Loading and Extracting HFM 11.1.2.4 data with ODI Knowledge Modules

Introduction

Oracle Hyperion Financial Management (HFM) is an Enterprise Performance Management (EPM) tool that provides financial consolidation and reporting that enable users to rapidly consolidate and report financial results, meet global regulatory requirements, reduce the cost of compliance, and deliver confidence in the numbers.

Like other EPM tools, HFM relies on dimensions (metadata) and fact measures (data) that are loaded into the tool and used to create reports. Prior to its latest version (11.1.2.4), HFM could use Oracle Data Integrator (ODI) to load such metadata and data into the tool, which led large companies to build large environments around HFM and ODI integrations.

However, in HFM's latest version, Oracle decided to remove its support for ODI, meaning that all HFM integrations would have to move from ODI to manual iteration with HFM, another integration tool would have to be used, or custom code would have to be created using the new Java HFM API. For new HFM implementations, the impact of Oracle's decision could perhaps be managed, but for existing integration processes it would have a great impact, since none of the available options are smoothly implemented within large existing environments.

This article focuses on how we can achieve data load/extract from/to HFM using ODI Knowledge Modules (KMs) built using the new Java HFM API. With this kind of ODI KMs, companies can continue with legacy ODI/HFM integrations without the need to drastically change integration processes or add a new tool into their environment. New HFM implementations could also benefit from these ODI KMs, since they allow any kind of complex ETL logic on metadata and data before easily loading it to HFM.

Since this article is the second of a two series that talks about HFM 11.1.2.4 integration with ODI (the first article may be found at the OTN webpage, which link is in the Appendix Section) we will skip the details around HFM integration's current state, HFM Java API architecture requirements and ODI agent configuration. Instead we will focus on how the ODI IKM for data load and the ODI procedure for data extract were created and how to use them.

ODI Data Integration Knowledge Module for HFM 11.1.2.4

This section will describe the concepts used to create the ODI Data IKM for HFM11.1.2.4. Although the majority of ODI/HFM developers will want only to import the ODI IKM (links to its download can be found in the APPENDIX section) and use it in their projects, this article will describe the main steps and decisions that were made to create this IKM. Understanding how it works behind the scenes will allow you to make changes to its code if necessary.

Similarly to the Metadata IKM shown in the first article, the Data IKM will also try to automate the manual steps that are used to load data to HFM. In resume, ODI will create a data file based on the source tables, send this data file to HFM and optionally ODI may send a consolidation task to HFM based on a consolidation POV.

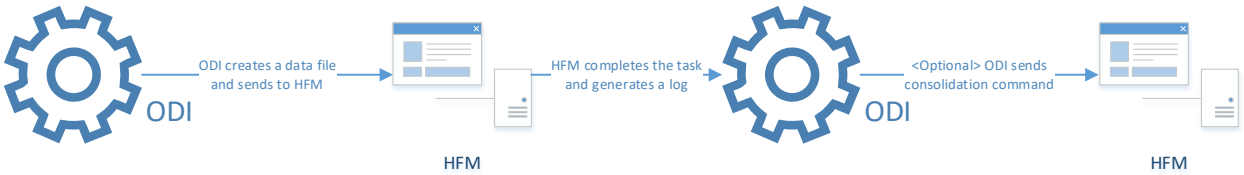

Figure 1- ODI data load to HFM

ODI Data IKM is fairly simple and contains only four steps. The first one (Generate File) is responsible to generate a valid HFM data file based on the source tables mappings. This gives the user the ability to create any kind of ETL process before loading that data to HFM, as is expected from any ODI interface.

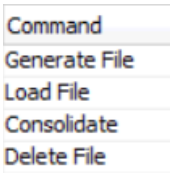

Figure 2- ODI IKM Data steps

The data file that is created by ODI is very simple and straight forward as it will only contain "!DATA" as a header and below that the POV where this data will be loaded into HFM. This file will be created temporarily on the location set by "DAT_FILE_LOCATION" IKM option and may be set to be deleted after the load is completed depending on the "DELETE_LOAD_FILE" option.

HFMData.dat

```
1 !DATA
2 ACTUAL_LOCAL;2016;Mar;Periodic;1575;<Entity Currency>;21117;1006;SUPP_Panama;Comm_Bus_Org_CH;[None];LOAD;1000
3 ACTUAL_LOCAL;2016;Mar;Periodic;1575;<Entity Currency>;21112;1375;SUPP_Panama;Comm_Bus_Org_CH;[None];LOAD;1000
4 ACTUAL_LOCAL;2016;Mar;Periodic;1030;<Entity Currency>;PSADJ;[ICP None];SUPP_Panama;Comm_Bus_Org_CH;[None];LOAD;1000
5 ACTUAL_LOCAL;2016;Mar;Periodic;1575;<Entity Currency>;87503;[ICP None];SUPP_Panama;Comm_Bus_Org_CH;[None];LOAD;1000
6 ACTUAL_LOCAL;2016;Mar;Periodic;1030;<Entity Currency>;GSADJ;[ICP None];SUPP_Panama;Comm_Bus_Org_CH;[None];LOAD;1000
7 ACTUAL_LOCAL;2016;Mar;Periodic;1525;<Entity Currency>;PSADJ;[ICP None];SUPP_Panama;Comm_Bus_Org_CH;[None];LOAD;1000
8 ACTUAL_LOCAL;2016;Mar;Periodic;1575;<Entity Currency>;21112;3750;SUPP_Panama;Comm_Bus_Org_CH;[None];LOAD;1000
9 ACTUAL_LOCAL;2016;Mar;Periodic;1525;<Entity Currency>;GSADJ;[ICP None];SUPP_Panama;Comm_Bus_Org_CH;[None];LOAD;1000
```

Figure 3- Example of HFM data file

The second step (Load File) is a little bit more complex and it is the place where the new HFM Java API is called to login into HFM application, set the load options based on what the user developed in the ODI interface, load the pre-generated file into HFM and finally getting any log that this data load may have generated (including "bad data" members).

The third step (Consolidate) is optional as you may select if you wish to run a consolidation task after the data load or not based on "CONSOLIDATE_AFTER_LOAD" option. If the user selects to run data consolidation, ODI will get "CONSOLIDATE_POV" option and send it as a parameter to HFM consolidation process. The pattern used for the consolidation POV is the same one as used in prior versions of HFM KMs, as for example "S#ACTUAL_USD.Y#2016.P#Mar.E#3450". This example would consolidate the data based on ACTUAL_USD Scenario, 2016 Year, Mar Period and 3450 Entity members.

The forth step (Delete File) is also optional, since you may select to delete or not the data file created for this data load process. This option is very useful for debugging processes where you may check exactly what was sent to HFM but probably the developers will set it to "True" on production environments, meaning that ODI will delete the file automatically after the data load process.

ODI IKM Data usage

After HFM data store information is correctly reversed inside ODI with the new ODI RKM (showed in the first article) and the new "IKM SQL to Hyperion Financial Management Data 11.1.2.4 Java API" is imported, it's time to build the data load interface. You'll notice that it has many more options than the old IKM for HFM Data. This is because the new IKM basically works in a two-step process: first it creates a data file in a folder location, then it gets this data file and loads it to HFM.

The creation of this data file has some setups that need to be defined in the ODI interface object, for which we use the IKM options. The table below lists all the available options and a description of how to use them.

Option	Description	E.g. Values
CONSOLIDATE_ONLY	Flag to indicate if this is Consolidate only process.	True False
IMPORT_MODE	Specifies one of the load options.	DATALOAD_DUPLICATE_ HANDLING.DATALOAD_MERGE
ACCUMULATE_WITHIN_FILE	Flag to indicate whether to accumulate values in the load data.	True False
CONSOLIDATE_AFTER_LOAD	Flag to indicate whether consolidate should be performed after the data load.	True False
CONSOLIDATE_POV	Specifies the parameters for consolidate operation.	S#ACTUAL_USD.Y#2016.P#Mar.E#3450
CONSOLIDATE_TYPE	Specifies the parameters for consolidate Type.	WEBOMDATAGRIDTASKMASKENUM. WEBOM_DATAGRID_TASK_CONSOLIDATE
DAT_FILE_DELIMITER	Defines which delimiter will be used in the .DAT file creation.	; , | *
DAT_FILE_LOCATION	Defines the folder location where the .DAT file will be created and stored before loading to HFM.	C:/Temp/HFM_Load.dat
DELETE_LOAD_FILE	Flag to indicate if .DAT file should be deleted in the end of the process or not.	True False
LOG_LOCATION	Log File Location and name.	C:/Temp/HFM_Load.log

Table 1- IKM Data options

Since all options are self-explanatory and have further detailed explanation on the ODI IKM “help” field of each of them, we may start to create the ODI interface itself. In this example we will load data from an Oracle table. Its creation is the same as any other interface: you may do joins, filters, ETL modifications to the mapped columns, and so on.

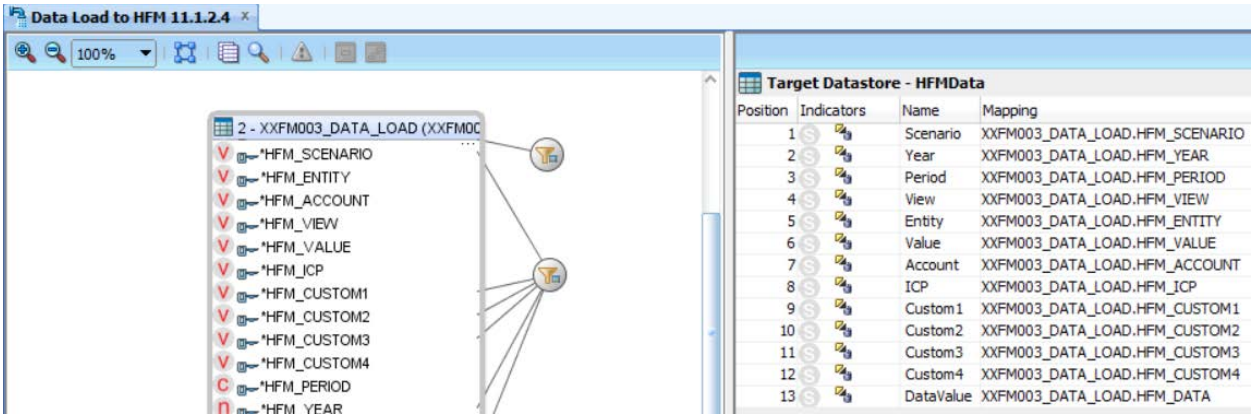

Figure 4- ODI interface example

Note that, since our target is an HFM application, we need to set our Staging Area to a valid Oracle Logical Schema (probably the same used in our source data store) in order to do any ETL to the data.

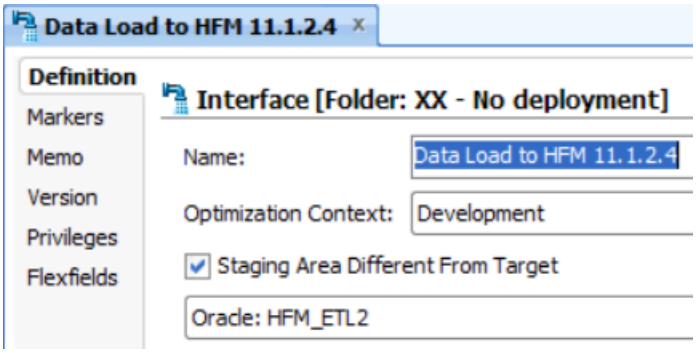

Figure 5- Set the correct Staging Area

In the Flow tab, select the options needed for your integrations and execute the interface. If all goes well, you will see green steps in the ODI Operator, indicating that the data was successfully loaded to the HFM application. You may also check the data file created on the folder that you added in the IKM option and the log that HFM generated for this process.

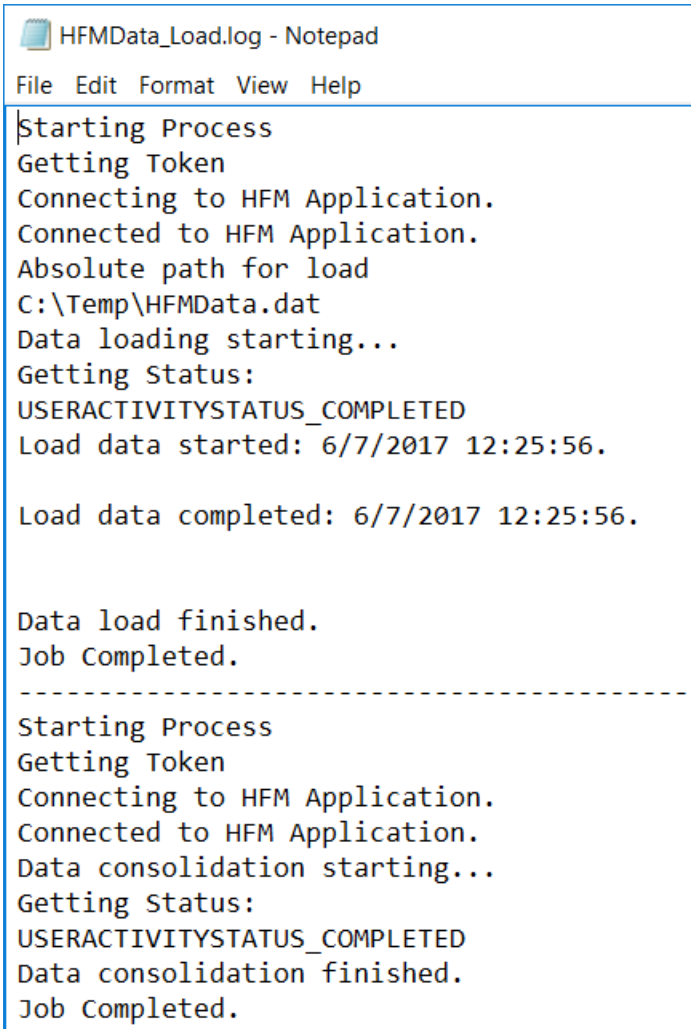

Figure 6- HFM data load log

As we can see, this is a very easy and transparent way to load data into HFM, using the same approach as the old version of ODI/HFM KMs.

ODI Data Extract procedure for HFM 11.1.2.4

First of all we need to understand why we are using an ODI procedure instead of an ODI KM for data extraction processes. The reason behind this decision was based on the nature of HFM data extracts, which automatically creates the outbound tables for the users containing all the extracted data and dimensions related to it. For example, when we send an extract command to HFM with a certain POV, HFM will create a fact table and a table for each dimension it contains using a pre-determined prefix (called “TIEOUT” in this article’s example).

TIEOUT_ACCOUNT
TIEOUT_CUSTOM1
TIEOUT_CUSTOM2
TIEOUT_CUSTOM3
TIEOUT_CUSTOM4
TIEOUT_ENTITY
TIEOUT_FACT
TIEOUT_ICP
TIEOUT_PARENT
TIEOUT_PERIOD
TIEOUT_SCENARIO
TIEOUT_VALUE
TIEOUT_VIEW
TIEOUT_YEAR

Figure 7- Tables automatically created by HFM Extract process

Since HFM create those tables automatically for the users, there is no reason to have an ODI interface to map it since the data would not go to the target table in the ODI interface but instead the data would go to those pre-defined HFM tables. Of course that we could elaborate an ODI KM that could send the extract command to HFM, which would populate the fact and dimension tables automatically and after that the KM code would read those tables and populate another target table that would be set in the ODI interface but the complexity to do that is not worth the work. It is way easier to run a procedure that will run the HFM extract and later on we may create another ODI interface that may read from those fact and dimension tables and do any ETL that we may want with them.

The "Extract Data from HFM" procedure is very simple and only contains two steps. The first step "Get HFM DB Connections" is used to get the DB user and password where the tables will be created. This DB connection needs to be the same one that is configured in HFM "Configure DNS" setup (more about that in a few).

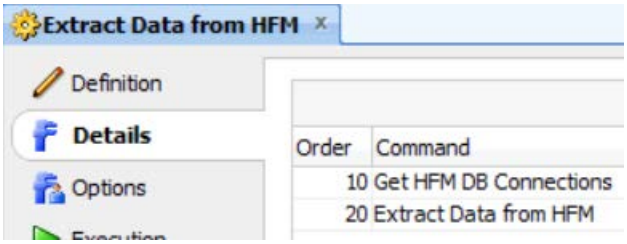

Figure 8- Extract procedure steps

The second step "Extract Data from HFM" contains the Java code that connects to HFM and send the extract command using a certain POV and a table prefix for the tables' creation.

ODI Extract Data Procedure usage

After importing "Extract Data from HFM" procedure into ODI you will need to create two ODI logical schemas. The first logical schema will point to the database where the extract tables will be created and

this DB needs to be the same one that you configured in HFM “Configure DNS” setup (found at “Consolidation Administration” on HFM Workspace) since this procedure uses the “Data Source Name” to get the information on where to extract the data. It seems really redundant (and it really is), but the extract procedure Java code needs both “Data Source Name” from HFM workspace and the database user and password from ODI Topology to work properly.

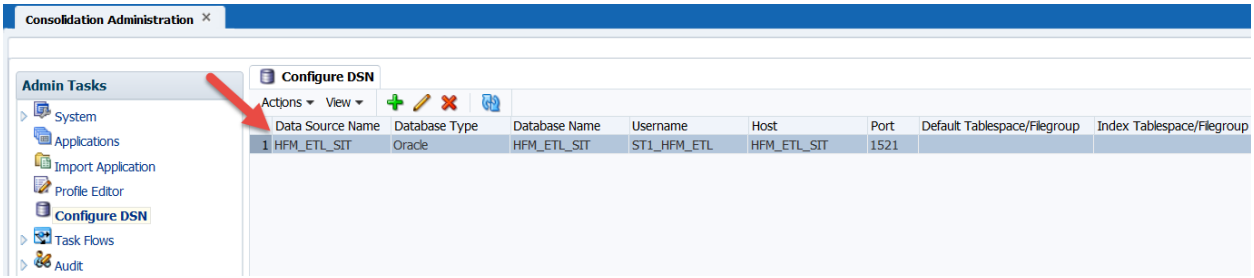

Figure 9- HFM workspace

The second logical schema may probably already exists in your environment (if you are following the examples given in the first article), which is related to the HFM application that you will extract data from. After both logical schemas are properly created and setup in ODI Topology, double click the export procedure in ODI, open the first step “Get HFM DB Connections”, click on “Command on source” and change the schema to the one that will hold the HFM extract tables.

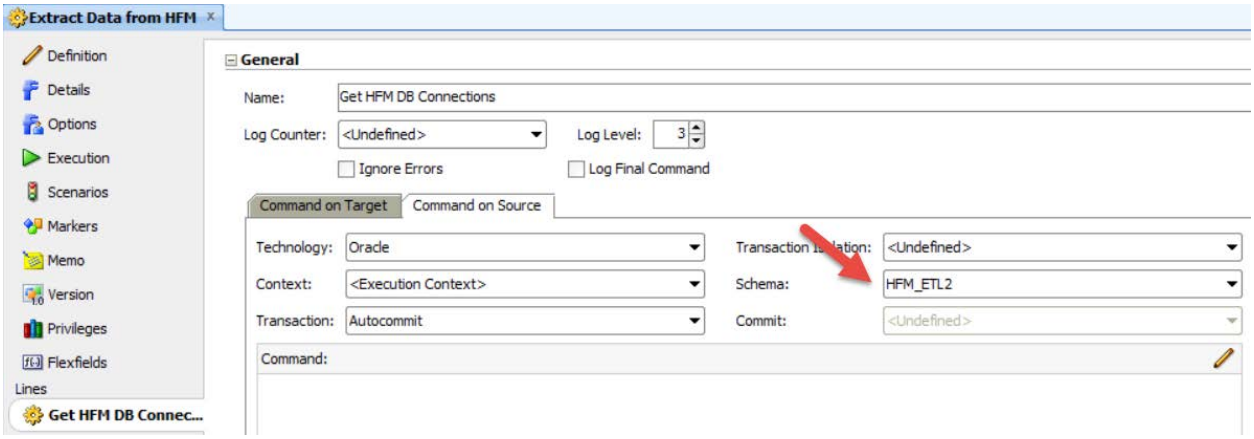

Figure 10- Change Extract DB logical schema

Do the same thing on “Extract Data from HFM” step and set the appropriated logical schema that is related to your HFM application.

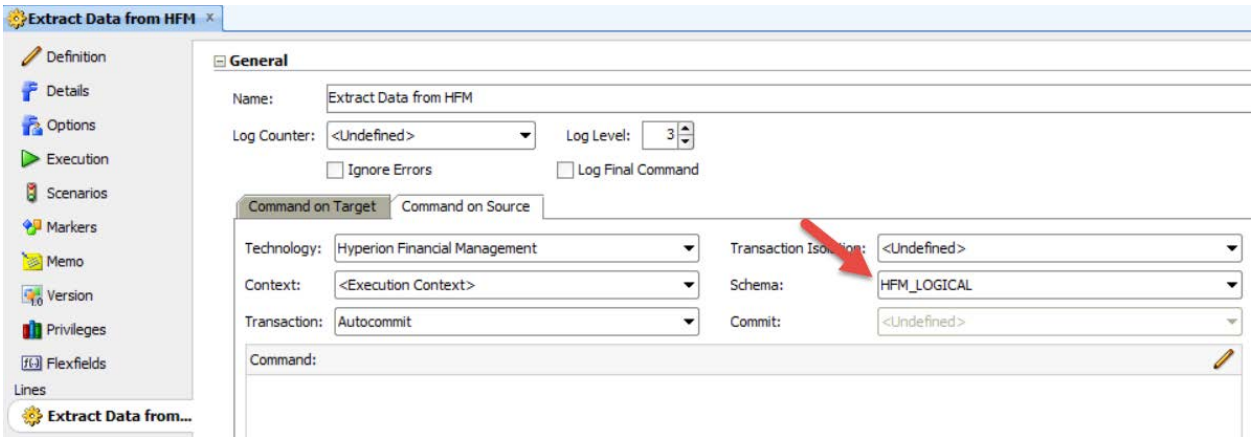

Figure 11- HFM logical schema setup

After this setup, we just need to populate the procedure options. The table below lists all the available options and a description of how to use them.

Option	Description	E.g. Values
EXTRACT_DNS	This option refers to the HOST attribute that will be mapped on mapDbConnectInfo API object. Its value is the same one that you have in "Consolidation Administration\Configure DNS\Data Source Name" inside your HFM application workspace.	HFM_ETL_DEV
EXTRACT_POV	Specifies the POV for extract operation.	A#{[Base]}; HEADCOUNTTOTPB. C1#TopCustom1. C2#TopCustom2. C4#TopCustom4. I#[ICP Top]. S#ACTUAL_USD. W#PERIODIC. Y#2017. V#<Entity Curr Total>. P#Nov. E#{TOPLF.[Base]}. C3#TopCustom3;PD
EXTRACT_PREFIX	Prefix that will be used to create the HFM extract tables inside the database.	TIEOUT TRG EXT
EXTRACT_TYPE	Specifies the parameters for extraction Type.	DATA_EXTRACT_TYPE_FLAG. EA_EXTRACT_TYPE_STANDARD
INCLUDE_CALCULATED	Defines if calculated members will be extracted or not.	True False
LOG_LOCATION	Log File Location and name.	C:/Temp/HFM_Extract.log

Table 2- Extract procedure options

Since all options are self-explanatory and have further detailed explanation on the ODI procedure “help” field of each of them if needed, we may just add this procedure in an ODI package and populate the options.

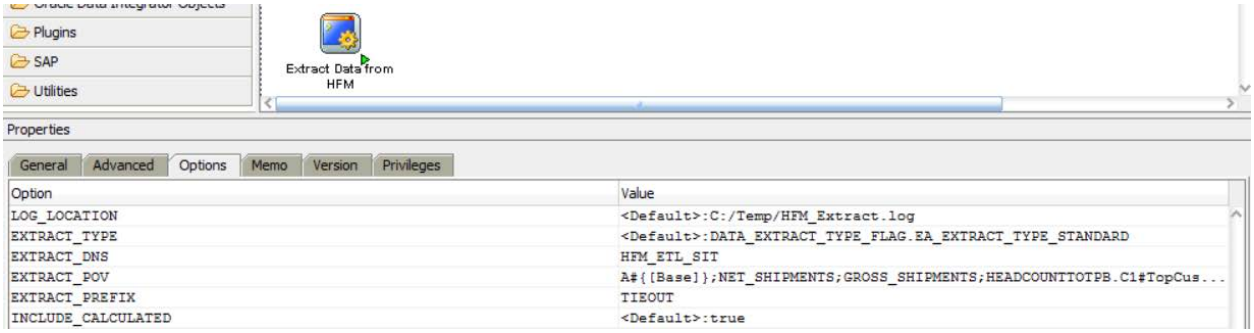

Figure 12- Extract procedure example

After the options are filled properly, we just need to run the ODI package. If all goes well, you will see green steps in the ODI Operator, indicating that the data was successfully extracted from the HFM application. You may also check the fact and dimension tables created on the database and the log that HFM generated for this process

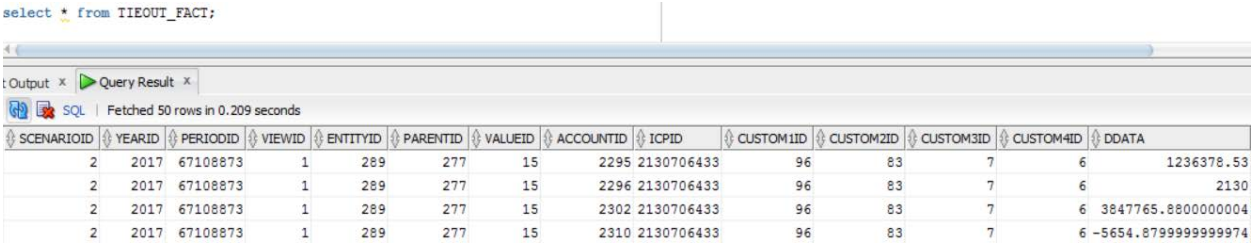

Figure 13- HFM Fact table example

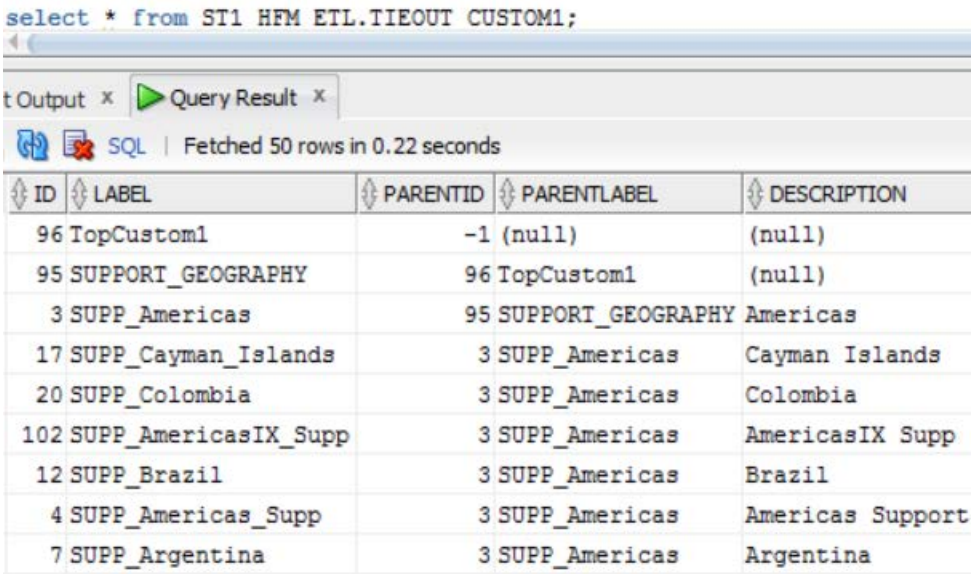

Figure 14- HFM Dimension table example

```
Starting Process
Getting Token
Connecting to HFM Application.
Connected to HFM Application.
Data extract starting...
Session ID:1803700720
USERACTIVITYSTATUS_COMPLETED


////////////////////////////////////////////////////////////////////////

//                   Financial Management Data Extract

//  Date:                        6/9/2017 6:51:04 PM

//  User:                        admin@Native Directory

//  Source Financial Management Application:    FINCDEV

//  Destination:        TIEOUT

//  Extract Type:                Financial Management Standard Star Schema

//  Delimiter:  ;

//  Include Data:       Yes

//  Dynamic Accounts:   Yes
```

Figure 15- HFM Extract log

As we can see, this is a very easy and transparent way to extract data from HFM. If you need to do some extra ETL on this extracted data, you just need to create other interfaces to manipulate the data even further.

Conclusion: HFM/ODI integration in a real environment

This article demonstrates the true power of ODI as a development platform. ODI is far from being just another ETL tool, as it goes way beyond this scope to give us the opportunity to implement any kind of connectivity between virtually any existing technologies. With a little setup and a couple of redesigned KMs, we were able to re-implement the old HFM/ODI connectivity, allowing us to do any kind of ETL on the necessary data before loading or extracting it, systematically and easily, to/from HFM.

These new KMs and procedures have already proven their worth on real projects where several ODI jobs created using the old HFM KMs needed to be replaced by the new ones, as the company was upgrading HFM to a newer version. This was a great accomplishment since there was no need to install any new application to the environment, like FDMEE; further, all the legacy ODI codes could be reused, since the

only change necessary to make ODI work in the new version of HFM was to replace the old KMs with the new ones and do some minor adjustments in its topology, data stores and interface options.

For new HFM implementations, the users could use ODI to create robust, scalable and enterprise integration solutions with very little development complexity, as the new ODI KMs abstract all the hard logic inside of them and can be reused as many times as necessary. Also, with ODI, users can integrate any kind of metadata and data sources into/from HFM, guaranteeing that all business requirements are fully implemented, even the more complex ones.

ODI is a dynamic and powerful tool that can provide an excellent platform for creating and maintaining EPM environments. This article demonstrates that we are limited in its use only by our imagination. With a few changes in the KMs we can overcome the boundaries of the default development, achieving a new level of excellence and thereby providing the increased flexibility, reliability and scalability to meet the challenges of a global and competitive environment.

About the Authors

Oracle ACE Rodrigo Radtke is a Software Development Consultant at Dell, where he specializes in ODI and EPM tools. A computer engineer experienced in software development, especially in the BI for Finance space, he is a certified professional on Oracle Data Integrator, Oracle SQL Expert, and Java (SCJP and SCWCD).

Oracle ACE Ricardo Giampaoli, a Master in Administration and system analyst, has been working in the IT environment for 20 years, the last nine years as an EPM consultant. He is a certified professional on Hyperion Planning, Essbase, OBIEE and ODI, and works with a great variety of Oracle tools.

Rodrigo and Ricardo frequently share their expertise by presenting at Kscope and other events and via their blog: https://devepm.com/

Appendix

IKM and Extract procedure downloads
https://drive.google.com/drive/folders/0B6qArlZmuBsaYnpRVVNzeVJvV2M

OTN Article - Integrating HFM 11.1.2.4 with ODI Metadata Knowledge Module (first article)
http://www.oracle.com/technetwork/articles/bi/giampaoli-hfm-odi-km-3748317.html

HFM Javadoc
https://docs.oracle.com/cd/E40248_01/epm.1112/hfm_javadoc/index.html

HFM Developer and Customization Guide
http://docs.oracle.com/cd/E57185_01/HFMCD/hfm_customdev.html

HFM Administrator's Guide
http://docs.oracle.com/cd/E57185_01/HFMAD/toc.htm

Eclipse Home Page
https://eclipse.org/