Agenda

• Consolidation from a Business perspective
• Consolidation from a Technical perspective
• CPU Utilization
• Intercompany
• Database Utilization
• Subcubes
• Rules
• Performance Tips
Who are We?

Jonathan Berry
Founder / CEO


Accelatis: Founded in 2008

Rob Poirier
Director Client Services

22 Years of Hyperion / EPM Experience

Tony Mitri
Architect

20 Years of Hyperion / EPM Experience

Development and Support Team

80 Years of Combined Hyperion Experience

© 2014 Accelatis, LLC
Our Focused Mission

Provide you the needed insight & control into Oracle EPM / Hyperion to efficiently & consistently manage your system to deliver maximum user benefit....
Our Solution

1. Empower Business Users & IT
2. Broad Approach to APM
3. Deep Look into Underlying Systems
Consolidation Process

1. The rule is run for the intersection of the child entity and the Value member that stores the child entity’s default currency (Entity Currency).

2. If the child entity’s AllowAdjs attribute is set to Y, the rule is run for the intersection of the child entity and the Value member that stores adjustments in the child entity’s default currency (Entity Curr Adjs).

3. If the child’s currency differs from the parent’s currency, the rule is run for the intersection of the child entity and the Value member that stores amounts translated to the parent’s home currency (Parent Currency).
4. If a child’s currency differs from the parent’s currency and the rules file contains a Translation rule, Financial Management executes the Translation rule before step 3.

5. If the child’s currency differs from the parent’s currency, and the child entity’s AllowAdjs attribute is set to Y, the rule is run for the intersection of the child entity and the Value member that stores adjustments translated to the parent’s home currency (Parent Curr Adjs).

6. If the parent entity’s AllowAdjFromChildren attribute is set to Y, the Logic rule is run for the intersection of the child entity and the Parent Adjs value.
7. The rule is run for the intersection of the child entity and the Proportion value.

8. The rule is run for the intersection of the child entity and the Elimination value.

9. If the parent entity’s AllowAdjFromChildren attribute is set to Y, the rule is run for the intersection of the child entity and the Contribution Adjs value.

10. For each additional child entity that contains previously unconsolidated data, repeat steps 1 through 9.

11. The rule is run for the intersection of the parent entity and the Value member that stores the parent entity’s default currency.
Consolidation vs Aggregation

- Aggregation is the process of calculating parent total in the HFM application server on the fly *without any action by the user*. Aggregation is performed within the HFM subcube.

- Consolidation is the process moving data across the Entity and value dimensions by physically persisting the calculated data to the database, *it is a user initiated action.*
What Happens Inside HFM?

1. A consolidation is initiated by the user
2. HFM will place a system lock to prevent metadata, rules, security, member lists from being loaded.
3. System prevents starvation
4. Chart logic is performed on the NONE member of the entity dimension.
5. Retrieve all the base level entities for the specified hierarchy and execute chart logic including prior periods with dirty calc status.
5. Start the consolidation process using a bottom-up approach.

1. This entity hierarchy is locked to prevent others consolidations from writing reporting transactions to the DB.
2. Enumerate all the parents under the specified hierarchy starting with the bottom most one.
3. For every parent, enumerate all the children and consolidate them into the parent using the HFM consolidation process described in the beginning.
   1. Each child is locked by HFM as calculation are performed and unlocked when done.
   2. Chart logic and translation are executed on all the children.
   3. The contribution of all the children is moved into the entity currency of the parent.
   4. Save the data to the database in a transactional way. Each subcube in a transaction.
What are my CPUs doing?

Look familiar?
Simple Scenario

- Consolidate hierarchy with 4 base entities, each with 100K records on a machine with 4 Cores
- Depending on the number of threads that are defined for consol, for this example let say that value is 4.
- HFM will load the 4 base entities into memory and will start running “Calculate” on them.
- They will all finish in roughly the same time due to the same number of records.
- All 4 CPUs will have high utilization and then all drop
- HFM can now move up the hierarchy.
HFM will enumerate those ID for those subcubes, they are not loaded into memory yet.

- Depending on the number of threads that are defined for consol, for this example let say that value is 4.
- HFM will load the first 4 subcube into memory from that list into memory and will start running “Calculate” on them.
- As soon as one finishes another will be run from that list, till that list is exhausted.
- At this stage all 4 CPUs should be fully utilized, but might drop when a subcube is loaded for calculation.
- Once the number of subcube is below the number of threads then are assigned for consol, then we will start to see a drop in CPU utilization.
- HFM can’t move up the hierarchy till all the base entities are done.
What are my CPUs doing? (cont)

- The HFM Application may have more impact on the performance profile than infrastructure.

- All children in a hierarchy need to be consolidated before consolidation moves up the hierarchy.
In a multi-CPU machine, children of a hierarchy are consolidated on separate threads and parent is processed when all children threads are complete.

Entities generally do not have equal amounts of data so some will be completed ahead of others. Most of the CPUs will be utilized for a brief amount of time while waiting for the entity with the most records to finish.
Intercompany

- Intercompany - Use rules or native
- Open Data unit - efficiencies gained
- Organizing rules to run IC elim by partner, not by account
HFM is an analytical engine, not a transactional engine

- This causes a lot of grief for DBAs and Analysts who treat it and try to analyze it like something that it is not. I.e: analyzing SQL statements.

All non-aggregated data points are read from and written to the database

- A high-performance database is a critical.
- Verify that your database has the properly allocated amount of memory and that your disk performance is optimal.
- Periodically run Oracle AWT or MSSQL SQL Profiler to help monitor system performance.
Large numbers of records are frequently added/deleted during normal usage

- This effects indexes and thus query performance. Verify that your DBMS performs periodic index statistics updates and/or index rebuilding after high data load cycles to maintain optimal performance.

- If your database is stored on a NAS or other network storage device, the performance of the network will directly affect your database performance. Periodically validate that your network is not becoming overloaded or adversely affected by other issues (resets, re-transmits, etc).
Subcubes

- Subcubes are units of memory inside HFM.
- All data for a given Scenario/Year/Entity/Value are in a single subcube.
- Get an understanding of the size of your subcubes.
- Understand the number of subcubes required for your largest branch.
- Understand the intercompany process.
- If you can get all the subcubes in RAM to allow intercompany to perform quickly there are performance benefits.
- Understand that 2 similar looking grids can perform very differently.

© 2014 Accelatis, LLC
Subcubes

Subcube Population
If #records > 100,000 consider changing org structure

SQL Example:
SELECT lEntity, lValue, count(*) as numRecs
FROM COMMA_DCE_1_2011
GROUP BY lEntity, lValue
Capturing Subcube Populations

select lEntity, lValue, COUNT (*) FROM <CURRENCY_TABLE>
group by lEntity, lValue

<table>
<thead>
<tr>
<th>Entity</th>
<th>Value</th>
<th>NumRecords</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>34</td>
</tr>
</tbody>
</table>

select lEntity, IParent, lValue, COUNT (*) FROM <NODE_TABLE>
group by lEntity, IParent, lValue

<table>
<thead>
<tr>
<th>Entity</th>
<th>Parent</th>
<th>Value</th>
<th>NumRecords</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>8</td>
<td>10</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>11</td>
<td>34</td>
</tr>
</tbody>
</table>

• Need to convert IDs to Metadata Labels
• Need to compare Before and After Consolidation
1. Multiple VBScript engines
   • Do not use global variables as counters
   • Total time calculated may be more than actual elapsed time due to multiple threads

2. Performance Impact of Instrumentation
   • If tracking function time, write data outside of timings
   • If using Objects, make calls asynchronous so writing of data happens in queue on background thread. Do not create thread for every call!

3. Eliminate unnecessary calls to routines from global scope. This can greatly increase load time and application startup time
4. Writing to file from multiple VBScript engines is dangerous.
   • Each instance may block access to the file and cause deadlocks as well as major performance issues.
   • Doing so effectively forces HFM Consolidation into single threaded operation.

5. Resolution of timing using the Timer function in VBScript is \( \geq 15 \text{ms} \). Using this built-in timing function can amplify timing calculation errors over many iterations.
Performance Tips
The HFM Rules Profiler is an automated tool for instrumenting, profiling and analyzing HFM Consolidation performance.

- Overall Inclusive Function Time
- Most Time Consuming Inclusive Functions
- Overall Exclusive Function Time
- Most Time Consuming Exclusive Functions
- Function Call Frequency
- Most Frequently Called Functions
Tip 1: OpenDataUnit

OpenDataUnit is powerful and efficient mechanism for retrieving data.
Resist using ‘for-next’ loops instead of OpenDataUnit to do calculation on the account dimension.

```
AccountList = HS.Account.List (""", "]")
for each Account in AccountList
  ‘do work
next
```
Use OpenDataUnit instead for the same task

Set MyDataUnit = HS.OpenDataUnit(""")
INumItems = MyDataUnit.GetNumItems
for i = 0 to INumItems-1
    ‘do work
next
Tip 2: Sub Calculate

- Sub Calculate is executed about 8 times for each entity and currency
- Limit the rules execution on that function
- Some data manipulation needs to occur only on the default currency, so adding that check in your if statement is a key to improving performance
Resist logic against the whole Value dimension

```
If HS.Entity.IsBase("","")  = True
   'do work
End if
```
Specify Entity and Value dimensions when possible

If HS.Entity.IsBase("","") = True And HS.Value.Member = <Entity Currency>
  ‘do work
End if
Tip 3: HS.EXP Resolution

HS.EXP is mechanism for copying blocks of data within a Subcube in an application.
Tip 3: HS.EXP Resolution (cont)

Resist implicitly copying all 4 custom data values from one account to the other.

HS.Clear “A#DestAccount”
HS.Exp “A#DestAccount = A#SourceAccount”
Tip 3: HS.EXP Resolution (cont)

Specify N custom data values between Accounts

HS.Clear “A#DestAccount.C1#DestC1”
HS.Exp “A#DestAccount.C1#DestC1 = A#SourceAccount.C1#SourceC1”
Tip 3: HS.EXP Data Explosion

The following causes data explosion by specifying Account on right side and not the left side.

```
HS.Clear "A#Sales.C4#Movement"
HS.Exp "C4#Movement = A#Sales.C4#Movement + A#Sales.C4#Increases - A#Sales.C4#Decreases"
```

© 2012 Accelatis, LLC
Tip 5: Global Logic

- Code at global scope is executed when applications load and when rules are loaded.
- Errors in code at global scope will not be reported at load time and may go unnoticed until consolidation failures result.

Example (code in RED is at global scope)

```vba
'Create global arrays
Global_Acclist_Statistical = HS.Account.List(,,"Statistical")
Global_Acclist_Validations = HS.Account.List("Validations","[Base]")
Global_Acclist_CAccounts = HS.Account.List("C_Validations","[Base]")

Sub NoInput()
    ' Make all input cells for the Budget Scenario, Year 2004, Sales account calculated (prohibit input)
    HS.NoInput "S#Budget.Y#2004.A#Sales"
    HS.NoInput "S#Budget.Y#2004.A#Adminexpenses"
    HS.NoInput "S#Budget.Y#2003.A#Adminexpenses"
End Sub

*** c1 list for security class C1_CORP
Global_C1_List = HS.Custom1.List("TotC1","[Base]")
For each StrMember in Global_C1_List
    If UCASE(HS.Custom1.SecurityClass(StrMember)) <> "C1_CORP" then
        aC1_CORP = ExactFilter(aC1_CORP,StrMember,False)
    End If
Next

Sub Consolidate()
    Set MyDataUnit = HS.OpenDataUnit(")
    dPCon = HS.Node.PCon(")
    lNumItems = MyDataUnit.GetNumItems
End sub
```
Tip 5: Global Logic (cont)

Performance times to open an application and load rules on sample application affected greatly by global logic to build member lists.

![Bar Chart]

- App Load Time: With 240, Without 30
- Rules Load Time: With 270, Without 10

© 2014 Accelatis, LLC
Questions?

Jonathan Berry  
President & CEO  
jberry@accelatis.com

Rob Poirier  
Director of Client Services  
rpoirier@accelatis.com