

# 11.1.2 ASO

## What you need to know

Jon Harvey, Sr. Management Consultant  
03/16/11



# Agenda

- Meet ASO – An intro
- When and how ASO should be used
- ASO Primer – What is different from BSO
- Important performance considerations when using ASO
- Performance improvements realized by ASO cubes
- “Calc functionality is limited in ASO” is a myth
- Where ASO is headed
- Q&A



# An Intro to ASO

---



# Meet ASO

- What is ASO?
  - A different type of cube available in Essbase designed for higher volume and aggregation speed
- What is MDX?
  - The calculation language used by ASO
- How does ASO relate to MDX?
  - Member formulas, no calcs in ASO



# When and How to use ASO

---



# When and how should ASO be used?

- ALWAYS!
- (unless...)
  - It's a Planning application
  - You have calcs that require looping
  - Tops down forecasting
  - Multi-step allocations, high level to low level



# Some reasons people ARE using ASO

- Volume of data
  - ASO engine aggregates lightning fast
  - Allows for parallel data loads
- “Incremental system” system
  - Ideal for a G/L or transactional data
- In tandem with Planning for reporting integration
  - Planning still can't use ASO as a backend



# An ASO Primer

---



# Hierarchies

- Stored
  - ‘+’ aggregation operator only
  - Doesn’t allow shared members in the hierarchy
  - Fastest hierarchy type
- Dynamic
  - Allows all aggregation operators
  - Allows all TimeBalance options
  - Allows shared members in the hierarchy



# Dimensions

- Stored
  - Entire dimension is one single Stored hierarchy
- Dynamic
- Multiple hierarchies
  - Leverage the speed of Stored where you can, get the flexibility of Dynamic where you need it
  - Effectively declares hierarchy root one generation lower than dimension root



# When are you required to use Dynamic hierarchy?

- You are using Shared members
- You have MDX calculations
- You want more than the '+' aggregation operator
- You want to use TimeBalance properties



# Compression Dimensions

- Not required to declare one (but you should)
- Improves performance
- Must be a Dynamic dimension
- Not always, but typically Time or Account



# Performance Considerations

---



# Optimal ASO Outlines

- This dimension has to be Dynamic...

☐ Demo Dimension Dynamic <1>

☐ Net Income (+) <2>

☐ Gross Margin (+) <2>

.....Sales (+)

.....COGS (-)

☐ Expenses (-) <3>

.....Marketing (+)

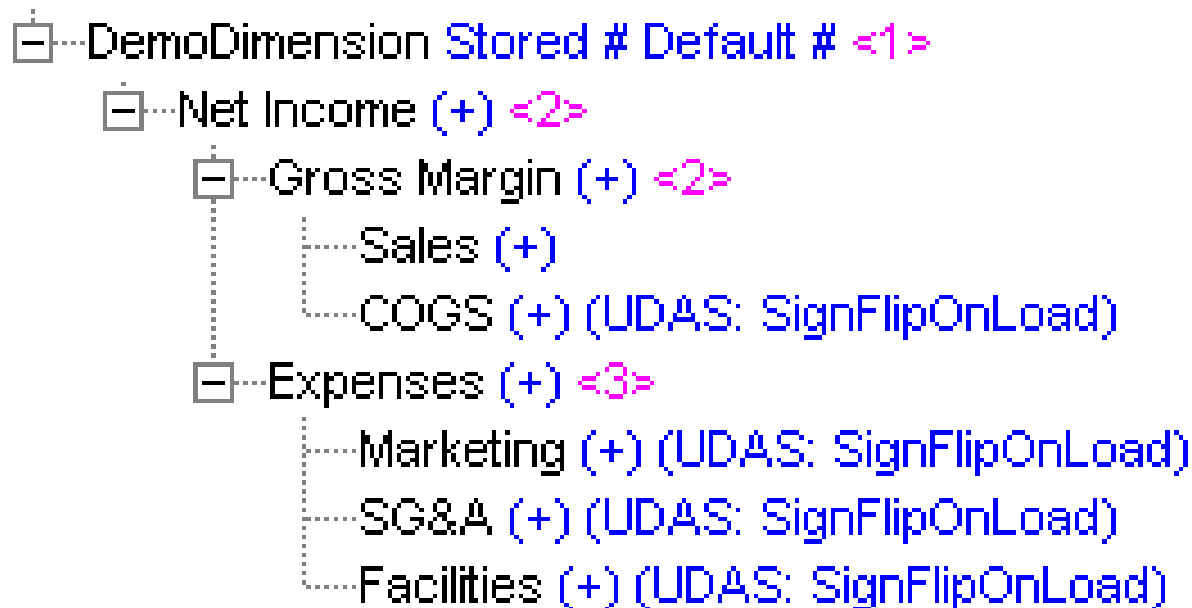
.....SG&A (+)

.....Facilities (+)



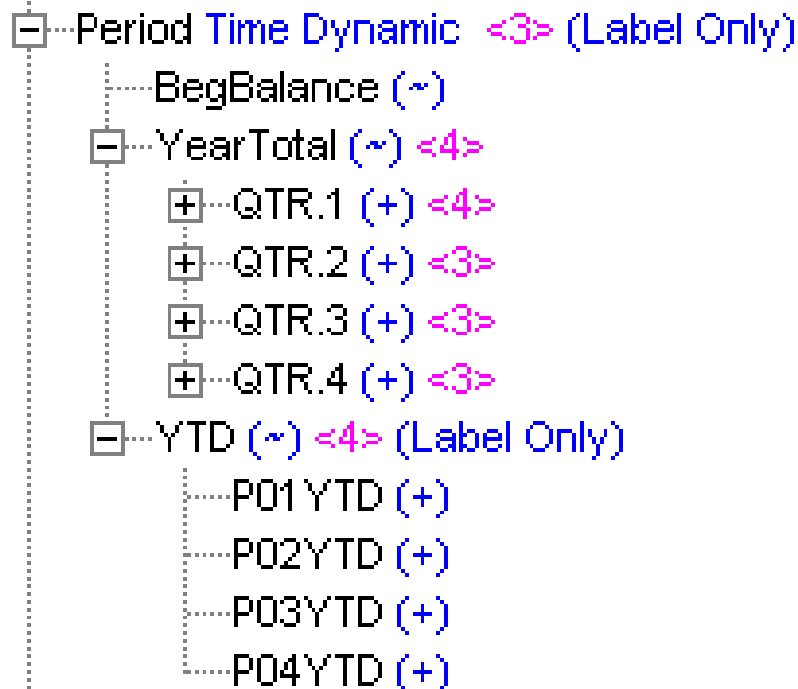
# Optimal ASO Outlines

- Or does it? Consider the following:

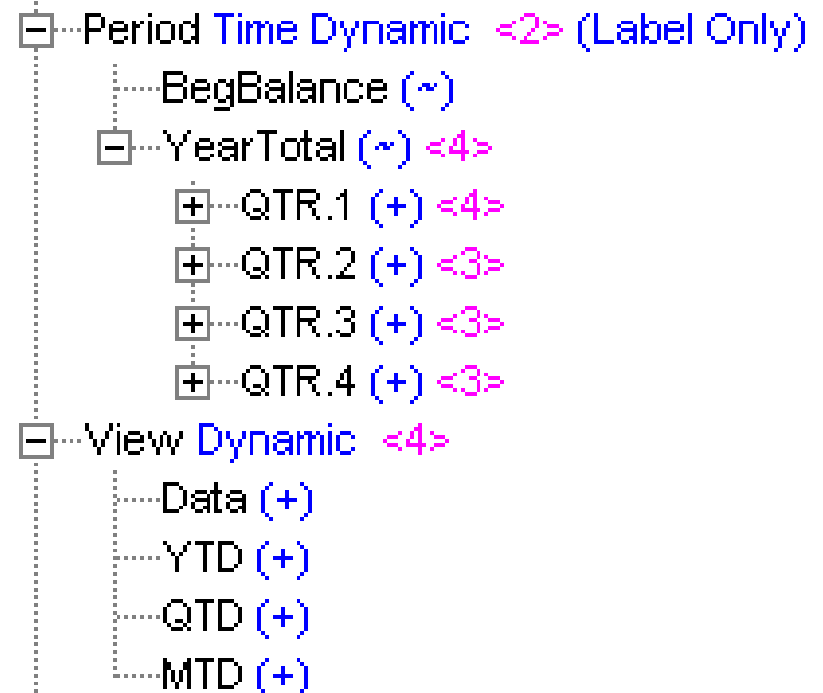


# Time Dimension

- Pre 9.3.1 recommended



- Post 9.3.1 recommended



# Optimizing MDX Calcs

- Use the provided Time/Period functions
- Use the NON EMPTY function
- Avoid CURRENTMEMBER references
- Use the built in calculations when possible
  - TimeBalance, Ranking, etc
- We'll cover in more detail in the calcs section



# Improvements Realized

---



# Performance Improvements

- Situation
  - GL Restructure Project
- Technical Requirement
  - 14 dimensional Planning cubes
- Results
  - BSO aggregation: 8 or more hours
  - ASO aggregation: 6 minutes
    - Outline build, data load AND aggregation



UnitedHealth Group®



# Performance Improvements

- Client: A \$3B food manufacturing firm in the Midwest
- Project: Operational Reporting cube
- Technical Requirements
  - 250+ MDX calcs
  - Very complex calcs (some hit character limit)
- Results
  - ASO aggregation and calculation: around 1 minute



# Debunking the myth – Calcs in ASO



# Debunking the myth – ASO calcs

BSO

ASO

Calcs/BRs

Member Formulas ONLY

FIX(), IF() statements

Case When statements

Two Pass

Solve Order



# Debunking the myth – ASO calcs

BSO

“mbr1”

“mbr1”->“mbr2”

(“mbr1”, “mbr2”)

ASO

[mbr1]

( [mbr1], [mbr2] )

{ [mbr1], [mbr2] }

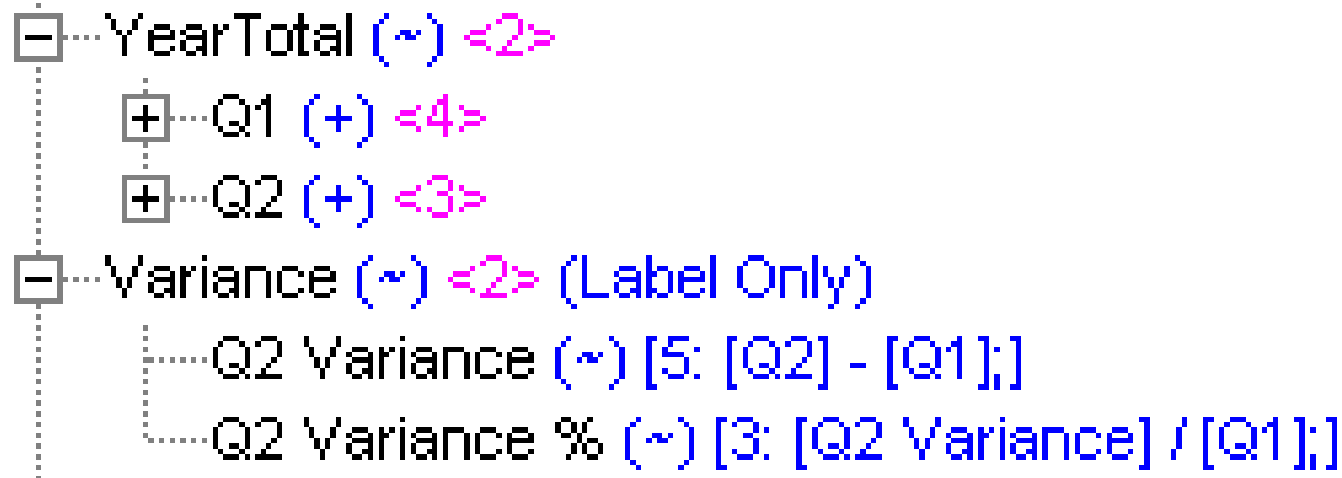


# Solve Order

- Replaces Two-Pass
- Defines the order of the calculations
- Is a value between 0 and 127
- Solve order solves BOTTOM UP
- Best practice
  - Put gaps between your solve order calcs
  - 10...20...30...40



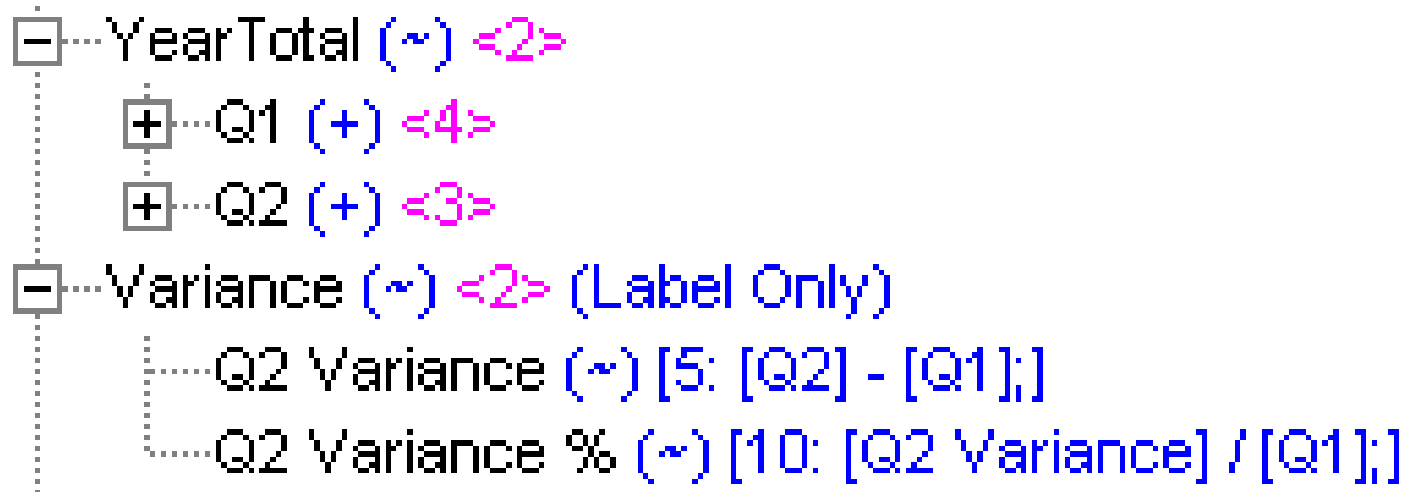
# Solve Order (example)



	Q1	Q2	Q2 Variance	Q2 Variance %
<b>(Finance)</b>	746,635	1,788,088	1,041,453	58.24%
<b>(Info Tech)</b>	563,120	1,264,742	701,622	55.48%
<b>(Total Expenses)</b>	1,309,755	3,052,831	1,743,075	113.72%



# Solve Order (example)



	Q1	Q2	Q2 Variance	Q2 Variance %
<b>(Finance)</b>	746,635	1,788,088	1,041,453	58.24%
<b>(Info Tech)</b>	563,120	1,264,742	701,622	55.48%
<b>(Total Expenses)</b>	1,309,755	3,052,831	1,743,075	57.10%



# MDX Member Set Functionality

- Allow for more precise, granular selections
  - OpeningPeriod, ClosingPeriod
  - PeriodsToDate
  - LastPeriods
  - FirstChild, LastChild
  - FirstSibling, LastSibling
- Use instead of LEAD, LAG, PARENT, CHILD



# MDX Member Set Functionality

- Consider the following:

```
Round(  
  (([Time].CurrentMember, [widget_purchases]) +  
  ([Time].CurrentMember.Lag(1), [widget_purchases]) +  
  ([Time].CurrentMember.Lag(2), [widget_purchases])) /  
  
  (([Time].CurrentMember, [total_dept], [total_expense]) +  
  ([Time].CurrentMember.Lag(1), [total_dept], [total_expense]) +  
  ([Time].CurrentMember.Lag(2), [total_dept], [total_expense]))  
,2)
```



# MDX Member Set Functionality

- Consider the following:

$(([\text{Time}].\text{CurrentMember}, [\text{widget\_purchases}]) +$   
 $([\text{Time}].\text{CurrentMember}.\text{Lag}(1), [\text{widget\_purchases}]) +$   
 $([\text{Time}].\text{CurrentMember}.\text{Lag}(2), [\text{widget\_purchases}])) /$

$(([\text{Time}].\text{CurrentMember}, [\text{total\_dept}], [\text{total\_expense}]) +$   
 $([\text{Time}].\text{CurrentMember}.\text{Lag}(1), [\text{total\_dept}], [\text{total\_expense}]) +$   
 $([\text{Time}].\text{CurrentMember}.\text{Lag}(2), [\text{total\_dept}], [\text{total\_expense}]))$



# MDX Member Set Functionality

- Could be rewritten as:

SUM( LastPeriods(3), [widget\_purchases] ) /  
SUM( LastPeriods(3), ([total\_dept], [total\_expense]) )



# NON EMPTY Functions

- Limits calcs to only cells that have data

	A	B	C	D	E	F
1		FY10	Promo	Forecast	Web Channel	Qtr.1
2						
3		Units	Avg Units/Transaction			
4	BusinessUnit1	#Missing	#Missing			
5	BusinessUnit2	#Missing	#Missing			
6	BusinessUnit3	4545	1.205			
7	BusinessUnit4	4100	1.187			
8	BusinessUnit5	#Missing	#Missing			
9	BusinessUnit6	#Missing	#Missing			
10	BusinessUnit7	#Missing	#Missing			
11	BusinessUnit8	#Missing	#Missing			
12	BusinessUnit9	3400	1.321			



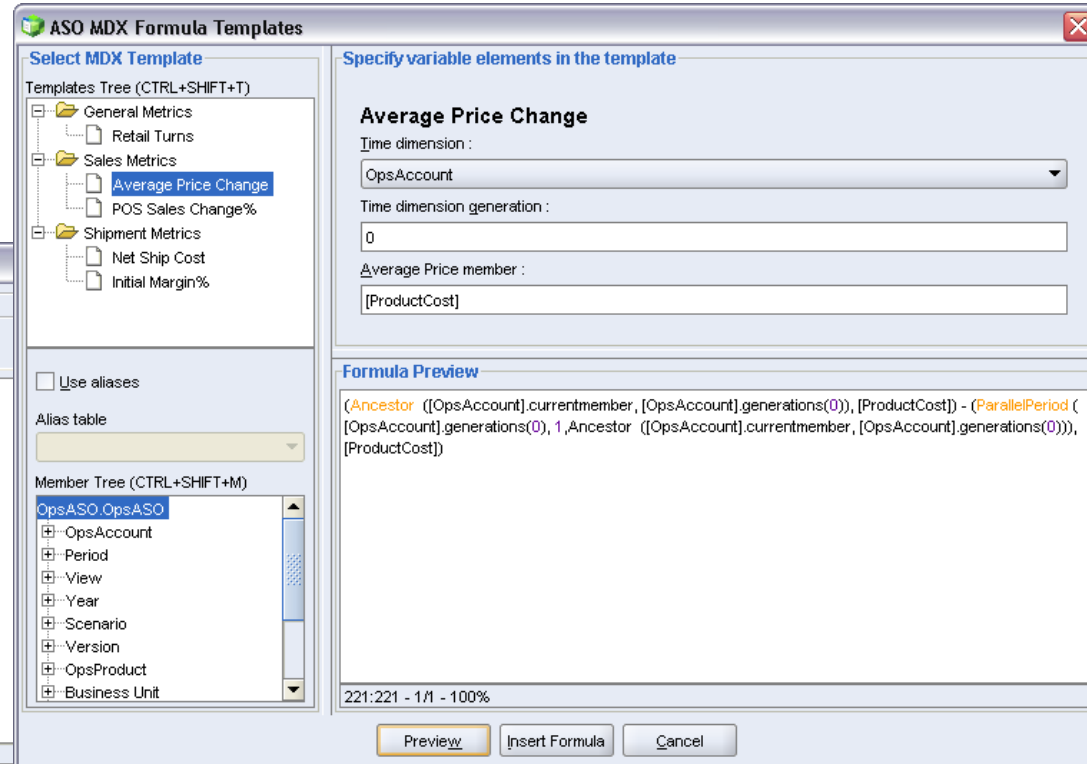
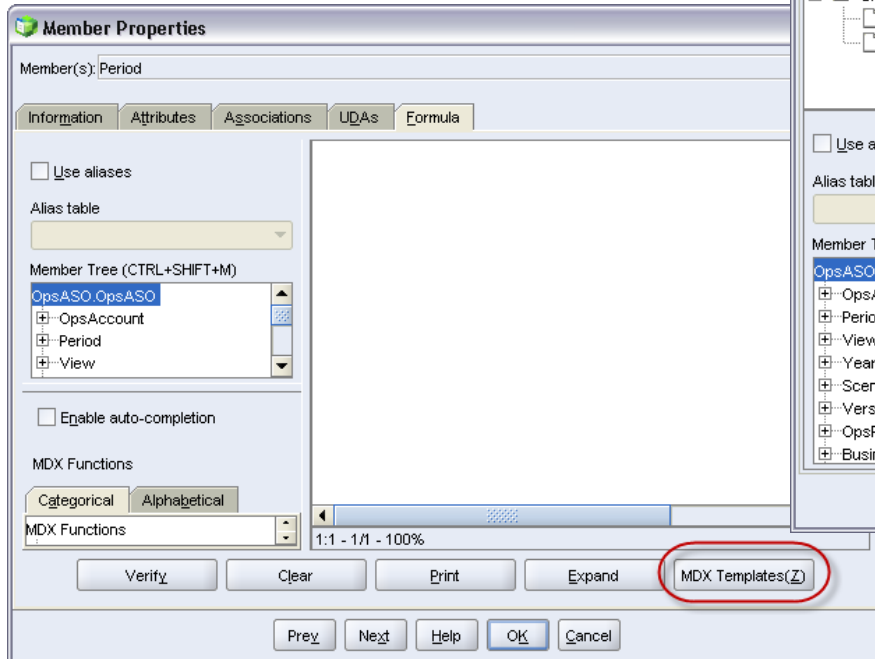
# Where ASO is heading

---



# Formula Templates

- What are they
- How to access them



# ASO Reporting Integration

- Map the planning application to reporting cube
  - Essbase BSO and Essbase ASO
  - Map dimensions to dimension
  - Map smart lists to dimension
  - Filtering data to be transferred

Planning Application			Reporting Application (PSB_ASO)
Mapping Type	Dimension / Smart List Name	Member Selection	Dimension Name
Dimension to Dimension	Account	Salary Element Cost,Business Unit Segm	Account
Dimension to Dimension	Period	ILvl0Descendants(Period)	Period
Dimension to Dimension	Year	ILvl0Descendants(Year)	Year
Dimension to Dimension	Scenario	Budget	Scenario
Dimension to Dimension	Currency	ILvl0Descendants(Currency)	Currency
Dimension to Dimension	Entity	ILvl0Descendants(Entity)	Entity
Smart List to Dimension	Business_Unit_List	Business Unit Segment	BusinessUnit
Smart List to Dimension	Fund_list	Fund Segment	Fund
Smart List to Dimension	Program_list	Program Segment	Program
Smart List to Dimension	Project_list	Project Segment	Project
Dimension to Dimension	Select Dimension	ILvl0Descendants(Employee)	Employee
Dimension to Dimension	Version	Governor_8	Version

Help Save Save As... Back Next Cancel



# Allocations in MDX

- Possible in 11.1.2
- Currently no interface available, run from MaxL
- Requires high level understanding of MDX member sets
- Worth looking into – very fast

```
execute allocation process on TestASO.TestDB
pov "CrossJoin({Descendants([Departments],[Departments].Levels(0)),
    {Descendants([Process],[Process].Levels(0))})"
amount "([Actual],[Total.Region],[&CurrYear])"
target ""
range "CrossJoin({[Sales]},
    {Descendants([Locations],[Locations].Levels(0))})"
spread;
```



# MDX as a Query Tool

- The problem: How to get data out of ASO cubes?
  - In BSO, you have options
    - Report Scripts
    - Level0 Columnar
    - DATAEXPORT Scripts
  - In ASO?



# MDX as a Query Tool

- Possible Solutions:
  - Report Scripts
    - Never Die Right
    - “Questionable” Performance
    - ANOTHER language?
      - Syntactically... well, ugly
  - ETL
    - Learning curve
    - Cost money



# The Catch

- Process has one major drawback
- We want to script this up, but how?
  - MaxL
    - Runs right from the shell
    - Output with spool command
    - Output is ugly
      - (Unusable?)



# MDX as a Query Tool

- ETL
  - (+) Great tools, tons of functionality
  - (-) Costs money, Learning curve
- MDX
  - (+) FREE (well, included anyway)
  - (+) Fast, Flexible
  - (-) Output is “not pretty”



Questions?

