ORACLE
INSURANCE

Oracle Health Insurance Back Office

Data Subsetting and Masking Technical Reference Guide

version 1.2

Part number: E54856_01

May 26, 2014

Contents

1 Introduction

Welcome to the data subsetting and data masking technical reference guide for the Oracle Health Insurance Back Office (OHI-BO) application.

1.1 Subsetting and Masking

This guide describes how you can create a subset of a given OHI-BO data store. The subset contains less data than the given data store but at the same time is still fully functional for the OHI-BO application (that is all data integrity rules are upheld). Using data subsetting you create smaller versions of, for instance, production-size OHI-BO databases. These smaller databases can then be provisioned to development teams, test teams, or deployed in user-acceptance environments. Through data subsetting you can significantly reduce the total amount of disk storage required to support the various, and often many, non-production OHI-BO environments in your organization.

Next to data subsetting, this guide describes how OHI-BO data stores can be masked. Due to privacy regulations, organizations are obliged to deal with privacy sensitive data in a secure manner. Production environments usually have stringent data access control and auditing mechanisms in place to ensure that only those who need to access privacy sensitive data can do so. Typically those accessing the various non-production environments are not authorized to see privacy sensitive data or the data access control and auditing mechanisms are less stringent, or even absent, in these environments. With data masking you can mask (scramble, anonymize, pseudonymize) the privacy sensitive data elements in non-production OHI-BO environments. Development and test teams that use these masked environments are therefore not able to see privacy sensitive data. The masked data store is still fully functional for the OHI-BO application (that is all data integrity rules are upheld).

The intended audience for this technical reference guide is the DBA-group that administers the various OHI-BO environments inside an organization. This technical reference guide contains four chapters.

1. *Introduction*
   The chapter you are currently reading. This chapter introduces you to the data subsetting and data masking packages of the OHI-BO application.
2. *High Level Design*
   In chapter 2 you find a high level design of the data subsetting and data masking processes as they have been designed to operate on a data store of the OHI-BO application.
3. *Detailed Design*
   In chapters 3 and 4 you find a step-by-step description of the data subsetting and data masking processes. By following these steps you can create a subset of the OHI-BO application data store and mask this data store.

Data subsetting and data masking are supported through the Quality Management submenu under the Enterprise menu of the Oracle Enterprise Manager. In this submenu you find three items to access the subsetting and masking packs:

1. Data Discovery and Modeling
2. Data Subset Definitions
3. Data Masking Definitions

See Figure 1.1 for a screenshot of this submenu.

Both the data subsetting and data masking packs depend upon an Application Data Model (ADM). ADMs are managed in the Data Discovery and Modeling menu. An ADM captures all tables of the OHI-BO data store involved in the subsetting and data masking processes. For these tables the ADM defines:

- *The referential relationships between these tables*
  Subsetting and masking processes use these to "walk through" the data model.
- *The type of table*
  Tables can either be of type transactional or config (configuration). The transactional are usually subset and the config are usually not subset.
- *Sensitive columns*
  Columns marked as sensitive in the ADM have a masking format defined for them in the masking pack.

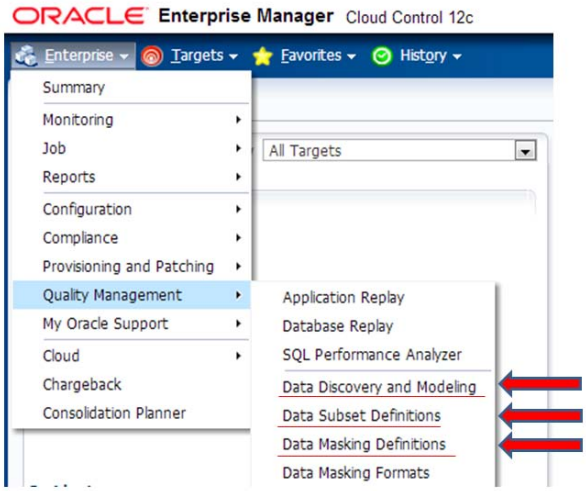

Figure 1.1: The Quality Management submenu in OEM12c.

Note: The OHI-BO masking process does not make use of the Data Masking Formats (the last entry in the submenu).

1.2 Prerequisites

The OHI-BO data subsetting and masking processes require the following components:

- *OHI-BO environment with the appropriate release*
  This environment is usually a dedicated copy of your production environment.
- *OEM Cloud Control 12c release 3, version 12.1.0.3.0*
- *Oracle Database Plug-in release 12.1.0.5.0*
  - *patch 18273816, EM DB PLUGIN BUNDLE PATCH 12.1.0.5.4 (System Patch) on top of this Plug-in*

  See section 1.3 for instructions on how to check and upgrade this component.

Note: In order to use the subsetting and masking packs (which are part of the Oracle Database Plug-in), you must have a license for these two packs.

It is recommended to use a dedicated OEM Cloud Control instance for use with subsetting and masking. A production monitoring instance of OEM Cloud Control might require a different release of the Database Plug-in which is not certified for use in combination with subsetting and masking.

1.3 Oracle Database Plug-in

You ensure that the data subsetting and data masking packs are at the required release level for OHI-BO subsetting and masking by installing the Oracle Database Plug-in release 12.1.0.5.

Select Setup → Extensibility → Plug-ins and verify the version that is installed on the Management Server. See Figure 1.2.

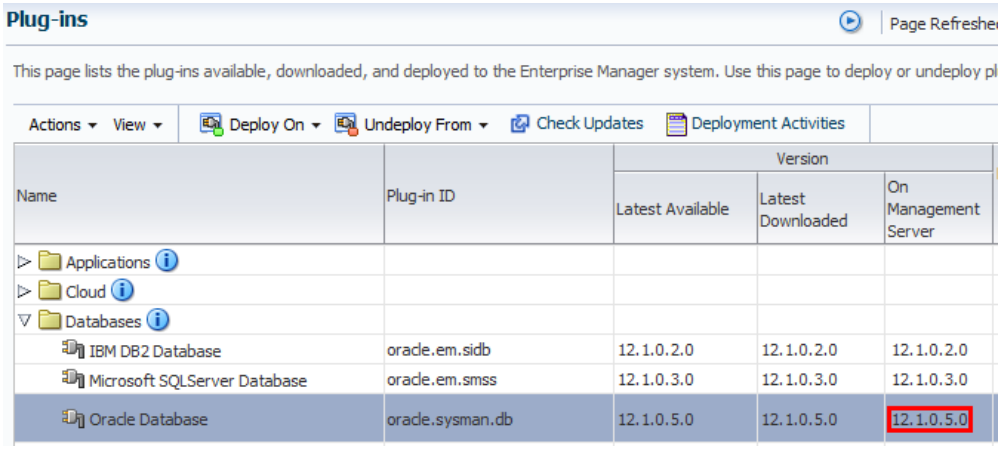

Figure 1.2: Verifying current version of database plug-in.

If the required version is not yet installed and not visible in the Latest Available column, you can use the Self Update functionality of OEM12c to download the new version. See Figure 1.3.

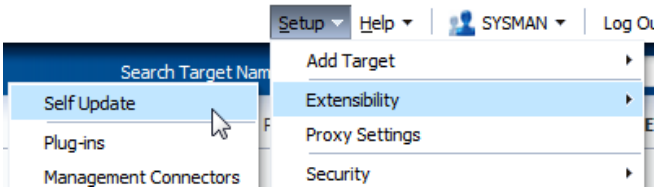

Figure 1.3: OEM12c Self Update.

When Oracle Database Plug-on 12.1.0.5.0 has been installed, bundle patch 12.1.0.5.4 needs to be installed ("*patch 18273816, EM DB PLUGIN BUNDLE PATCH 12.1.0.5.4 (System Patch)"*)

Please note that during upgrade of the Oracle Database Plug-in on the Management Server you are required to schedule downtime of the OEM12c instance.

2 High Level Design

This section contains a high level overview of the data subsetting and data masking processes as they apply to OHI-BO application.

2.1 Subsetting Process

Subsetting can be performed in two distinct modes:

1. *All data that is not part of the subset is deleted from an existing data store.*
   In this case database reorganization is required after running the subset process to reclaim the empty free space in the data files.
2. *All data that is part of the subset is exported from an existing data store.*
   The export file is imported into an empty (smaller) database after running the subset process.

The OHI-BO subsetting process runs in the second mode.

Before starting with the subset process, you first have to import two XML-files into OEM12c:

- The ADM xml file holding a description of the tables to be subset. This file is named *SDM_OHI_[release]_SUBSET_ADM.xml.* This file should be generated.
- The subsetting xml file holding the specification of the subset process. This file is named *SDM_OHI_[release]_SUBSET_DEF.xml*. This file is delivered in a Back Office release.

Through the subsetting pages in OEM12c you can start the export job. This export job creates a dump file that contains the relevant subset of the source OHI-BO data store. You then have to prepare an

empty target OHI-BO data store in a smaller database. Finally run an import job to load the subset into this smaller database. Figure 2.1 shows a high level overview of this process.

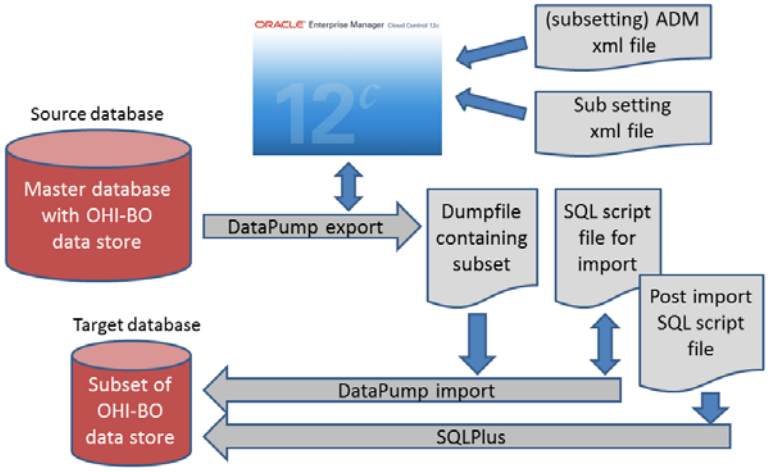

Figure 2.1: High level overview of subsetting process.

Next to the dump file, the subsetting export job also generates a SQL script file for import and a post import SQL script file (as shown in Figure 4). This import SQL script file contains the commands that import the dump file into the empty subset database. This script file can be run using SQLPlus. The post import SQL script file needs to be run after the data pump import has completed.

Note: The source database needs to be a quiescent database, as the export job runs for some time and the dump file needs to be a read-consistent snapshot of the source database. The source database is usually a dedicated RMAN copy (or Dataguard copy) from your production environment.

The target database needs to be correctly prepared before starting the import job. This is described in Chapter 3.

2.2 Data Masking Process

Data masking is performed in-place. The masking process is run on a source database, which is then masked. Before starting the masking process, you first have to import two XML-files into OEM12c:

- The ADM xml file holding the tables, their relationships and sensitive columns to be masked (named *SDM_OHI_[release]_MASK_ADM.xml*).
  Note that this is a dedicated ADM xml file for Masking; it is not the same ADM xml file as mentioned in Section 2.1 which is used for subsetting. This ADM xml file should be generated.
- The masking xml file holding a masking definition for each sensitive column (named *SDM_OHI_[release]_MASK_DEF.xml*). The definition xml is delivered in a Back Office release.

Through the masking pages in OEM12c you can start a job that generates a masking script. This masking script can be run from SQLPlus and performs the actual masking of the sensitive columns in the target database. Figure 2.2 shows a high level overview of this process.

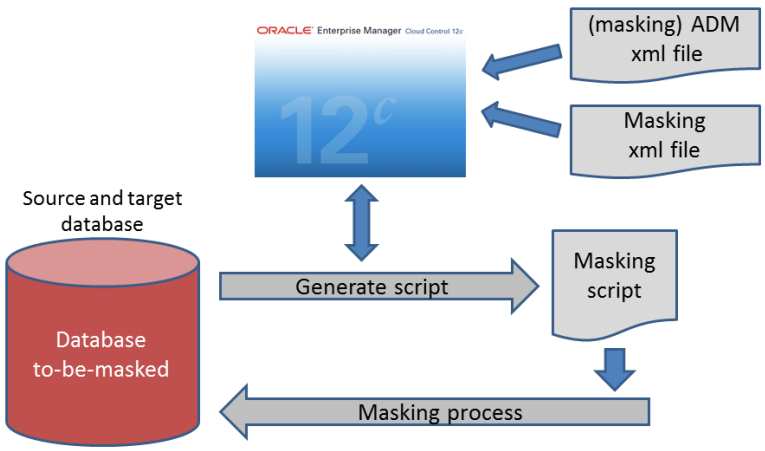

Figure 2.2: High level overview of masking process.

Chapter 4 contains a detailed description of the data masking process.

3 Detailed Process - Subsetting

As mentioned in Chapter 2.1 you need to acquire the two XML files that describe how the OHI-BO data store is to be subset, before starting the subsetting process.

- Subsetting application data model (*SDM_OHI_[release]_SUBSET_ADM.xml*)
- Subsetting definitions (*SDM_OHI_[release]_SUBSET_DEF.xml*)

As part of the subsetting process you need to generate the ADM xml, and import both this generated ADM and the subsetting definition into OEM12c.

3.1 Provision Source Database to be Subset

The subsetting process starts by designating a source database as the database from which the subset is to be created. Oracle strongly advises that you do not use a live production database for the following reasons:

- The subsetting process places considerable load on the source database.
- The data pump export sub process that performs the subset extraction, needs a single snapshot of the source database. Therefore, if you use a non-quiescent database as the source database, the data pump export produces more load as rollback segments are heavily used to recreate a read-consistent snapshot.
- Subsetting can run in a *delete* mode as mentioned in Section 2.1.You run the risk of accidentally choosing the *delete* mode instead of the *data pump export* mode as described in step 6 of the subsetting process. In which case you would submit a job that starts deleting data from your live production database.

You would usually use a dedicated RMAN backup copy of the production database as your subsetting source database. Another alternative could be to use a Data Guard environment of your production database.

Note: As part of the subsetting process, you have to load a few test data management packages into the source database. This implies that the Data Guard environment has to be converted into a Snapshot Standby database first. When the subsetting process has completed, you can convert the Data Guard environment back into a Physical Standby database. See the "Data Guard Concepts and Administration" guide for more information.

Separate database user for subsetting

We recommend to create a separate database user for running the subset export process with. The subset export creates temporary working tables during the subsetting job. These tables are created in the *default tablespace* of the user credentials you supply when submitting the subset export job.

The user can be created as follows:

```
create user <USER> identified by <PASSWORD> default tablespace <SCRATCH TABLESPACE>;
grant dba to <USER>;
grant execute on DBMS_AQADM to <user>;
```

By employing a separate database user, you can easily reclaim the disk space occupied by the working tables by dropping the tablespace.

3.2 Define Policy Selection in the Source Database

OHI-BO subsetting currently uses a policy selection concept to determine which rows should be included in the subset. A policy selection is a set of policy numbers. You have to create a policy selection that includes all policy numbers that need to be in the subset. There are two ways you can do this:

1. Use the OHI Back Office Policy Selection window to setup a policy selection.
2. Use SQLPlus to directly populate the underlying two tables of a policy selection.

Figure 3.1 shows the Policy Selection window. You have to enter a name for the policy selection (this name will be input for one of the steps later on). The description is optional. All other items can be left empty or at their default value. In the second block you enter all required policy numbers.

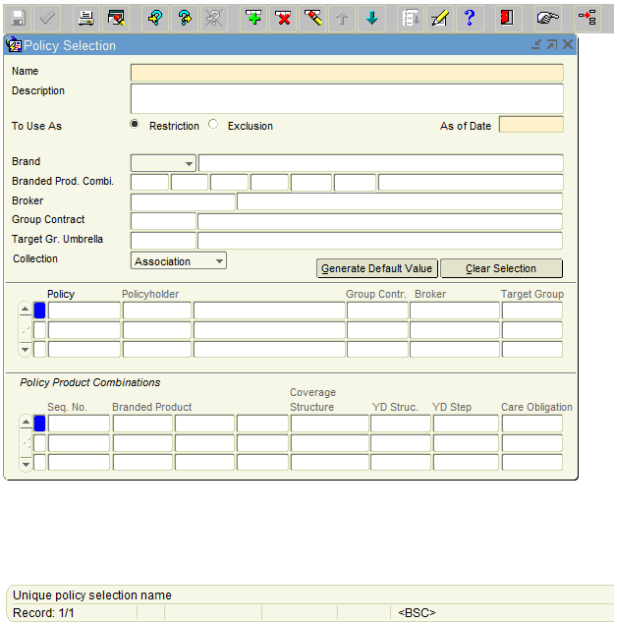

Figure 3.1: The Policy Selection window.

The two underlying tables for the Policy Selection screen are depicted in Figure 3.2. They are:

- **VER#POLICY_SELECTIONS_** (Dutch name **VER_POLIS_SELECTIES**), and
- **VER#POLICY_PER_POLICY_SELECTN_** (Dutch name **VER_POL_PER_POS**).

It might be more convenient to use an SQL tool, such as SQLPlus, to create the necessary rows manually in these tables.

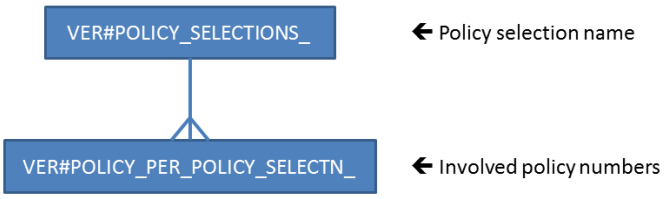

Figure 3.2: Policy selection tables.

For example, the following two insert statements set up a policy selection named SAMPLE_OF_5_PERCENT that holds a random sample of 5% of the total number of policies available in the source database.

```
insert into ver#policy_selections_ (name) values('SAMPLE_OF_5_PERCENT');


insert into ver#policy_per_policy_selectn_ (pos_id, pol_no)
select (select id from ver#policy_selections_ where name = 'SAMPLE_OF_5_PERCENT')
      , no
from ver#policies_ SAMPLE(5);


commit;
```

Note: The subsetting process uses a policy selection named OHI_SDM_POS_SUBSET internally. This name is reserved by the OHI-BO subsetting development team. You cannot use this name as it would interfere with the subsetting process.

3.3 Generate and import Subsetting Application Data Model into OEM12c

The ADM contains the data model for the OHI-BO application. This model is used by the subsetting process to calculate the subset. The ADM is specific to an OHI-BO release and must be generated for each new release. Prior to importing you should make sure that the ADM to be imported corresponds with the OHI-BO release installed for the source database.

To generate the ADM for subsetting, connect with SQL*Plus to the source database. Invoke the following command (as application owner):

```
exec sdm_adm_drv_pck.write_adm_files('DB_DIR');
```

Replace DB_DIR with the database directory of your choice.

Two files will be written to this directory:

- *SDM_OHI_[release]_MASK_ADM.xml*
- *SDM_OHI_[release]_SUBSET_ADM.xml*

Once these files have been generated, you may want to transfer these to a file system that is accessible from your Desktop, so you can import them using Enterprise Manager.
Open the *Data Discovery and Modeling* page. See Figure 3.5.

Figure 3.5: Open Data Discovery and Modeling

Make sure the ADM XML file is on your local desktop. Then select Actions → Import → File from Desktop. See Figure 3.6.

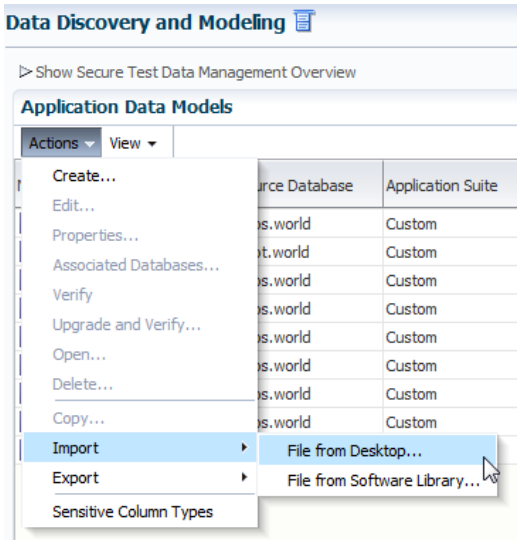

Figure 3.6: Starting the File from Desktop import.

Next enter a name for this ADM. Specifying the release number as part of the name of the ADM will help identify the correct ADM later on. Enter a description and select the subsetting source database. Then press Choose File and navigate to the ADM XML file on your local desktop. Finally press the OK button to start the import. See Figure 3.7.

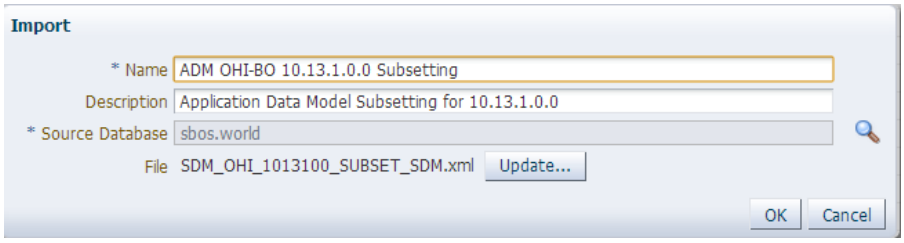

Figure 3.7: Specifying Subsetting ADM XML file to be imported.

The ADM import process may take a while.

After the import process has completed, the ADM must be verified against the source database. Select the imported ADM in the list of available ADMs. Then select Actions → Verify, and click Create Verification Job. See Figure 3.8.

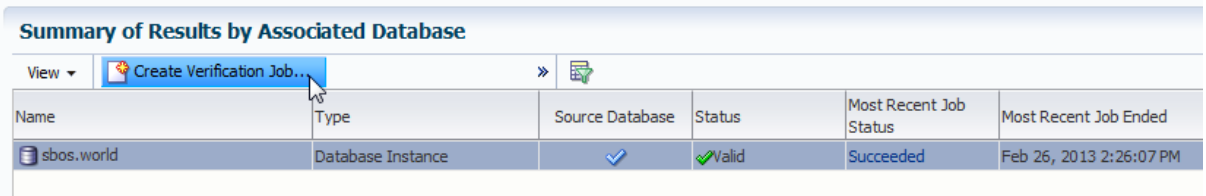

Figure 3.8: Create Verification Job.

Make sure to **uncheck** Synchronize Application Data Model before submitting, see Figure 3.9.

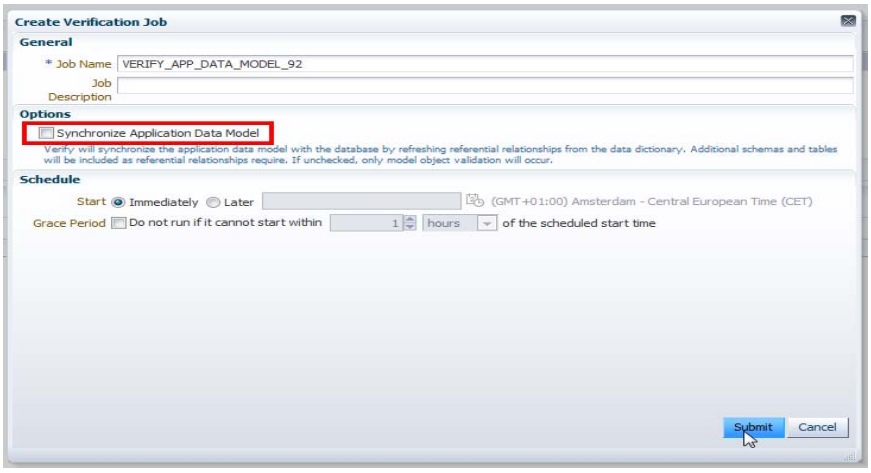

Figure 3.9: Uncheck Synchronize Application Data Model.

When the verification job has completed, you need to check the verification results. For this select the ADM from the list of available ADMs, and select Action → Verify. Now select the row with the database that you associated with the verification job. Look at the Verification Results. It should say: No verification results. See Figure 3.10.

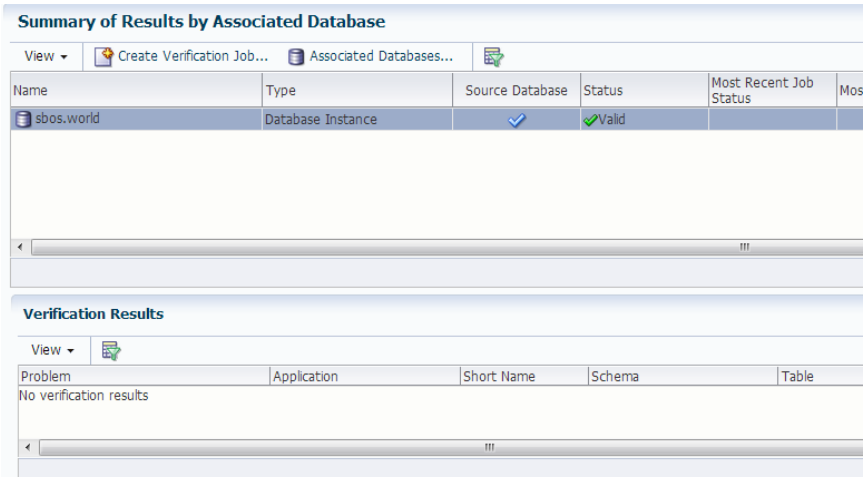

Figure 3.10: Checking verification results.

3.4 Import Subsetting Definitions into OEM12c

The subsetting definitions XML file is imported from the Data Subset Definitions page. Navigate to this page by selecting Quality Management->Database Subset Definitions as shown in Figure 3.11. Make sure you have the XML file loaded on your desktop.

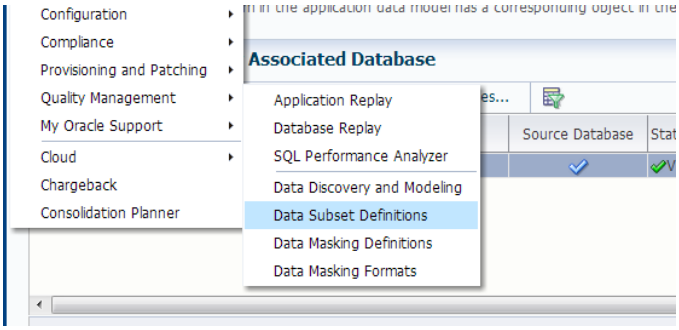

Figure 3.11: Open Data Subset Definitions

Select Actions → Import → File from Desktop in the Data Subset Definition page. See Figure 3.12.

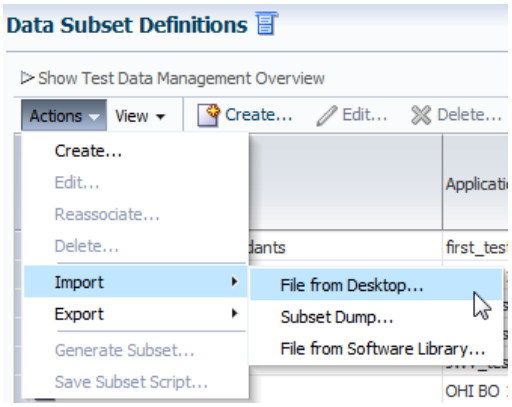

Figure 3.12: Starting the File from Desktop import.

In the Import Data Subset Definition page enter a name (preferably include release name here too) and optionally a description. Select the **corresponding ADM** and select the source database. Finally select the subsetting XML file from your local desktop and press the Continue button. See Figure 3.13.

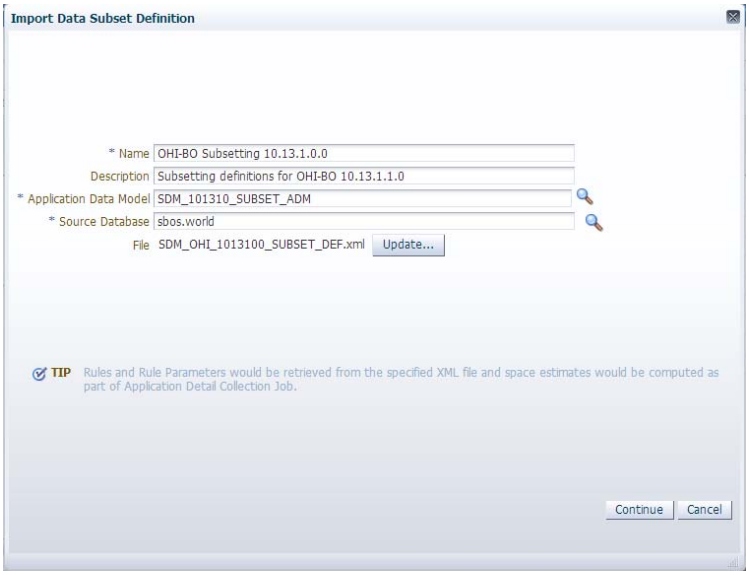

Figure 3.13: Import Data Subset Definition.

Specify the subset user credentials of the source database in the next page and press the Submit button See Figure 3.14.

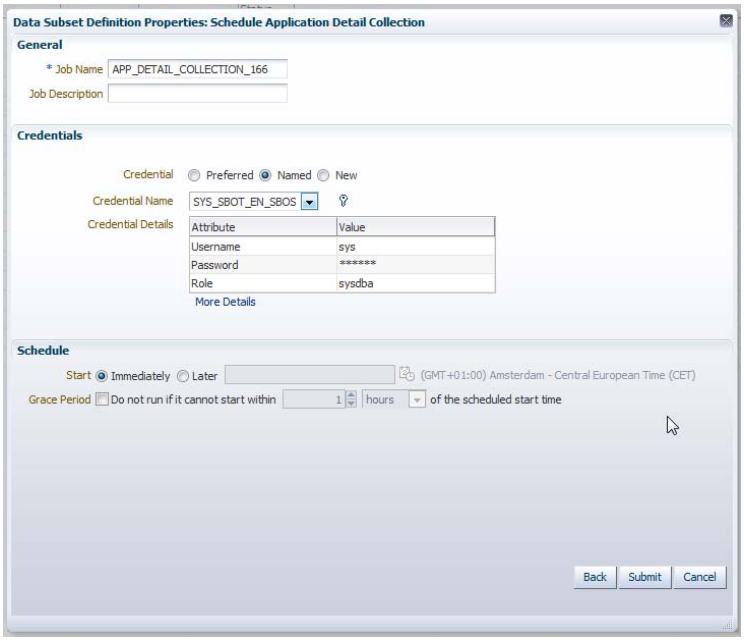

Figure 3.14: Specifying credentials for import job.

A job is created to import the subsetting definitions from the XML file. Verify the job succeeded. This job should only run for a couple of minutes.

Note: when there is a need to import the subset definition for a second time after a subset export has been run, purging the internal SDM selection tables will speed up the import job. During the import, an estimation on the size of the subset is performed. When these internal tables are filled, this estimation will take considerable amount of time.

The internal SDM tables can be purged with the following command (as application owner):

```
exec sdm_util_pck.purge_sdm_data;
```

3.5 Generate Subset Export File

Recommended database configuration during export

Before starting the actual export job, the source database should be configured to maximize performance. The following parameters are recommended (and deviate from parameters required for Back Office runtime):

- `parallel_degree_limit` = IO
  To limit the number of parallel query workers to the optimum found during **IO Calibration**. This calibration must be performed before setting this parameter to 'IO' (see below).
- `parallel_max_servers` - default value (>1), to enable parallel query during subset export
- `optimizer_mode` – default value (ALL_ROWS)
- `optimizer_index_cost_adjust` – default value (100)

With respect to memory allocation, make sure the SGA (in particular the BUFFER_CACHE value) and PGA (for in-memory sorting and hash-area) are generously sized. As an indication: for a high volume environment (5+ TB database size) a `sga_target` of 16GB and a `pga_aggregate_target` of 12GB are found to be suitable values. These values of course depend on several system properties and the optimal values should be determined when running the subset export. The Memory Advisor included in Oracle Enterprise Manager can be a guide to find the optimum.

IO Calibration

To effectuate the 'parallel_degree_limit=IO' setting, IO Calibration needs to be performed.
There are a number of ways to perform this calibration. One way is to start the IO Calibration from OEM: navigate to the 'Performance Home' of the database, click on the "I/O" tab and click the "I/O Calibration button" (see Figure 3.15). After specifying the number of disks and the maximum tolerable latency (10ms minimum), the calibration is started. The calibration process will take about 15 minutes.

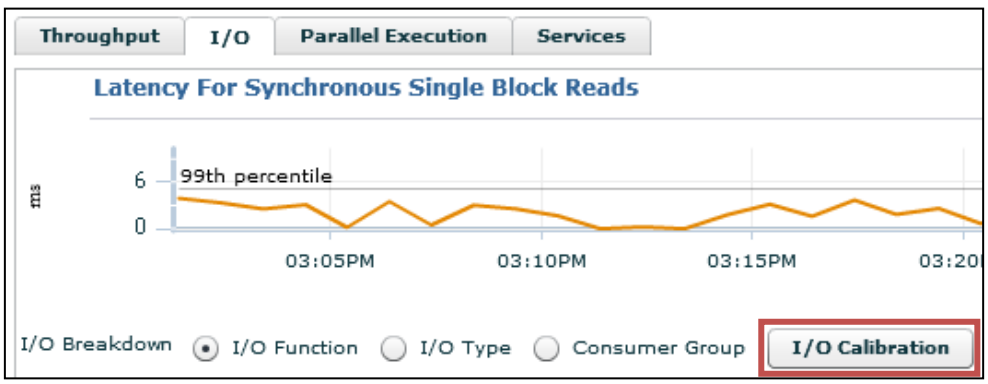

Figure 3.15: Start IO Calibration from OEM

Another possibility is to start calibration from SQLPlus. See http://docs.oracle.com/cd/E11882_01/server.112/e16638/iodesign.htm#PFGRF94384 for more information.

In MOS note **"Automatic Degree of Parallelism in 11.2.0.2 (Doc ID 1269321.1)"** a method is described to set the IO calibration data manually, without the need to run the IO Calibration. This could be useful when the used production-copy runs on hardware with different characteristics.

Temporary tablespace requirements

Make sure the temporary tablespace of the subset user is set to auto extend, and enough disk space is available for it to grow. During the export, tables can be written to temp first before being written to the export file. This can consume large amount of temp space. As a guideline: make sure the temporary tablespace can hold at least the "subset %" of the size of the largest application table. For example, 0.05 times the largest application table when a 5% subset is created.
When using multiple worker threads during export, temporary table space can be claimed by each worker. Multiply the subsetted size of the largest table by the number of workers.

$$TEMP\ size\ \approx (\#workers) * (size\ of\ largest\ application\ table) * (estimated\ subset\ fraction)$$

For example, when using 2 worker threads for a 5% subset and the largest table being 1000GB: expect 2 * 1000 * 0.05 = 100GB of required temporary tablespace.

Submitting the export job

The generation of the subset export file can now be started. From the Data Subset Definitions page select the subset definition to use for generating the subset and then select Actions → Generate Subset. See Figure 3.17.

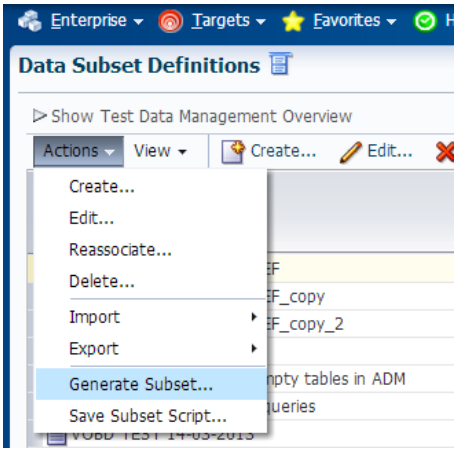

Figure 3.17: Generate Subset.

Specify the database from which the subset is to be exported (this is called target database here). Keep the 'Create Subset By' radio button default (Writing Subset Data to Export Files). Select the database and host credentials and specify the name of the policy selection to drive the subset generation. You created this policy selection in step 2 (Section 3.1.2). Then press the Continue button. See Figure 3.18.

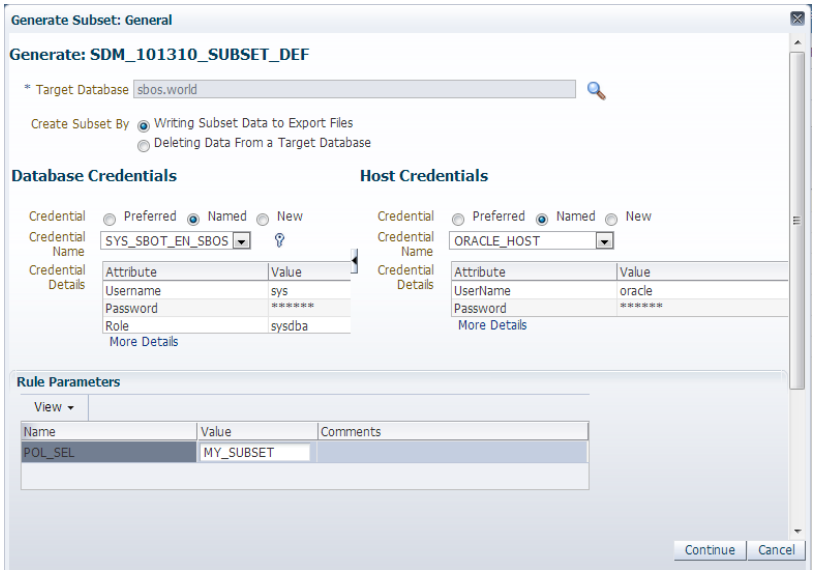

Figure 3.18: Specify general parameters for subset process.

You enter the data pump parameters in the next page and specify the directory object that data pump must use to create the dump file in. This must be an existing directory object (one that the DBA must have created on beforehand, using the CREATE DIRECTORY command) inside the source database and the application owner schema, usually OZG_OWNER, must have read/write access on this directory object.

Provide the export file name, maximum file size and the number of worker threads that data pump must use whilst querying the source database and generating the export files.

The maximum file size should not be set too low, because the total export size is limited by the of files, which is 100 files maximum, can be used and those files should be large enough to contain the subset export dump. The exported database dump should fit into 100 * {maximum file size}. Hence the default file size of 100M accommodates for a 10G maximum export size.

Consider setting the number of worker threads to 2. Depending on the number of available CPU's and IO capacity it will decrease the time needed for the export. When using two workers the table data is exported in parallel.

Please note that the Advanced Compression and Advanced Security licenses are required in order to use the compress and encrypt options. Specify the log file name and press the Continue button. See Figure 3.19.

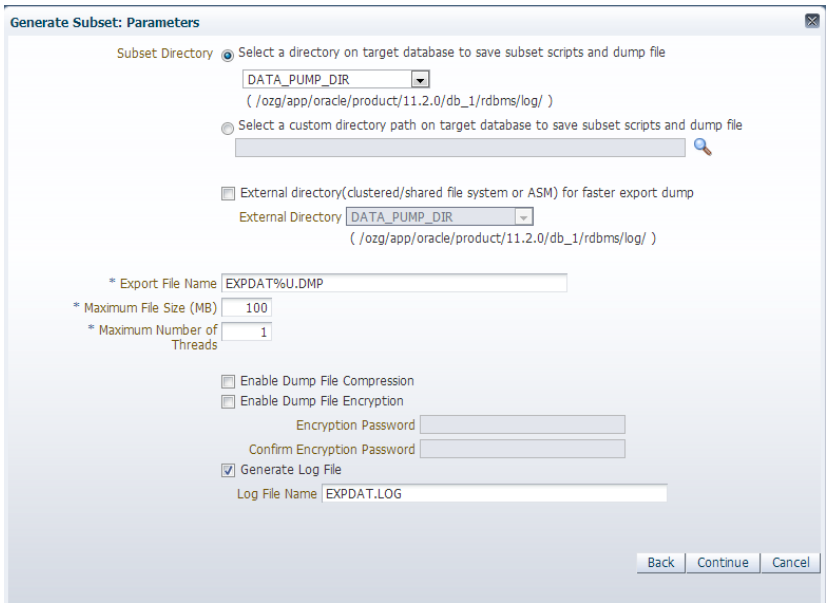

Figure 3.19: Specify data pump parameters for the subset process.

Finally schedule the job and submit it. See Figure 3.20.

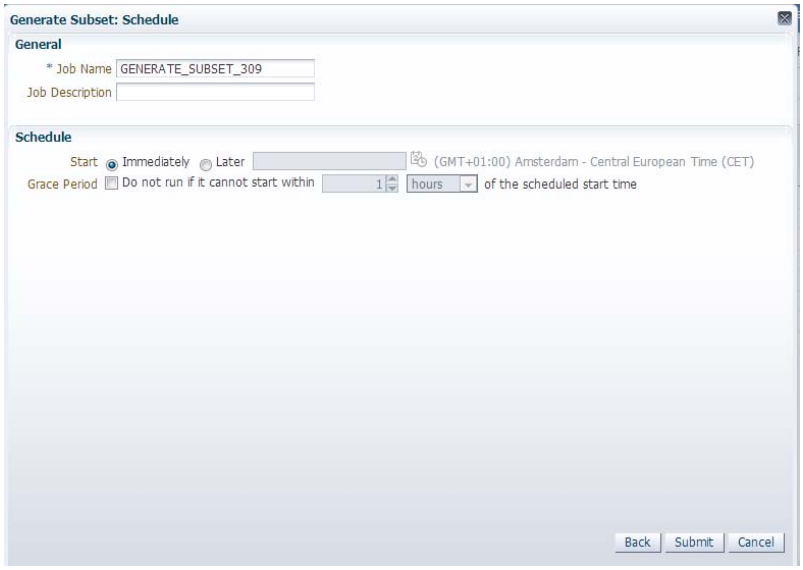

Figure 3.20: Schedule and submit the subset process job.

This job generates the following:

- *One or more data pump export files and a corresponding log file*
  These contain the subset of the data in the source database.
- *An import sql script (tdm_import.sql)*
  This script contains the commands that import the data pump export files
- *A post import sql script (subset_post_script.sql)*
  This script contains commands that must be run on the target database after the data pump import has completed.

Once the subset generation job has completed, check the data pump log file for any errors.

Note: don't forget to revert the changes made to the database parameters when using the source in a Back Office runtime environment.

3.6 Provision Target Database

The data pump export that was created in step 6, will have to be imported into a smaller target database. The export contains a user-mode export of the owner-schema of the OHI-BO application. This is usually the OZG_OWNER schema. You have to prepare an empty target database so that this user-mode export file can be successfully imported.

- *Make sure the database is loaded with the required options.*
  Those are currently the XDB and JVM options.
- *Make sure the instance is running with all the required (s)pfile settings.*
  Refer to the Oracle Health Insurance : Installation, Configuration and DBA Manual to verify these settings.
- *Make sure the database has all the necessary OHI-BO tablespaces.*
  In addition to the SYSTEM, TEMP, UNDO, USER and SYSAUX tablespaces, there are currently eighteen required application tablespaces (ozg_fact_rel_tab, ozg_fact_rel_ind, etc.). Create these as small tablespaces and ensure that they can grow (autoextend on).
- *Make sure the database has the required OHI-BO schemas and roles.*
  Running script *$OZG_BASE/sql/*OZGI001S*.sql* will create these. The import job ensures that these schemas and roles receive all the object privileges again.
- *Optionally create your own custom schemas and roles.*
  When your OHI-BO environments have additional (custom) schemas, roles or both that have object grants from the OHI-BO application owner schema, the import job successfully restores the object grants to these schemas also when they created in advance.

It is also advisable to disable the archivelog mode on the target database prior to starting the import job.

Note: The above describes only the database-side of the target environment. Of course you also need a client environment (contents of $OZG_BASE) with the forms, reports, batch scheduler and so forth. This client environment can simply be one-on-one copy of the client environment for the source.

3.7 Import Export File

The subset can now be loaded into the target database that was prepared in the previous section. You have to transfer the export dump files from the source database host to the target database host and create a dictionary object in the target database that points to the directory on the host holding the export files. Also transfer the *tdm_import.sql* and *subset_post_script.sql* scripts that were generated in step 6.

Next start up SQLPlus on the target host, connect as SYS and start the *tdm_import.sql* script. This script creates a few objects and then requests two inputs: the state of the schemas and the directory object name. Select option 2 for the first input and enter the name of the directory object that holds the export files. See Figure 3.21.

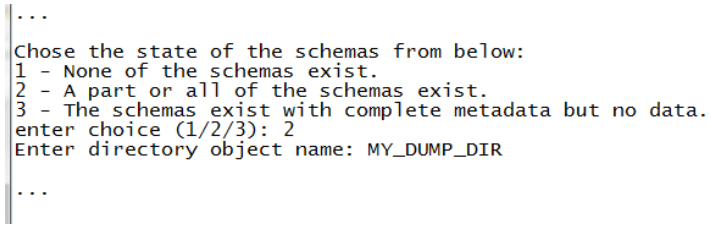

Figure 3.21: Enter data pump import parameters.

Errors are logged during the import which causes objects in the OHI-BO application schema to remain invalid due to a known issue in the way the subsetting pack interacts with data pump. These errors are resolved by the post import script.

You supplied a password for the application owner schema (OZG_OWNER) during preparation of the target database (as described in Section 3.7). The tdm_import.sql script drops this schema and has it recreated by the data pump import process. This means that you lose the password that you have

supplied: after the import, it is again set to the password that was in place in the source environment.

3.8 Run the Post Import Script File

Go into SQLPlus on the target database, connect as SYS and run the post import script: *subset_post_script.sql*. To run the post import script on the target, first set the environment correctly, i.e.:

. ozg_init.env <env>

. ozg_init.env $OZG_ORATAB_DB11204

Make sure the [SID]_install wallet entry exists and connects to the application owner schema of the target database (you may want to reset the application owner password in the target database at this stage).

After running this script all stored PL/SQL objects should be valid. You now have a subset of the OHI-BO data store. Note however that some tasks still remain. For instance, you still need to set up the application users. This target database will have the ALG#USERS_ table contents (Dutch name: ALG_FUNCTIONARISSEN) of the source database. You may want to update this table and create the corresponding Oracle application user schemas for it.

As a final verification you can run the object check script for the OHI-BO application (script *SYS9006S*). Note: This script assumes that the batch scheduler has been started. Whether the client and database are in a correct state is verified by Script *SYS9006S*.

3.9 Install and Configure Other Custom Localizations

Any adjacent schemas should be installed and configured as the last step for local customizations.

4 Detailed Process - Data Masking

Contrary to subsetting, where an empty target database is filled with a subset of data, the data masking process executes "in-place" within an existing OHI-BO data store. As mentioned in Section 2.2 you need to acquire the two XML files that describe how the OHI-BO data store is to be masked before starting the masking process.

- Masking application data model (*SDM_OHI_[release]_MASK_ADM.xml*)
- Masking definitions (*SDM_OHI_[release]_MASK_DEF.xml*)

4.1 Prepare to-be-masked Database (Target Database)

Data masking can be performed on a full-size OHI-BO data store, or on a subset OHI-BO data store. Obviously the masking process takes more time on full-size data stores. During masking, tables with sensitive columns are temporarily duplicated (this is further explained in Section 4.4). For this reason it is necessary to check that tablespaces holding the table segments either have enough free space available, or are able to grow (autoextend) during the masking process. Upon completion of data masking a database (or tablespace) reorganization of some kind would have to be performed to reclaim the then remaining free space. Oracle describes an approach in Section 4.5 to prevent having to execute such reorganization.

Before masking the target database, it is advisable to create a backup of the database. Also depending upon the size of the archivelog destination file system, it may be advisable to disable archive logging during the masking process, and re-enable it afterwards.

4.2 Generate and import Data Masking Application Data Model into OEM12c

The ADM contains the data model for the OHI-BO application. This model is used by the masking process to identify the sensitive columns and relationships between the tables. The ADM is specific to an OHI-BO release and must be generated using the source database.
Prior to importing you should make sure that the ADM to be imported corresponds with the OHI-BO release installed at the source database.

To generate the ADM for masking, connect with SQL*Plus to the source database. Invoke the following command (as application owner):

```
exec sdm_adm_drv_pck.write_adm_files('DB_DIR');
```

Replace DB_DIR with the database directory of your choice.

Two files will be written to this directory:

- *SDM_OHI_[release]_MASK_ADM.xml*
- *SDM_OHI_[release]_SUBSET_ADM.xml*

(same inserted sentence here….) Open the *Data Discovery and Modeling* page. See Figure 4.1.

Figure 4.1: Open Data Discovery and Modeling

Make sure the ADM XML file is on your local desktop. Then select Actions → Import → File from Desktop. See Figure 4.2.

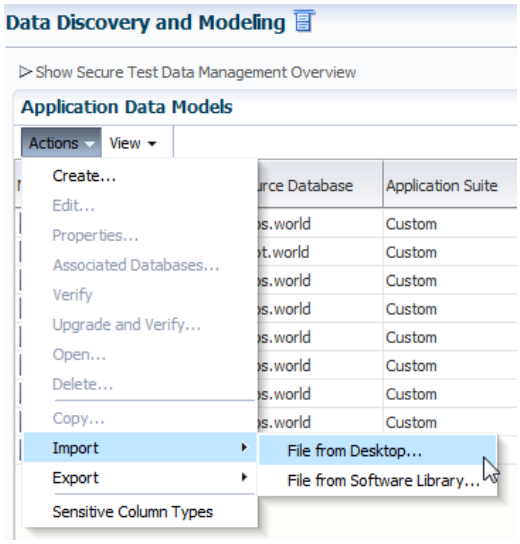

Figure 4.2: Starting the File from Desktop import.

Next enter a name for this ADM. Specifying the release number as part of the name of the ADM will help identify the correct ADM later on. Enter a description and select the masking source database. Then press Choose File and navigate to the ADM XML file on your local desktop. Finally press the OK button to start the import. See Figure 4.3.

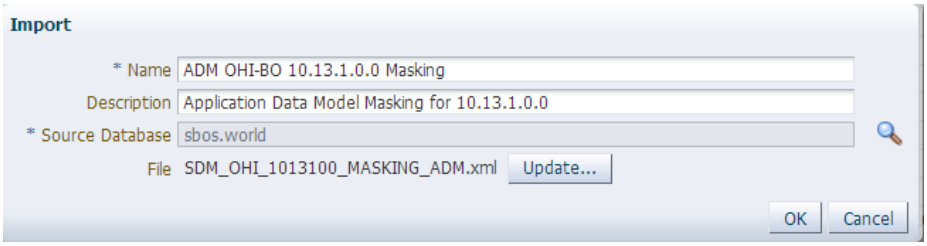

Figure 4.3: Specifying masking ADM XML file to be imported.

The ADM import process may take a while. The masking ADM assumes that the standard application schema OZG_OWNER is used in your OHI-BO environments.

After the import process has completed, the ADM must be verified against the source database. Select the imported ADM in the list of available ADMs. Then select Actions → Verify, and click Create Verification Job. See Figure 4.6.

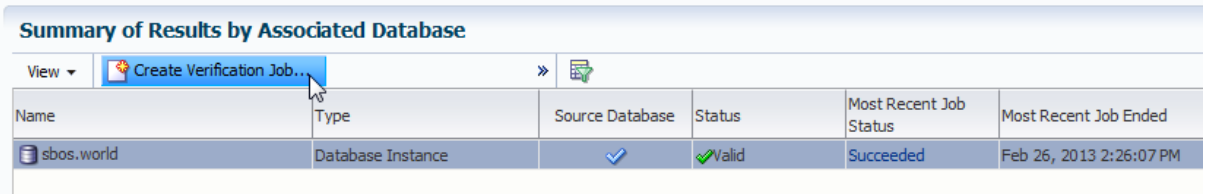

Figure 4.6: Create Verification Job.

Make sure to **uncheck** Synchronize Application Data Model before submitting. See Figure 4.7.

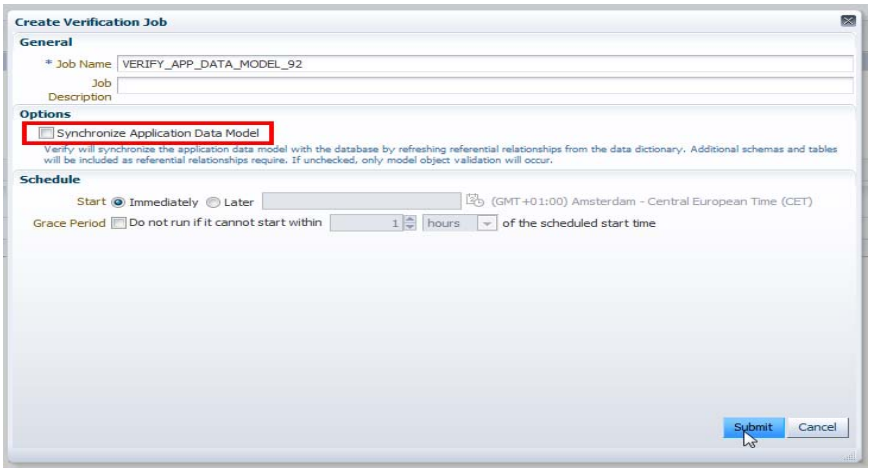

Figure 4.7: Uncheck Synchronize Application Data Model.

When the verification job has completed, you need to check the verification results. For this select the ADM from the list of available ADMs, and select Action → Verify. Now select the row with the database that you associated with the verification job. Look at the Verification Results. It should say: No verification results. See Figure 4.8.

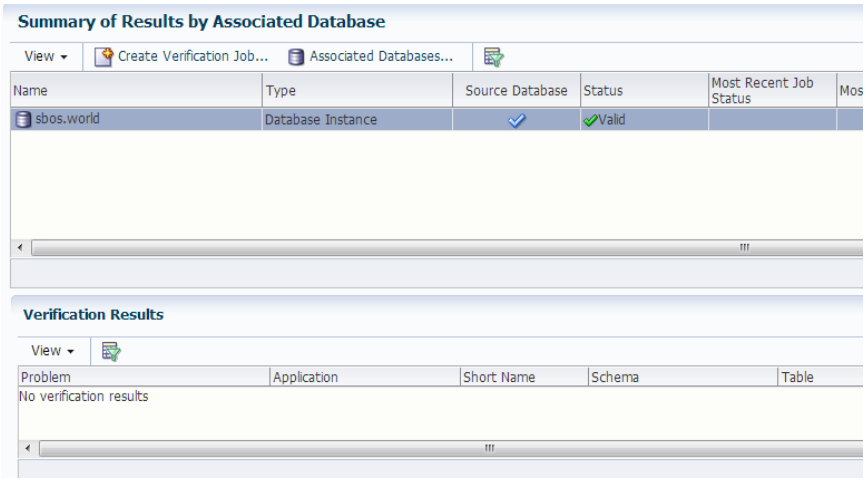

Figure 4.8: Checking verification results.

4.3 Import Data Masking Definitions into OEM12c

The masking definitions XML file is imported from the Data Masking Definitions page. Press the Import button at the top right (Figure 4.9). Make sure you have the XML file loaded on your desktop.

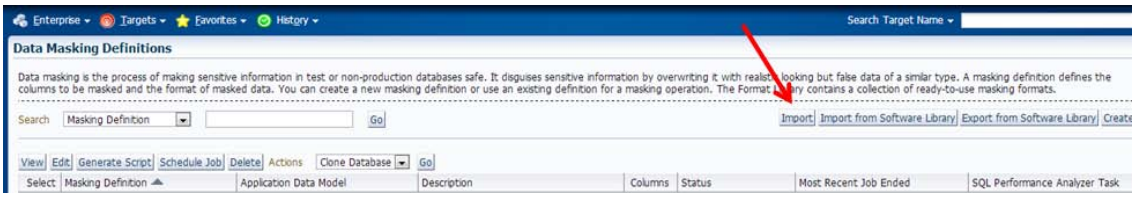

Figure 4.9: Starting masking definitions import.

Adapting the definition for a specific application schema name

When the name of the application schema is other than 'OZG_OWNER', the subset definition needs to be modified before being imported.
All references to OZG_OWNER in the SDM_.._MASK_DEF.xml must be replaced with the actual name of the application schema.

This renaming can be performed using the Linux command 'sed', for example for alternative schema 'OZADMIN':

```
$> sed -i.bak -e 's/OZG_OWNER/OZADMIN/g' <masking_definition.xml>
```

It will create a backup of the original.

Import the (modified) definition

In the Import Masking Definitions page enter a name (preferably include release name here too) and optionally a description. Select the **corresponding masking ADM** and select the source database (which is also the target database). Finally select the masking XML file from your local desktop and press the Continue button. See Figure 4.10.

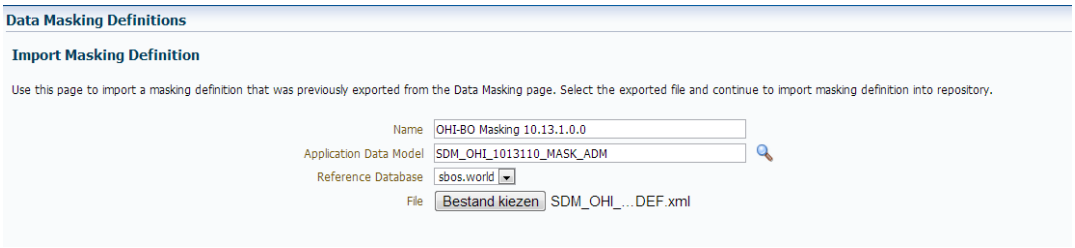

Figure 4.10: Import Masking Definition.

The masking definitions xml file will now be imported and should be available on the Data Masking Definitions page.

4.4 Generate Masking Script

Once the masking definitions have been imported, the next step is to generate a masking script from these definitions. On the Data Masking Definitions page, select the masking definition and press the Generate Script button. See Figure 4.11.

Important: it is necessary to generate this script against the to-be-masked database. Running the script against a different target might result in errors.

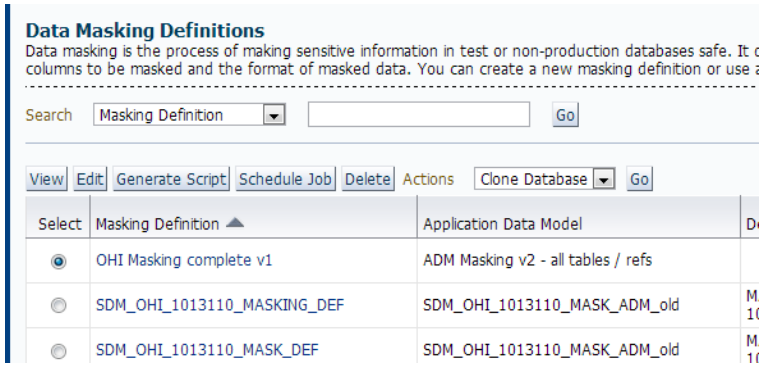

Figure 4.11: Generate Script (Masking).

The Schedule Script Generation Job page displays. Specify the SYS credentials and submit the job. This job runs for a considerable amount of time during which all tables with sensitive columns are inspected. The end result of this job is a SQL script file which holds the actual commands to perform the data masking later in Section 4.6.

As mentioned earlier in Section 4.1, tables with sensitive columns are temporarily duplicated during the actual masking. The way the masking script performs is as follows:

- Tables with sensitive columns are renamed. Indexes and triggers are removed from the renamed tables.
- For each renamed table a "create table … as select …" (CTAS) is performed to recreate the table under the original name. During this CTAS the actual data masking is performed.
- Indexes and triggers are restored on the newly created tables. Also object-privileges for these tables are restored.
- The renamed original tables are dropped.
- Invalid objects are recompiled.

The CTAS statements are performed with the space definition clauses that were in place on the original table during the generation of the masking script. This means that there are two segments for these tables in the tablespaces allocated to these tables during masking. When data masking is performed on a full-size OHI-BO data store this could substantially increase the tablespaces holding

tables with sensitive columns. This is the reason that why in Section 4.1 it was mentioned that during masking these tablespaces should either have enough free space available or be able to extend as required.

Take a close look at the output of the 'Generate Masking Script' Job. The Job will check whether enough free space is available in all tablespaces, it will report issues in the output log.

For example, it may report:

```
RESOURCE WARNING
TABLESPACE OZG_FACT_FIN_TAB
Insufficient free space in Tablespace OZG_FACT_FIN_TAB even with
automatic extension.

Some operations involving Tablespace OZG_FACT_FIN_TAB may fail.

Starting Freespace: 25610MB.

Ending Freespace (assuming increase in size): 9536MB.

Lowest Freespace: -18960MB.

Increase size of Tablespace OZG_FACT_FIN_TAB to at least 163418MB.
```

Make sure to add additional data files when reported, before running the masking script.

4.5 Move Large Tables with Sensitive Columns (Optional)

A way to prevent the necessary tablespace reorganizations to reclaim the free space after running data masking is as follows:

- *After* the masking script has been generated, inspect this script and locate all tables with sensitive columns. A way to do this is to use the grep utility:

  `grep "CREATE TABLE \"OZG_OWNER\""  [generated-masking-sql-script]`

  This assumes that OZG_OWNER is the application owner schema.
- Temporarily create a tablespace into which these tables can be moved. Also assign a tablespace quota to the application owner on this new tablespace.

- Using the output of the grep command generate "alter table … move … tablespace" commands.
- Run these commands *prior* to starting the masking script.

The CTAS statements creates new table segments in the corresponding table tablespaces which should now have enough free space, due to the fact that the original tables have been moved out of them during data masking. After completion of the masking script the temporary created tablespace should be empty again and can be dropped thus reclaiming the extra space that was required during data masking.

4.6 Run Masking Script on Target Database

Recommended database configuration during masking

Before running the masking script on the target, it should be configured to maximize performance. The following parameters are recommended (and deviate from parameters required for Back Office runtime):

- `parallel_degree_limit` = IO (See chapter 3.5 for more details about this parameter and the importance of IO calibration)
- `parallel_max_servers` - default value (>1), to enable parallel query during subset export
- `optimizer_mode` – default value (ALL_ROWS)
- `optimizer_index_cost_adjust` – default value (100)

As noted in the subsetting chapter, the sizing of both PGA and SGA is important to achieve a good performance.

The data masking script that was generated in Section 4.4, can be downloaded from OEM12c by selecting the masking definition in the Data Masking Definitions page, and by selecting Save Script. See Figure 4.12.

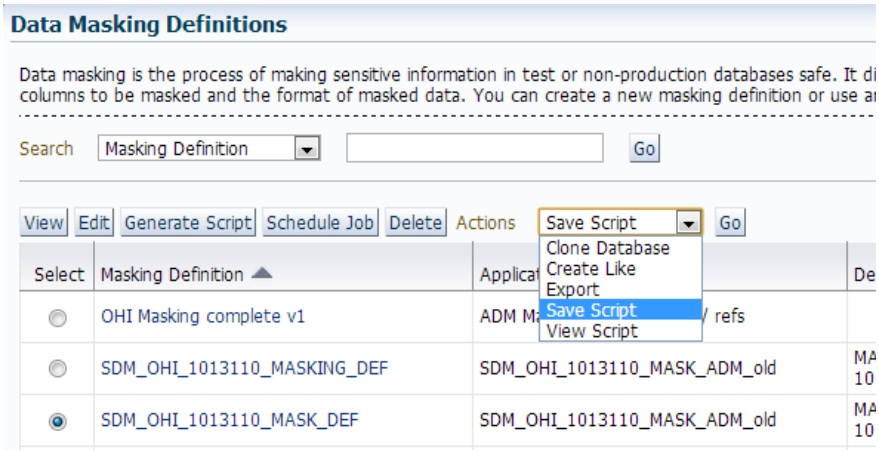

Figure 4.12: Save masking script.

Next, press the Go button. The masking script downloads to your desktop. Transfer the masking script to the target database server and start it from SQLPlus as SYS.

After that the masking proceeds. Depending on the size of the data store that is to be masked, this can take a considerable amount of time.

The masking script spools a log file in the current directory. This script runs in the "whenever sqlerror exit" mode. Upon completion of the masking script execution, inspect the log file and check for unexpected "ORA-" errors. Depending on the situation and the stage at which the error occurred, you might be able to fix the issue and rerun the script. Alternatively you have to restore the target database first and then rerun the masking script.

If you had moved the tables with sensitive columns to a temporarily created tablespace (as described in Section 4.5) then you can drop that tablespace now.

Note: revert the changes made to the database parameters for use in a Back Office runtime environment.