

Formula Optimization

A guide on identifying and optimizing Anaplan formulas



Mark Warren

Manager – Operational Excellence Group

/anaplan

“

**3 easy steps to follow if you
want to improve your
model performance”**

Golden Rules

Calculate once and refer many times

- Avoid repetition!
- Why calculate the same thing in multiple places when we can avoid it
- Use System modules – learn how to DISCO

Calculate at lowest cell count possible

- Why calculate more than we need to?
- Calculate over just the dimension the formula applies to
- Break up formulas to calculate at the right cell count for each part

Test as you go!

- Spotting poor performance early makes it easier to address
- Test different ways of achieving the same goal
- Test and prove optimizations

Calculate once and refer many times



Repetition

- **Where it occurs**
 - Generally find it within the same module
 - Between similar modules and functional areas
 - Simple common elements like list items and dates
 - Conditional checks are a common area
- **Why is it bad?**
 - Because it is a calculation we can avoid
 - They add up, increasing overall calculation time
 - Can lead to more recalculation

Calculate once and refer many times



Common example of repeated functions

- ITEM(X)
- NAME(ITEM(X))
- CODE(ITEM(X))
- PARENT(ITEM(X))
- START()
- END()
- ITEM(Time) = TIME.'Current Period'
- CURRENTPERIODSTART() < START()
- Boolean checks in IF statements

Calculate once and refer many times

| Formula | Parent | Is Summary | Format | Organization, Hou |
|---|--------------------|-------------------------------------|--------|-------------------|
| START() > CURRENTPERIODSTART() THEN 0 ELSE IF Call Center Forecast | 1 | <input type="checkbox"/> | Number | - |
| IF START() > CURRENTPERIODSTART() THEN 0 ELSE IF Call Center Forecast | | <input type="checkbox"/> | Number | - |
| "Current Period".Actuals Through - START() + 1 | | <input type="checkbox"/> | Number | - |
| IF START() > CURRENTPERIODSTART() THEN 0 ELSE IF Import Call Data.Inbound Y_Calls | 2 | <input type="checkbox"/> | Number | - |
| IF Y_Calls < 0 THEN 0 ELSE 0 | | <input type="checkbox"/> | Number | - |
| CUMULATE(1) | | <input type="checkbox"/> | Number | - |
| IF Y_Calls < 0 THEN CUMULATE("1_Max_Calls") ELSE 0 | X_Total_Calls | <input type="checkbox"/> | Number | - |
| X_Max_Calls | | <input checked="" type="checkbox"/> | Number | - |
| POWER(X_Max_Calls, 2) | | <input type="checkbox"/> | Number | - |
| Total Actual Calls | | <input checked="" type="checkbox"/> | Number | - |
| X_Max_Calls * Y_Calls | | <input type="checkbox"/> | Number | - |
| (X_Max_Calls[SELECT: TIME.All Periods] * XY_Calls[SELECT: TIME.All Periods]) | | <input type="checkbox"/> | Number | - |
| (Y_Calls[SELECT: TIME.All Periods] - m_Calls * X_Total_Calls[SELECT: TIME.All | | <input type="checkbox"/> | Number | - |
| m_Calls * X_All_Calls + c_Calls | | <input type="checkbox"/> | Number | - |
| IF START() <= CURRENTPERIODSTART() THEN Y_Calls ELSE MOVINGSUM(") | | <input type="checkbox"/> | Number | - |
| IF CUMULATE(1) = 1 THEN Y_Calls ELSE Trend Line Method.a for Exponential t | | <input type="checkbox"/> | Number | - |
| IF START() <= CURRENTPERIODSTART() + 1 AND END() > CURRENTPERIOD | | <input type="checkbox"/> | Number | - |
| IF START() <= CURRENTPERIODSTART() THEN Total Actual Calls ELSE IF ST/ | | <input type="checkbox"/> | Number | - |
| IF START() > CURRENTPERIODSTART() THEN 0 ELSE IF Import Call Data.Average Y_Avg Handle Time | 3 | <input type="checkbox"/> | Number | - |
| IF Y_Avg Handle Time < 0 THEN 0 ELSE 0 | | <input type="checkbox"/> | Number | - |
| CUMULATE(1) | | <input type="checkbox"/> | Number | - |
| IF Y_Avg Handle Time < 0 THEN CUMULATE("1_Max_Avg Handle Time") ELSE X_Total_Avg Handle | X_Total_Avg Handle | <input type="checkbox"/> | Number | - |
| X_Max_Avg Handle Time | | <input checked="" type="checkbox"/> | Number | - |

Calculate once and refer many times

Use a Time Management Module

Future Period: START() > CURRENTPERIODSTART()

Time Management

| | W/c 1 Jan 18 | W/c 8 Jan 18 | W |
|-----------------|-------------------------------------|-------------------------------------|-------------------------------------|
| Current Period? | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Future Period? | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| History Period? | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |

Cell Count = 105

VS

280,908 x 3
= 842,724

**99.9%
reduction in
calculated
cells**

| | Formula |
|--|---|
| Call Data Forecast | |
| Override Call Flag | IF Time Management.Future Period? THEN 0 ELSE IF Call Ce |
| Override Avg Handle Time Flag | IF START() <= CURRENTPERIODSTART() THEN 0 ELSE IF C |
| Distance from Actuals | 'Current Period'.Actuals Through - START() + 1 |
| - | |
| Total Actual Calls | IF Time Management.Future Period? THEN 0 ELSE Import Ca |
| 1_Max_Calls | IF Y_Calls <> 0 THEN 1 ELSE 0 |
| X_All_Calls | CUMULATE(1) |
| X_Max_Calls | IF Y_Calls <> 0 THEN CUMULATE('1_Max_Calls') ELSE 0 |
| X_Total_Calls | X_Max_Calls |
| X_Sqr_Calls | POWER(X_Max_Calls, 2) |
| Y_Calls | Total Actual Calls |
| XY_Calls | X_Max_Calls * Y_Calls |
| m_Calls | (X_Max_Calls[SELECT: TIME.All Periods] * XY_Calls[SELECT |
| c_Calls | (Y_Calls[SELECT: TIME.All Periods] - m_Calls * X_Total_Calls) |
| E(Y) - Straight Line_Calls | m_Calls * X_All_Calls + c_Calls |
| E(Y) - Moving Average_Calls | IF START() <= CURRENTPERIODSTART() THEN Y_Calls EL |
| E(Y) - Exponential Smoothing_Calls_Formula | IF CUMULATE(1) = 1 THEN Y_Calls ELSE Trend Line Method |
| E(Y) - Exponential Smoothing_Calls | IF START() <= CURRENTPERIODSTART() + 1 AND END() > 1 |
| Total Inbound | IF START() <= CURRENTPERIODSTART() THEN Total Actual |
| -- | |
| Total Actual Avg Handle Time | IF Time Management.Future Period? THEN 0 ELSE Import Ca |
| 1_Max_Avg Handle Time | IF Y_Avg Handle time <> 0 THEN 1 ELSE 0 |
| X_All_Avg Handle Time | CUMULATE(1) |
| X_Max_Avg Handle Time | IF Y_Avg Handle Time <> 0 THEN CUMULATE('1_Max_Avg H |

Calculate at lowest cell count possible



A common example is adding text together to form a unique code

In this case we are adding a Company code to Product code with an underscore separator

| | Formula | Parent | Is Summary | Format | Applies To | Time Scale |
|--------------------|--|--------|--------------------------|--------|------------------|------------|
| REP01 Data Summary | | | | | Company, Product | Month |
| Code | <code>CODE(ITEM(Company)) & "_" & CODE(ITEM(Product))</code> | | <input type="checkbox"/> | Text | - | Month |

There are two text additions here (two &'s) done at a combined cell count of **128,895,624**

Calculate at lowest cell count possible

- **Product** has 3531 items
- **Company** has 1014 items

So we do one text addition at the lowest cell count possible with no timescale

| | Formula | Parent | Is Summary | Format | Applies To | Time Scale |
|-----------------------|---------------------|--------|--------------------------|--------|------------|----------------|
| SYS05 Company Details | | | | | | |
| Code | CODE(ITEM(Company)) | | <input type="checkbox"/> | Text | - | Not Applicable |
| Code to Use | Code & " _" | | <input type="checkbox"/> | Text | - | Not Applicable |
| SYS06 Product Details | | | | | | |
| Code | CODE(ITEM(Product)) | | <input type="checkbox"/> | Text | - | Not Applicable |

Note that both are done in System modules for each list (DISCO)

The main formula is now



| | Formula | Parent | Is Summary | Format | Applies To | Time Scale |
|--------------------|--|--------|--------------------------|--------|------------|------------|
| REP01 Data Summary | | | | | | |
| Code | 'SYS05 Company Details'.Code to Use & 'SYS06 Product Details'.Code | | <input type="checkbox"/> | Text | - | Month |

Calculate at lowest cell count possible

We can take it a step further here...
The formula does not apply to time

| Jan 14 | Feb 14 | Mar 14 | Apr 14 | May 14 | Jun 14 |
|----------|----------|----------|----------|----------|----------|
| 840_1274 | 840_1274 | 840_1274 | 840_1274 | 840_1274 | 840_1274 |
| 840_5337 | 840_5337 | 840_5337 | 840_5337 | 840_5337 | 840_5337 |
| 840_6990 | 840_6990 | 840_6990 | 840_6990 | 840_6990 | 840_6990 |
| 840_4184 | 840_4184 | 840_4184 | 840_4184 | 840_4184 | 840_4184 |
| 840_1501 | 840_1501 | 840_1501 | 840_1501 | 840_1501 | 840_1501 |
| 840_1580 | 840_1580 | 840_1580 | 840_1580 | 840_1580 | 840_1580 |
| 840_8034 | 840_8034 | 840_8034 | 840_8034 | 840_8034 | 840_8034 |
| 840_1900 | 840_1900 | 840_1900 | 840_1900 | 840_1900 | 840_1900 |

Doing the calculation without a timescale reduces cell count to 3,580,434

| | Formula | Parent | Is Summary | Format | Applies To | Time Scale |
|--------------------|--|--------|--------------------------|--------|------------------|----------------|
| REP01 Data Summary | | | | | Company, Product | Month |
| Code Optimised |  'SYS05 Company Details'.Code to Use & 'SYS06 Product Details'.Code | | <input type="checkbox"/> | Text | - | Not Applicable |
| Code |  Code Optimised | | <input type="checkbox"/> | Text | - | Month |

Calculate at lowest cell count possible



- The first example calculation took 5.68 sec
 - (two text additions at 128,895,624 cells)
- The last example took 0.52 sec
 - (one text addition at 3,580,434 cells)
- **91%** reduction in duration

Demo

1. Identifying large formulas
2. How to edit large formulas
3. Splitting up formulas to **reduce repetition and complexity**
4. Testing the optimizations
5. Performance analysis

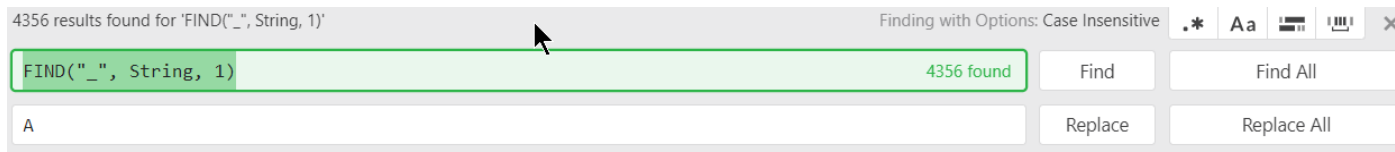
- [illegible]

Demo

- Finding and highlighting repeated parts in the text editor

```
(IF ISNOTBLANK(LEFT(String, FIND("_", String, 1) - 1)) THEN LEFT(String, FIND("_", String, 1) - 1) ELSE String) & (IF LE  
LENGTH(LEFT(String, FIND("_", String, 1) + 1))) - LENGTH(LEFT(String, FIND("_", String, 1))) < 0 THEN 1000 ELSE FIND("_"  
LENGTH(LEFT(String, FIND("_", String, 1))) < 0 THEN 1000 ELSE FIND("_", String, LENGTH(LEFT(String, FIND("_", String, 1)  
FIND("_", String, LENGTH((IF ISNOTBLANK(LEFT(String, FIND("_", String, 1) - 1)) THEN LEFT(String, FIND("_", String, 1) -  
FIND("_", String, LENGTH(LEFT(String, FIND("_", String, 1) + 1))) - LENGTH(LEFT(String, FIND("_", String, 1))) < 0 THEN  
LENGTH(LEFT(String, FIND("_", String, 1))) < 0 THEN 1000 ELSE FIND("_", String, LENGTH(LEFT(String, FIND("_", String, 1)  
THEN 1000000 ELSE FIND("_", String, LENGTH((IF ISNOTBLANK(LEFT(String, FIND("_", String, 1) - 1)) THEN LEFT(String, FIND  
String, 1) + 1, IF FIND("_", String, LENGTH(LEFT(String, FIND("_", String, 1) + 1))) - LENGTH(LEFT(String, FIND("_", Str  
String, 1) + 1))) - LENGTH(LEFT(String, FIND("_", String, 1))) < 0 THEN 1000 ELSE FIND("_", String, LENGTH(LEFT(String,  
1))) - 1)) + 2)) = BLANK THEN BLANK ELSE " " & MID(String, LENGTH((IF ISNOTBLANK(LEFT(String, FIND("_", String, 1) - 1))  
MID(String, FIND("_", String, 1) + 1, IF FIND("_", String, LENGTH(LEFT(String, FIND("_", String, 1) + 1))) - LENGTH(LEFT
```

- Using the Find or Count function of the editor to look for part with most repetitions
- Replacing that part with a reference to a new line item named A



Demo

- Repeating until all repetition is removed...

| | | | | | |
|---------------------|---|--|--------------------------|--------|---|
| Optimised Formula | C & D & G & H & K & M | | <input type="checkbox"/> | Text | - |
| Substitute Function | SUBSTITUTE(String, "_", "") | | <input type="checkbox"/> | Text | - |
| A | FIND("_", String, 1) | | <input type="checkbox"/> | Number | - |
| B | FIND("_", String, LENGTH(LEFT(String, A + 1))) - LENGTH(LEFT(String, A)) | | <input type="checkbox"/> | Number | - |
| C | IF ISNOTBLANK(LEFT(String, A - 1)) THEN LEFT(String, A - 1) ELSE String | | <input type="checkbox"/> | Text | - |
| D | IF LENGTH(MID(String, A + 1, IF B < 0 THEN 1000 ELSE B)) = 0 THEN BLANK ELSE | | <input type="checkbox"/> | Text | - |
| E | LENGTH(C & D & MID(String, A + 1, IF B < 0 THEN 1000 ELSE B - 1)) + 2 | | <input type="checkbox"/> | Number | - |
| F | MID(String, E, IF FIND("_", String, E) - E <= 0 THEN 1000000 ELSE FIND("_", String, E)) | | <input type="checkbox"/> | Text | - |
| G | MID(String, A + 1, IF B < 0 THEN 1000 ELSE B - 1) | | <input type="checkbox"/> | Text | - |
| H | IF F = BLANK THEN BLANK ELSE " " & F | | <input type="checkbox"/> | Text | - |
| I | LENGTH(C & D & G & H) + 2 | | <input type="checkbox"/> | Number | - |
| J | MID(String, I, IF FIND("_", String, I) - I <= 0 THEN 1000000 ELSE FIND("_", String, I)) | | <input type="checkbox"/> | Text | - |
| K | IF J = BLANK THEN BLANK ELSE " " & J | | <input type="checkbox"/> | Text | - |
| L | MID(String, LENGTH(C & D & G & H & K) + 2, 10000) | | <input type="checkbox"/> | Text | - |
| M | IF L = BLANK THEN BLANK ELSE " " & L | | <input type="checkbox"/> | Text | - |

Demo

- Calculation analysis – comparison of calculation times before and after
- Test by comparing formulas and how long did it take before and after

Calculations by Line Item

| | Module ▾ | Line Item ▾ | Total calculation time (ms) ▾ | % of total calculation time ▾ |
|----|------------|---------------------|-------------------------------|-------------------------------|
| 1 | Complexity | Original Formula | 1,170,317.36 | 96.93 |
| 2 | Complexity | B | 5,580.09 | 0.46 |
| 3 | Complexity | F | 5,160.04 | 0.43 |
| 4 | Complexity | String | 5,079.23 | 0.42 |
| 5 | Complexity | A | 4,738.79 | 0.39 |
| 6 | Complexity | G | 3,483.94 | 0.29 |
| 7 | Complexity | D | 3,392.21 | 0.28 |
| 8 | Complexity | Substitute Function | 3,289.74 | 0.27 |
| 9 | Complexity | C | 2,866.46 | 0.24 |
| 10 | Complexity | L | 1,414.55 | 0.12 |
| 11 | Complexity | H | 616.94 | 0.05 |
| 12 | Complexity | Optimised Formula | 462.33 | 0.04 |

Key takeaways

- Large performance gains can be made
- Using a text editor can aid in finding repetition
- The gain in model size can be worth it for the improved performance
- Difficult to write an optimized formula – **testing and optimising after build is key**

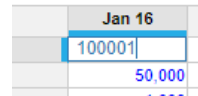
Complexity

Complexity

- **Where it occurs**
 - Multiple functions in one line item
 - Nested IF statements
 - Copy and pasted or Excel formula generation
- **Why is it bad?**
 - **Large calculations**
 - **Frequent recalculation**
 - Difficult logic to understand & maintain

Complexity

User makes a cell change



| |
|--------|
| Jan 16 |
| 100001 |
| 50,000 |
| 4,000 |

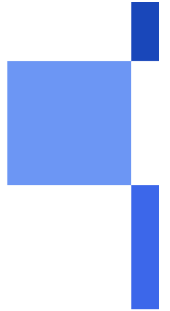


Products.Purchase Price[SUM: Region] + **Marketing.Total Costs**[SUM: Region]
/ **Sales.Total Sales**[LOOKUP: Region] * (**Currency Modifier.USD** / 1000)

In this scenario all four functions will recalculate if any reference changes

Calculation Time: 30,320 ms + 29,780 ms + 21,320 ms + 4560 ms = 85,980 ms

Complexity



Functions split out to their own line items

A = Products.Purchase Price[SUM: Region]

B = Marketing.Total Costs[SUM: Region]

C = Sales.Total Sales[LOOKUP: Region]

D = (Currency Modifier.USD / 1000)

If B, Marketing Costs, is the only reference affected by the changes then only this calculates

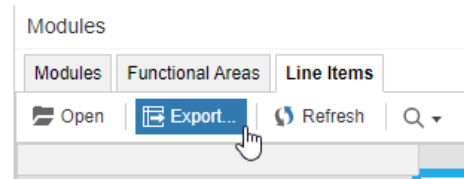
A + B / C * D

Calculation Time: **1 ms + 29,780 ms + 1 ms + 1 ms = 29,783 ms**

Giving us a 65% reduction in calculation time!

Complexity

- Export the line items from settings



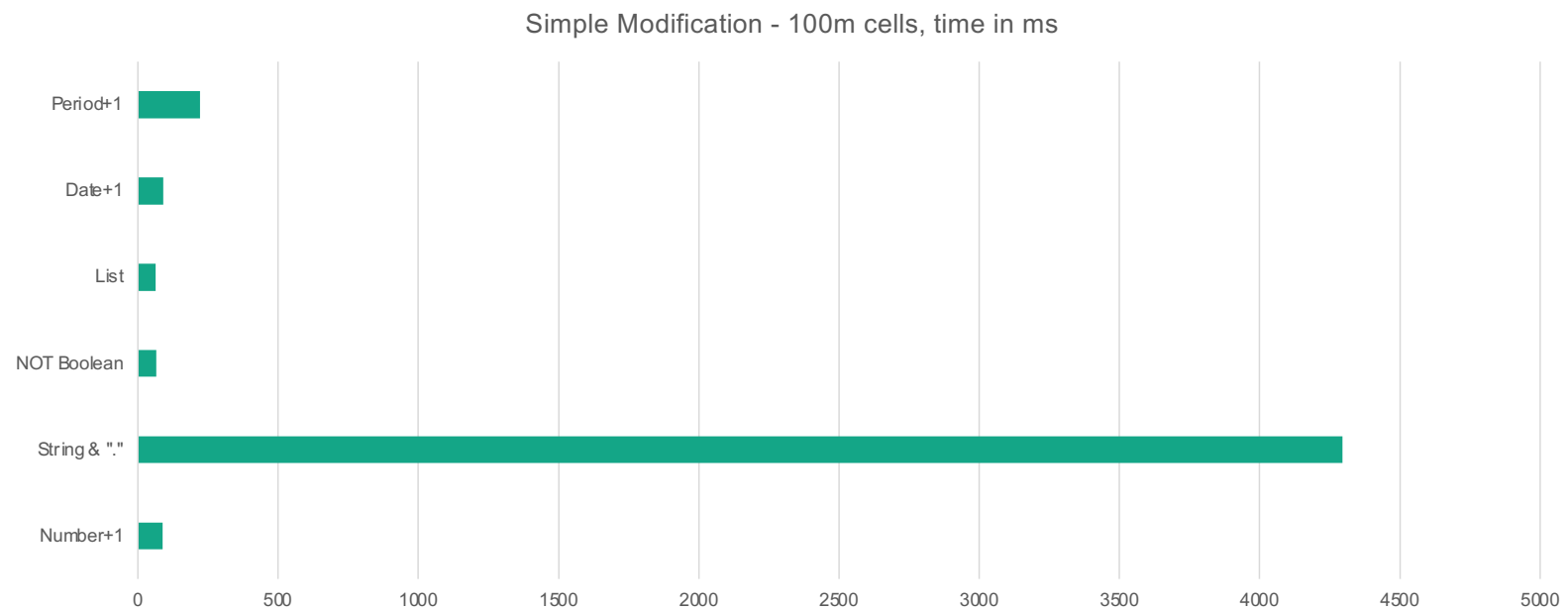
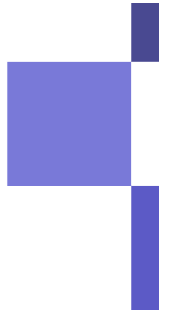
- Create a column in the export to check formula length using: **=LEN(B3)**
- Anything longer than 500 to 1000 characters is worth looking at
- Combine with cell count to work out which line items to focus on first

The screenshot shows an Excel spreadsheet with the following data:

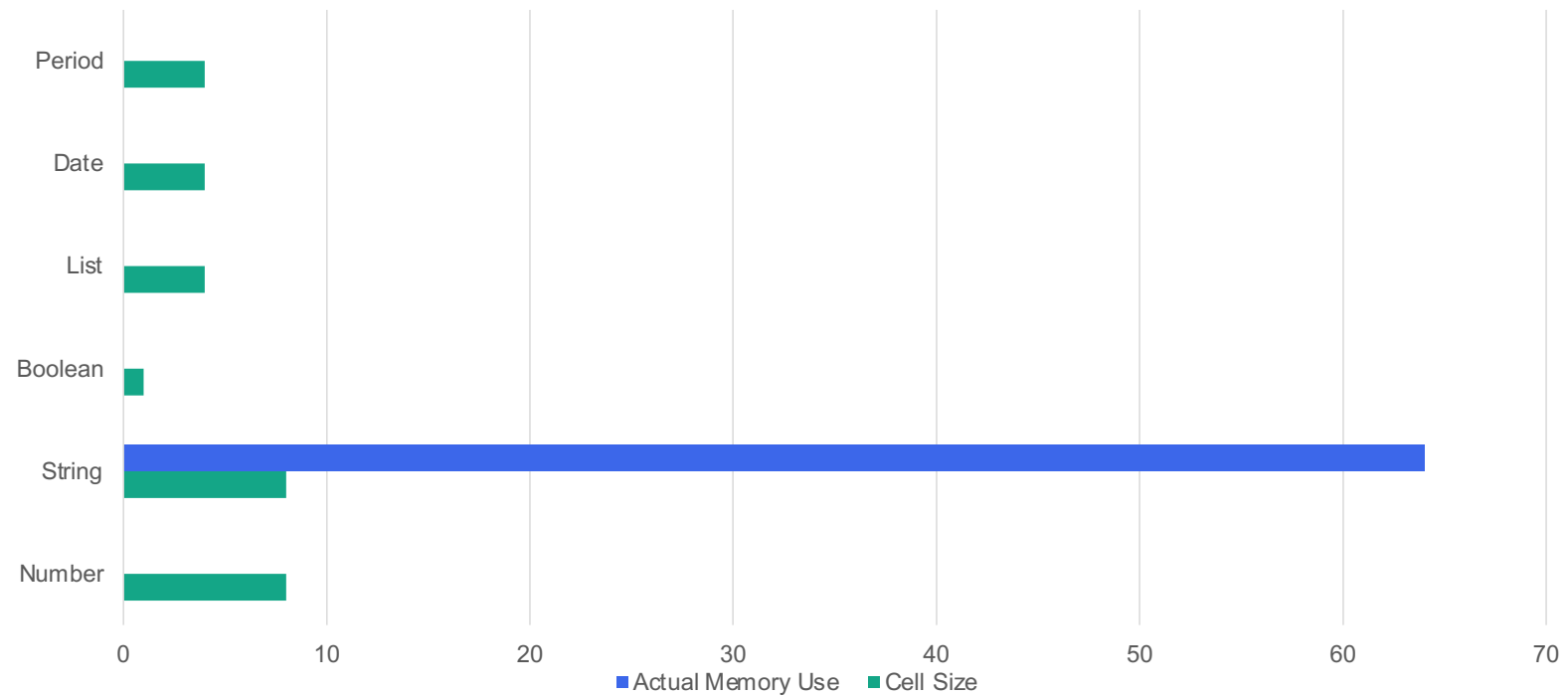
| | A | B | C |
|----|--------------------------|------------------|------|
| | | Formula | |
| 1 | | | |
| 2 | Model Drivers | | |
| 3 | Open Period Months | | |
| 4 | Date | | |
| 5 | Date +1 | Date + 20 | 9 |
| 6 | Month | PERIOD(START()) | 15 |
| 7 | Month Mapping | S(START(), -1)) | 29 |
| 8 | Versions Compare Mapping | | 0 |
| 9 | LISS Mapping | ision Compare))) | 65 |
| 10 | Comparison - Original | | 0 |
| 11 | Data | ers.Open Period | 3820 |
| 12 | Comparison - LISS | | 0 |
| 13 | Collect | COLLECT() | 9 |
| 14 | Data | 9 / 1000 ELSE 0 | 171 |
| 15 | Comparison - New | | 0 |
| 16 | Data | 0 ELSE 0 ELSE 0 | 371 |
| 17 | HD Product Planning | | 0 |
| 18 | Net Revenue | | 0 |
| 19 | Gross Margin | | 0 |
| 20 | Volume | | 0 |
| 21 | | | |

Text Concatenation

Modification Performance



Memory Use



Finding text concatenation

| Modules | |
|---------------------|----------------------|
| Modules | Functional Areas |
| Open | Export |
| Refresh | Q |
| Text Concatenation | |
| Original Formula | "CODE1:" & CODE(I |
| Optimised Formula | "Text 1' & Text 2' |
| Text 1 | "CODE1:" & CODE(I |
| Text 2 | "_CODE2:" & CODE(I |
| Text 3 | "_DATE:" & NAME(I |
| Complexity | |
| String | "ABC_DEF_GHI_JKL |
| Original Formula | (IF ISNOTBLANK(LE |
| ----- | |
| Optimised Formula | C & D & G & H & K & |
| Substitute Function | SUBSTITUTE(String |
| A | FIND("_", String, 1) |
| B | FIND("_", String, LE |
| C | IF ISNOTBLANK(LE |
| D | IF LENGTH(MID(Str |



| C3 | |
|-------------------------------------|--------------------|
| =LEN(B3)-LEN(SUBSTITUTE(B3,"&","")) | |
| A | B |
| 1 | Formula |
| 2 | Text Concatenation |
| 3 | Original Formula |
| 4 | Optimised Formula |
| 5 | Text 1 |
| 6 | Text 2 |
| 7 | Text 3 |
| 8 | Complexity |
| 9 | String |
| 10 | Original Formula |
| 11 | |

Excel formula for & count:
 =LEN(B3)-LEN(SUBSTITUTE(B3,"&",""))

- Combine with formula length and cell count to decide which line items to focus on first

Demo

1. Identifying text concatenation formulas
2. Splitting formulas to **calculate at lowest cell count possible**
3. Testing the optimization
4. Performance analysis

Demo

- Breaking down formula (at each applies to)
- "CODE1:" & CODE(ITEM('500')) & "_" & "CODE2:" & CODE(ITEM('200')) & "_" & "DATE:" & NAME(ITEM(Time))

| Original Formula | "CODE1:" & CODE(ITEM('500')) & "_" & "CODE2:" & CODE(ITEM('200')) & "_" & "DATE:" & NAME(ITEM(Time)) | | | | | |
|--------------------|--|--------|------------|--------|------------|----------|
| | Formula | Parent | Is Summary | Format | Applies To | Time |
| Text Concatenation | | | | | 500, 200 | Month |
| Original Formula | "CODE1:" & CODE(ITEM('500')) & "_" & "CODE2:" & CODE(ITEM('200')) & "_" & "DATE:" & NAME(ITEM(Time)) | | | Text | - | Month |
| Optimised Formula | "Text 1" & "Text 2" | | | Text | - | Month |
| Text 1 | "CODE1:" & CODE(ITEM('500')) | | | Text | 500 | Not Appl |
| Text 2 | "CODE2:" & CODE(ITEM('200')) & "Text 3" | | | Text | 200 | Month |
| Text 3 | "DATE:" & NAME(ITEM(Time)) | | | Text | | Month |

- New line items with lowest cell counts possible

| Cell Count |
|------------|
| 2,402,912 |
| 1,200,000 |
| 1,200,000 |
| 500 |
| 2,400 |
| 12 |

Demo

- Test optimisation – compare original and optimised
- Filter out any items that don't match – should be none

The screenshot displays a software interface for comparing formulas. The main window is titled "compare" and shows a table with columns for "Original Formula", "Optimised Formula", and "compare". The table lists 33 rows of data, each with a unique identifier (e.g., CODE1.001_CODE2.001_DATE:Jan 20) and a corresponding date. A "Filter" dialog box is open in the foreground, showing a search criteria of "Text Concatenation: compare" and "Time: -- Current Page --". The filter is set to "is not equal to" and "Show items that match all of the following". The "Enable filter" checkbox is checked, and the "Clear" button is visible. The "OK", "Cancel", and "Clear All" buttons are at the bottom right of the dialog.

| | Original Formula | Optimised Formula | compare |
|-----|---------------------------------|---------------------------------|---------|
| #1 | CODE1.001_CODE2.001_DATE:Jan 20 | CODE1.001_CODE2.001_DATE:Jan 20 | |
| #2 | CODE1.001_CODE2.002_DATE:Jan 20 | CODE1.001_CODE2.002_DATE:Jan 20 | |
| #3 | CODE1.001_CODE2.003_DATE:Jan 20 | CODE1.001_CODE2.003_DATE:Jan 20 | |
| #4 | CODE1.001_CODE2.004_DATE:Jan 20 | CODE1.001_CODE2.004_DATE:Jan 20 | |
| #5 | CODE1.001_CODE2.005_DATE:Jan 20 | | |
| #6 | CODE1.001_CODE2.006_DATE:Jan 20 | | |
| #7 | CODE1.001_CODE2.007_DATE:Jan 20 | | |
| #8 | CODE1.001_CODE2.008_DATE:Jan 20 | | |
| #9 | CODE1.001_CODE2.009_DATE:Jan 20 | | |
| #10 | CODE1.001_CODE2.010_DATE:Jan 20 | | |
| #11 | CODE1.001_CODE2.011_DATE:Jan 20 | | |
| #12 | CODE1.001_CODE2.012_DATE:Jan 20 | | |
| #13 | CODE1.001_CODE2.013_DATE:Jan 20 | | |
| #14 | CODE1.001_CODE2.014_DATE:Jan 20 | | |
| #15 | CODE1.001_CODE2.015_DATE:Jan 20 | | |
| #16 | CODE1.001_CODE2.016_DATE:Jan 20 | | |
| #17 | CODE1.001_CODE2.017_DATE:Jan 20 | | |
| #18 | CODE1.001_CODE2.018_DATE:Jan 20 | | |
| #19 | CODE1.001_CODE2.019_DATE:Jan 20 | | |
| #20 | CODE1.001_CODE2.020_DATE:Jan 20 | | |
| #21 | CODE1.001_CODE2.021_DATE:Jan 20 | | |
| #22 | CODE1.001_CODE2.022_DATE:Jan 20 | | |
| #23 | CODE1.001_CODE2.023_DATE:Jan 20 | | |
| #24 | CODE1.001_CODE2.024_DATE:Jan 20 | | |
| #25 | CODE1.001_CODE2.025_DATE:Jan 20 | | |
| #26 | CODE1.001_CODE2.026_DATE:Jan 20 | | |
| #27 | CODE1.001_CODE2.027_DATE:Jan 20 | | |
| #28 | CODE1.001_CODE2.028_DATE:Jan 20 | | |
| #29 | CODE1.001_CODE2.029_DATE:Jan 20 | | |
| #30 | CODE1.001_CODE2.030_DATE:Jan 20 | | |
| #31 | CODE1.001_CODE2.031_DATE:Jan 20 | | |
| #32 | CODE1.001_CODE2.032_DATE:Jan 20 | | |
| #33 | CODE1.001_CODE2.033_DATE:Jan 20 | | |

Demo

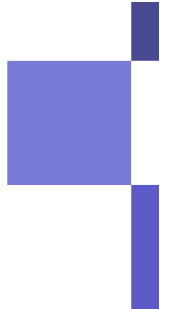
- Performance analysis

Calculations by Line Item

| | Module ▾ | Line Item ▾ | Total calculation time (ms) ▾ | % of total calculation time ▾ |
|---|--------------------|-------------------|-------------------------------|-------------------------------|
| 1 | Text Concatenation | Original Formula | 22,253.84 | 90.29 |
| 2 | Text Concatenation | Optimised Formula | 2,170.20 | 8.81 |
| 3 | Text Concatenation | Text 3 | 120.56 | 0.49 |
| 4 | Text Concatenation | Text 1 | 56.31 | 0.23 |
| 5 | Text Concatenation | Text 2 | 45.60 | 0.19 |

Key takeaways

- Text formula performance is poor
- Combine as many text parts as possible
- Split apart by each 'applies to'
- Use System modules where possible



Session takeaway

- Following the golden rules will lead to improvements
- Use Excel to analyse blueprints
- Learn to use the text editor to break down formulas

Key factors in reducing formula complexity

- Calculate once and refer many times
- Calculate at lowest cell count possible

Don't forget

- Test as you go!

Further reading

- <https://community.anaplan.com/t5/Best-Practices/Reduce-Calculations-for-Better-Performance/ta-p/33667>
- <https://community.anaplan.com/t5/Best-Practices/Formula-Optimization-in-Anaplan/ta-p/41663>
- <https://community.anaplan.com/t5/Best-Practices/Formula-Structure-for-Performance/ta-p/33177>
- **PLANS** - <https://community.anaplan.com/t5/Best-Practices/PLANS-This-Is-How-We-Model/ta-p/33530>
- **DISCO** - <https://community.anaplan.com/t5/Best-Practices/Best-Practices-for-Module-Design/ta-p/35993>
- **Planual** - <https://community.anaplan.com/t5/Best-Practices/The-Planual/ta-p/49773>
- My text editor <https://atom.io/>

Exercise 1

- An empty model with the complex formula
- Try to recreate the demo by finding repeated elements in a text editor and populating the model
- Remember to add in a line item to compare the results

| | Formula | Parent | Is Summary | Format | |
|-------------------|--|--------|--------------------------|---------|-----------|
| Complexity | | | | | 1000, 500 |
| String | "ABC_DEF_GHI_JKL_MNO" | | <input type="checkbox"/> | Text | - |
| Original Formula | (IF ISNOTBLANK(LEFT(String, FIND("_", String, 1) - 1)) THEN LEFT(String, FIND("_", String, 1) - 1) ELSE RIGHT(String, FIND("_", String, 1) - 1)) | | <input type="checkbox"/> | Text | - |
| ----- | | | <input type="checkbox"/> | No Data | - |
| Optimised Formula | | | <input type="checkbox"/> | Text | - |
| A | | | <input type="checkbox"/> | Number | - |
| B | | | <input type="checkbox"/> | Number | - |
| C | | | <input type="checkbox"/> | Text | - |
| D | | | <input type="checkbox"/> | Text | - |
| E | | | <input type="checkbox"/> | Number | - |
| F | | | <input type="checkbox"/> | Text | - |
| G | | | <input type="checkbox"/> | Text | - |
| H | | | <input type="checkbox"/> | Text | - |
| I | | | <input type="checkbox"/> | Number | - |
| J | | | <input type="checkbox"/> | Text | - |
| K | | | <input type="checkbox"/> | Text | - |
| L | | | <input type="checkbox"/> | Text | - |
| M | | | <input type="checkbox"/> | Text | - |

Exercise 2

- Add the line items needed to separate out the text concatenation
- Remember to change dimensions to get lowest cell counts
- Use a line item to compare the two formulas

| | Formula | Parent | Is Summary | Format | Applies To |
|--------------------|---|--------|--------------------------|--------|------------|
| Text Concatenation | | | | | 500, 200 |
| Original Formula | "CODE1-" & CODE(ITEM('500')) & "-" & "CODE2-" & CODE(ITEM('200')) & "-" & | | <input type="checkbox"/> | Text | - |
| Optimised Formula | | | <input type="checkbox"/> | Text | - |

A decorative graphic in the bottom left corner consisting of a grid of colored squares in shades of blue, green, and purple, arranged in a pattern that tapers to the right.

Thank you!

**CONNECTED
PLANNING
XPERIENCE**

The Anaplan logo, which is a stylized green arrow pointing upwards and to the right.

anaplan