Monitoring EPM systems performance with Oracle Data Visualization

Alex Znyk
Ingersoll Rand
Disclaimer

This presentation is based on my experience. It does not reflect Ingersoll Rand or Oracle official positions. Any errors are my own.

This presentation is based on the following versions (on-premise):

- HFM: 11.1.2.4.204.5267  [Workspace: Help => About]
- Planning: 11.1.2.4.005.08  [Workspace: Help => About]
- Essbase: 11.1.2.4.0  [EAS or MaxL]
- Databases on SQL Server (old version)
- Oracle Data Visualization Desktop: 12.2.3

Things may differ if you are using other versions.
Introduction

This presentation is about getting from...

“If my system is not down, it means it’s working!”

“If users are not screaming, surely performance is good?”
Introduction

To a place where you know how your systems are behaving, and you can identify performance trends:
We will achieve this fine goal by:

• Exploring examples of information that can be gathered from Planning, Essbase and HFM; reviewing how to collect and prepare the data for consumption by Oracle DV

• Understanding how to connect Oracle DV to data sources

• Creating visualizations that allow to monitor EPM systems performance, and discussing some examples of how visualization can help understand system behavior and drive actions

Questions are welcome at any time !
Introduction

Who am I to talk about such things?

Alex Znyk  alexandre_znyk@irco.com

Sr Manager, Hyperion Systems Administration and Data Management at Ingersoll Rand, a global diversified manufacturing company

https://company.ingersollrand.com/

Prior to this, I oversaw global support of the Hyperion Systems, and had been on the team that performed the initial roll-out of the EPM systems.

My past experiences include:

• Consulting (Hyperion integration, Financial Processes Optimization),
• I.T. Audit (internal and external),
• Financial Planning & Analysis (know thy enemy).

At Ingersoll Rand, we use HFM, FDMEE, Planning, Essbase, EPM Maestro, DRM. Our user population is 1000+, located all over the world.
Monitoring EPM systems performance with Oracle Data Visualization

Part 1: Where is my data?
Part 1 - Game plan

We will take a look at the different systems we are interested in, and where to find useful data for our purpose:

• Planning: Job Status tables
• Essbase: using MaxL text log files to capture calculation execution times
• HFM: ErrorLog SysInfo records, Consolidation records

Along the way, I will share different techniques that I have found useful to prepare the data for better analytics.

This list is not meant to be exhaustive. It is based on my experience, and is there to serve as a starting point. If you find something useful and are willing to share, let me know, I am interested!
Hyperion Focus 17

Monitoring EPM systems performance with Oracle Data Visualization

Part 1: Where is my data?

Hyperion Planning

Job Status Tables
Information on business rule execution is available in the job console:

Detailed information about each job is available by clicking on the Run Status:
Part 1 – Hyperion Planning Job Status Tables

We can find this information in the database in the following tables:
HSP_JOB_STATUS: for recent jobs
HSP_HISTORICAL_JOB_STATUS: for older records

Let’s take a look:
The data has the same structure in the two tables:

• A parent record for the rule execution
• Children record for each parameter

We will use a query that joins the parent records and the parameters useful for analysis.

This will be useful when using the data to filter out records for specific business rules run with the same parameters.

The query is provided on the next page, and a sample of the output follows.
Part 1 – Hyperion Planning Job Status Tables

Query:

```sql
SELECT PJ.[JOB_ID],
    PJ.[JOB_NAME],
    PJ.[START_TIME],
    PJ.[END_TIME],
    DATEDIFF(ss, [START_TIME], [END_TIME]) AS [ExecTime],
    PJ.[RUN_STATUS],
    PJ.[ATTRIBUTE_1],
    PJ.[ATTRIBUTE_2],
    REPLACE(CAST(CJ1.[DETAILS] AS nvarchar(max)),"", ") AS [Entity],
    REPLACE(CAST(CJ2.[DETAILS] AS nvarchar(max)),"", ") AS [Product]
INTO zHISTJOB
FROM [HSP_HISTORICAL_JOB_STATUS] AS PJ
LEFT OUTER JOIN (
    SELECT [JOB_ID],
        [PARENT_JOB_ID],
        [JOB_NAME],
        [DETAILS]
    FROM [HSP_HISTORICAL_JOB_STATUS]
    WHERE [PARENT_JOB_ID] IS NOT NULL
    AND ([JOB_NAME] = 'Entity' OR [JOB_NAME] = 'Entity')
) AS CJ1
    ON PJ.[JOB_ID] = CJ1.[PARENT_JOB_ID]
LEFT OUTER JOIN (
    SELECT [JOB_ID],
        [PARENT_JOB_ID],
        [JOB_NAME],
        [DETAILS]
    FROM [HSP_HISTORICAL_JOB_STATUS]
    WHERE [PARENT_JOB_ID] IS NOT NULL
    AND ([JOB_NAME] = 'Product')
) AS CJ2
    ON PJ.[JOB_ID] = CJ2.[PARENT_JOB_ID]
WHERE PJ.[PARENT_JOB_ID] IS NULL
    AND PJ.[JOB_TYPE] = 1
ORDER BY [START_TIME] DESC
```
Sample output:

<table>
<thead>
<tr>
<th>JOB_ID</th>
<th>JOB_NAME</th>
<th>START_TIME</th>
<th>END_TIME</th>
<th>ExecTime</th>
<th>RUN_STATUS</th>
<th>ATTRIBUTE_1</th>
<th>ATTRIBUTE_2</th>
<th>Entity</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>1774205</td>
<td>FUNCSPND-SGA-F....</td>
<td>2017-10-07 13:16:35.357</td>
<td>2017-10-07 13:16:37.983</td>
<td>2</td>
<td>2</td>
<td>PLAN2</td>
<td>FUNCSPND</td>
<td>Entity_Total</td>
<td>P11000</td>
</tr>
<tr>
<td>1774155</td>
<td>REV_CST-AGG-AOP</td>
<td>2017-10-07 13:00:48.507</td>
<td>2017-10-07 13:01:04.250</td>
<td>16</td>
<td>2</td>
<td>PLAN2</td>
<td>REV_CST</td>
<td>Entity_Total</td>
<td>P0022</td>
</tr>
<tr>
<td>1774150</td>
<td>REV_CST-FX-AOP</td>
<td>2017-10-07 12:57:45.120</td>
<td>2017-10-07 12:58:55.553</td>
<td>70</td>
<td>2</td>
<td>PLAN2</td>
<td>REV_CST</td>
<td>Entity_Total</td>
<td>P0022</td>
</tr>
<tr>
<td>1774147</td>
<td>REVCST-SCM-Cont...</td>
<td>2017-10-07 12:56:17.750</td>
<td>2017-10-07 12:56:18.813</td>
<td>1</td>
<td>2</td>
<td>PLAN2</td>
<td>REV_CST</td>
<td>E9258</td>
<td>NULL</td>
</tr>
<tr>
<td>1774144</td>
<td>REVCST-SCM-Serv...</td>
<td>2017-10-07 12:55:05.413</td>
<td>2017-10-07 12:55:12.683</td>
<td>7</td>
<td>2</td>
<td>PLAN2</td>
<td>REV_CST</td>
<td>E9258</td>
<td>NULL</td>
</tr>
<tr>
<td>1774140</td>
<td>FUNCSPND-AGG-....</td>
<td>2017-10-07 12:44:19.973</td>
<td>2017-10-07 12:44:37.207</td>
<td>18</td>
<td>2</td>
<td>PLAN2</td>
<td>FUNCSPND</td>
<td>Entity_Total</td>
<td>P11000</td>
</tr>
<tr>
<td>1774136</td>
<td>FUNCSPND-SGA-F....</td>
<td>2017-10-07 12:43:48.160</td>
<td>2017-10-07 12:44:00.393</td>
<td>12</td>
<td>2</td>
<td>PLAN2</td>
<td>FUNCSPND</td>
<td>Entity_Total</td>
<td>P11000</td>
</tr>
</tbody>
</table>
Hyperion Focus 17

Monitoring EPM systems performance with Oracle Data Visualization

Part 1: Where is my data?

Hyperion Essbase
MaxL Script logs
A common practice for Essbase cubes is to run scheduled calculation scripts. For example, perform a nightly calculation or aggregation.

This can take the form of a MaxL script similar to this:

```maxl
set timestamp on;
SPOOL ON to 'PLAN-ConsFX\PLANConsFX.log';
LOGIN $1 $2 on $3;
execute calculation $5.'BS_OTHIS'.ConsFX';
execute calculation $5.'REV_CST'.ConsFX';
execute calculation $5.'FUNCSPND'.ConsFX';
execute calculation $5.'BS_OTHIS'.ConsFXAg';
execute calculation $5.'REV_CST'.ConsFXAg';
execute calculation $5.'FUNCSPND'.ConsFXAg';
SPOOL OFF;
EXIT;
```
The output into the log file will be similar to this:

essmeh timestamp: Sun Oct 08 22:30:00 2017

EXEC> execute calculation PLAN2.'BS_OTHIS'.>'ConsFX';

OK/INFO - 1200481 - Formula for member [Entity_Currency] will be executed in [TOPDO]
OK/INFO - 1200481 - Formula for member [Entity_Currency] will be executed in [TOPDO]
OK/INFO - 1012558 - Clearing all data blocks from [(ALL)] partition with fixed mem
OK/INFO - 1012675 - Commit Blocks Interval for the calculation is [[0]].
OK/INFO - 1017023 - No message for message #1017023 in message database.
OK/INFO - 1012675 - Commit Blocks Interval for the calculation is [[0]].
OK/INFO - 1012668 - Calculating [Currency(Entity_Currency)] with fixed members [Entity_Currenc
OK/INFO - 1012679 - Calculation task schedule [1].
OK/INFO - 1012680 - Parallelizing using [1] task dimensions...
OK/INFO - 1012553 - Copying data from [Working->FCST_09_03->Entity_Currency] to [Working]...
OK/INFO - 1012675 - Commit Blocks Interval for the calculation is [[0]].
OK/INFO - 1012675 - Commit Blocks Interval for the calculation is [[0]].
OK/INFO - 1012574 - Datacopy command copied [57849] source data blocks to [57849] target da
OK/INFO - 1012675 - Commit Blocks Interval for the calculation is [[0]].
OK/INFO - 1012569 - Formula on (or backward dependence from) member [Working] forced.
OK/INFO - 1012668 - Calculating [Version(Working)] with fixed members [Period(Jan)]
OK/INFO - 1012677 - Calculating in serial.
OK/INFO - 1012675 - Commit Blocks Interval for the calculation is [[0]].
OK/INFO - 1012668 - Calculating [Currency(Entity_Currency)] with fixed members [Entity_Currenc
OK/INFO - 1012679 - Calculation task schedule [1].
OK/INFO - 1012680 - Parallelizing using [1] task dimensions...
OK/INFO - 1012553 - Copying data from [Working->FCST_09_03->Entity_Currency] to [Working]...
OK/INFO - 1012675 - Commit Blocks Interval for the calculation is [[0]].
OK/INFO - 1012675 - Commit Blocks Interval for the calculation is [[0]].
OK/INFO - 1012574 - Datacopy command copied [57849] source data blocks to [57849] target da
OK/INFO - 1012675 - Commit Blocks Interval for the calculation is [[0]].
OK/INFO - 1012569 - Formula on (or backward dependence from) member [Working] forced.
OK/INFO - 1012668 - Calculating [Version(Working)] with fixed members [Period(Jan)]
OK/INFO - 1012677 - Calculating in serial.
OK/INFO - 1013274 - Calculation executed.

essmeh timestamp: Sun Oct 08 22:32:35 2017
In my case, our process creates a separate log file each time the script is executed (on a daily basis).

I collected the files into a folder:
To extract the data, I use an Excel VBA script which reads through the files, captures the execution time and outputs it to a spreadsheet page using different columns for the different cubes.

The output is the following:

<table>
<thead>
<tr>
<th>Date</th>
<th>IRPLAN.BSCAP.ConsFX</th>
<th>IRPLAN.PANDL.ConsFX</th>
<th>IRPLAN2.BS_OTHIS.ConsFX</th>
<th>IRPLAN2.REV_CST.ConsFX</th>
<th>IRPLAN2.FUNCSPND.ConsFX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/02/16</td>
<td>1.50</td>
<td>13.11</td>
<td>0.26</td>
<td>1.57</td>
<td>0.38</td>
</tr>
<tr>
<td>01/03/16</td>
<td>0.60</td>
<td>7.36</td>
<td>0.25</td>
<td>1.17</td>
<td>0.34</td>
</tr>
<tr>
<td>01/04/16</td>
<td>4.43</td>
<td>13.94</td>
<td>0.47</td>
<td>1.16</td>
<td>0.39</td>
</tr>
<tr>
<td>01/05/16</td>
<td>2.68</td>
<td>19.82</td>
<td>0.27</td>
<td>1.55</td>
<td>0.40</td>
</tr>
<tr>
<td>01/06/16</td>
<td>0.89</td>
<td>10.03</td>
<td>0.28</td>
<td>1.21</td>
<td>0.35</td>
</tr>
<tr>
<td>01/07/16</td>
<td>3.33</td>
<td>29.10</td>
<td>0.45</td>
<td>1.29</td>
<td>0.34</td>
</tr>
<tr>
<td>01/08/16</td>
<td>5.79</td>
<td>30.84</td>
<td>0.36</td>
<td>1.19</td>
<td>0.35</td>
</tr>
<tr>
<td>01/09/16</td>
<td>1.82</td>
<td>12.92</td>
<td>0.26</td>
<td>1.44</td>
<td>0.34</td>
</tr>
<tr>
<td>01/10/16</td>
<td>6.59</td>
<td>25.10</td>
<td>0.54</td>
<td>2.10</td>
<td>0.64</td>
</tr>
<tr>
<td>01/11/16</td>
<td>7.90</td>
<td>49.25</td>
<td>0.67</td>
<td>1.36</td>
<td>0.33</td>
</tr>
<tr>
<td>01/12/16</td>
<td>5.00</td>
<td>36.91</td>
<td>0.32</td>
<td>5.23</td>
<td>0.35</td>
</tr>
<tr>
<td>01/13/16</td>
<td>7.87</td>
<td>38.02</td>
<td>0.33</td>
<td>1.20</td>
<td>0.33</td>
</tr>
<tr>
<td>01/14/16</td>
<td>4.01</td>
<td>24.03</td>
<td>0.35</td>
<td>1.15</td>
<td>0.34</td>
</tr>
<tr>
<td>01/15/16</td>
<td>5.67</td>
<td>26.12</td>
<td>0.30</td>
<td>1.40</td>
<td>0.33</td>
</tr>
<tr>
<td>01/16/16</td>
<td>1.26</td>
<td>13.17</td>
<td>0.25</td>
<td>1.26</td>
<td>0.35</td>
</tr>
<tr>
<td>01/17/16</td>
<td>2.53</td>
<td>19.01</td>
<td>0.29</td>
<td>1.26</td>
<td>0.42</td>
</tr>
<tr>
<td>01/18/16</td>
<td>5.77</td>
<td>37.85</td>
<td>0.27</td>
<td>1.26</td>
<td>0.37</td>
</tr>
<tr>
<td>01/19/16</td>
<td>6.50</td>
<td>36.33</td>
<td>0.25</td>
<td>1.23</td>
<td>0.33</td>
</tr>
<tr>
<td>01/20/16</td>
<td>5.67</td>
<td>32.84</td>
<td>0.27</td>
<td>1.21</td>
<td>0.35</td>
</tr>
<tr>
<td>01/21/16</td>
<td>6.55</td>
<td>37.59</td>
<td>0.26</td>
<td>1.56</td>
<td>0.39</td>
</tr>
<tr>
<td>01/22/16</td>
<td>6.89</td>
<td>40.97</td>
<td>0.26</td>
<td>1.22</td>
<td>0.44</td>
</tr>
<tr>
<td>01/23/16</td>
<td>1.26</td>
<td>12.59</td>
<td>0.15</td>
<td>1.43</td>
<td>0.44</td>
</tr>
<tr>
<td>01/24/16</td>
<td>0.54</td>
<td>7.22</td>
<td>0.17</td>
<td>1.30</td>
<td>0.43</td>
</tr>
<tr>
<td>01/25/16</td>
<td>5.82</td>
<td>33.22</td>
<td>0.12</td>
<td>1.24</td>
<td>0.33</td>
</tr>
<tr>
<td>01/26/16</td>
<td>3.83</td>
<td>33.23</td>
<td>0.53</td>
<td>1.20</td>
<td>0.34</td>
</tr>
<tr>
<td>01/27/16</td>
<td>4.95</td>
<td>39.52</td>
<td>0.80</td>
<td>1.16</td>
<td>0.34</td>
</tr>
<tr>
<td>01/28/16</td>
<td>5.89</td>
<td>39.58</td>
<td>0.38</td>
<td>1.26</td>
<td>0.33</td>
</tr>
</tbody>
</table>
The vba script:

Sub LoopThroughFiles()
    Dim MyObj As Object, MySource As Object, file As Variant
    Dim StrFile, strRange As String
    Dim i, j As Integer
    Dim fso As New FileSystemObject
    Dim dataArray
    Dim strScript, strExec As String
    i = 2
    strPath = "C:\Path\"
    StrFile = Dir(strPath & "*PLANCONSFX.LOG")
    Do While Len(StrFile) > 0
        strRange = "A" & i
        Sheets("LogLoad").Range(strRange) = Replace(ReplaceStrFile, "_.PLANCONSFX.LOG", "")
        StrFile = Dir
        If StrFile <> "" Then
            Set fs0 = CreateObject("Scripting.FileSystemObject")
            Set fil = fs0.GetFile(strPath & StrFile)
            Set txt = fil.OpenAsTextStream(1)
            strtxt = txt.ReadAll
            txt.Close
            dataArray = Split(strtxt, vbCrLf)
            j = 1
            strExec = ""
            For Each strline In dataArray
                j = j + 1
                If Left(strline, 26) = "MAXL> execute calculation " Then
                    strScript = Trim(Mid(strline, 27, 5000))
                End If
                If Instr(strScript, "BSCAP.'ConsFX') Then
                    strRange = "B" & i
                    Sheets("LogLoad").Range(strRange) = Replace(Replace(strExec, "] seconds.", ""), "[ConsFX.csc]:"", "") * 1 / 60
                    End If
                If Instr(strScript, "PANDL.'ConsFX') Then
                    strRange = "C" & i
                    Sheets("LogLoad").Range(strRange) = Replace(Replace(strExec, "] seconds.", ""), "[ConsFX.csc]:"", "") * 1 / 60
                    End If
                If Instr(strScript, "BS_OTHIS.'ConsFX') Then
                    strRange = "D" & i
                    Sheets("LogLoad").Range(strRange) = Replace(Replace(strExec, "] seconds.", ""), "[ConsFX.csc]:"", "") * 1 / 60
                    End If
                If Instr(strScript, "REV_CST.'ConsFX') Then
                    strRange = "E" & i
                    Sheets("LogLoad").Range(strRange) = Replace(Replace(strExec, "] seconds.", ""), "[ConsFX.csc]:"", "") * 1 / 60
                    End If
                If Instr(strScript, "FNCSPND.'ConsFX') Then
                    strRange = "F" & i
                    Sheets("LogLoad").Range(strRange) = Replace(Replace(strExec, "] seconds.", ""), "[ConsFX.csc]:"", "") * 1 / 60
                    End If
            Next
            Set fso = Nothing
            End If
        End If
    End While
End Sub
Monitoring EPM systems performance with Oracle Data Visualization

Part 1: Where is my data?

HFM

Overview
HFM relies on a database that contains system tables, and application specific tables. The tables we are interested in are:

### HFM Database

#### “System” tables
- HFM_ERRORLOG
- HSV_USERS_ON_SYSTEM
- HSV_ACTIVITY_USERS

#### Application tables
For each application, table names are prefixed with the application name:
- [app name]_RUNNING_TASKS
- [app name]_TASK_AUDIT
Monitoring EPM systems performance with Oracle Data Visualization

Part 1: Rounding up your source data

HFM
Consolidation Times
HFM Consolidation information is captured by the system and stored in the database.

It can be accessed through Workspace in the Task Audit:

![Task Audit](image)
Workspace allows to extract this information into a text file. We could certainly use this as a source, however, this would require to re-export through Workspace each time we want to refresh our dataset.

We will see how to query the same information from the database. The main table for this is [app name]_TASK_AUDIT:

```
SELECT TOP (100) [strGUID],[ActivityUserID],[ActivitySessionID],[ActivityCode],[ServerName],[AppName],[StartTime],[EndTime],[strDescription],[strModuleName]
FROM [app name]_TASK_AUDIT
```
There are a couple of transformations to perform to get the information in a more useable format.

UserID: the user IDs are identifiers and need to be matched to table [HSV_ACTIVITY_USERS] to display the username.
Part 1 – HFM Consolidation Times

Timesteps: Start and End time timestamps are stored as numbers (floats), representing the number of days since January 1st, 1900. Casting to datetime and adjusting for timezone difference makes this much nicer:

```sql
CAST(StartTime - 2.1666666666606034 as datetime) AS StartTimestamp,
CAST(EndTime - 2.1666666666606034 as datetime) AS EndTimestamp
```

<table>
<thead>
<tr>
<th>Start Time</th>
<th>Start Timestamp</th>
<th>End Time</th>
<th>End Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>43010.927025463</td>
<td>2017-10-02 18:14:55.000</td>
<td>43010.927025463</td>
<td>2017-10-02 18:14:55.000</td>
</tr>
<tr>
<td>43010.9270138889</td>
<td>2017-10-02 18:14:54.000</td>
<td>43010.9270138889</td>
<td>2017-10-02 18:14:54.000</td>
</tr>
<tr>
<td>43010.9237962963</td>
<td>2017-10-02 18:10:16.000</td>
<td>43010.9237962963</td>
<td>2017-10-02 18:10:16.000</td>
</tr>
<tr>
<td>43010.9236689815</td>
<td>2017-10-02 18:10:05.000</td>
<td>43010.9236689815</td>
<td>2017-10-02 18:10:05.000</td>
</tr>
<tr>
<td>43010.9235763889</td>
<td>2017-10-02 18:09:57.000</td>
<td>43010.9235763889</td>
<td>2017-10-02 18:09:57.000</td>
</tr>
<tr>
<td>43010.9235648148</td>
<td>2017-10-02 18:09:56.000</td>
<td>43010.9235648148</td>
<td>2017-10-02 18:09:56.000</td>
</tr>
<tr>
<td>43010.9234606481</td>
<td>2017-10-02 18:09:47.000</td>
<td>43010.9234606481</td>
<td>2017-10-02 18:09:47.000</td>
</tr>
<tr>
<td>43010.9234606481</td>
<td>2017-10-02 18:09:47.000</td>
<td>43010.9234606481</td>
<td>2017-10-02 18:09:47.000</td>
</tr>
</tbody>
</table>
Filtering records for Consolidation only:
The table contains all types of events. In this case, we are only interested in Consolidation (other events could be used for further analysis), which Activity Code is 4.

(Refer to next page for the table of Activity Codes)
Furthermore, two records are created, one when Consolidation start, the other when it completes. We will only use the one marking completion.

```sql
WHERE ActivityCode = 4
AND strDescription Like 'Consolidation Completed%'"
### Part 1 – HFM Consolidation Times

#### Table of HFM Activity codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Idle</td>
<td>21</td>
<td>MetadataLoad</td>
</tr>
<tr>
<td>1</td>
<td>RulesLoad</td>
<td>22</td>
<td>MetadataExtract</td>
</tr>
<tr>
<td>2</td>
<td>RulesScan</td>
<td>23</td>
<td>MemberListLoad</td>
</tr>
<tr>
<td>3</td>
<td>RulesExtract</td>
<td>24</td>
<td>MemberListScan</td>
</tr>
<tr>
<td>4</td>
<td>Consolidation</td>
<td>25</td>
<td>MemberListExtract</td>
</tr>
<tr>
<td>5</td>
<td>ChartLogic</td>
<td>26</td>
<td>SecurityLoad</td>
</tr>
<tr>
<td>6</td>
<td>Translation</td>
<td>27</td>
<td>SecurityScan</td>
</tr>
<tr>
<td>7</td>
<td>CustomLogic</td>
<td>28</td>
<td>SecurityExtract</td>
</tr>
<tr>
<td>8</td>
<td>Allocate</td>
<td>29</td>
<td>Logon</td>
</tr>
<tr>
<td>9</td>
<td>DataLoad</td>
<td>30</td>
<td>LogonFailure</td>
</tr>
<tr>
<td>10</td>
<td>DataExtract</td>
<td>31</td>
<td>Logoff</td>
</tr>
<tr>
<td>11</td>
<td>DataExtractviaHAL</td>
<td>32</td>
<td>External</td>
</tr>
<tr>
<td>12</td>
<td>DataEntry</td>
<td>33</td>
<td>MetadataScan</td>
</tr>
<tr>
<td>13</td>
<td>DataRetrieval</td>
<td>34</td>
<td>DataScan</td>
</tr>
<tr>
<td>14</td>
<td>DataClear</td>
<td>35</td>
<td>ExtendedAnalyticsExport</td>
</tr>
<tr>
<td>15</td>
<td>DataCopy</td>
<td>36</td>
<td>ExtendedAnalyticsSchemaDelete</td>
</tr>
<tr>
<td>16</td>
<td>JournalEntry</td>
<td>37</td>
<td>TransactionsLoad</td>
</tr>
<tr>
<td>17</td>
<td>JournalRetrieval</td>
<td>38</td>
<td>TransactionsExtract</td>
</tr>
<tr>
<td>18</td>
<td>JournalPosting</td>
<td>39</td>
<td>DocumentAttachments</td>
</tr>
<tr>
<td>19</td>
<td>JournalUnposting</td>
<td>40</td>
<td>DocumentDetachments</td>
</tr>
<tr>
<td>20</td>
<td>JournalTemplateEntry</td>
<td>41</td>
<td>CreateTransactions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>EditTransactions</td>
</tr>
<tr>
<td>42</td>
<td>DeleteTransactions</td>
</tr>
<tr>
<td>43</td>
<td>PostTransactions</td>
</tr>
<tr>
<td>44</td>
<td>UnpostTransactions</td>
</tr>
<tr>
<td>45</td>
<td>DeleteInvalidRecords</td>
</tr>
<tr>
<td>46</td>
<td>TaskAuditPurged</td>
</tr>
<tr>
<td>47</td>
<td>PostAllTransactions</td>
</tr>
<tr>
<td>48</td>
<td>UnpostAllTransactions</td>
</tr>
<tr>
<td>49</td>
<td>DeleteAllTransactions</td>
</tr>
<tr>
<td>50</td>
<td>UnmatchAllTransactions</td>
</tr>
<tr>
<td>51</td>
<td>AutoMatchbyID</td>
</tr>
<tr>
<td>52</td>
<td>AutoMatchbyAccount</td>
</tr>
<tr>
<td>53</td>
<td>IntercompanyMatchingReportbyID</td>
</tr>
<tr>
<td>54</td>
<td>IntercompanyMatchingReportbyAcct</td>
</tr>
<tr>
<td>55</td>
<td>IntercompanyTransactionReport</td>
</tr>
<tr>
<td>56</td>
<td>ManualMatch</td>
</tr>
<tr>
<td>57</td>
<td>UnmatchSelected</td>
</tr>
<tr>
<td>58</td>
<td>ManageICPeriods</td>
</tr>
<tr>
<td>59</td>
<td>Lock/UnlockICEntities</td>
</tr>
<tr>
<td>60</td>
<td>ManageICReasonCodes</td>
</tr>
<tr>
<td>61</td>
<td>Null</td>
</tr>
</tbody>
</table>
Remove unnecessary text from the Description field:

```sql
strDescription
Consolidation Completed. The Scenario is ACTUAL; the Year is 2017; the Start Period is Sep; the End Period is Sep; the Entity is CB...
```

```sql
(replace(replace(replace(replace(replace(replace(replace(strDescription,'The Parent is ','')
    ,'ConsolidationType: ',''))
    ,'The Entity is ',''))
    ,'The End Period is ',''))
    ,'The Start Period is ',''))
    ,'The Year is ',''))
    ,'The Scenario is ',''))
    ,'Consolidation Completed. ',''))
    ,'Consolidation Completed',''))
```

```sql
AS POV
```

```sql
POV
ACTUAL: 2017; Sep; Sep; CBC_TAX; IMPACTED
```
Transform the POV field into separate columns:

We will perform this parsing by using the following trick (the old version of T-SQL I use does not have string parsing):

- Transform the Error Info field by replacing ‘;’ with XML tags
- Convert it to an xml field
- Extract the different values by XML lookups

This generates an XML field with multiple records, so will need to join tables to build our output.

The final query is provided on the next page, and the sample output follows
SELECT [sUserName],
[ServerName],
[AppName],
StartTS,
EndTS,
Duration_s,
Scenario=n.v.value('/e[1]', 'varchar(20)')
,Year=n.v.value('/e[2]', 'varchar(20)')
,StartPer=n.v.value('/e[3]', 'varchar(20)')
,EndPer=n.v.value('/e[4]', 'varchar(20)')
,Entity=Replace(n.v.value('/e[5]', 'varchar(20)'), '.', '')
,TypeOrPar=Replace(n.v.value('/e[6]', 'varchar(20)'), '.', '')
,Type=Replace(n.v.value('/e[7]', 'varchar(20)'), '.', '')
FROM ((SELECT [sUserName],
[ServerName],
[AppName],
CONVERT(datetime, (StartTime - 2.166666666693345)) AS StartTS
,CONVERT(datetime, (EndTime - 2.166666666693345)) AS EndTS
,DATEDIFF(SECOND, CONVERT(datetime, (StartTime - 2.166666666693345)), CONVERT(datetime, (EndTime - 2.166666666693345))) AS Duration_s
,(replace(replace(replace(replace(replace(replace(replace(replace(replace...
,'The Parent is ',''),'
,'The Entity is ',''),'The End Period is ',''),'
,'The Start Period is ',''),'
,'The Year is ',''),'
,'The Scenario is ',''),'
,'Consolidation Completed. ',''),'
,'Consolidation Completed','') AS POV
FROM [HYPFMR12_TASK_AUDIT]
LEFT OUTER JOIN [HSV_ACTIVITY_USERS] ON [HSV_ACTIVITY_USERS].[lUserID] = [ActivityUserID]
WHERE ActivityCode = 4 AND strDescription Like 'Consolidation Completed%') AS tt
CROSS APPLY
(SELECT CAST('<e>' + Replace(tt.POVOXML, '; ', '</e><e>')+'</e>' AS XML) AS POVOXML
CROSS APPLY
i.[POVOXML].nodes('/') AS v)
ORDER BY StartTS DESC
### Sample output:

<table>
<thead>
<tr>
<th>sUserName</th>
<th>ActivityCode</th>
<th>ServerName</th>
<th>AppName</th>
<th>StartTS</th>
<th>EndTS</th>
<th>Duration_s</th>
<th>POV</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc-hyper</td>
<td>4</td>
<td>IGREPMNDC</td>
<td>HYPFM</td>
<td>2017-10-09 15:09:58.997</td>
<td>2017-10-09 15:09:58.997</td>
<td>0</td>
<td>ACTUAL; 2017; Sep; Sep; CBC_TAX; IMPACTED</td>
</tr>
<tr>
<td>svc-hyper</td>
<td>4</td>
<td>IGREPMNDC</td>
<td>HYPFM</td>
<td>2017-10-09 15:09:54.997</td>
<td>2017-10-09 15:09:54.997</td>
<td>0</td>
<td>ACTUAL; 2017; Sep; Sep; TAX; IMPACTED</td>
</tr>
<tr>
<td>svc-hyper</td>
<td>4</td>
<td>IGREPMNDC</td>
<td>HYPFM</td>
<td>2017-10-09 15:09:51.997</td>
<td>2017-10-09 15:09:51.997</td>
<td>0</td>
<td>ACTUAL; 2017; Sep; Sep; LEGAL; IMPACTED</td>
</tr>
</tbody>
</table>
Part 1: Where is my data?

HFM
System Information
HFM records automatically System Information. This information can be accessed from the workspace:

Navigate => Administer => Consolidation Administration

Once in Consol. Administration, access:
Administration => System => Messages
Part 1 – HFM System Information

Select Diagnostic messages:
Clicking on a message displays its full contents:

That’s what I call some good information!

Now, if only it could be fed into a data visualization tool...
Where can we find that?
Let’s take a look in the database:

The information is definitely there...

But will require a bit of work:
• Reformating the Timestamps
• Transforming the “XMLError” message
Transforming the timestamp into a datetime format

Timestamps are stored as numbers (floats), representing the number of days since January 1st, 1900. Casting to datetime and adjusting for timezone difference makes this much nicer:

```sql
CAST([dTimeStamp] - 2.166666666606034 as datetime) AS TimeStamp
```

<table>
<thead>
<tr>
<th>dTimeStamp</th>
<th>TimeStamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>43011.7137156468</td>
<td>2017-10-03 13:07:45.030</td>
</tr>
<tr>
<td>43011.7132007162</td>
<td>2017-10-03 13:07:00.540</td>
</tr>
<tr>
<td>43011.7129523842</td>
<td>2017-10-03 13:06:39.083</td>
</tr>
</tbody>
</table>
We can generate additional fields derived from the timestamp:
- the date in a text format (will allow us to do day averages)
- the month number (month averages)
- Week day number
- Hour

```sql
CAST(CONVERT(date,[TimeStamp]) AS nvarchar(10)) AS strDate,
DATEPART(MONTH,[TimeStamp]) AS numMth,
DATEPART(dw,[TimeStamp]) numWeekDay,
DATEPART(HOUR,[TimeStamp]) AS numHour
```

These can be very useful to generate more analytics.

We will see later that Oracle DV also has the capability to derive different fields from the data source.
The next step is to transform the message field. This is an XML field, so let’s first take a look at an example to understand the structure:

The good news is that it’s well structured, and SQL provides functions to process XML.
To use SQL XML functions, we need to transform this field from its text format to XML.

```sql
SELECT TOP 10
    [sServerName],
    [sAppName],
    dTimeStamp,
    CAST([dTimeStamp] - 2.16666666666606034 AS datetime) AS TimeStamp,
    (CAST([sXMLLError] AS XML)).value('(/EStr/AppName/node())[1]', 'varchar(10)') AS XML_App,
    (CAST([sXMLLError] AS XML)).value('(/EStr/ESec/Svr/node())[1]', 'varchar(20)') AS XML_Svr,
    (CAST([sXMLLError] AS XML)).value('(/EStr/ESec/DTme/node())[1]', 'varchar(50)') AS XML_TS,
    (CAST([sXMLLError] AS XML)).value('(/EStr/ESec/Ver/node())[1]', 'varchar(50)') AS XML_Ver,
    (CAST([sXMLLError] AS XML)).value('(/EStr/ESec/DStr/node())[1]', 'varchar(500)') AS XML_Info
FROM [HFM_ERRORLOG]
WHERE LogType = 6
ORDER BY dTimeStamp DESC
```
Much better!

But we are not quite there yet: the information is bundled in a single field.

Let’s parse this into separate fields, so that we have the data we want to graph nicely laid out.
We will perform this parsing by using the following trick (the old version of T-SQL I use does not have string parsing):

- Transform the Error Info field by replacing ‘;’ with XML tags
- Convert it to an xml field
- Extract the different values by XML lookups

This generates an XML field with multiple records, so will need to join tables to build our output.

Once this is done, we will remove the extra text from each field, and cast the results to the appropriate numeric type.
Here is the resulting query with select output fields (the query with all fields is provided later for reference):

```sql
SELECT [sServerName],
[sAppName],
[TimeStamp],
CAST(CONVERT(date,[TimeStamp]) AS nvarchar(10)) AS strDate,
DATEPART(MONTH,[TimeStamp]) AS numMth,
DATEPART(dw,[TimeStamp]) numWeekDay,
DATEPART(HOUR,[TimeStamp]) AS numHour,
NumUsers=CAST(REPLACE(n.v.value('/e[3]','varchar(50)'),'NumUsers=','') AS int),
[UsedPhysicalMem]=CAST(REPLACE(n.v.value('/e[8]','varchar(50)'),'UsedPhysicalMem=','') AS bigint),
[UsedVirtualMem]=CAST(REPLACE(n.v.value('/e[11]','varchar(50)'),'UsedVirtualMem=','') AS bigint),
[UsedCPU]=CAST(REPLACE(n.v.value('/e[13]','varchar(50)'),'UsedCPU=','') AS float),
NumDataRecordsInRAM=CAST(REPLACE(n.v.value('/e[16]','varchar(50)'),'NumDataRecordsInRAM=','') AS bigint),
[NumRecordsInLargestCube]=CAST(REPLACE(n.v.value('/e[17]','varchar(50)'),'NumRecordsInLargestCube=','') AS bigint) FROM

(SELECT
[sServerName],
[sAppName],
CAST([dTimeStamp] - 2.166666666606034 as datetime) AS [TimeStamp],
(CAST([sXMLError] as XML)).value('//EStr//ESec//DStr//node()')[1], 'varchar(1000)') AS [XML_Info]
FROM [HFM_ERRORLOG]
WHERE [LogType] = 6
ORDER BY [dTimeStamp] DESC) AS tt
CROSS APPLY
(SELECT CAST('<e>'+Replace(tt.XML_Info,' ','<e><e>'+'</e>') AS XML) AS [XML_Info_XML]) AS i
CROSS APPLY
i.[XML_Info_XML].nodes('/') AS n(v)

)
# Part 1 – HFM System Information

<table>
<thead>
<tr>
<th>Server Name</th>
<th>sAppName</th>
<th>Timestamp</th>
<th>strDate</th>
<th>numMth</th>
<th>numWe</th>
<th>numHour</th>
<th>NumU...</th>
<th>UsedPhysicalMem</th>
<th>UsedVirtualMem</th>
<th>UsedCPU</th>
<th>NumData</th>
<th>Records In RAM</th>
<th>NumRecords In LargestCube</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 11:25:24.277</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>11</td>
<td>0</td>
<td>4667387352</td>
<td>3095322524</td>
<td>0.322513</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 11:21:24.233</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>11</td>
<td>0</td>
<td>4663661952</td>
<td>3095322524</td>
<td>0.354949</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 11:15:24.177</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>11</td>
<td>0</td>
<td>48709545408</td>
<td>3095322524</td>
<td>0.590429</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 11:11:24.363</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>11</td>
<td>0</td>
<td>4746604544</td>
<td>2387615744</td>
<td>6.3994</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 10:59:23.980</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>0</td>
<td>4255387072</td>
<td>3039307104</td>
<td>0.312431</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 10:54:23.950</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>0</td>
<td>4255672256</td>
<td>3039307104</td>
<td>0.32595</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 10:49:23.910</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>0</td>
<td>4245134784</td>
<td>3039307104</td>
<td>0.630395</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 10:44:24.127</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>0</td>
<td>4138749952</td>
<td>2384850240</td>
<td>6.3994</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 10:00:23.840</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>0</td>
<td>427563176</td>
<td>3080830376</td>
<td>0.441155</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 09:55:23.890</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>427168256</td>
<td>3088891904</td>
<td>0.295171</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 09:50:23.860</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>4283647648</td>
<td>3088891904</td>
<td>0.376585</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 09:45:23.820</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>4269178880</td>
<td>3088891904</td>
<td>0.613580</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 09:40:24.040</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>4156655552</td>
<td>2388140032</td>
<td>6.3994</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 09:38:23.423</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>4673261696</td>
<td>3076204080</td>
<td>0.357143</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 09:33:23.367</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>4515715520</td>
<td>3076204080</td>
<td>0.51989</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 09:08:23.277</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>8</td>
<td>0</td>
<td>4860772352</td>
<td>3076204080</td>
<td>0.362828</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 08:53:23.177</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>8</td>
<td>0</td>
<td>4861227008</td>
<td>3076204080</td>
<td>0.421818</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 08:48:23.133</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>8</td>
<td>0</td>
<td>4854956032</td>
<td>3091259352</td>
<td>0.649328</td>
<td>301571</td>
<td>65623</td>
<td></td>
</tr>
<tr>
<td>IGREPMM...</td>
<td>HYPPM...</td>
<td>2017-10-07 08:43:23.287</td>
<td>2017-10-07</td>
<td>10</td>
<td>7</td>
<td>8</td>
<td>0</td>
<td>4732264448</td>
<td>2388920000</td>
<td>9.5194</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Part 1 – HFM System Information

For reference, this is the full final query that I use to generate the transformed data set for analytics:

```sql
SELECT
    [sServerName],
    [sAppName],
    [TimeStamp],
    CAST(CONVERT(date,[TimeStamp]) AS nvarchar(10)) AS strDate,
    DATEPART(MONTH,[TimeStamp]) AS numMth,
    DATEPART(dw,[TimeStamp]) numWeekDay,
    DATEPART(HOUR,[TimeStamp]) AS numHour,
    PID=REPLACE(n.v.value('/e[2]', 'varchar(50)'),'PID=',''),
    NumUsers=CAST(REPLACE(n.v.value('/e[3]', 'varchar(50)'),'NumUsers=','') AS int),
    CPU_Cores=REPLACE(n.v.value('/e[4]', 'varchar(50)'),'CPU(Cores)=',''),
    NumTasks=CAST(REPLACE(n.v.value('/e[5]', 'varchar(50)'),'NumTasks=','') AS int),
    NumErrors=CAST(REPLACE(n.v.value('/e[6]', 'varchar(50)'),'NumErrors=','') AS int),
    PhysicalMem=CAST(REPLACE(n.v.value('/e[7]', 'varchar(50)'),'PhysicalMem=','') AS bigint),
    UsedPhysicalMem=CAST(REPLACE(n.v.value('/e[8]', 'varchar(50)'),'UsedPhysicalMem=','') AS bigint),
    ProcUsedPhysicalMem=CAST(REPLACE(n.v.value('/e[9]', 'varchar(50)'),'ProcUsedPhysicalMem=','') AS bigint),
    VirtualMem=CAST(REPLACE(n.v.value('/e[10]', 'varchar(50)'),'VirtualMem=','') AS bigint),
    UsedVirtualMem=CAST(REPLACE(n.v.value('/e[11]', 'varchar(50)'),'UsedVirtualMem=','') AS bigint),
    ProcUsedVirtualMem=CAST(REPLACE(n.v.value('/e[12]', 'varchar(50)'),'ProcUsedVirtualMem=','') AS bigint),
    UsedCPU=CAST(REPLACE(n.v.value('/e[13]', 'varchar(50)'),'UsedCPU=','') AS float),
    ProcUsedCPU=CAST(REPLACE(n.v.value('/e[14]', 'varchar(50)'),'ProcUsedCPU=','') AS float),
    NumCubesInRAM=CAST(REPLACE(n.v.value('/e[15]', 'varchar(50)'),'NumCubesInRAM=','') AS bigint),
    NumDataRecordsInRAM=CAST(REPLACE(n.v.value('/e[16]', 'varchar(50)'),'NumDataRecordsInRAM=','') AS bigint),
    NumRecordsInLargestCube=CAST(REPLACE(n.v.value('/e[17]', 'varchar(50)'),'NumRecordsInLargestCube=','') AS bigint),
    FROM [HFM_ERRORLOG]
WHERE lLogType = 6
ORDER BY dTimeStamp DESC AS tt
CROSS APPLY (SELECT CAST('<e>'+Replace(tt.XML_Info,';', '</e><e>')+'</e>' AS XML) AS XML_Info_XML
CROSS APPLY i.XML_Info_XML.nodes='/') AS n(v)
```

ORDER BY TimeStamp DESC
We now have a query that outputs the data we will use for visualizations.

We can do one of the following:

• Export and save to a spreadsheet. This makes sense for ad-hoc analysis that may require data ‘post-processing’.

• Write the output to a table. This has the benefit of better performance, but requires a process to refresh the data to the output table.

• Create a database view

  *At this time, I have not been able to connect to views in Oracle DV due to a technical issue. In process of investigating.*
To output to a view:

- Add `CREATE VIEW view_name AS (` before the query
- Remove the ORDER BY Statement at the end and close the parenthesis

To output to a table:

- Add `INTO table_name` after the SELECT section and before the FROM
Some areas to explore if your appetite is not satiated:

- **Planning and Essbase log files:**
  These could be scanned and relevant information extracted using a script and written to a database table

- **Server level metrics:**
  Depending on your technology, you can scan services, memory, CPU... and write that to a file or DB table (similar to what HFM SYSINFO provides for the HFM server)

If you are using other systems (ETL, Database), the same approach can be extended to them.
Monitoring EPM systems performance with Oracle Data Visualization

Part 2: How do I connect to my data?
Now that we have our data, we want to connect to it in Oracle DV. Because the process is streamlined and does not depend on the data content, we will look at two examples:

- Connecting to a SQL Database table
- Connecting to an Excel Spreadsheet table

My personal experience is primarily with these two types of connections. Oracle DV offers a number of options, and keeps expanding available sources. Some connectors are in beta release. Please refer to Oracle DV documentation for further information.
Part 2: How do I connect to my data?

Connecting to a SQL Table
Quick recap of what we will use:

- A source HFM database that we have access to (in my case, on SQL Server, but Oracle DV can handle all kinds of databases)
- In this HFM database, a table named vSYSINFO_Full

Over the next slides, we will go through the steps of creating the connection and setting it up for easy building of our visualizations.
Step 1: Creating the connection

In Oracle DV:

- Navigate to “Data Sources”
- Click on Create Connection
Step 1: Creating the connection

- Select the Connection Type: (for me, SQL Server)

- Enter the connection settings:
Connection name: this can be any name you want. Chose something that makes sense for this database. I chose “HFM”.

Host: the name of the database server.

Port: for SQL Server, this is typically 1433. Verify with your database administrator if this does not work.

Username and password: this must be a username and password with access to the database. In my experience, DB accounts work better.

Database name: the name of the database.

If all works well, you will see a message “Connection created successfully”
Part 2 – Connecting to a SQL Table

Oracle DV validates the connection when saving. If the settings are not correct, you will be notified and the connection will not be saved:

![Create Connection](image)

Failed to save the connection.
Invalid Host Name, Port Number or Database Name.
Step 2: Creating the data source

Once we have a connection, we can create a data source:

This can be done from the left pane by clicking on Create Data Source.

It can also be done from the connection list, by clicking on your database connection and selecting Create Data Source from the contextual menu, as well as by clicking on the database connection.
Step 2: Creating the data source

This will bring up the data source creation screen:

The data source creation screen shows the schemas and allows you to drill-down into tables.
Step 2: Creating the data source

By selecting the schema, you will access the list of tables, which will allow you to select the table you want to be a data source:
Step 2: Creating the data source

Once the table is selected, you will see the list of columns and be able to select the ones that are useful (in our case, all, since we use a specially designed table). Selected columns will appear in the right pane.
Step 2: Creating the data source

In the background, Oracle DV creates a SQL query. You can access it and update if you would like to do so:

```
SELECT sServerName, NumDataRecordsInRAM, numHour, numMth, NumRecordsInLargestCube, NumUsers, numWeekDay, sAppName, strDate, Timestamp, UsedCPU, UsedPhysicalMem, UsedVirtualMem FROM dbo.zSYSINFO_Full
```
Part 2 – Connecting to a SQL Table

Step 2: Creating the data source

You can add filters to the data from the filter screen:

You can select any of the columns in scope, and set a filter through the Oracle DV interface.

Once again, this becomes part of the underlying SQL query and can be reviewed and edited there.
Part 2 – Connecting to a SQL Table

Step 2: Creating the data source

Once ready, click on Preview Data to verify the connection:
Step 2: Creating the data source

The next step is to review the different columns settings:

Each column can be set to “Attribute” (a property that can be used for filtering, data grouping…) or a “Measure” (a value that will be graphed).

A measure can be assigned an Aggregation type (Sum, Average, Count…). (This can be later updated).
Step 2: Creating the data source

Finally, set the data source parameters and save the data source:

In particular, you define on this screen whether the data will be refreshed Live, or “Auto”:

- Live ensures that data is always pulled from the Database, but can impact performance
- Auto relies on a cache. This provides better performance but requires refreshes
Step 2: Creating the data source

After saving the data source, you can add derived and calculated columns.

One set of derived columns that can be very useful are the derivations from a timestamp column:

There are more advanced functionalities for creating derived columns, which are not in scope of this presentation.
Monitoring EPM systems performance with Oracle Data Visualization

Part 2: How do I connect to my data?

Connecting to a Spreadsheet file
Part 2 – Connecting to a Spreadsheet file

This process is simpler than connecting to a database. Essentially, we only perform step 2:

In Oracle DV:

- Navigate to “Data Sources”
- Click on Create Data Source
- Select “File”:
  You can also drag an drop a file on this tile
Part 2 – Connecting to a Spreadsheet file

The spreadsheet is loaded and interpreted based on column titles and formats.

As with a database table, you have to review and where needed adjust the column types:
Hyperion Focus 17

Monitoring EPM systems performance with Oracle Data Visualization

Part 3: Let’s visualize!
Monitoring EPM systems performance with Oracle Data Visualization

Part 3: Let’s visualize!

Creating a visualization
With data sources created, we can build visualizations.

We will first go through the steps of creating one, then discuss how the different visualizations can provide us insights into EPM systems performance.

To begin, from the Home page, Click on Create Project
The first step is to add a data source. DV will present the list of defined data sources, select the source or sources that you want to use for the project. You can change sources later. For the purpose of this walkthrough, I will use the example of the Planning Job History.
Once you have selected a data source, you are brought to the project screen. Click on Visualizations to start creating one:

This will bring you to the list of visualizations and allow to select the one you want. Let’s select a Line graph:
Part 3 – Creating a visualization

With the visualization created, select data elements, and set up your visualization by adding data elements by dragging and dropping into the main pane.
Part 3 – Creating a visualization

As you drag and drop, you will see different areas allowing you to define how the data element is used:
Once you have added a data element, it will appear in the left column. You can drag and drop or remove it from there:
We will use the following for this Line graph:

- Execution Time (in seconds. Integer) as Values (Y-Axis)
- START_TIME as Category (X-Axis)
- JOB_NAME, ATTRIBUTE_1, ATTRIBUTE_2, Entity, Product as Filters

With my data set, the output is the following:
While this gives a line graph, let’s consider what it shows: All business rules execution times by date, for all our applications, and all parameters.

We can change a bit the presentation by putting JOB_NAME as Color:
We are now getting into the analytics: this new view allows to identify which business rules have the longest execution time. We can use filters to focus on specific datasets, identify areas of concerns, or patterns.

It is useful to first spend some time exploring your dataset to find the most useful selections, filters and visualization.

As you identify them, create multiple Canvases set to the views that are most useful.

Remember to save your project before closing, or you will lose your work.
Monitoring EPM systems performance with Oracle Data Visualization

Part 3: Let’s visualize!

Visualizing data to understand EPM performance
Part 3 – Visualizing data to understand EPM performance

Count of rules execution and Average time by Application and Cube.

In order to design this, we will add a calculated column to the data source, named Count, containing a fixed value of 1. From the data source, click on the contextual menu of an existing column, and select Create column:

Also verify that the measure Execution time has its aggregation property set to Average:
Part 3 – Visualizing data to understand EPM performance

With the count created, we will create a Pivot time visualization with the following settings:
Understanding which rules are executed most often as well as which rules have the highest average time can provide insights. This is particularly useful as an input to further analyze execution times over time, i.e. as a filter for the chronological view:
Execution time of Calculation scripts over time:

This type of visualization allows to identify oddities, as well as the general evolution of the process. This can allow to take actions before the process evolution causes adverse effects.
Analyzing HFM System Information:
Looking at the correlation, or lack of, between different system metrics can provide insights. For example, in the following comparison, we can see memory usage grow during month end close and correlate to number of records in memory.

In this case, we can see how the limit of physical memory on the server is reached during peak phases and the process relies on Virtual memory.
Part 3 – Visualizing data to understand EPM performance

The definitions of the graphs are as follows:
HFM Consolidation times
In this example, the two visualizations are the same, but for different years, allowing a year over year comparison.
Part 3 – Visualizing data to understand EPM performance
There are many possible visualizations, and which one to use depends on your data and the analysis you are performing.

In real life cases, the order of steps may be the following:

- Identify an area of interest or concern
- Identify datasets that may be relevant to this area
- Obtain data and use visualizations to understand patterns and trends, gaining insights that can lead to appropriate actions or provide evidence in support of requests for resources
Thank you