A custom data rule applies a SQL expression that you specify to its input parameters. For example, you can create a custom rule called VALID_DATE with two input parameters, START_DATE and END_DATE. A valid expression for this rule is: "THIS"."END_DATE" > "THIS"."START_DATE".

Table 5: Data Rule types

4.3.1 Understanding Oracle regular expression

Domain pattern list rules are written using Oracle regular expression. Oracle Database supports regular expressions and uses them to search for patterns in string data in general. Regular expressions use standardized conventions with both meta-characters and literals. A quick reference table for meta-characters used in the regular expressions is provided in Table 6. However, for a more complete reference guide to Oracle regular expressions please refer to the Online Database Documentation.

Syntax	Description	Example
.	Matches any character in the database character set.	The expression a.b matches the strings abb, acb, and adb, but does not match acc.
+	Matches one or more occurrences of the preceding subexpression.	The expression a+ matches the strings a, aa, and aaa, but does not match bbb.
?	Matches zero or one occurrence of the preceding subexpression.	The expression ab?c matches the strings abc and ac, but does not match abbc.
*	Matches zero or more occurrences of the preceding subexpression.	The expression ab*c matches the strings ac, abc, and abbc, but does not match abb.
{m}	Matches exactly m occurrences of the preceding subexpression.	The expression a{3} matches the strings aaa, but does not match aa.
{m,}	Matches at least m occurrences of the preceding subexpression.	The expression a{3,} matches the strings aaa and aaaa, but does not match aa.
{m,n}	Matches at least m, but not more than n occurrences of the preceding subexpression.	The expression a{3,5} matches the strings aaa, aaaa, and aaaaa, but does not match aa.
|	Matches one of the alternatives.	The expression a|b matches character a or character b.
^	Matches the beginning of a string (default).	The expression ^def matches def in the string defghi but does not match def in abcdef.
$	Matches the end of a string (default).	The expression def$ matches def in the string abcdef but does not match def in the string defghi.

Table 6: Table of regular expression meta-characters

Below are a few examples of regular expression and what they denote:

4.4 Looking at Data Rule Compliance

Derived data rules are directly applied to the corresponding attributes. However, for manually created rules, you must bind the rule parameters and apply it to specific attributes in your data. You can bind a data rule to multiple tables within the project in which the data rule is defined.

To do this, go to the Data Rules panel in the bottom part of the Data Profile editor for the table in which you want to apply the rules. Click on **Apply Rule** to open the wizard. First, choose the data rule you wish to apply. There will be a BUILT_IN list of data rules, a list of DERIVED data rules and any custom Data Rule folder that have been created.

Continuing to the next step, you can rename the data rule for this particular instance of the rule being applied, and then bind the rule parameters to corresponding columns. For example, For example, you can bind a Common Format rule for emails to the EMAIL_ADDR column in your system, which you would like to check if they truly follow the email address common format.

After successfully creating your new data rule, you can go to the Data Rule tab in the Profile Results Canvas to evaluate data compliance to your new data rule. Figure 10 shows both the Data Rule tab (at the top) and the Data Rule Panel (at the bottom). The results tab shows the following information for each rule applied:

- ***Rule Name and Rule Type***, as specified during data rule creation;
- ***Origin***, this is CUSTOM or DERIVED, depending if it was manually created or derived respectively; and,
- ***% Compliant and # Defects***, this shows what percentage of the row set are compliant with the rule, and how many actual rows are not.

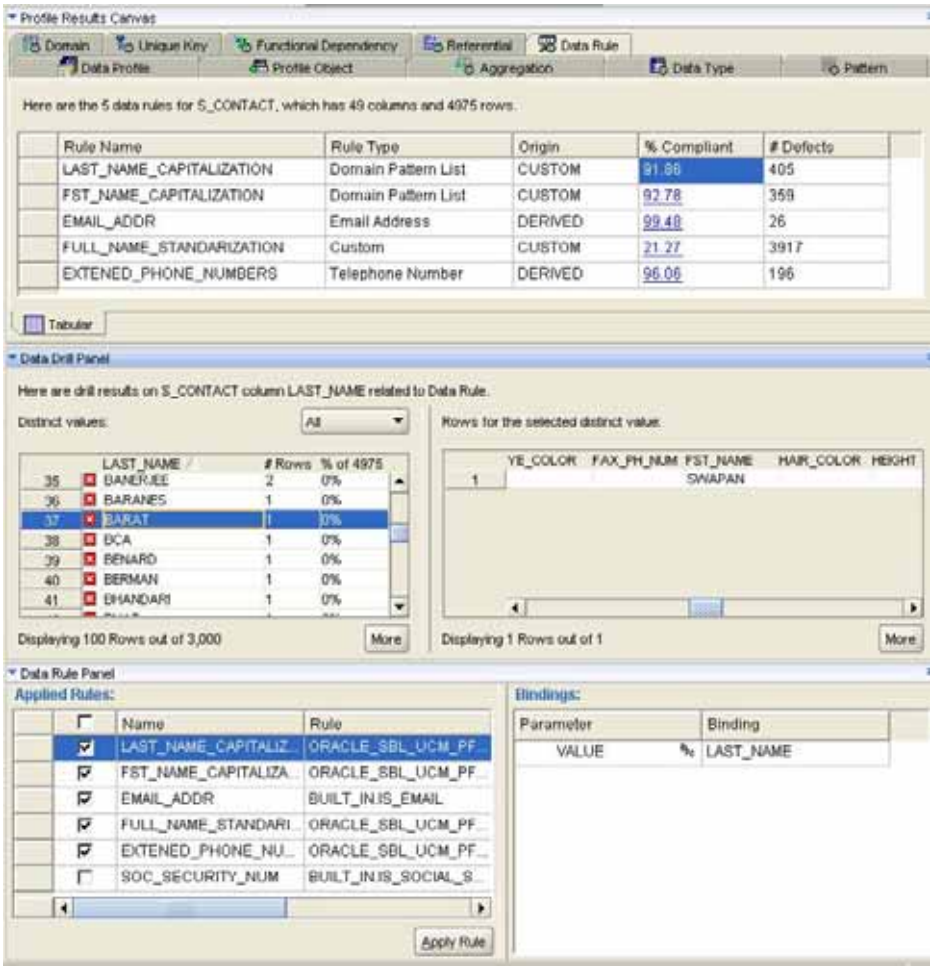


Figure 10: Examining Applied Data Rules

Like all other data profiling results, each of these results can be drilled down into. All records are shown, and they also be filtered to view only the compliant or only the non-compliant entries. Also, as the data changes, the percentage of compliance and number of defects get automatically updated to reflect the most updated state of the data.

In the Data Rule Panel, the bindings for each parameter are also shown, as well as the actual rule applied. In this panel, you can also open any applied data rule and edit the data rule definition directly.

5. Creating a Correction Mapping

After creating and applying any data rule, the next step is to create a correction. For Universal Customer Master (UCM), the tool can then rewrite these corrections back into the UCM database. For all MDM hubs, you can create this correction mapping in the DWR tool to examine how your corrected dataset would appear. Once the data is corrected, you can re-profile the data to get quantitative knowledge of how the data has improved.

5.1 Using the Correction Mapping Wizard

You can create a correction mapping for a profiled object by clicking on **Profile >> Create Correction…**, within the corresponding data profile. Alternatively, you can click on the **Create Correction** icon (^^^) in the . This will bring up the **Correction Mapping Wizard**. In the wizard, first select an existing target module, or create a new one to contain the corrected version of the object: This will by default have the same definition as the original object, except with the new data rules (implemented as check constraints or unique keys) additionally present. Then, select which data rules will be used to create the correction, as shown in Figure 11 below.

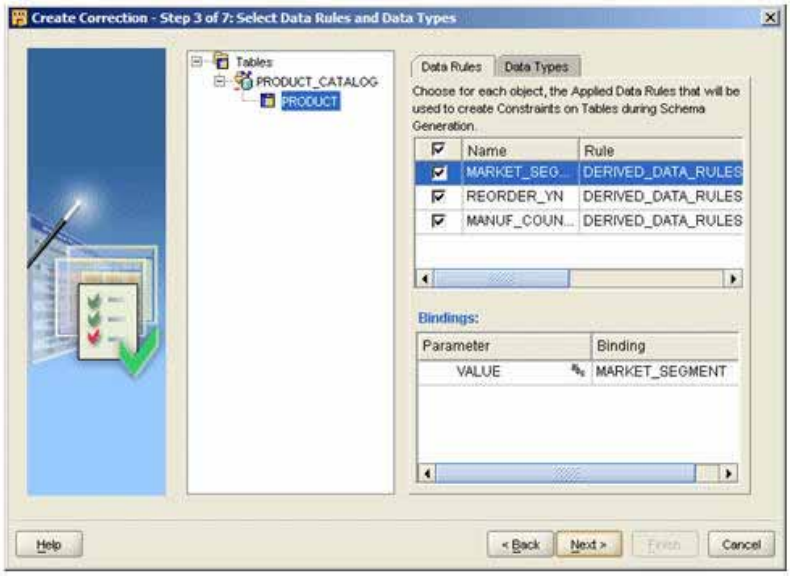


Figure 11: Choosing data rules used in correction

Next, click on the drop-down menu under *Action* for each data rule, and specify one of the following actions,

- ***Ignore:*** Do not take any action
- ***Report:*** Log, in the ERR$$$ columns in the table, the instances when the data rule would have been violated
- ***Cleanse:*** Apply a data cleansing strategy to correct data that violates the data rule

For those rules that you have chosen to cleanse the data, click on the drop-down menu under *Cleanse Strategy* and select the correction strategy to be used. These drop-down lists and the correction strategies are shown in Figure 12 below. You can either**,**

- ***Remove:*** Excludes from the corrected object those rows that fail this data rule. We will use this rule for rows that break the REORDER_YN = "Y" rule.
- ***Similarity Match:*** Uses the built-in Match-Merge functionality in Oracle Warehouse Builder to change the erroneous value to the one that is most similar to it within the column domain.
- ***Soundex Match:*** Uses the SOUNDEX function to change erroneous values to the one within the domain that is the closest phonetic match to it.
- ***Custom:*** Used where the logic to cleanse the column is more complex, involves other PL/SQL functions, uses conditional logic, or otherwise requires the creation of a custom PL/SQL function.

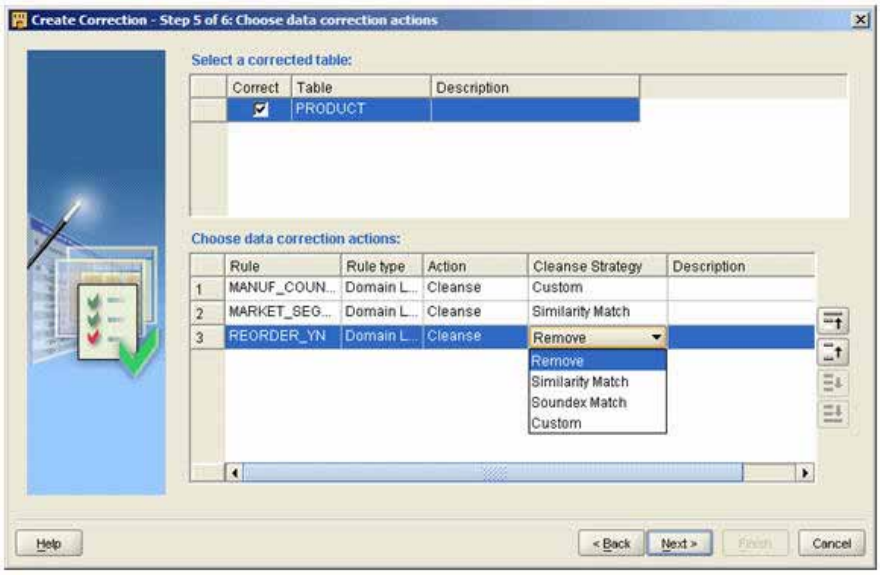


Figure 12: Choosing data correction actions

When you complete the selection of your cleansing actions and strategy, the wizard will take the specification and create a mapping within the target module to implement the corrections. A new target module will appear in the Project Explorer. This module contains the table definitions that hold the corrected data and the mapping and transformations that implement the data correction. Figure 13 shows the new module expanded, with its mappings, tables and transformations.

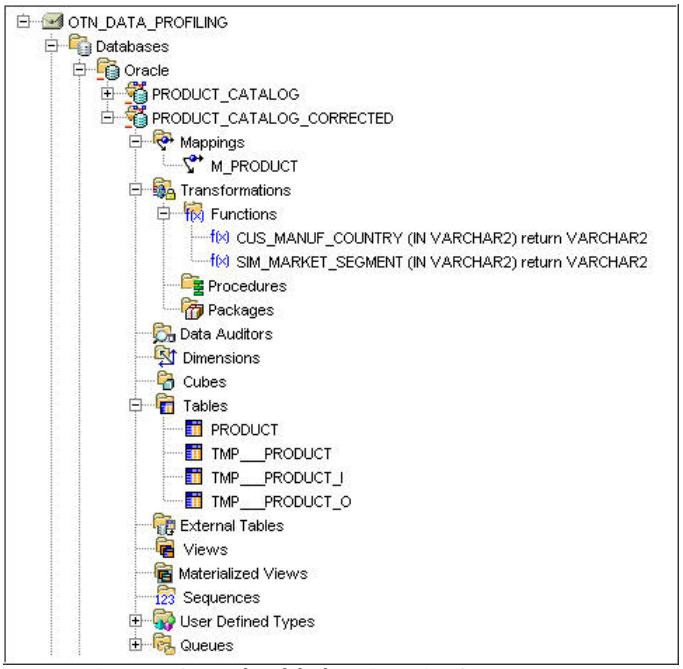


Figure 13: Corrected module shown in Design Center

Under Transformations, you will see the transformations that the correction wizard has created. If you specified any cleansing strategy to be custom, you will need to write the PL/SQL function for this custom cleansing. To add the program logic that will implement the custom cleansing action, double-click on that function and add the required logic. An example of the Code Editor and a sample function is shown below in Figure 14.

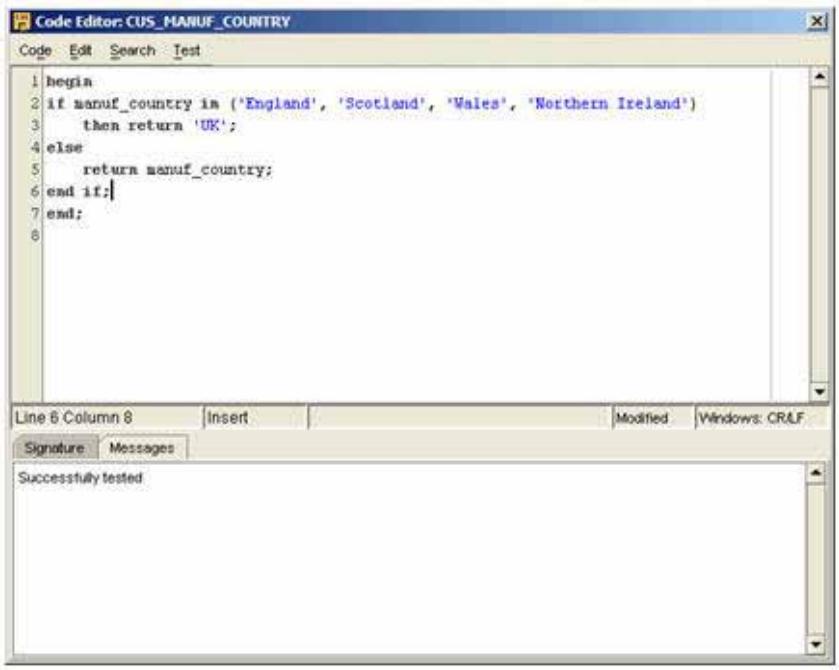


Figure 14: Code Editor for custom cleansing function, written by user

Also, if you examine the correction mapping in the **Mapping Editor** under the Mappings node of the corrected module, you can look at the automatically generated correction mapping that first reads from the original source table and then loads in into a staging copy of the table with the data rule applied to it. Those rows that pass the data rule are then copied into the corrected table. Those that fail any of the rules are then cleansed. You can then examine the contents of the pluggable mapping to see how it has been implemented. Figure 15 below shows the correction mapping, as well as the contents of the mapping that corrects the non-compliant data.

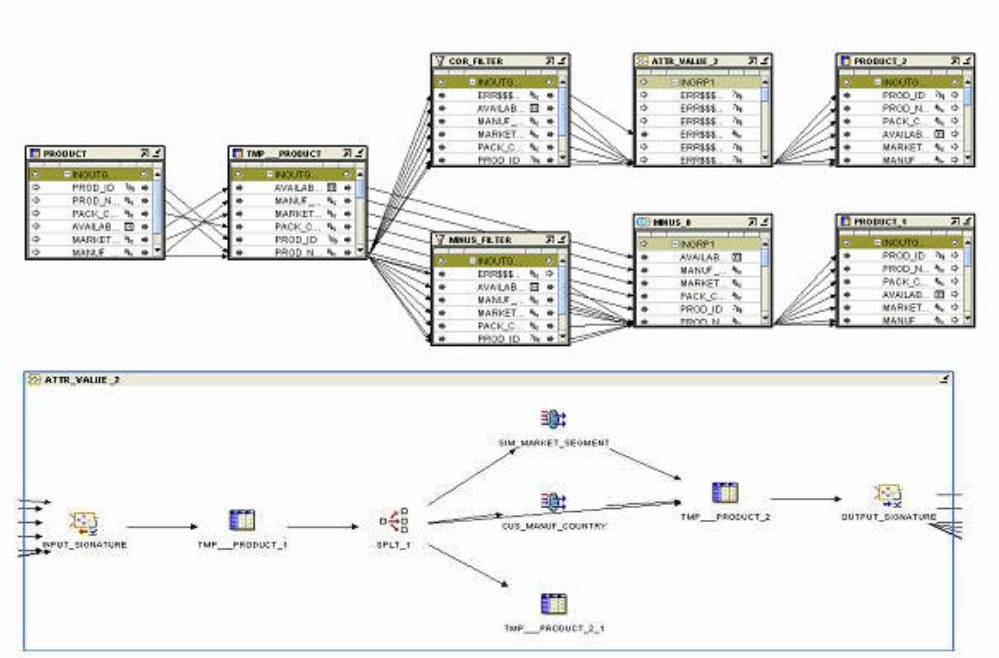


Figure 15: Automatically generated correction mapping and pluggable mapping

Finally, the new correction objects, transformations and mappings need to be deployed in the Control Center to test the correction. After deployment, the correction mapping should also be run. To do this, follow these steps:

- Open the Control Center, in the Design Center go to **Tools** >> **Control Center Manager**
- Expand the location specified for your correction, and then expand the newly created corrected module. All the objects created in the process should appear on the right-hand side in the Object Details panel. The Design Status for these objects should be New.
- For each of these objects, click on the drop-down menu for Deploy Action and select **Create**. If the correction object had previously been deployed (after a new modification of the correction mapping or custom transformations, for example) you can re-deploy the correction by selecting the **Replace** action. This will drop the existing objects and create new ones with the new definitions.
- Once there is at least one action specified, the Deploy icon should be (^^^) enabled. Click on it to deploy new items and wait for the job to be over. You can monitor the job in the **Control Center Jobs** panel below.

- Finally, check the contents of the corrected tables with the Data Viewer and verify that the results are as you expected.

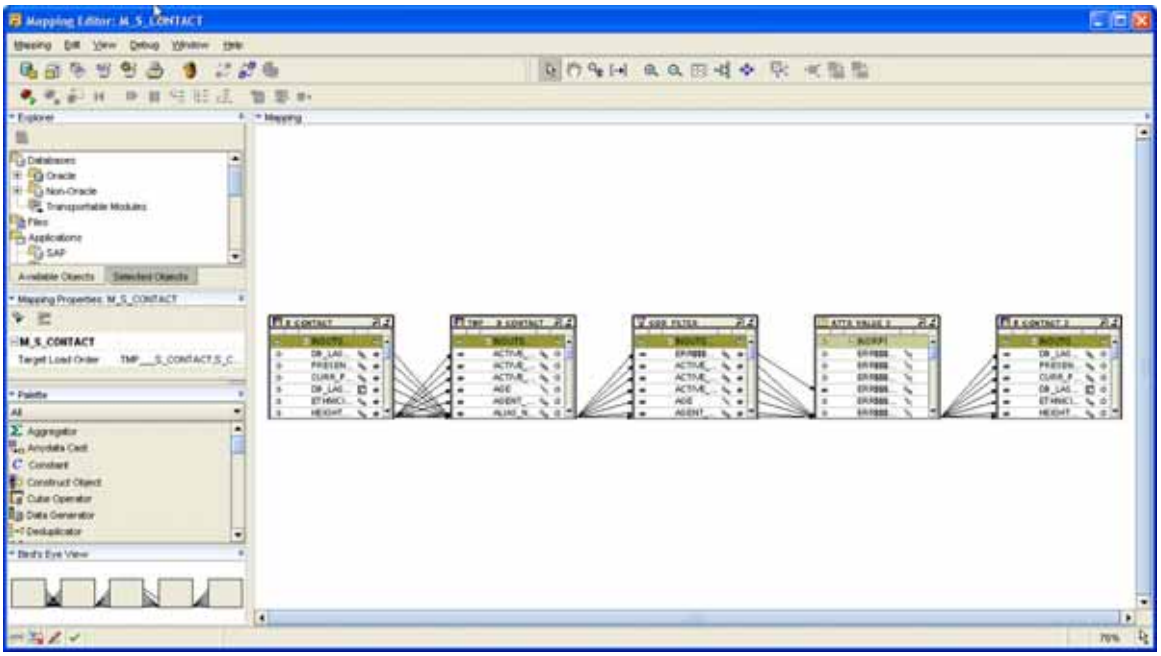
6. Writing back corrections into UCM

The correction wizard generated a map that would process the rows in the source table, and:

1. Corrects the records that don't comply with the rule(s), by applying the correction function(s)
2. Lets the rest of the records to pass through unchanged.

Thus the target table will contain all the rows in the source table. You can re-profile this table to see the new data rule compliance percentages and possibly refine the correction maps. This can be an iterative process.

After you are satisfied with the corrected data it's time to write the results back into the source system (UCM). First, in order to be more efficient, not all rows need to be written back, but only those that are to be corrected. To achieve this you can delete from the correction map the branch that passes through the compliant rows unchanged (i.e. the branch that contains the minus filter and the minus set operators), leaving only the corrected rows processing branch. After running this map, the target S_CONTACT table will contain only the corrected rows. For our S_CONTACT example, the resulting map will look like this:



Next you need to create a map that copies these corrected rows back into the source system. For UCM we use the interface tables as the target of this map. Several steps are necessary to accomplish this task:

1. Identify the interface table that corresponds to the base table you are working with. You can use the MDM application tools and documentation to achieve this (for example, in UCM you can use Siebel Tools). In our case, the table that corresponds to S_CONTACT is the interface table EIM_CONTACT.

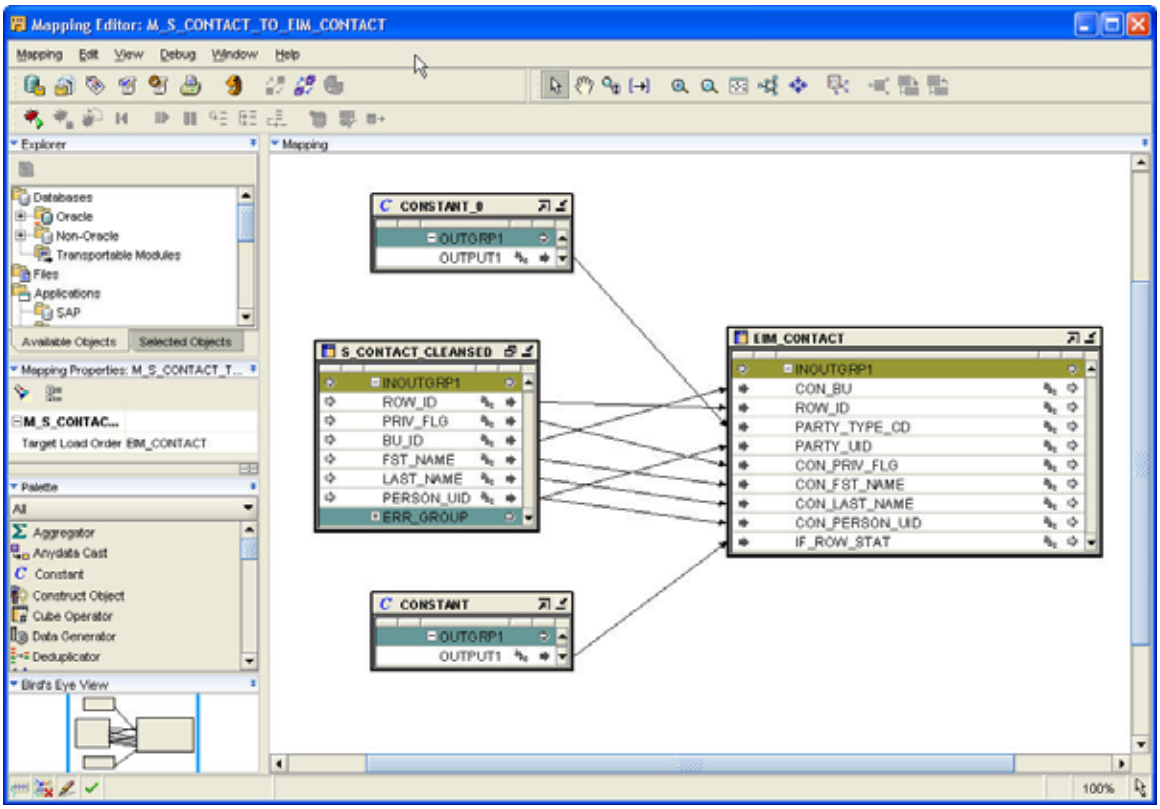
2. Map the necessary columns from the corrected table to the interface table. To be able to do that, you need to identify how columns in the base table map to the corresponding columns in the interface table. Again, you can use the MDM application tools and documentation to achieve this (in UCM you can obtain this mapping from Siebel Tools). The columns that are necessary to be mapped are:

    a. The corrected columns. In our example: FST_NAME and LAST_NAME map to EIM_CONTACT.CON_FST_NAME and EIM_CONTACT.CON_LAST_NAME, respectively.

    b. The columns contained in the user key that uniquely identify a record for update in the base table. In our example, those are BU_ID, PERSON_UID and PRIV_FLAG. They are mapped to CON_BU, CON_PERSON_UID and CON_PRIV_FLAG, respectively, in EIM_CONTACT table.

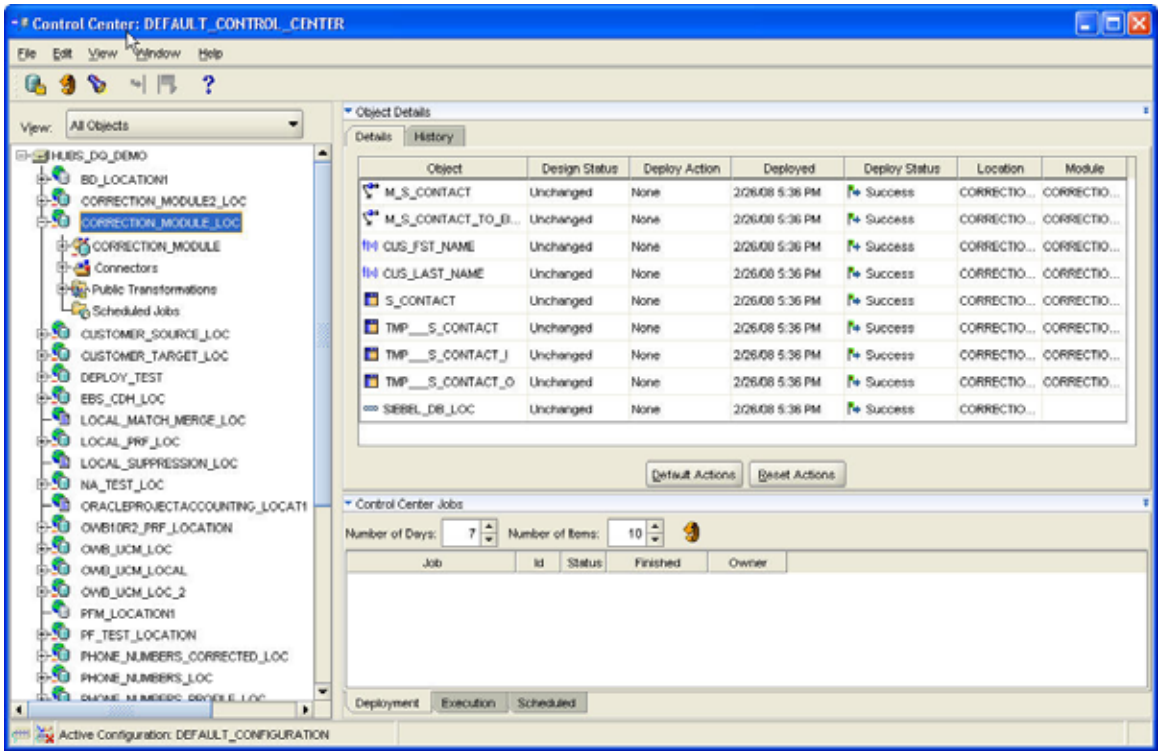
    c. Any other column that must be NOT NULL in the interface table.

The resulting map will look like this:



Note that in the map above table operator S_CONTACT_CLEANSED is bound to table S_CONTACT in our correction module and table operator EIM_CONTACT is bound to table EIM_CONTACT in our source UCM module. When deployed, OWB will generate the necessary database links to these schemas.

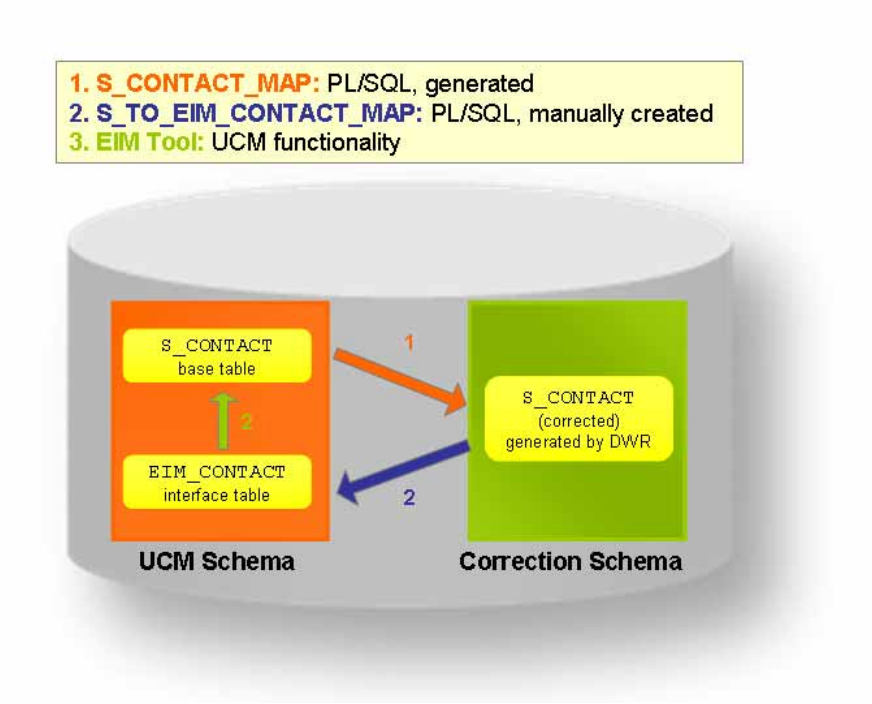
The next step is to deploy the objects to an Oracle database. For that you need a target location to be associated with the correction module. The easiest way to do deployment OWB is to go to Control Center Manager and navigate in the location tree to the correction module location. By selecting the location, the object details panel will be populated with the objects to be deployed. In our example, Control Center Manager looks like:



Next you choose to deploy all objects (Default Actions) and run the deployment.

After you deployed all objects, you first run our correction map (M_S_CONTACT in our example). That will populate table S_CONTACT in our correction database schema. You can verify the results by using OWB data viewer or by connecting using SQLPlus.

Next you run the map that copies the results into the interface table in UCM schema. In our case, M_S_CONTACT_TO_EIM_CONTACT is the map to run, which populates table EIM_CONTACT in UCM schema. Again you can verify the result using OWB data viewer or SQLPlus.



At this point you have the corrected data back in UCM system, in the interface table (EIM_CONTACT). The final step is to update the base table from the data in the interface table. For this purpose we use Siebel Enterprise Integration Manager (EIM). EIM can be run either from GUI or from command line interface. You only need to create a configuration file and specify it as a configuration parameter. In our example we can use the following configuration file (owb.ibf):

```
;
; owb.ibf -Sample Siebel EIM IFB file used for Data Watch and Repair
; for Master Data Management example
;

[Siebel Interface Manager]

; The following two lines make sense when doing an initial import or if
; no transaction logging is desired. Otherwise, read the comments below

;LOG TRANSACTIONS = FALSE
;SET BASED LOGGING = TRUE

; When doing ongoing import(s), if transaction logging is desired,
; comment out the above two lines, and uncomment the following two
```

```
; lines.

LOG TRANSACTIONS = TRUE
SET BASED LOGGING = FALSE

[Import_Contacts]
      TYPE  = IMPORT
      BATCH = 0
      TABLE = EIM_CONTACT
      ONLY BASE TABLES = S_PARTY, S_CONTACT
      ONLY BASE COLUMNS = S_CONTACT.FST_NAME, S_CONTACT.LAST_NAME,
S_CONTACT.BU_ID, S_CONTACT.PERSON_UID, S_CONTACT.PRIV_FLG
      INSERT ROWS = FALSE
      UPDATE ROWS = FALSE
      UPDATE ROWS = S_CONTACT, TRUE
      COMMIT EACH TABLE = FALSE
      COMMIT EACH PASS = FALSE
      ROLLBACK ON ERROR = TRUE
```

To run this file you need to start Siebel srvrmgr program in the command-line mode and run the following command:

run task for component eim with config=owb.ifb

For more information on Siebel EIM, please see *Siebel Enterprise Integration Manager Administration Guide*.

At this point the rows the base table (S_CONTACT) have been updated with the corrected values. You can check the results using OWB data viewer or SQLPlus.