Anaplan入門

2019/08/28
Anaplan Japan株式会社

Anaplan

INDEX

Anaplan

はじめに

内容が多岐に渡り量も多く、一気にすべてを理解するのは大変かもしれません。

実プロジェクトを行う際に、資料を見返して活かしてもらえるとうれしいです。

Anaplan

Scope

Model Builder

Designer

▽Part 1

- 設計観点(基本)
- 規約等
- 複数Modelでの設計観点

▽Part 2

- 設計観点(応用)
- パフォーマンス
- セキュリティ
- 容量(Sizing)

Anaplan

Out of Scope

以下の内容はこのセッションでは扱いません。

- 基本機能の紹介
- ユースケースの紹介
- Project Management/要件定義の仕方
- インフラ系の内容(ALM、WS管理、アカウント管理、SSO等)
- データ連携(Anaplan Connect等)

Target

Model Builder | Designer

Model Builder(実装者)からDesigner(設計者)のレベルまで、様々な内容を取扱います。

スライド毎の対象者を右上のアイコンで表示するので、ご自身に関する内容に集中してください。

Part 1は、Learning Course L1(旧102,201)を修了した方を対象に、「これさえ抑えておけば、大きな失敗はしなくなる。」という予防接種的な内容を扱います。

Part 2は、Anaplanプロジェクト経験者を対象に、高難度な案件に対応するための知識と観点を扱います。

Expectation

Model Builder | Designer

健全なModelが育てられるようになります。

- 拡張性のある設計
- 高い生産性
- 無駄がない
- 引き継ぎしやすい
- 容量やパフォーマンスなどでトラブルにならない

設計観点
(基本)

健全なModelになるかどうかは、設計こそが最初の運命を左右します。

設計次第で、急な仕様変更や手戻り発生時の対応にかかる工数も最小化できます。

ここでは以下を扱います。

- Moduleの設計
- Listの設計
- Dashboardの設計
- Importの設計
- Modelをどうわけるか

Anaplan

Moduleの設計

Data, Input, System, Calculation, Output

Anaplan

Moduleの単位?

Model Builder

Designer

Moduleでは、入力/計算/表示などの様々なことができます。

Moduleをわけずに1つのModuleでなんでも実装してしまうこともできてしまいます。

しかし、1つのModuleで実装してしまってもよいのでしょうか?

Anaplan

1つにするデメリット

Model Builder | Designer

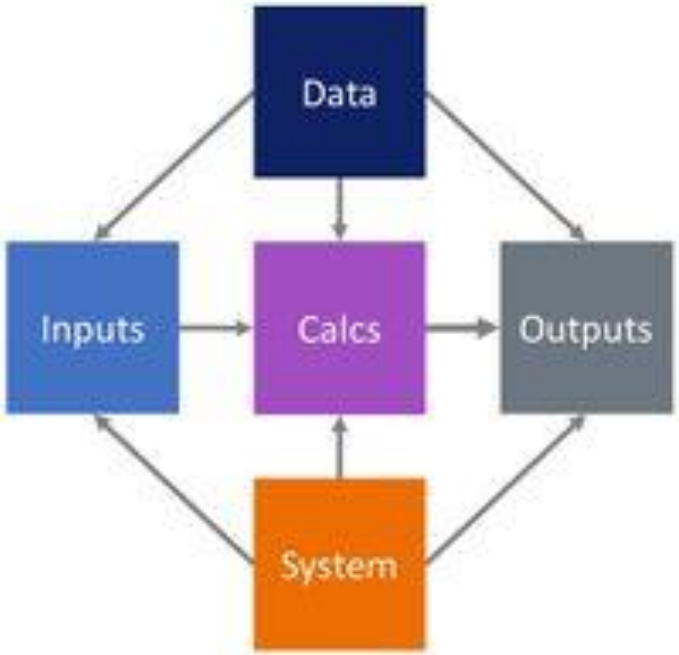
1つのModuleになんでもかんでも詰め込んでしまうと、以下のようなデメリットが発生してしまいます。

- どのModuleになにが入っていたのかわからず、生産性が下がる。
- ModuleをDashboardにPublishした後に、毎回Hide、Show。
- 誤ってエンドユーザーに見えてはいけない計算が見えてしまう。
- Model Mapがデータや参照の流れにならない。
- 不要になったLine Itemがどれだかわからなくなる。

これはAnaplanに限らず、Excelを含めたどんなシステムでも同じことが言えます。

D・I・S・C・O

Model Builder | Designer



Moduleは機能別、DISCO別、Dimensionの組み合わせ別に分けましょう。

Data – 実績データ等のトランザクションデータ、マスタの元にするデータ。
Inputs – ユーザーの入力項目。
System – Filterや設定。
Calculations – 計算。
Outputs - 計算結果。

https://community.anaplan.com/t5/Best-Practices/Best-Practices-for-Module-Design/ta-p/35993

Anaplan

基本の考え方

Model Builder | Designer

エンドユーザーのDashboard上に表示するもの

- 見込値や意思入れなどの入力は、Input
- 計算結果や参考情報などの表示は、Output

エンドユーザーのDashboardに表示しないもの

- InputからOutputの間の計算を行うものは、Calculation
- 取り込んだ外部データやデータ加工、過去計算値の保持などは、Data
- FilterやDCAの条件になるもの、時間や為替や配賦ドライバの設定は、System

DISCOのメリット

Model Builder | Designer

▽メリット

- 変更の影響範囲がわかりやすい。
- 機能追加がしやすい。
- レビューしやすい。
- Dashboard上でのHide、Showなどをしなくて良くなる。
- アクセス権の設定が楽になる。
- 容量削減時の分析がしやすい。
- 不要なLine Itemが一目瞭然なので消せる。(複数人開発だと特に。)

よくある疑問

Model Builder | Designer

- InputとOutputのLine Itemが入っているModuleはわけたくないけどどうすればいい?
- Dashboardには載せないけれど、Chartの元にするModuleは、Output? それともCalculation?
- 計算中の数値も見たいと言われたけどCalculationをDashboardに載せていい?
- User別Filterの条件入力はInput?それともSystem?
- 1つの機能のOutputが次の機能の計算に使われるけど、OutputがCalculationから参照されていいの?

回答する前に

Model Builder

Designer

「このModuleはDISCOでいうどれなのか?」と迷うことは多くあります。
しかし、重要なのは「DISCOのラベルをどれにするか?」ではなく、

「用途や目的に応じて、適切にModuleをまとめたりわけたりする。」

という考え方です。
プロジェクト毎にルールを決めて、必要に応じてDISCO以外の分け方を追加することもアリです。

回答(1/3)

Model Builder

Designer

InputとOutputのLine Itemが入っているModuleは1つにしたいという要望は少なくありません。

その通りにModuleを作成するのが良いです。その場合、NotesやFunctional Areaなどで「このModuleは、InputとOutputが同じになったModuleなんだ。」ということがわかるようにしてあれば良いです。

Dashboardには載せないけれど、Chartの元にするModuleは、基本はOutputとして扱いつつ、NotesやFunctional AreaでChartだということがわかりやすければ尚良いです。

回答(2/3)

Model Builder

Designer

計算中の数値も見たいと言われた場合、それはCalculationを参照するOutputのように実装しましょう。CalculationはDashboardに載せないものという基本は守ったほうが、レビューや引き継ぎが楽になります。

User別Filterの条件入力などは、Dashboard上に載せるものということがわかりやすいようにInputでも良いですし、NotesやFunctional Areaで条件設定なんだということがわかりやすければ良いです。

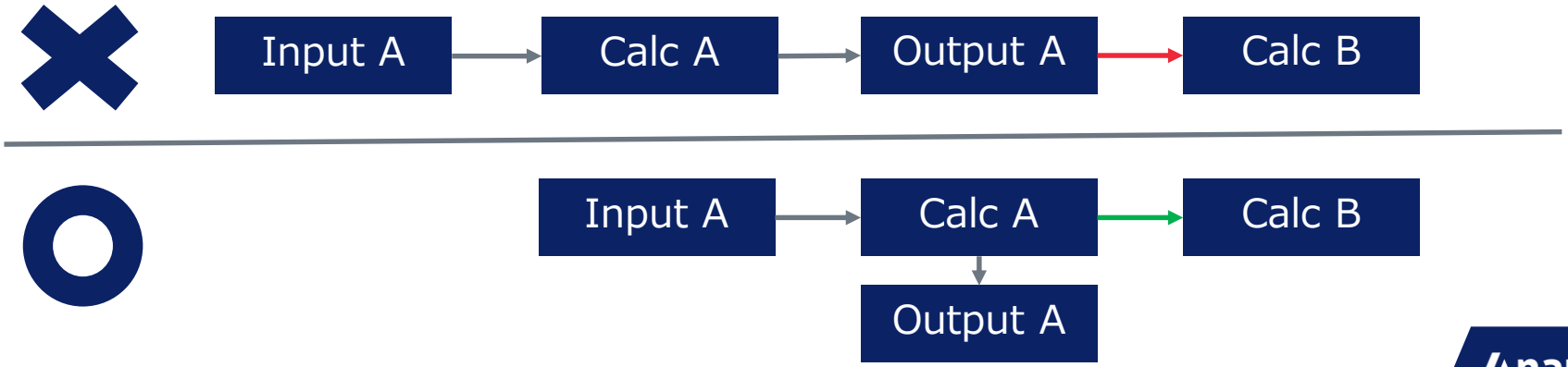
Anaplan

回答(3/3)

Model Builder | Designer

機能Aの計算結果を機能Bの計算で参照したい場合、Calc BがOutput Aを参照するよりも、Calc BはCalc Aを参照する形にしてあげるほうが、影響範囲調査や仕様変更が容易になります。

例えば、後から「数値の表示は単位を切り替える機能をつけてほしい。」と言われた場合、OutputがCalcから参照されていると機能追加が難しくなります。



Anaplan

Listの設計

健全なModelを育てる第一歩

Anaplan

Listを作る観点

Model Builder | Designer

Listには以下のようなものがあります。

- 処理したいデータを保持する軸になるMaster Data(製品、顧客等)
- Master Dataの属性情報(カテゴリなどの分析軸等)
- 構造を作るList(StatusやLock/Unlockなど)
- Custom Time/Custom Version
- Transaction Data
- レポートや表示を作成するための縦軸や横軸

誤ったListの設計

Model Builder

Designer

「このような表示にしたい。」というExcelを受け取って、その縦軸横軸にあるものをいきなりListにしてModelを作り出すのはやめましょう。

Anaplanは、データを持つのも計算するのも表示するのも、すべて”Module”で行うことになります。しかし、「最適なデータ保持方法」と「表示したい画面のあり方」が必ずしも一致するとは限りません。

例えば、Excelの取引先の列の中に、”その他”や”Unallocated”等の項目が混ざっていた場合、これをListの中にいれてしまって良いかどうかは見極めなければいけません。

Anaplan

Listを設計する基本(1/5)

Model Builder

Designer

まず、計画業務の中でどのデータを扱いたいのかを洗い出してください。
ここでは「製品価格や売上」などを例とします。

次に、「そのデータは何をキーに一意に決まるのか?」を考えてください。
将来の製品価格データのキーは、
「製品によって決まるのか?」
「いや、月別で製品によって決まる。」
「いやいや、取引先毎に月別に製品毎に異なる。」
という場合まであります。
(※将来の製品価格が、”受注毎に変わる”ことも要件に入っている場合、それは本当に計画業務なのかを疑ってください。実行系業務かもしれません。)

Listを設計する基本(2/5)

Model Builder

Designer

売上の計画見込値のキーはもっと複雑な可能性があります。

toCの業態の計画見込では「月別製品別で良い。」というものや「さらに販売地域別にしたい。」という場合もあります。

という一方で、toBの業態では、
「得意先別、製品別、支社別にしたい。」
「さらに得意先は、主要顧客は取引先別で、その他はまとめたい。」
「いや、得意先以外にも代理店を挟む場合もあるから、代理店別にも。」
など、キーになるものを特定するまでにヒアリング事項が多くなる場合もあります。

Listを設計する基本(3/5)

Model Builder

Designer

これらキーになるデータがListになって、これらのListをDimensionにしたModuleでデータを保持できている状態が、「最適なデータ保持」です。

少なくとも外部から取り込むデータや保存する過去分の計画値などは、この状態で保持するようにしてください。

しかし、ここで以下のような困ることが出てくるかもしれません。

「入力画面では、取引先の列の中に”Unallocated”を混ぜたいらしい。」

「入力画面表示では、時間軸の中に合計値だけでなく、”前年比”や”増減”などもいれたい。」

こういったものをListのItemに加えても良いのでしょうか?

Anaplan

Listを設計する基本(4/5)

Model Builder

Designer

ListのItemの中に、画面要求により「本来はそのMaster Dataとは異なるもの。」を混ぜることは可能な限り避けましょう。

こういう例外を混ぜてしまうと、「IF分岐」や「SELECTでの値取得」が増えて、「実装工数が増える。」「引き継ぎ後の罠になる。」などの弊害が生まれます。

“前年比”や” Unallocated/その他”などの項目は、まずは他のModuleのLine Itemにする仕様にできないかどうかを確認してください。

AnaplanのBest Practiceは、こういった実装例の紹介が多くあるので参考にしてください。

Anaplan

Listを設計する基本(5/5)

Model Builder

Designer

それでもそれでも「いやどうしてもすべての数値が1つのModuleに表示されていてほしい。」という場合もあります。

この場合、データ保持とは別に、入力や表示用のListを新規作成、またはSubsetとして作るようにして、データ保持とはわけるようにしましょう。
画面要求で作成したListで、データ保持までしてしまうことは避けましょう。

別Listにした上で、Propertyなどでマッピング情報を保持してLOOKUPとSUMで参照可能にしましょう。

Data, Input, Calcまでは通常のListで、Outputだけ表示用のListでModuleを作成して、レポート用にする。などは良い例です。

Anaplan

List vs Line Item

Model Builder | Designer

同一Formatかつ同一Formulaで扱えるCellの集合がLine Itemです。
なので、Line Itemは、基本的には扱うデータの種類毎にわけましょう。

ListにすべきかLine Itemにすべきか悩むものの代表に勘定科目があります。
勘定科目に紐づく費用は、データの種類としては同一ですが、勘定科目別に異なるFormulaが必要です。
しかし、Listの要素毎にIFで分岐していると生産性が低くなります。

こういうケースでは、Line ItemをList化できるLine Item Subsetの利用を検討しましょう。

https://community.anaplan.com/t5/Anaplan-Platform-Discussions/Line-Item-or-Line-Item-Subset-or-Line-Item-Subset-mapping-to/td-p/24653

Selective Access

Model Builder | Designer

Listの要素の名称も見せてはいけないようなセキュリティ要件がある場合は、Selective Accessが必須になるので、Listの階層構造を考慮して設計しましょう。

これ以外のセキュリティ要件は、FilterとDCAでなんとかできるので、最初の設計段階では最低限上記の観点を押さえて確認してください。

※)アクセス権限の使い分け方は別スライドで。

※)ただし、あらゆるものをDCAに詰め込むとメンテナンスの際に混乱するので、バランスを取ってください。

Listの階層構造

Model Builder | Designer

Listの階層関係をつける場面は以下のようなものがあります。

- Grid上に小計を表示したいという要件

製品に対して、カテゴリ別の小計を同一Moduleに表示する場合などは、カテゴリなどを入れた階層構造が必要です。

- Dashboard上で、1クリックでの表示切替

親階層をクリックすることで、瞬時に子階層の詳細を表示するSynchronizeは階層構造が必要です。

- アクセス権限を設定する

Selective Accessで、グループのようにアクセス権限を設定するには、階層構造が必要です。

やってはいけない階層

Model Builder | Designer

階層構造を持つとSummaryでの計算が発生します。

「SUMのFormulaを書くのが苦手だから」などの実装都合でListの階層関係をつけるのはやめましょう。データの対応関係が1:nであるというだけであれば、階層構造ではなくPropertiesでMappingしてください。

また、Versionに階層構造を使うのもやめましょう。

各Versionでの数値を足し上げるという計算処理は不要なはずです。

余計な階層構造をつけると、以下のデメリットが発生します。

- 後から小計に使いたいものは別だと言われて、作り直しになる。
- 気が付かないうちにSummaryで容量を使ってしまっている。
- DashboardにPublishするたびに、Select Levels to Showする。

List内の親子階層

Model Builder | Designer

Listの階層構造の付け方には、2つの方法があります。

1つは、複数のListに対して階層構造をつけて、中のItemに親子関係を付ける方法です。

もう1つは、1つのList内で、Itemに親子関係を付ける方法です。

基本的には前者の実装を行うことで不都合になることはないので、前者での実装を推奨しています。

後者の場合、親階層での数値入力などはできなくなってしまいます。

ListのTop Hierarchy

Model Builder | Designer

「全製品の合計」などを計算したい場合は、製品ListのTop Hierarchyに「製品合計」などを設定しましょう。

たまに「 Formula を入れようとしたら、Top Hierarchyが未設定という理由でエラーがでました。」という理由で、「とりあえずTop Hierarchyを設定しました。」というケースを見かけることがあります。
こういった場合、そもそもFormulaの書き方が誤っているケースがほとんどです。

StatusやLock/Unlockなどの業務フローの構造を作るListなど、本当は足し上げる必要がないListには、 Top Hierarchyをいれないことをルール化して誤ったFormulaに気がつけるようにしましょう。

ListのCODE

Model Builder | Designer

Master Dataには、必ずCODEを設定するようにしましょう。

Numbered ListでCODEがないと、Modelを跨ぐImportができません。
1つのModel内での問題は発生しませんが、将来的なデータ連携時に困ります。

※)元データにCODEがなくても、Anaplan内でCODEを採番する方法もあります。(実装例は、”逆引きAnapedia”を参考に。)

※)他システム内ではCODEとされているものでも、運用上、更新がはいるようなものをAnaplanのCODEにするのはやめたほうがいいです。最初に可変か不変かを確認しましょう。

Anaplan

List Itemの名称で困る

Model Builder | Designer

Listを作成している際に、以下のようなことで困ることがあります。

- ListのItemの名称が一意ではない。
- ListのItemの名称が一意なのかどうか判断できない。
- 製品名称が60文字を超えてしまってデータが取り込めない。
- 製品ListのItemの名称に、CODEも一緒に表示したい。
- 設定に応じて動的に名称が変わってほしい。

こんなときは、Numbered Listを使いましょう!

Numbered List

Model Builder | Designer

▽Numbered Listにするとできること

- 同一名称のListのItemを作ることができる。(CODEは異なる必要がある。)
- List名称が動的に変更できる。
- 60文字以上のItemの名称を扱えるようになる。
- 通常はNAMEにいれられない特殊記号も扱える。
- 必ず一意なシステム内部変数が振られるので、採番などに利用できる。
- Item追加ボタンやCopy BranchなどのActionが使える。

▽Numbered Listにするとできないこと

- FileまたはModelからModuleへのImportで、NAMEで一致できなくなる。(CODE一致のImportで代用可能。)

Numbered Listの判断

Model Builder | Designer

基本的な考え方として、以下のような場合には、最初からNumbered Listにしておくほうが無難です。

- 外部からListに取り込むデータで、将来的に想定外のデータが来る可能性がある。
- 動的に名称が変わってほしい。
- エンドユーザーがDashboard上でItem追加を行う。

Line ItemかPropertiesか

Model Builder | Designer

DimensionがList1つだけのModuleと、ListのPropertiesは、構造上同じようにデータを保持できます。しかし、 ListのPropertiesから参照されていてもReferenced Byには表示が出てこないので、可能な限りModuleにしましょう。

ListのPropertyは、以下の用途以外は1次元のModuleで作りましょう。

- Listの属性情報(ListのImport時に付帯してくるもの。)
- 他Listの属性情報をLOOKUPで取得
- Numbered ListのDisplay Name
- Dependent Drop Downでのセレクター
- Import時のIdentifiersで、Combination of Propertiesにする
- ExportのLabels
- ActionのOpen DashboardとAssign

Dashboardの設計

Anaplan

Landing Dashboard

Model Builder | Designer

メニュー画面として、ボタンがたくさん配置されているDashboardを作るのはやめましょう。

左側にあるレフトナビを活用して業務フローに沿う導線にしましょう。

Landing Dashboardには、Model全体が対象としている業務のKPIなどのChartを載せましょう。
または、業務の進捗率などを表示するDashboardしましょう。

▽実際の例
https://community.anaplan.com/t5/Knowledge/Designing-a-Landing-Dashboard/ta-p/33563

ModuleとChartの数

Model Builder

Designer

Dashboardを開く際のパフォーマンスは、1つのDashboardに載せているModuleとChartの数に依存します。多いほど遅くなります。

1つのDashboardでのModuleやChartの数は可能な限り5個以下にしましょう。

※)サーバー側の計算負荷ではなく、クライアント側(ブラウザ)のレンダリングの負荷によるものなので、Model自体の容量やパフォーマンスとは異なる理由でパフォーマンスへ影響がでます。

Anaplan

Line ItemとPage Selector

Model Builder

Dashboard上で、Listの要素を選択して絞り込みを行う方法は2つあります。

- Page SelectorのSyncとFilterで絞り込みする方法
- Line ItemをFilterの条件にして絞り込みする方法

これらがDashboard毎にバラバラに使われていると、「あっちのDashboardで選択した内容が、こっちのDashboardに反映されない。」などの指摘を受けることになります。

Page Selectorは、リフレッシュボタン不要かつ全Dashboardで条件変更が同期されるので、とてもユーザビリティが高いです。ただし、複数選択などの複雑な条件設定ができません。

一方、Line Itemを条件に指定する方法は、リフレッシュボタンは必要ですが、あらゆることができます。

Importの設計

Importとは

Model Builder

Designer

Importとは、ListやModuleにデータを取り込むActionのことです。
Model外からのデータを取り込むことや、Model内でデータをコピーするかのように利用することもできます。

Importを利用することで、実行系に近い実装をAnaplan上で行うことまでできてしまいます。

一方で、AnaplanはFormulaによるリアルタイム計算が強みのPlatformなので、リアルタイム計算(Formula)とデータ連携(Import)のバランスをとるように設計しましょう。

Importの種類

Model Builder | Designer

Importには以下のようなものがあります。

- Dashboard上でのファイル取込
- 外部システムからのデータの取込
- Anaplan上のModel間のデータのやり取り
- 1つのModel内で、データをコピーするImport
- 1つのModel内で、ListのItemを生成するImport

Importの注意点と推奨

Model Builder | Designer

データの整合性を保つため、ImportやExportが実行される際には、Modelにロックがかかります。(Processing…のダイアログがでます。)

データ量が少なければ数秒で終わりますが、データ量が多いと数十秒かかる場合もあります。

エンドユーザーが利用するModelでの業務時間帯にImportが実行される作りは可能な限り避け、夜間や休日にデータが更新される作りにしましょう。

https://community.anaplan.com/t5/Best-Practices/Imports-and-Exports-and-Their-Effects-on-Model-Performance/ta-p/33552

※Data HubやExport用Modelなどのデータ連携用Modelを利用することで、エンドユーザーへの影響を最小化する方法があります。(応用編)

FormulaかImportか(1/2)

Model Builder

Designer

Model内でのImportで、以下のような機能が実装されがちですが、可能であればFormulaにしましょう。(特にUser数が20を超えてきた場合必須。)

- 入力値を承認者へ提出

→提出済か否かをBooleanの入力欄にしてIF分岐などで承認者に表示されるようにしましょう。

- データ保持用と表示用に中身が重複したListを作成した場合に、データをImportでコピーする

→マッピングを条件にしてLOOKUPやSUMでFormulaを書きましょう。

FormulaかImportか(2/2)

Model Builder

Designer

- 設定によってList Item名称を更新する

→Numbered Listにしましょう。

- 機能を跨ぐデータの連携

→機能間のデータも参照できるのであればFormulaで参照しましょう。

機能間でリアルタイム計算してほしくない場合などは、夜間バッチなどでデータがImportされる仕組みにしましょう。

または、別Modelにすることも検討しましょう。

Processの単位

Model Builder | Designer

1トランザクションでやりきるデータ加工などの、他のデータ更新が間に入ってほしくないような複数のImportはProcessにまとめましょう。

例

- CODEのないデータを取り込んだ直後に、CODEを採番する。
- データ保持用と表示用に中身が重複したListを作成した場合に、中身を常に同期する。

Modelの分け方

Connected Planningのはじまり

Anaplan

Modelの技術的制約

Designer

▽技術的制約

- 1Modelの容量の上限はWorkspaceの上限に合わせて130GB
- Time設定の有効範囲(Time Rangeでできることは除く)
- Versionの範囲
- UsersのModelへのアクセス可否
- Dashboardでリンクできる範囲
- Import/Exportでのロック範囲
- Formulaで直接参照可能なのは1Model内
- HistoryのRestoreの単位
- ALMの同期の単位

Modelをわける単位

Designer

Modelの制約条件に従ってModelをわけることになりますが、以下の観点でModelをわけるのか1つにするのかを検討してください。

- 業務用途が異なる場合は、Modelをわける。
- リアルタイム計算を行いたい処理は1Model内にする。
- 同一人物が一連の業務フローで行う処理は、1Model内にしないとModelを開いたり閉じたりするので、ユーザビリティを考えると1Model内にする。
- 外部とFileを入出力するModelは、パフォーマンスを高く維持するために、Data Hubとして分けてModelをわける。
- 一連の業務フローを1つのModelにすべて入れると、容量が130GBを超えてしまう場合は、Distributed Modelにわける。

データ連携とData Hub

Designer

Data Hubを作ることのメリットです。

- 複数のユースケース(Model)でマスタデータ(List)を共有できるので、データの粒度が揃い、Connected Planningができる。
- マスタデータ(List)の変更管理を一箇所にできる。
- パフォーマンス負荷の高いデータ加工処理をData Hubに寄せられる。
- 簡易ETLツールのような使い方ができる。

https://community.anaplan.com/t5/Best-Practices/Data-Hubs-Purpose-and-Peak-Performance/ta-p/48866

Anaplan

Anaplan

お作法

一定の規約やルールがないまま実装をしていると実装が散らかってしまい、プロジェクトが進むに連れて生産性が低下してしまいます。(特に複数人だと顕著。)

Anaplanの参照を辿れる機能の強みを活かして弱みを補う規約はどのようなものがあるのかをここで紹介します。

- 命名規則
- Formulaの書き方
- Notesの書き方
- Importの管理の仕方
- その他

命名規則

名前は大事?

Anaplan

命名規則の基本

Model Builder

Designer

これから紹介する命名規則は、「最低限これさえ抑えておけば後で痛い目を見ることはない。」というものです。

案件毎、プロジェクト毎、Model毎に、メンバーのコミュニケーションが楽になる命名規則や、保守性をあげるための命名規則を追加してください。

※)下記の記号は、命名規則上で利用するのは避けてください。

- . (ドット)
- ‘ (シングルクォーテーション)

Anaplan

命名規則(Functional Area)

Model Builder

まずFunctional Areaは、ModuleとDashboardで別のものを設定しましょう。

DashboardのFunctional Areaは、業務フローに沿ってわけましょう。
名前は、業務要件に合わせてください。

ModuleのFunctional Areaは、 基本的に「DISCOと機能別」に分けましょう。
こうすることで、Model MapがDISCOにわかれており、見やすくなります。

1機能のModule数が少ない場合、機能別だけでわけるのもアリです。

命名規則(Module)

Model Builder

DISCOがしっかり守られているのであれば、なんでもいいです。
Dashboardに表示されるものであれば、業務要件に合わせてください。

Anaplan

命名規則(Saved View)

Model Builder

用途がわかる名前にしましょう。

- ImportのData Source
- DashboardにPublishする元にする
- Exportの元にする
- Chartの元にする
- (New UXで利用する。※正式版は未リリース)

命名規則(Line Item)

Model Builder

DISCOがしっかり守られているのであれば、基本的にはなんでもいいです。
データの意味がわかる名称にすることをおすすめします。

例外として、 Filter, Conditional Formatting, DCAといった見た目系Line Itemに使う場合は、命名規則を決めてその用途がわかる名前にしてください。

※)Formulaで参照される場合は、Referenced Byで参照が辿れますが、 Filter, Conditional Formatting, DCAといった見た目系Line Itemは、どこで利用されているのかが辿れません。

※)他Moduleの見た目系Line Itemは利用せず、FilterとDCAも「利用するModule内」で必ずそれぞれ定義して、System Areaの共通Line Itemを参照して作る、というテクニックもあります。

命名規則(List)

Model Builder

List自体の名称は、業務上の用途がわかりやすければなんでもいいです。
階層がある場合は、何階層目かがわかる名前だとより良いです。

Subsetを利用する場合、そのSubsetがどのListに属するものなのかが分かる名前にしてください。

Listを分類して、中身は空のセパレータのListを配置するのも、可読性があがるのでおすすめです。

命名規則(Action)

Model Builder

なんでもいいです。業務要件にあわせてください。
ただし、デフォルトの名前で放置するのはやめてください。

また、Processに登録するImportは、先頭や末尾に順番をつけると実行時に順番がわかって便利です。

例:
Import into Product Category (1/3)
Import into Product (2/3)
Import into Product Price (3/3)

命名規則(Dashboard)

Model Builder

なんでもいいです。業務要件にあわせてください。

Formula

読みやすい書き方。

Anaplan

SELECT

Model Builder

SELECTの条件は、Listの特定の要素を指定することになります。
ここに、動的な要素をいれるのはやめましょう。
条件にするのはStructural DataまたはTop Hierarchyのみにしましょう。
動的要素は条件のLine Itemを作った上で、LOOKUPしましょう。

例:
[SELECT: 'Time'.'FY19']
↓
[LOOKUP: 'Time Settings'.'Current FY']

Boolean

Model Builder

Booleanの判定に 「= TRUE」などは不要です。パフォーマンスも改善!

IF xx = TRUE THEN yy ELSE zz

↓

IF xx THEN yy ELSE zz

IF xx = yy THEN TRUE ELSE FALSE

↓

xx = yy

分岐条件に#

Model Builder

×IF文の条件をNumbered Listの#にするのはやめましょう。

IF ITEM(List) = List.#35 THEN ~~

Structural Dataであっても、#では可読性が低いです。
条件フラグはBoolean FormatのPropertiesなどにすると見やすいです。

Anaplan

大量のIF

Model Builder

1つのLine Item内にIFをたくさんいれるのはやめましょう。
めあすとしては5個以上でなにかおかしいと思って手を止めてください。

特に、Listの要素別にIFとSELECTを書くような実装は、別の実装方法があるはずです。(LOOKUPを利用する。Listの構造を変える。などで。)

※)ここでは、可読性の意味合いで記載しますが、パフォーマンスの面でもデメリットがあるので、別スライドでも述べます。

PropertiesのFormula

Model Builder

ListのPropertiesのFormulaは、Line ItemのFormulaと同等のものが書けます。(Listを1次元のModuleとみなして。)

しかし、ここで複雑なFormulaを書くのはやめましょう。
以下のようなデメリットがあります。

- Formula Barがでないので書きにくい。
- Referenced Byにでてこないので参照を辿れない。
- Model Mapにもでてこない。

Notes

他の人のため。来週の自分のため。

Anaplan

Notes

Model Builder

- 量より質!
- Notesがなくてもわかるくらいのキレイな実装が理想的ですが、現実問題常にそうはいかないので、必要に応じて書きましょう。
- 基本はWhatではなく、Whyを書きましょう。（やってることが複雑な場合はWhatも。）

Notes改善の例

Model Builder

例1：ROUND(xx,2)
小数点以下を2桁で切っている。→外部連携先が数値を小数点以下2桁までしか受け付けていないので切る。

例2：OFFSET(xx,12,0)
前年度の実績を取得する。→ダッシュボード上で比較に利用するため。
または、
前年度の実績を取得する。→過去実績から係数を作るために取得。

Import

ImportはPublishしない

Model Builder

ImportをそのままDashboardにPublishするのはやめましょう。
必ず、Processに登録して、ProcessをPublishしましょう。
※)ExportはProcessに登録しないとPublishできません。

▽メリット

- 必要なImportを誤って削除できなくなる。
- 仕様変更があった際に、「いちいちImportのPublishし直し」ではなく、「Processへの登録し直し」で済む。

Anaplan

不要なImportは削除

Model Builder

Importは勝手に増えていくのですが、不要なImportを残していると、なにがなんだかわからなくなります。一覧の可読性も下がります。

不要なImportは常々削除しましょう。

以下をルーティンにしましょう。

- 必要なImportを作成した場合は、名称をわかりやすいものに変更して、Notesを記入して、Processに登録する。
- 残す必要のないImportを作成した場合は、削除する。

SourceはSaved View

Model Builder

Module to ModuleのImportを行う際は、ModuleをそのままSourceに選択するのではなく、必要なFilterなどをかけてSaved Viewを作ってSourceにしましょう。

▽メリット

- SourceとTargetの組み合わせが同じModuleでも、複数のImportを定義できる。(意図せず既存のImportを上書きするリスクが減る。)
- List階層構成の仕様変更があった場合に、Import定義を作り直す必要なくSourceを変更できる。
- 無駄なImportエラーの発生をなくせる。

Saved Viewの名称変更

Model Builder

Import定義を作成後にSourceになっているSaved Viewの名称を変更すると、Importを実行しても「Sourceが見つからない。」というエラーが発生するようになり、定義を再作成せざるを得なくなります。

Importの定義では、Anaplan内のSaved ViewであってもSourceの情報はFileと同等に扱われるので、このような事象が発生します。

Saved Viewの名称は変更を行わない前提で作るようにしましょう。

Anaplan

Default Set

Model Builder

FileをImport Data Sourceにする際、Default SetをPrivateのままにしておくと、数日後にサーバー上のFileが自動的に削除されます。

Default SetをAdmins OnlyまたはEveryoneにすることで、削除されなくなります。(GDPR対応の影響です。)

Fileが削除されると、”File is no longer Available~~”の警告がでて、Import定義の変更ができなくなります。

セキュアな情報が入ったFileをImport定義に使うのではなく、ヘッダーだけのFileでImport定義を作る、などをして、Privateを使う必要のない状態にしましょう。

Encoding

Model Builder | Designer

Fileをアップロードする際に、Encodingを選択できます。

データ連携を行う場合も、手動でファイルアップロードする場合でも、Encodingはプロジェクト全体で最初に統一しましょう。

エンドユーザーがアップロードするFile毎にEncodingを意識しなければいけない、というのはユーザビリティが低いです。

Encodingは、「パソコンの世界共通語」であるUTF-8で統一することを強く推奨します。

https://www.graffe.jp/blog/1278/

Other

その他

Anaplan

Show & Hide

Model Builder

ShowとHideを多用するのは控えましょう。

特定条件で表示を絞りたい場合は、Filterを利用しましょう。

Time Filterなどの知識は必須です。Line ItemはLine Item SubsetでFilterできます。

また、DISCOを守っていれば、「計算用Line ItemをHideする。」なども発生しません。

ShowやHideはAdmin側で設定するものではなく、エンドユーザーの便利機能として紹介するほうがBetterです。

※)見た目系Line ItemをHideするのは致し方ありません。

不要なModuleとLine Item

Model Builder

言うまでもありませんが、試しに作ったModuleやLine Item、昔の実装として残しているModuleやLine Itemは、Go-Liveの前には削除しましょう。

実装の参考のために置いておきたいものもあるかと思うので、そういうものは削除予定用のFunctional Areaを1つ用意してそこに設定しておくのも手です。

Anaplan

品質保証

Anaplanは実装面だけでなく、品質保証にも優位点があります。

Anaplanの特徴を活かした品質保証手法を紹介します。

テスト観点と評価手法

Japan Quality

Anaplan

テストシナリオ作成の観点

Model Builder

Designer

一般的なテストシナリオの作成観点に加えて、Anaplanでは以下のような観点を追加してください。

- Platformが保証している関数などは、評価対象外。
- Time(会計年度)やVersion(計画期)を切り替えても正しく動くか。
- Admin権限がないユーザーでも正しく動くか。(※AdminはWriteの権限設定を無視してImportが実行できてしまいます。)
- Role別で正しく動くか。

デグレを防ぐ

Model Builder | Designer

評価期間以降にFormulaを変更するのは、意図しないデグレードのリスクがあります。

いきなり変更するのではなく、以下のようなステップを踏むことで、意図しない変更がないかを確認してください。

1. OutputのModuleまたはLine ItemをCopyする。
2. Copyした側のFormulaを変更する。
3. 修正前と修正後の値を全件比較し、Filterで異なるもののみ表示するなどして、意図した変更だけが行われているかを確認する。
4. 意図通りの修正ができていれば、元のLine Itemを修正する。

※)参照先のLine ItemのOutputまでバックアップとって比較するとより正確。

Test Driven Development

Designer

実装前に計算結果の期待値を先に別Moduleや別Line Itemで作っておき、期待値通りの計算ができているかを確認しながら実装する手法です。

1. Input, Calculation, OutputのModuleを作成する。
2. Input Moduleに、業務上のパターン網羅したテストデータを入れておく。
3. Output ModuleにFormulaをいれるModule/Line Itemとは別に、Formulaのない期待値をいれるModule/Line Itemを作成する。
4. CalculationやOutputを実装する。
5. 期待値とOutputを比較し、すべての計算が意図通りになっていることを確認する。

TDDのメリット

Designer

TDD(Test Driven Development)のメリットです。

- UAT前に高品質。
- 期待値を作成している間に(実装前に)、要件や仕様の漏れに気がつける。
- 納品時の評価のエビデンスとして提出できる。

※)全機能に対して行わなくても良いです。業務上、特にパターンが複雑なものを対象に行うだけでも効果があります。

Anaplan

設計観点
(応用)

ここでは以下を扱います。

- Standard Time vs Custom Time
- Standard Version vs Custom Version
- Modelの分け方

Anaplan

Listの設計(応用)

Anaplan

特別なList

時間と計画期の軸は、計画業務での利用頻度が高いので、いくつかの便利な専用の機能(Formula等)を備えたTimeとVersionという専用のListを提供しています。

これらはとても便利なものですが、便利故にいくつかの制約が発生しています。

通常のListでも時間と計画期を実装することができ、こちらはいろいろな制約が外れています。

ここでは、標準機能をStandard Time/Standard Versionと呼び、通常のListで実装したものをCustom Time/Custom Versionと呼び、それぞれのメリット・デメリットを検討してみましょう。

Timeの設定(1)

網羅的な機能の詳細は、下記のサイトを見てください。
ここでは、要件に応じてどの設定を選べばよいのか、どのような注意をすべきか、を記載します。

https://help.anaplan.com/anapedia/Content/Modeling/Dimensions/Model%20Calendar.htm

▽Calendar Months/Quarters/Years
最も使われる設定。
週単位のデータを扱いたい、という要望がなければこれ一択。
あとの設定はすべて週をどう扱うか、が異なるだけのもの。

Timeの設定(2)

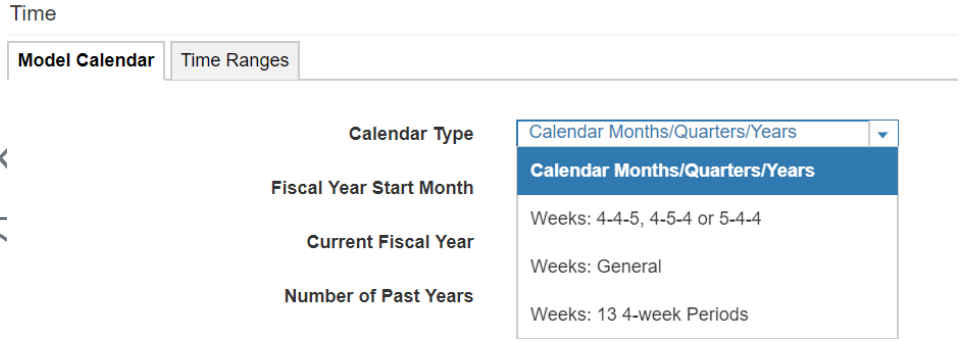
▽Weekすべてに共通すること

月初月末などが、本当の月とズレるので注意が必要です。
他のWeekの設定も月と日のマッピングはできません。
月も週もどちらも扱いたい場合は、「月のModel」と「週のModel」にわけて作る。または、週はCustom Timeにしましょう。

▽Weeks: 4-4-5, 4-5-4, or 5-4-4

Weekの中では1番汎用度の高いWeek

特殊要件がなければこれを使いましょ



Timeの設定(3)

▽Weeks: General
FYの概念がなくなったもの。ただ単に週がずらずらならんでいる。
便利なようで、使えなくなる機能が多いので要注意。
ModuleのDimensionにTimeを設定してもDAYとWeekしか使えなくなる。
Line ItemのFormatでWeek以外が使えなくなる。
Time Rangeが使えなくなる。

▽Weeks: 13 4-Week Periods
1Quarterは、13週をもっているという前提で、これは、Quarterに対して
Weekをどのようにもたせるのか?を重視した設定。
3,3,3,4をP1からP4にグループ分けした合計値などを表示できるようになる。
この場合、MonthのTime Periodが、このPを意味するようになる。
このP以外はWeeks: 4-4-5, 4-5-4, or 5-4-4と同じ。

Anaplan

Standard Time(1)

Model Builder | Designer

▽Standard Timeだからできること(とCustomでの代用可否)

- 月と日付等のカレンダー機能。
- 日次、週次、月次、四半期、年次などの単位が標準で利用できる。
- Time RangeとしてSubsetに近いものが利用できる。
- PREVIOUSとNEXT - 別Line Itemの値を取得する動きはLOOKUPで代用可能。Line Item自身を参照するのは代用不可能。

Standard Time(2)

Model Builder | Designer

▽Standard Timeだと不便なこと

- 1年以下のDAYの場合、Subsetが作れない。
- 日次で、年数が長いとパフォーマンス劣化が発生する。
- 表記される名称が変更できない。英語固定。
- 時間軸の列の中に、前年比などを入れたくても入れられない。(YTDだけ可能。)
- ALMと併用すると、設定変更のたびにSyncが必要。
- 週次だと、月と日のマッピングがズレる。

Custom Timeの検討(1)

Model Builder

Designer

絶対に容量問題やパフォーマンス問題にならないと判断できる軽めのユースケースであれば、Customを作る手間がかからないので、Standard Timeで良いです。

どうしても譲れない見た目の要件として「比率を別Line Itemで出すのではなく、FYの合計欄の隣に出したい」や「Aprではなく4月と表示したい」などがあれば、メンテナンスの手間が増えることの了承を得た上で、Custom Timeにしましょう。

Custom Timeの検討(2)

Model Builder

Designer

月と週をどちらも1つのModelで扱うような要件がある場合は、まず、Modelをわけることを検討してください。

その上で、1Modelでやる必要があれば、Standardを月、Customを週にして実装してください。

1年未満の日次などの数量計算が必須で、容量やパフォーマンス問題が発生しそうな場合には、DAYのCustom Timeにしましょう。

Custom Timeの補足

Model Builder

Designer

※)PREVIOUSとNEXTが必須の場合も、CustomとStandardのハイブリッドにすることで、Customの利用が可能。(実装例は逆引きAnapediaを参考。)

※)Current Periodは、設定画面以外での変更を許可したほうがあらゆる面で都合が良いので、ALMの利用有無やCustomの利用有無によらず、Line Itemで作ることを推奨します。

Standard Version(1)

Model Builder | Designer

▽Standardだからできること(と代用可否)

- Switch Over - ユーザビリティは少し劣化するが、DCAで代用可能
- Formula Scope - ユーザビリティは少し劣化するが、DCAで代用可能
- Version Formula - IF関数で代用可能
- Edit from Edit to - DCAで代用可能
- Versionに対してFormulaを設定できる - ユーザビリティは少し劣化するが、処理は代用可能
- PREVIOUSVERSIONとNEXTVERSION -別Line Itemの値を取得する動きはLOOKUPで代用可能。Line Item自身を参照するのは代用不可能。

Standard Version(2)

Model Builder | Designer

▽Standardだと不便なこと

- Line ItemやPropertyのFormatにできない。
- LOOKUPの条件にできない。
- Subsetが作れないので、容量対応ができない。
- Versionに対してFormulaを設定するとパフォーマンス劣化の原因になり得る。
- ALMと併用すると、設定変更のたびにSyncが必要。

Custom Versionの検討

Model Builder | Designer

Standard Versionでできることのほとんどは Custom Versionで代用可能です。

容量問題やパフォーマンス問題にならないと判断できる軽めのユースケースであれば、入力欄のユーザビリティをリッチにできるメリットを受けられるので、Standard Versionがいいです。

マスタ件数が多いまたは計算処理の複雑性が高いのであれば、Custom Versionにしましょう。特にSCM系のユースケースなどはマスタ件数が多く計算も複雑なので、Custom Version必須になります。

または、ユーザビリティはリッチにしたまま、容量削減やパフォーマンス改善も両立させるために、ハイブリッドで利用する方法もあります。

複数Modelの設計(応用)

Anaplan

Export用Model

Designer

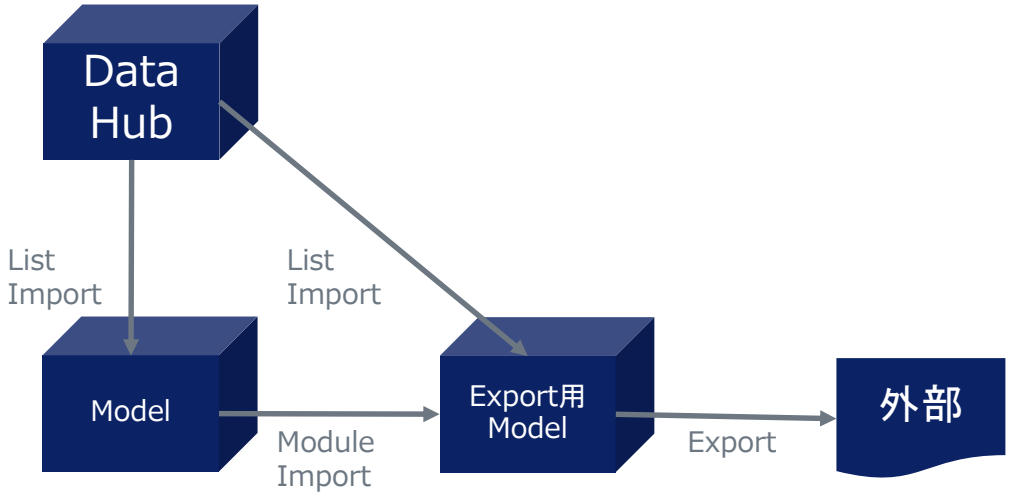
Export時は、Model全体がロックされます。
しかし、他ModelからImportされる場合は、ロックされません。

データ連携で頻繁にデータをAnaplanから出力したい場合は、業務用のModelから直接Exportするのではなく、
業務ModelからExport用Modelへデータを Import後、Export用Modelからデータを Exportするように設計しましょう。

2つのModelでマスタデータが一致している必要があります。
業務ModelからExport用Modelにマスタを同期するという手もありますが、Data Hubの利用を推奨します。

Anaplan

Export用Modelの構成図



Distributed Model

Designer

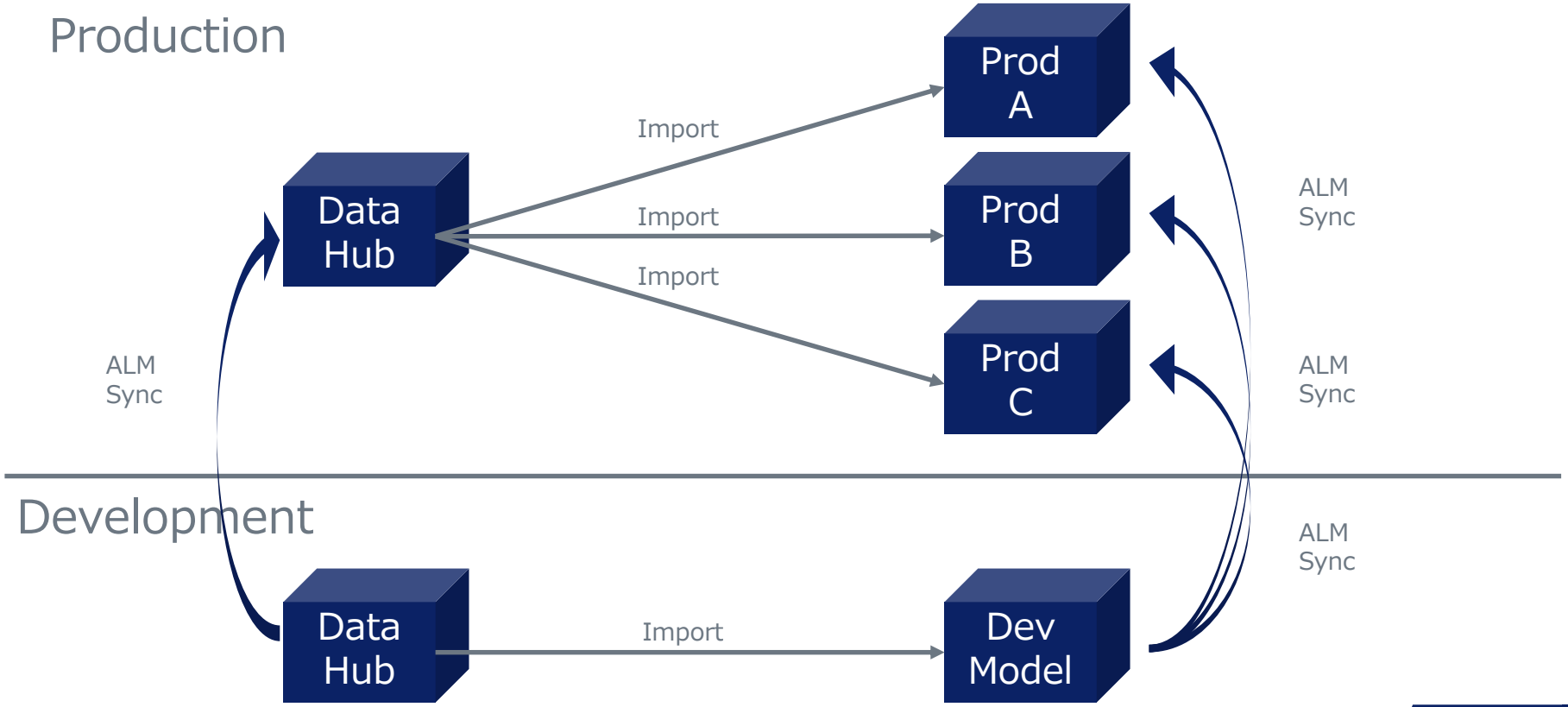
ALM機能を応用することでModelを冗長化できます。

容量が130GB以上のModelは作れませんが、
- 構造情報は同一でマスタデータは異なる

というようにModelを冗長化することで対応可能です。

構造情報のダブルメンテナンスは保守性が低いので、ALMを利用して、開発は1つのModelで行い、本番は複数Modelというように運用します。

Distributed Modelの構成図



非機能要件
前提知識

Anaplan

Anaplanのデータ保持と計算処理

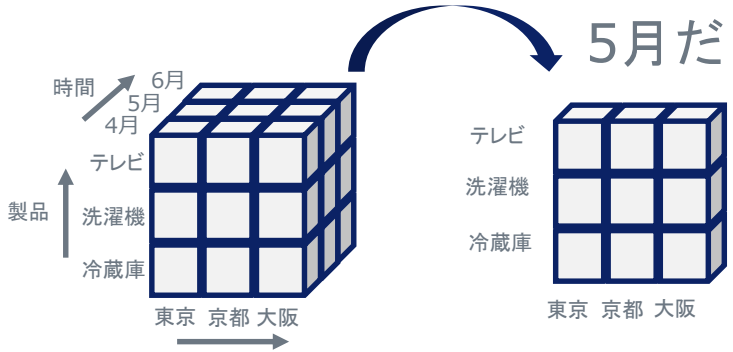
Moduleという多次元構造のキューブでデータを保持します。Cellのことです。
数値やテキストの1つ1つのデータがこのCellに入っています。

Line Itemに記述したFormulaは、このCellの1つ1つで実行されます。

1つのLine Itemの見方

上から見ると月別の支店の売上になる。

横から見ると
月別の製品の
売上になる。



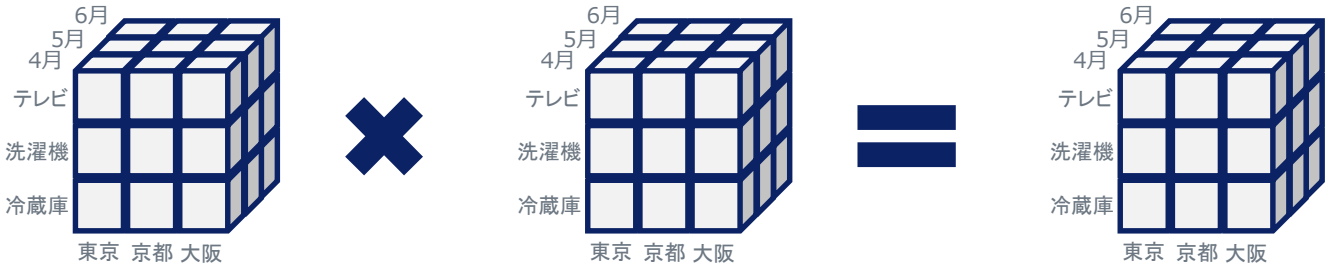
5月だけ取り出す。

手前から見ると年間の支店別製品別の売上になる。

Anaplan

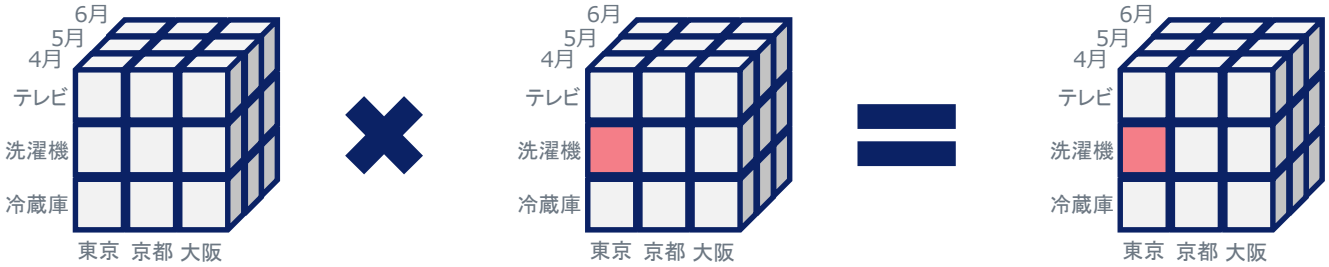
3つのLine Itemの見方

「単価 × 数量 = 売上」の3つのLine Itemを図で表すと下のようになります。



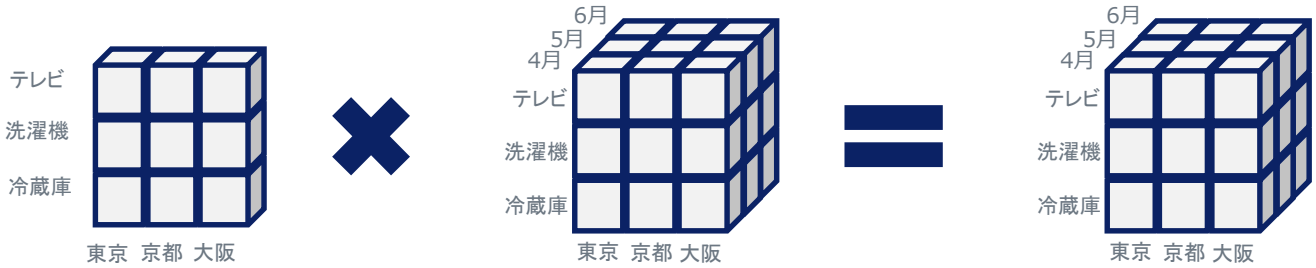
Hyperblockの計算処理

Cellの値に変更が入ると、最小限の再計算が行われます。
4月に東京で洗濯機が売れた例が以下になります。



異なるDimensionの参照

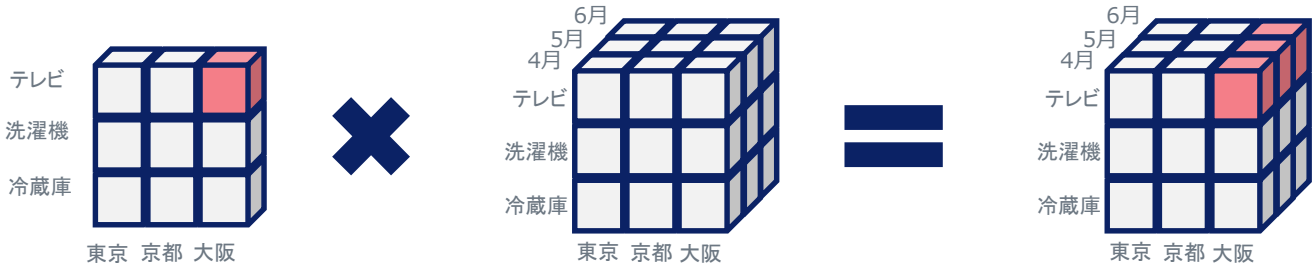
単価が毎月同じ場合の参照も可能。
単価Line Itemは製品と支店の2つのListで表現できる。



異なるDimensionの計算

単価が毎月同じ場合の計算。

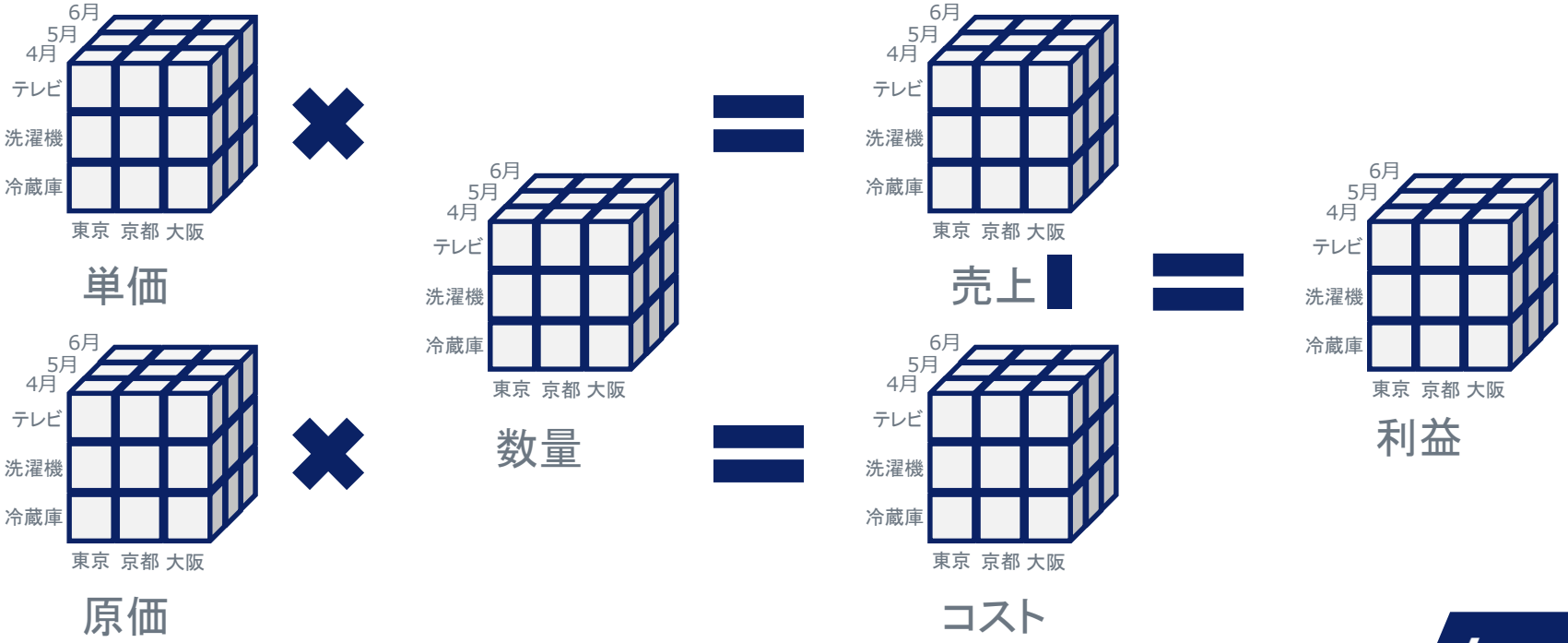
大阪のテレビの値段を変更すると、大阪のテレビのすべての月の売上が再計算されます。



単価　数量　売上

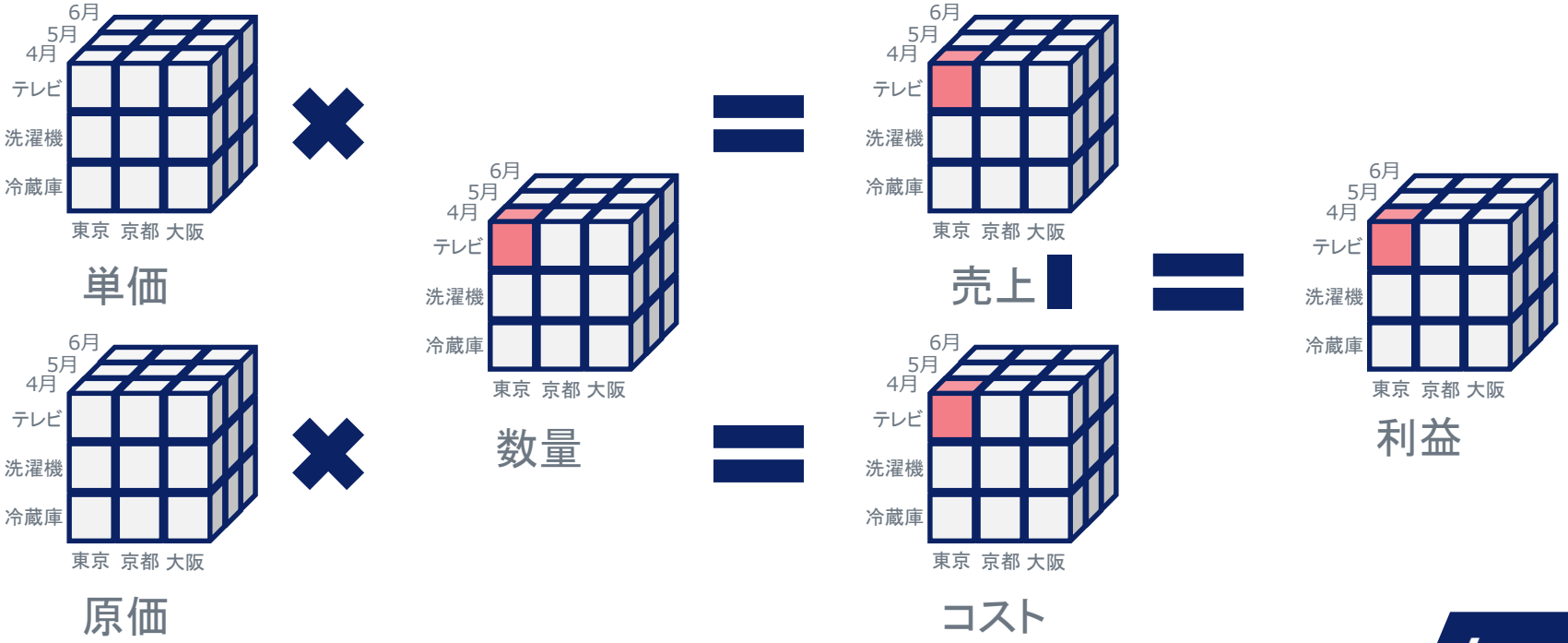
多くのLine Itemでの計算

「単価 × 数量 = 売上」「原価 × 数量 = コスト」「売上 – コスト = 利益」の場合。



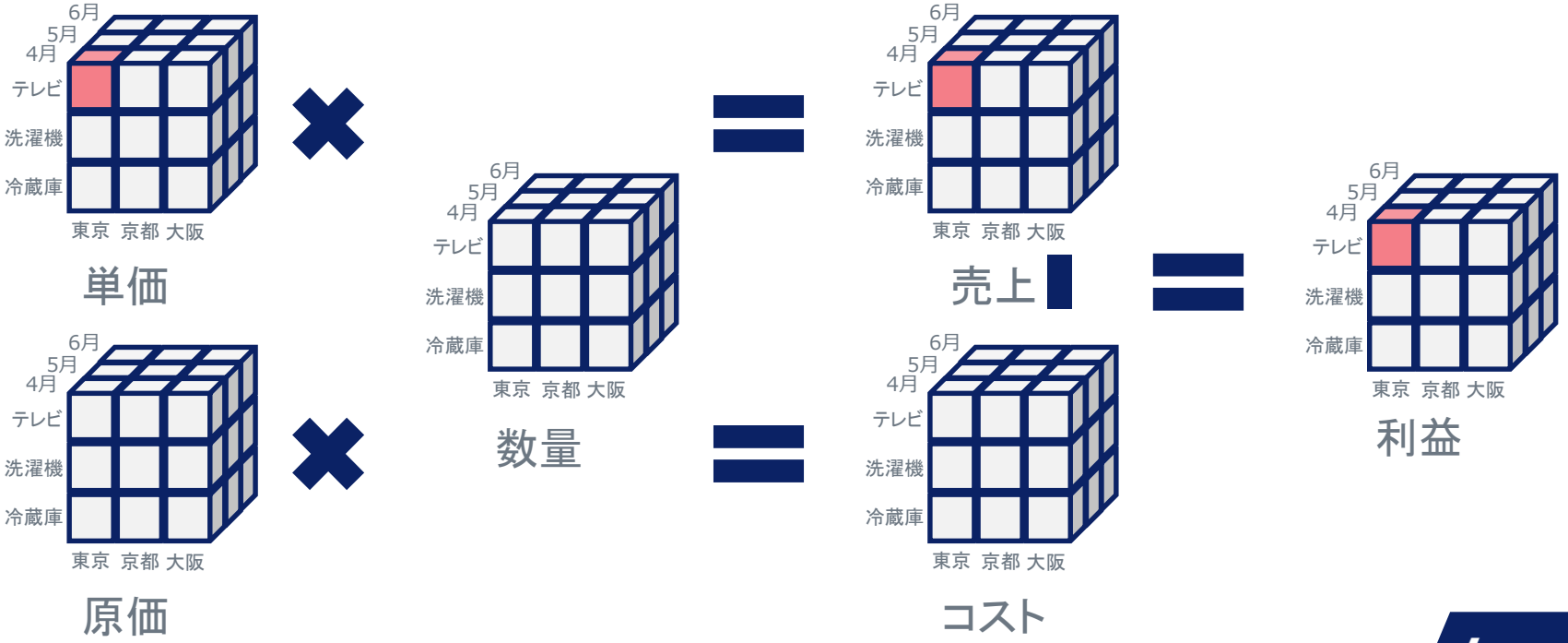
多くのLine Itemでの再計算

数量に変更があった場合の再計算対象。参照先は一瞬で再計算される。



多くのLine Itemでの再計算

単価に変更があった場合の再計算対象。コストは再計算されない。



非機能要件テクニック

機能実装に必要なテクニックではなく、パフォーマンスや容量、アクセス権限といった非機能要件を満たすための重要テクニックを紹介します。

軽めのユースケースでは、ここまで気にしなくてもいいですが、重たいユースケース、特にSCMではここまで気にしてください。

Anaplan

Performance

快適な速度を保つためにできること

Anaplan

パフォーマンスとは?

ここでいうパフォーマンスとは、Anaplan Platformの応答速度を意味しています。

下記それぞれのパフォーマンスに対して、影響する要因と改善方法を解説します。

- Cellに入力する
- ListのItem(要素)を増やす
- Importする
- Dashboardを開く
- Modelを開く
- ログインする

Cellに値を入力する際のパフォーマンス

Cellに値を入力後、そのCellを参照しているCellが連鎖的に再計算されます。

[1 Cellあたりの計算時間]を再計算対象になるCellすべての分を合計したものが、再計算の時間になります。

なので、以下2つのアプローチでパフォーマンスを改善します。

- 1 Cellあたりの計算時間/処理を減らす。
- 再計算されるCellの数を減らす。→これは後半の容量削減で触れます。

大量のIF

Model Builder

AnaplanのLine Itemに入れたFormulaは、Cellの数だけ実行されます。
1万CellあるLine Itemに10個のIFを入れると、最大で10万回のIF判定が行われます。
様々な工夫で、IFは減らすことが可能です。

ListのItem別にIF判定をするなどはMapping情報を元にLOOKUPに変更可能です。

IF ITEM(List X) = List X.aaa THEN VALUE[SELECT:List Y.bbb] ELSE
IF ITEM(List X) = List X.ccc THEN VALUE[SELECT:List Y.ddd] ELSE
IF ITEM(List X) = List X.eee THEN VALUE[SELECT:List Y.fff] ELSE …

↓

VALUE[LOOKUP:'List XにMappingしたList Y']

(※ List Xに対して、List YのMapping情報をもたせます。)

Anaplan

Text処理

Model Builder

Text処理の関数は、数値処理の関数より10倍処理負荷が高いです。
Anaplanが使っているjavaの仕様上避けられません。
それでもIn Memoryなので一般的なアプリケーションより処理は早いです。

TEXT, SUBSTITUTE, LEFT, RIGHT, MIDなどを多用する際は、以下を検討してください。

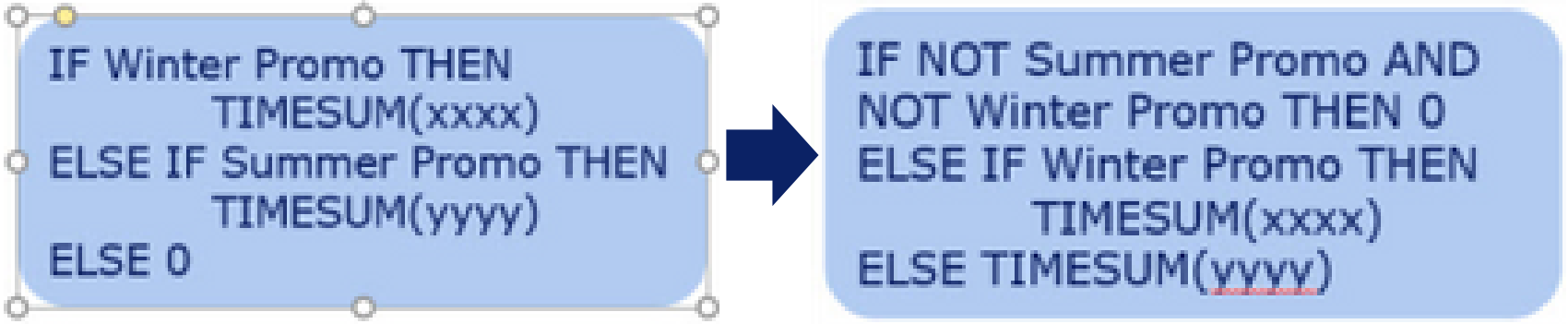
- 数値をTEXTにして処理している。→MODやROUNDで数値処理として検討。
- データ連携上避けられない。→Data Hubの検討
- Textの取込や入力のたびに再計算する必要がない。→ReferenceではなくImportに変更

Early Exit(1)

Model Builder

IFの条件句には、Hit率の高いものを先に置いて、Early Exitすることで、IF判定の実行回数を減らすようにしましょう。

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Summer Promo						✓	✓	✓				
Winter Promo	✓											✓



Early Exit(2)

Model Builder

先程の例で、IFの条件句を”NOT Summer Promo AND NOT Winter Promo”と改善していましたが、ここも、No Promoとして別Line Itemを作っておくと、さらにパフォーマンスが改善します。

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Summer Promo						✓	✓	✓				
Winter Promo	✓											✓
No Promo		✓	✓	✓	✓				✓	✓	✓	

```
IF No Promo THEN 0
ELSE IF Summer Promo THEN
TIMESUM(yyyy)
ELSE TIMESUM(xxxx)
```

Anaplan

FINDITEMの前にIF

Model Builder

検索値がBLANKになっている可能性が高い場合、以下のようにFormulaを改善しましょう。

FINDITEM(List, Value)
↓
IF ISBLANK(Value) THEN BLANK ELSE FINDITEM(List, Value)
または
IF ISNOTBLANK(Value) THEN FINDITEM(List, Value) ELSE BLANK

BLANKでのFINDITEMはすべてのITEMを検索した上で、BLANKを返すので、検索対象数が多い場合は、パフォーマンスが低くなります。

LAG, OFFSET, POSTの前にIF

Model Builder

LAG, OFFSET, POST関数を実行しなくて良い条件になっている可能性が高い場合は、以下のようにFormulaを改善しましょう。

OFFSET(xx, DAYS,0)

↓

IF DAYS = 0 THEN xx ELSE OFFSET(xx, DAYS,0)

または

IF DAYS <> 0 THEN OFFSET(xx, DAYS,0) ELSE 0

RANKの多用

Model Builder

RANKやRANKCUMULATEは、Cellの数だけList全件に対しての比較処理をするので、通常の数値計算より処理負荷が高いです。
計算内容に代替方法がある場合は、そちらを利用しましょう。

業務要件上、必須である場合もあるので、可能な限り処理負荷が減るように実装しましょう。

※)RANKやRANKCUMULATEの再計算が行われるListが10000件以上新規追加されるようなImportは、行えないように制御されています。

Applies Toの順番

Model Builder

計算結果の精度にはなにも影響ありませんが、ModuleのApplies Toの順番は参照元と参照先で揃えるようにしましょう。

- Module作成時のGeneral Listの順番でApplies Toは作られます。 General Listの順番を変更しても、 Applies Toは作成時の順番のままです。
- Applies Toの三点リーダーから「ダイアログを開いてOK」をすることで、各ModuleのApplies Toの順番は、 General Listの順番も揃います。

参照はApplies ToにあるDimensionの順でMappingが行われます。

技術的な詳細は以下のリンクから。

https://community.anaplan.com/t5/Knowledge/Dimension-Order/ta-p/32934

Line Itemは少なければいい?

Model Builder

Formulaを書く際に、Line Itemを複数使わずに、IF文をたくさん使ってLOOKUPやSUMも含めて、1つのLine Item内でいろいろな計算をやってしまう書き方もできます。

これは容量節約にもなるので、一見メリットに見えます。

しかし、四則演算するLine Item, IFするLine Item, LOOKUPするLine Item, SUMするLine Itemをそれぞれわけて実装した方が圧倒的にパフォーマンスが改善します。

https://community.anaplan.com/t5/Knowledge/Formula-Structure-for-Performance/ta-p/33177

Line Itemをわけましょう。

Model Builder

◆1つのLine Itemで実装した場合

Total Profit = aaa[LOOKUP:bbb] + ccc[SUM:ddd] - eee*fff[LOOKUP:ggg] – (IF hhh THEN III ELSE jjj) - kkk[SELECT:lll]

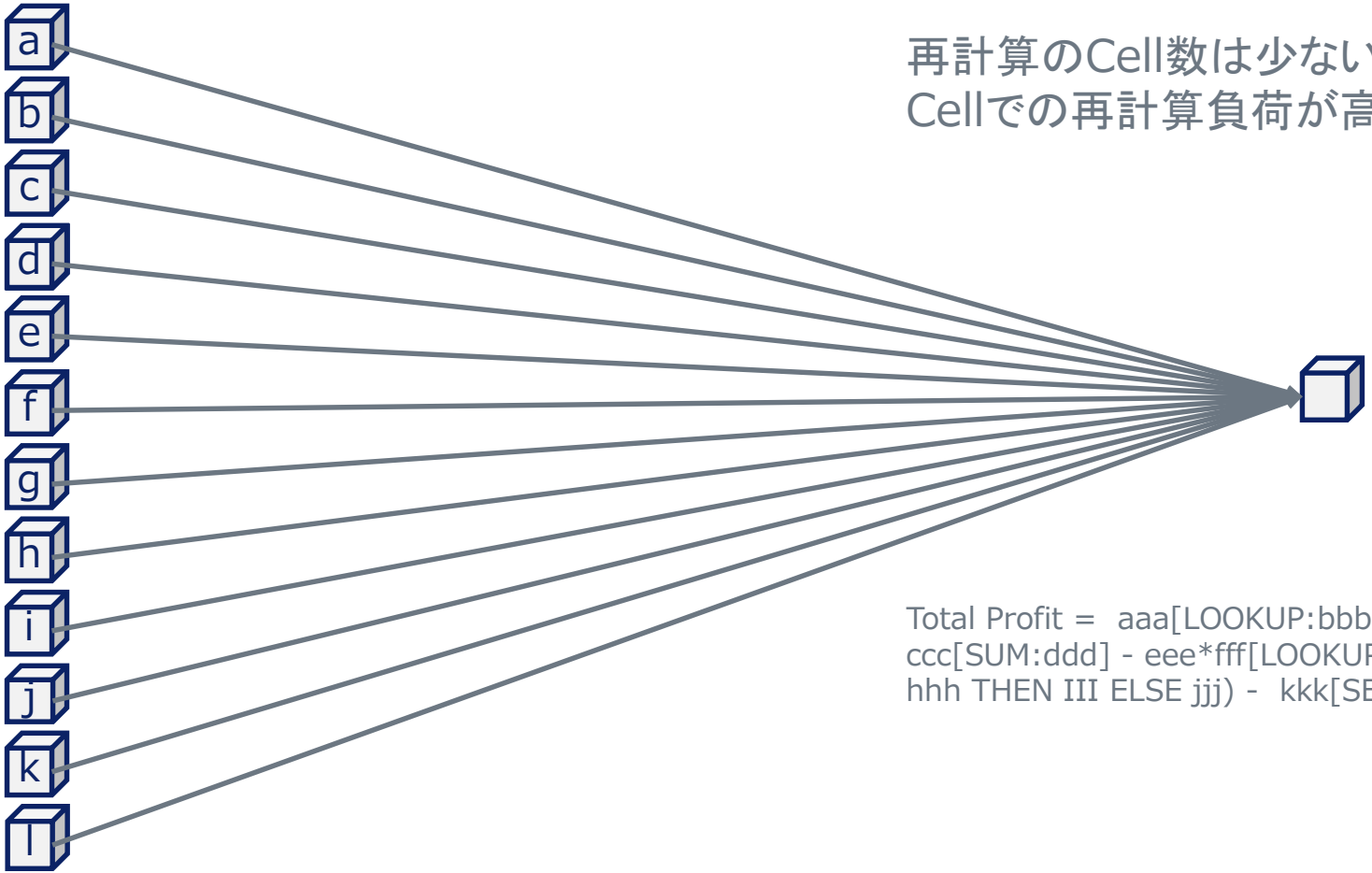
◆関数の単位でLine Itemをわけた場合

Product Sales = aaa[LOOKUP:bbb]
Services Sales = ccc[SUM:ddd]
Opex = eee*fff[LOOKUP:ggg]
COGS = IF hhh THEN III ELSE jjj
Rent&Utilities = kkk[SELECT:lll]

Total Profit = Product Sales + Services Sales - Opex - COGS - Rent&Utilities

Anaplan

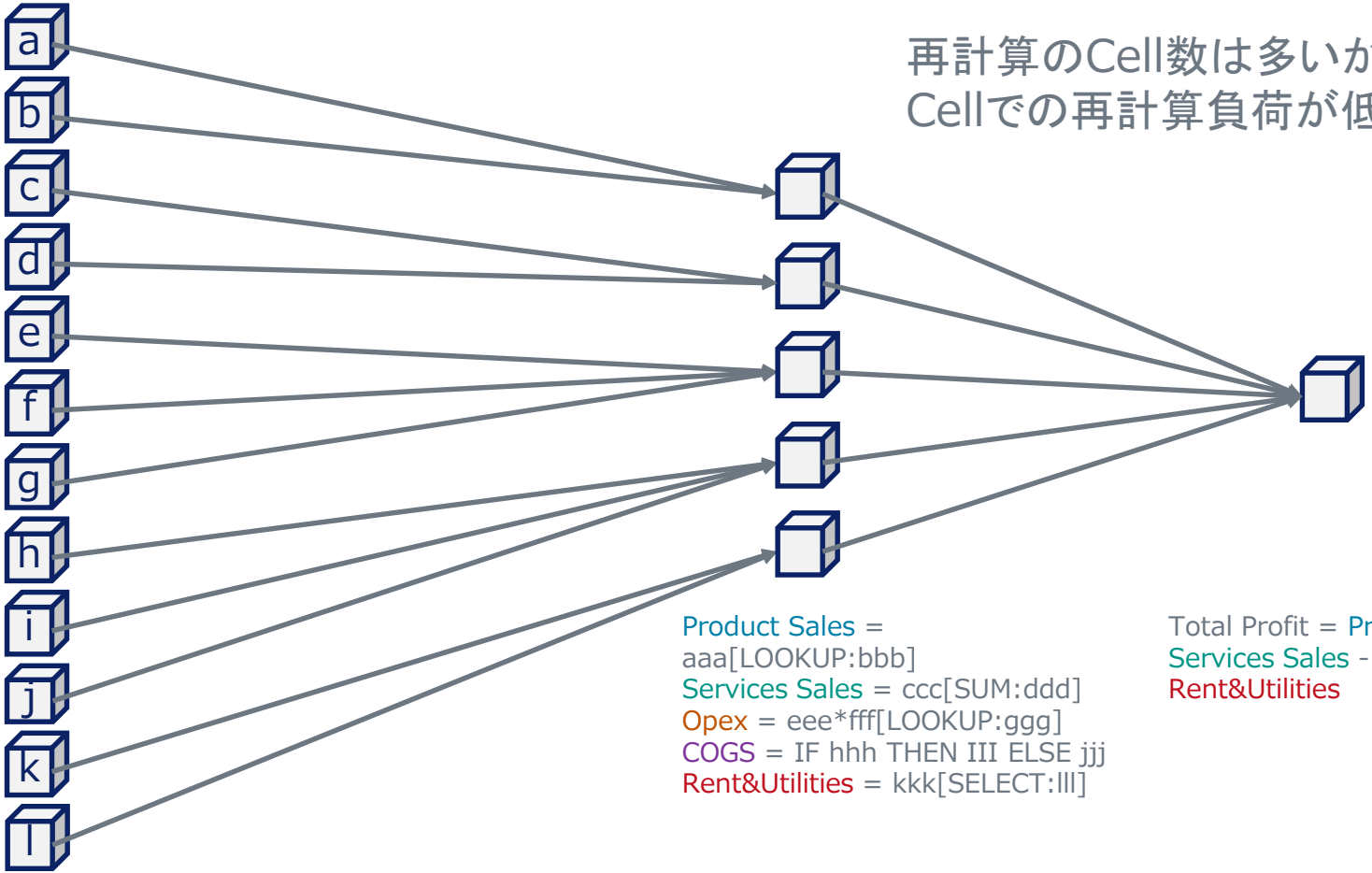
Before



再計算のCell数は少ないが、1つのCellでの再計算負荷が高い。

Total Profit = aaa[LOOKUP:bbb] + ccc[SUM:ddd] - eee*fff[LOOKUP:ggg] – (IF hhh THEN III ELSE jjj) - kkk[SELECT:lll]

After



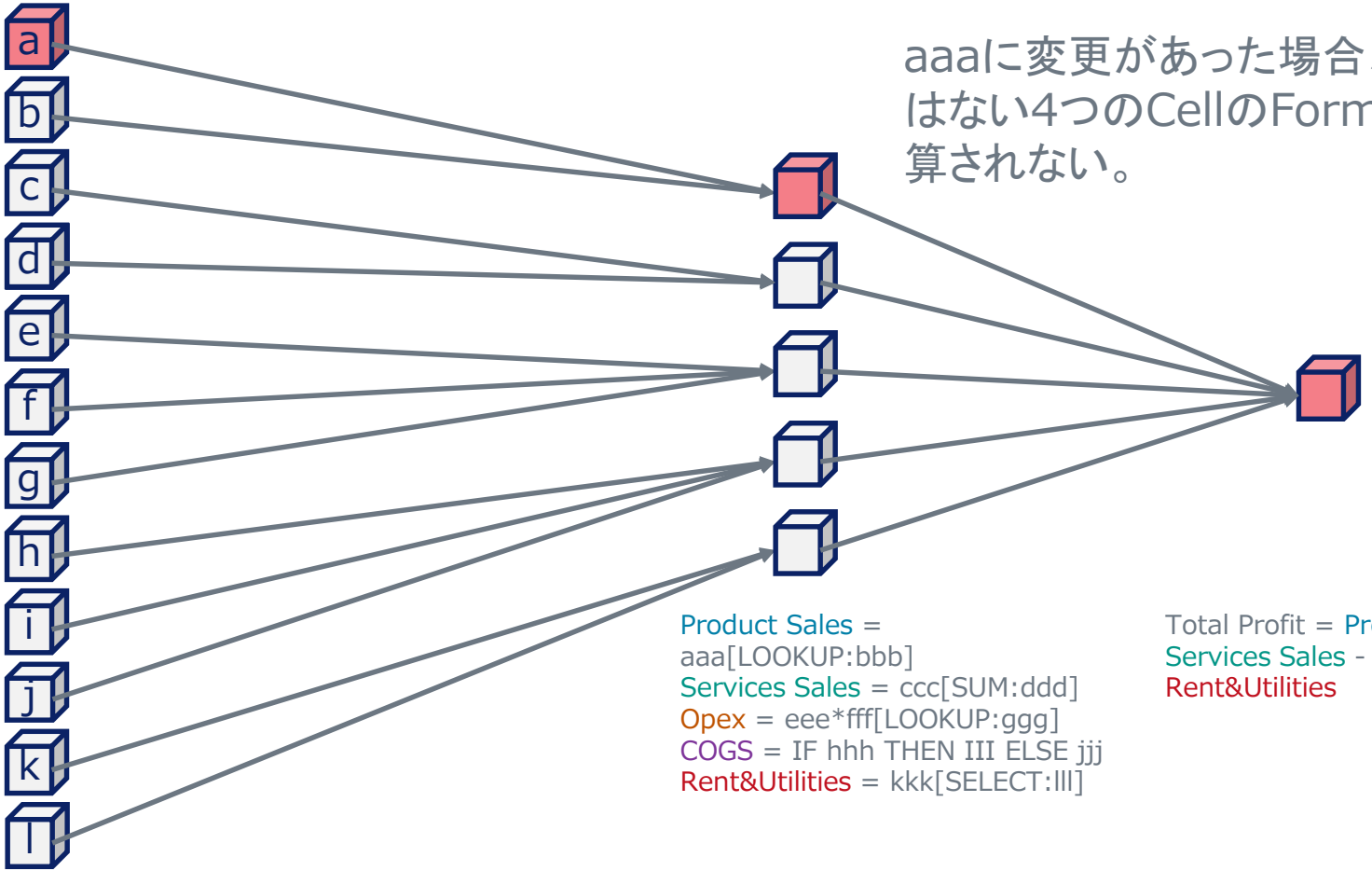
再計算のCell数は多いが、1つのCellでの再計算負荷が低い。

Product Sales = aaa[LOOKUP:bbb]
Services Sales = ccc[SUM:ddd]
Opex = eee*fff[LOOKUP:ggg]
COGS = IF hhh THEN III ELSE jjj
Rent&Utilities = kkk[SELECT:lll]

Total Profit = Product Sales + Services Sales - Opex - COGS - Rent&Utilities

Anaplan

Afterの再計算



aaaに変更があった場合、参照先ではない4つのCellのFormulaは再計算されない。

Product Sales = aaa[LOOKUP:bbb]
Services Sales = ccc[SUM:ddd]
Opex = eee*fff[LOOKUP:ggg]
COGS = IF hhh THEN III ELSE jjj
Rent&Utilities = kkk[SELECT:lll]

Total Profit = Product Sales + Services Sales - Opex - COGS - Rent&Utilities

Anaplan

わけるとどう変わる?

Model Builder

Line Itemをわけてもわけなくても初回の計算時のパフォーマンスはあまり変わりません。
しかし、計算元になっている数値が1つでも変わった場合の再計算処理の量が異なります。

Anaplanでは、数値変更後は、Referenced Byで繋がっているLine Itemのみが再計算されていきます。

1つのLine Itemですべての関数を実行すると、参照先のすべてのLOOKUPやSUMが再計算されますが、Line Itemを分けておくと、再計算する必要のない関数は実行されません。

Line Itemをわける簡単な例

以下のFormulaの場合、cに変更があった場合に、LOOKUPが実行されず、再計算の量が減ります。

IF a[LOOKUP: b] = c THEN d ELSE 0

↓

X = a[LOOKUP: b]

IF X = c THEN d ELSE 0

Listの要素を追加/削除する際にかかる時間

Actionボタンを利用して、ListのItemを追加する場合、Itemが増えたことによって、ModuleのCellが増加します。

そのCellに関連する再計算の時間がかかります。

Importする際のパフォーマンス

◆外部ファイルからImportする場合
Importにかかる時間 = ファイル転送にかかる時間 + Cellが再計算する時間

◆Anaplan内でImportする場合
Importにかかる時間 = データを取得する時間+ Cellが再計算する時間

以下のような特徴があります。
- Moduleへの取込はListへの取込よりはやい。
- Moduleからの取込はFileからの取込より5~6倍はやい。

Import時の再計算とは

Model Builder

Importの速度はデータを取り込む時間だけでなく、再計算の時間も必要です。

- 更新された値が参照されている場合、都度再計算される。
- 同じ値を取り込んでも再計算される。

「大容量の全件洗い替えのトランザクションデータをFileからListへ直接取込していて、Propertyが直接参照されていて再計算されるようになっている。」などの条件が重なると取込速度が遅いです。

(ここに複雑なテキスト処理やエラーが多発、など加わると、極端に遅くなります。)

Import時の再計算の速度改善

Model Builder

以下のような実装を行って、Importのパフォーマンスを改善できます。

- 元ファイルを全件ではなく差分にする。
- 元ファイルが全件の場合、参照のない仮取込のListとModuleを作成し、全件取込後に差分を洗い出して、差分のみをFilterで抽出し、再度Importする。
- 外部システムのFileのデータなどは、Data Hubへの取込にする。

※) メンテナンス性が下がり過ぎない程度にしましょう。

※) 参照もPropertyもないListであれば100万件の取込でも約3秒で終わります。

Anaplan

Import Errorは0に

Model Builder

Import速度は、Errorが増えるほど劣化します。
以下のようにエラーをなくすようにしましょう。

- キーにNo Valueがないようにする。(ISNOTBLANK)
- 重複取込は無くす。(ISFIRSTOCCURRENCE)
- 手動マッピングの設定では、不要なものには必ずIgnoreを設定する。
- Source側で、可能な限り不要な情報をないようにしておく。(Hide)

※) 劣化といっても数秒~数分単位の話です。
※) パフォーマンスもありますが、Import後の警告表示は無いほうが見栄えがいいです。

Dashboardを開く際にかかる時間

Dashboardを開く際のパフォーマンスは、1つのDashboardに載せているModuleとChartの数に依存します。多いほど遅くなります。

1つのDashboardでのModuleやChartの数は可能な限り5個以下にしましょう。

※) 表示するCellの値などは、サーバー側で計算済みなので、「Dashboardを開くたびに再計算」などはありません。

※) サーバー側の計算負荷ではなく、クライアント側(ブラウザ)のレンダリングの負荷によるものなので、Model自体の容量やパフォーマンスとは異なる理由でパフォーマンスへ影響がでます。

Filterの条件はBoolean

Model Builder

Filterの条件で、Value >100などの判定をして表示するよりも、

Boolean FormatのLine ItemでValue >100の判定を済ませておいて、そこのTRUE,FALSEをFilterの条件にした方が、Dashboard描画のパフォーマンスが向上します。

Anaplan

Sortの多用

Model Builder

Sortは、Moduleをブラウザで描画する際の処理を増やします。
Sortを多用すると、Dashboardを開く際のパフォーマンスが劣化します。

Anaplan

Moduleの大きさとDashboard?

1万CellのModuleと100万CellのModuleがそれぞれDashboardにPublishされている時に、どちらの方がはやくDashboardが開くでしょうか?

結論としては、Module全体のCellの数はDashboardを開く際のパフォーマンスに影響しません。

Dashboardを開く際のパフォーマンスに影響するのは、Dashboard上で目に見えるCellの数です。Page Selectorで隠れている分の読み込み時間はかかりません。

また、Cellが多く、スクロールが発生している場合、スクロールで隠れている分はスクロールするタイミングで読み込まれるので、Dashboardを開く際にかかる時間も上限はあります。

AnaplanのModelを開く際にかかる時間(1)

1時間以内にModelにアクセスがあったかどうかで、応答時間が異なります。

①1時間以内に誰もModelを開いていない場合
Model Loadの時間 + ブラウザで開く時間 + Landing Dashboardを開く時間

②1時間以内に誰かがModelを開いていた場合
ブラウザで開く時間+ Landing Dashboardを開く時間

AnaplanのModelを開く際にかかる時間(2)

- Model Loadの時間

Anaplanのサーバー上で用意をする時間。Modelをメモリ上に展開して、Model上のすべてのロジックを再計算している時間。

格納しているデータ量、計算ロジックの複雑さや煩雑さ、データの粒度などに依存。

- ブラウザで開く時間

Anaplanのサーバー上のデータを、インターネット経由でブラウザに読み込んでブラウザでレンダリングするのにかかる時間。

要因は多くあるものの、多くの場合、ネットワークの帯域と利用しているPCとブラウザのスペックに依存。

ログインにかかる時間

ネットワークによる影響が最も大きいです。

クライアントのネットワーク環境(モバイルWifiの利用や、セキュリティ対策、社内でのルーティング)に依存いたします。

また、Anaplanのデータセンターが日本国外にあることもあり、現在日本では距離による時間もかかってしまっています。

Performance改善の参考

https://community.anaplan.com/t5/Knowledge/What-are-some-good-practices-to-improve-model-performance/ta-p/33550

https://community.anaplan.com/t5/Knowledge/Reduce-Calculations-for-Better-Performance/ta-p/33667

https://community.anaplan.com/t5/Knowledge/Formulas-and-their-effect-on-model-performance/ta-p/33556

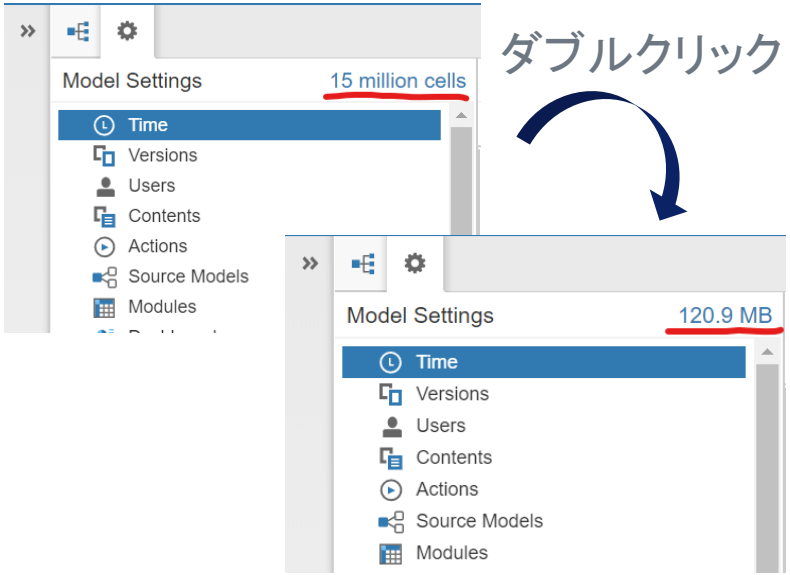
Model Sizing

87%の人が1ヶ月で3GB以上痩せた!

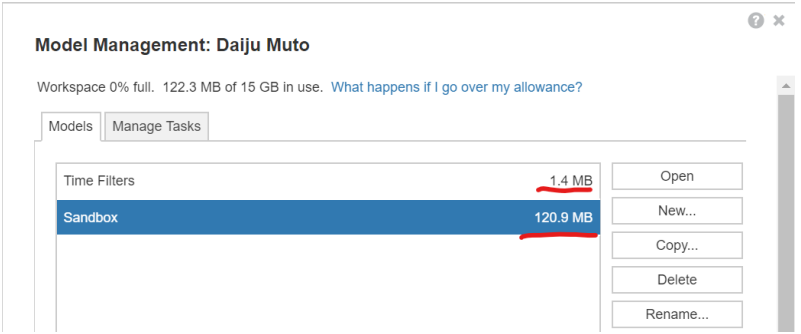
Anaplan

使用している容量の確認方法

1. Model Settingsで確認



2. Manage Modelで確認



容量概算の計算方法

[容量] = [Cellの数] * [Format別のByte]/2^30 GB （※1GB=2^30 Byte）

[Cellの数] = List Aの数 * List Bの数 * List Cの数・・・
[Format別のByte]は、以下の表。

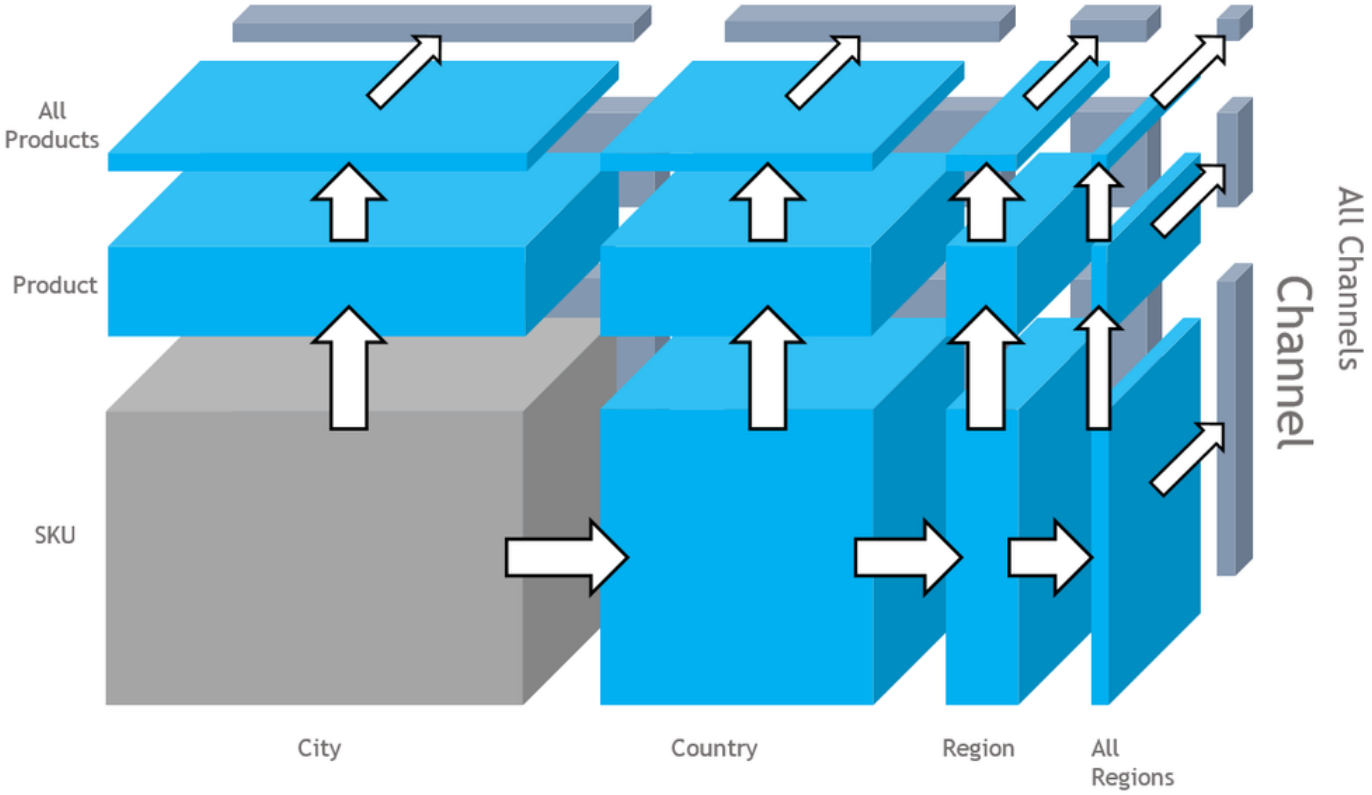
	Parent	Code	Data Type	Byte
Number	All		NUMBER	8
Boolean	All		BOOLEAN	1
Date	All		DATE	4
Time Period	All		TIME_ENTITY	8
List	All		"ENTITY"	4
Text	All		TEXT	8
No Data	All		NONE	0
All				

容量算出の例

製品:100
顧客:100
月数:12
Version:3
受注数、売上額という2種類のデータ(Line Item)を持つ場合

Cellの数 = 100*100*12*3*2=720000
容量 = 720000*8/2^30=0.00536GB=5.36MB

ParentがCellの数に与える影響



ListのItem(要素)

Listの中のItem(要素)それぞれは、約500 byte(= 0.5KB)の容量を使います。
Cellの数にカウントされる容量とは別にカウントされます。

ListのItemが、100万件で0.5GB程度です。
※)全体容量にほぼ影響を与えないので、概算を出すときには無視しています。

Properties追加による容量は、ModuleのCellと同じ計算になります。
Listは1次元なので、100万件のListにText FormatのPropertyを1つ追加しても8MBにしかなりません。

容量計算に含まれないもの

計算時にMemoryに展開されているものが容量に含まれるので、以下のようなものは容量計算対象外です。

- Dashboard(Personal Dashboard含む)
- Action
- Import Data Source

容量を削減するとは?

容量を削減するためにできることは大きく以下の2つです。

- Cellの数を減らす。
- 1つのCellに必要なByteを減らす。

後者でできることと言えば、0または1のフラグになっているNumberを、Booleanに変更するくらいしかできません。
前者の「Cellの数を減らす。」ことに注力して容量を考慮しましょう。

多次元データベースでは、「本来存在しない組み合わせ」による無駄領域(Sparsity)をいかに減らすか?が容量削減のポイントになります。

https://community.anaplan.com/t5/Blog/The-Truth-About-Sparsity-Part-1/ba-p/44493

Anaplan

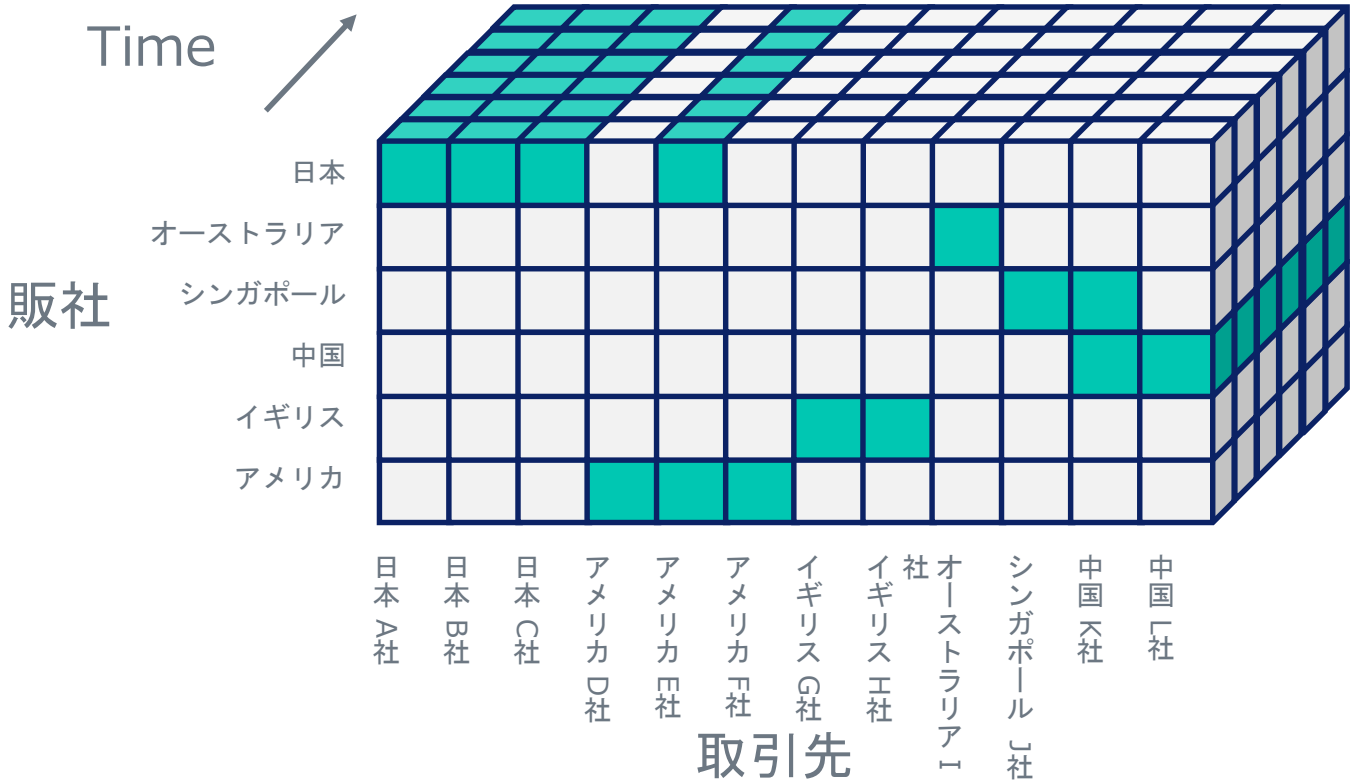
中身がスカスカなLine Item

データのキーになるListをすべて選んでModuleのDimensionにすると、Cell数は膨大な一方で中身がスカスカになってしまう場合があります。

例えば、販社と取引先の場合を考えてみてください。
「日本の販社は基本的に日本の会社」「アメリカの販社は基本的にアメリカの販社」のように、基本的には国別で販社と取引先が決まっているけれど、日本の販社も一部のアメリカの会社と取引がある、というような場合があります。

こういった場合に、販社Listと取引先ListをDimensionにModuleを作ると、本来持つ必要のない無駄領域(Sparsity)が大量に発生してしまいます。

中身がスカスカな例



販社 6
＊
取引先 12
＊
6ヶ月
＝ 432 Cell

Anaplan

Combination List

Model Builder | Designer

キーになるListで作ったModuleでデータを保持する方法が、最初の観点となりますが、そうすると「データはスカスカなのに容量をたくさん確保してしまう。」という場合があります。

こういった場合に、要素の組み合わせだけを保持するListを別に作る(Combination Listと呼びます。)ことで、大幅に容量を削減することができます。

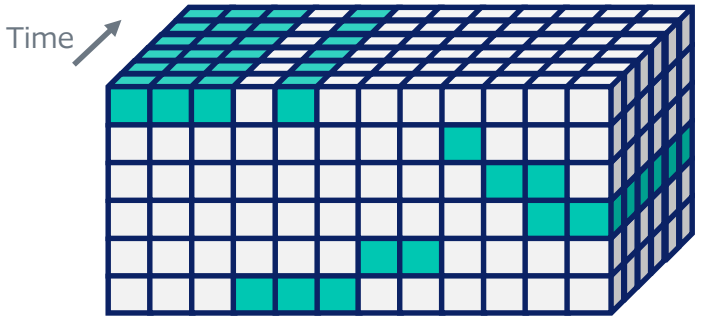
実装サンプルは逆引きAnapediaにもあるので、参考にしてください。

https://community.anaplan.com/t5/Anaplan-Discussions/Create-list-which-is-a-combination-of-multiple-lists/m-p/30227

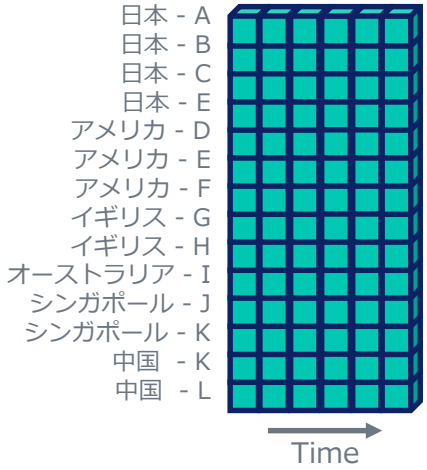
Anaplan

Combination Listにすると?

Before



After



Combination Listを作成すると、実在する販社と取引先の組み合わせは14なので、14 * 6 = 86 Cellになります。

これで、432 Cellが 86 Cellになったので、約80%の容量削減になります。

Subset

Model Builder | Designer

Listの要素すべてに対して計算する必要がない場合、Subsetを利用しましょう。
メンテナンス性が下がらない程度にSubsetを用意しましょう。

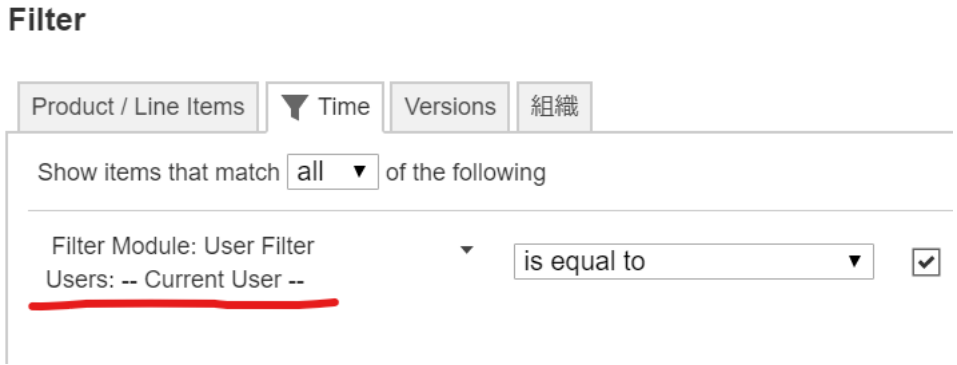
Timeに対してもTime Rangeの機能でSubsetと同様のことができます。

VersionsはSubsetを持てないので、Custom VersionにしてSubsetを使いましょう。

User別のFilterやDCA

Model Builder | Designer

User別に表示を制御したい場合、FilterやDCAの条件側のLine ItemのDimensionにUsersを加えることで、FilterやDCAをかけた際に、利用ユーザーが自動的に特定(Current Userとして)されて、ユーザー別の表示/非表示やWrite/Readの制御が可能です。



User別のFilterやDCAの容量

Model Builder

Designer

この機能はとても便利ですが、1Model内のユーザー数に比例して容量が大きくなるので、Userが多い場合は、Page Selectorでの絞り込みやFilterをエンドユーザーに直接触ってもらうようにするか、または、Distributed Modelを検討しましょう。

※)　Usersを加えるのは条件側のLine Itemだけで、FilterやDCAをかけられる側のLine ItemにUsersを追加する必要はありません。

Line ItemのSummary

Model Builder

CellはListのItemの数の掛け算で作られますが、ListのParentのItemについても同様にCellが作られます。

この部分のCellは、SummaryによってFormulaとは異なる動きが設定できます。

InputやOutputでは数値の合計値を表示することが多いですが、Calculationでは、多くの場合で不要です。(SummaryのSELECTやLOOKUPを除く。)

不要な場合は、Noneに設定することでCellが作られず容量が削減できます。

	Formula	Parent	Is Summary	Format	Applies To	Time Scale	Time Range	Versions	Start of Section	Brought-Forward	Summary	Cell Count
入力Module					Department, Product	Month	Model Calendar	All				227,670
数量			☐	Number	-	Month	Model Calendar	All	☐	☐	None	83,616
売上			☐	Number	-	Month	Model Calendar	All	☐	☐	Sum	144,054

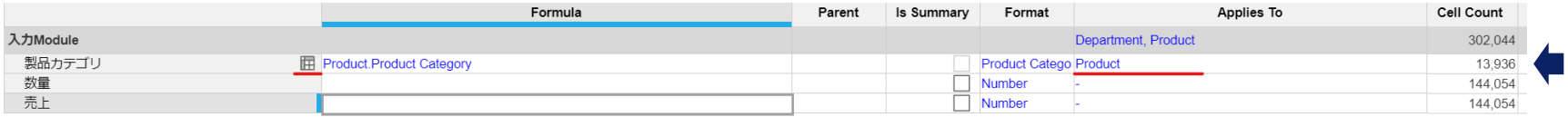
Anaplan

Subsidiary View

Model Builder

Module内のLine Item毎にDimensionを増やしたり減らしたりすることができます。

データに対してそのDimensionが不要な場合は、減らすことで容量が節約できます。



	Formula	Parent	Is Summary	Format	Applies To	Cell Count
入力Module					Department, Product	302,044
製品カテゴリ	Product.Product Category			Product Catego	Product	13,936
数量				Number	-	144,054
売上				Number	-	144,054

Cell数の少ないで計算する

Model Builder

最もDimensionの掛け算の多いModuleに、あらゆるものを取得して計算ロジックを書くのではなく、可能な限りDimensionの掛け算の少ないほうで前処理の計算をしておいて、 Dimensionの掛け算の多いModule側では、その粒度の細かさが必要な計算のみを行うようにしましょう。

User roles

Model Builder

定義されたUser roleの種類と、それぞれに与えられているListへのアクセス権限に対して、ListはMemory上に展開されて容量を消費します。

Role毎に必要なListへの最低限なアクセス権限を設定するようにしましょう。

※) Model Settingsに表示されている容量には反映されていません。

※) Listは100万件でも0.5GBなので、そこまで重要ではないです。

不要なものは消す

Model Builder

以下のようなものがないか、今一度確認して、消せるものは消しましょう。
特に、複数人で開発していると気が付かないうちに同じものを作っている場合があります。

- Viewを変更すれば同じ見た目を作ることのできるOutputのModule。
- どこにも参照されていないCalculationのModuleとLine Item。
- SUMの条件にするために追加したLine Itemが重複している。
- IFの条件になるBooleanのLine Itemが重複している。

Access Control

かなり細かく設定できます。

アクセス権の機能

Model Builder | Designer

ここではModelに入ってからのアクセス権限についてまとめます。
エンドユーザー別にデータのRead/Writeを制限する機能は4つあります。

▽ListのItem毎でそれに紐づくCellの値のRead/Writeを設定したい場合
- Selective Access
- Filter ※Write制御不可。Read制御のみ可。
- DCA

▽Dashboard/Module/List/Actionの単位でアクセス権限を設定したい場合
- Role

例えば

各部門別で、「営業員が数字を提出し、上司が意思入れや承認を行うと、本部に数字が渡る。」という利用方法の場合、

Selective Accessで、部門別(Item別)のアクセス権限を設定し、
Roleで、提出者と承認者の権限を設定する。(承認Moduleは承認者Roleだけ編集可)
などの使い方をします。

他には、組織と勘定科目の組み合わせで、
人事部だけが人件費の詳細を閲覧できて、他部門は人件費の閲覧権限はない。
などのCell単位でのアクセス権限が必要な場合は、DCAを利用します。

アクセス権機能の選び方

Model Builder | Designer

Selective Access, Role, Filter, DCAのどれを利用するかの観点を記載します。

Model全体で一括で設定可能なのは、 Selective AccessとRoleです。
Dashboard毎やModule毎の設定が不要なので、実装工数を減らすには、この2つがオススメです。

一方で、最も細かくアクセス権限を設定できるのは、DCAです。
しかし、DCAはLine Item単位で設定しなければならないので、実装負荷が高いです。

可能な限り必要な権限設定を初期設計時に洗い出し、 Selective AccessとRoleに寄せて、細かい部分をDCAで拾う形で進めましょう。

アクセス権の制約事項

Model Builder | Designer

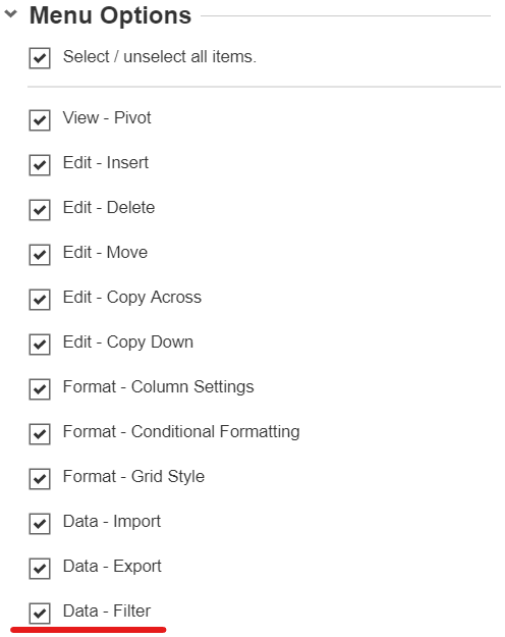
アクセス権限関連の機能の制約事項です。

- ListのItemの名称すら見せてはいけない。
  →Selective Accessのみ可。
- ListのItemに紐づくすべてのCellのデータのReadを許可しない。
  → Selective Access, Filter, DCAで可。
- Cell別のアクセス制限
  →DCAのみ可。
- Dashboard別のアクセス制限
  →Roleのみ可。

Filterでの権限制御?

DashboardでModuleを選択後、左側にでてくる設定欄(Properties Panel)で、Menu Optionを減らすことができます。

ここで、Filterを変更不可にすることで、Filterを外せなくなるので、閲覧制限行うことが可能です。



Anaplan

DCAとFilterの注意点

Model Builder | Designer

Selective Accessは設計時から考慮が必要ですが、DCAとFilterは後からいくらでも追加可能です。条件も柔軟に設定可能です。

しかし、 DCAとFilterは検索/進捗管理/過去入力値のロックなどの別機能の用途もあります。

なので、あまり条件を複雑にすると、なぜRead/Writeの権限がないのかわからなくなり、実装者もエンドユーザーも混乱してしまう可能性があるので、あまり複雑に多用するのは避けてください。

Roleの注意点

Model Builder | Designer

Roleの設定後に新規追加されたDashboard/Module/List/Actionは、デフォルトでReadもWriteもNoneになります。

また、Dashboard上のModuleに1つでもアクセス権がないと、Dashboardへのアクセス権を設定していても表示されなくなります。

▽よくあるお問い合わせ

「Dashboardにアクセス権限を設定しているのに、急にDashboardが表示されなくなりました。」

→新規作成したModule/ChartをDashboardにPublishした際に、よく起こります。新規作成後は忘れずアクセス権を設定しましょう。

導入プロジェクトでは、実装期間中ではなく、評価直前に設定しましょう。

Anaplan

Appendix

この資料で推奨されている内容を理解するための、製品仕様や技術の裏側をまとめました。

全体に影響のあるAppendix

1番難しい基本的な内容

Anaplan

PLANS

この資料で言われている「健全なModel」とは、以下に紹介されるPLANSと同等のことです。

Performance / 耐障害性
Logical / 論理性
Auditable / 可読性
Necessary / 必要性
Sustainable / 拡張性

https://community.anaplan.com/t5/Knowledge/PLANS-This-Is-How-We-Model/ta-p/33530

個別機能のAppendix

知っていると便利

Anaplan

前提知識のLink集(1)

User別のFilter

https://help.anaplan.com/anapedia/Content/Modeling/Working_with_Data/Current%20User%20Filter.html

DCA

https://help.anaplan.com/anapedia/Content/Modeling/Working_with_Data/DynamicCellAccess.htm
https://community.anaplan.com/t5/Knowledge/Dynamic-Cell-Access-Tips-and-Tricks/ta-p/33871

Conditional Formatting

https://help.anaplan.com/anapedia/Content/Modeling/Working_with_Data/Conditional_Formatting.html

前提知識のLink集(2)

Line Item Subset

https://help.anaplan.com/anapedia/Content/Modeling/Dimensions/Line_Item_Subsets.html

Publish(Publish Line Items to a Dashboard)

https://help.anaplan.com/anapedia/Content/Dashboards_and_Visualization/Build%20Dashboards/Publishing_to_a_Dashboard.html

https://help.anaplan.com/anapedia/Content/Dashboards_and_Visualization/Build%20Dashboards/Lists_Properties_and_Subsets.html

Dependent dropdown list

https://help.anaplan.com/anapedia/Content/Modeling/Build%20Models/Dropdowns/Many-to-many_Filtered_Picklists.htm

前提知識のLink集(3)

View/Saved View

https://help.anaplan.com/anapedia/Content/Modeling/Navigate%20Anaplan/View.html

Import/Combination of properties

https://help.anaplan.com/anapedia/Content/Import_and_Export/Import_Data_into_Models/Import_Data_into_Lists.html

Importの挙動

2種類あります。ListへのImportとModuleへのImportです。
(例外として、BlueprintへのImportもあります。)

Data SourceがAnaplan内のListやModuleであっても、Fileと同じ扱いになります。
なので、ListがParentを持っていて、Sourceでその部分が表示されていると、Parentの部分についてもImport対象になってしまいます。
List内に同じ要素がある場合は、数値は合算され、ない場合はImportの警告になります。

Import定義の単位

Import定義は、Import Data SourceとTargetの名前を元に、一意に定義されます。

同名の組み合わせのImportを行うを、定義が上書きされます。

Saved Viewを使って、Importはわけましょう。

Admin権限でのImport

Importを実行する際にWorkspace Admin権限があると、Selective AccessとDCAで設定されているWrite権限を無視して、すべてを更新します。

▽Dynamic Cell Access

黒字(Read Only)やInvisibleのセルに対しても、Adminは書き込み可能。

▽Selective access

Admin権限のアカウントにSelective Accessの対象が設定されている場合、Target側のWrite権限は無視してすべてのItemが更新対象になります。

一方で、Source側のRead権限は有効なままなので、閲覧権限があるデータのみがImportの対象になります。

ListのCODE

以下のような技術上の制約があるので、テキストを結合してCODEを作るような場合は、気をつけてください。

- 60文字以内
- 特殊な記号は使えない。
- アンダーバー_を末尾にはできない。
- 重複できない。

‘が必要な場合と不要な場合

AnaplanのFormulaでは、システム予約語と変数を区別するためのDelimiterとして、’が使われます。

以下のような場合は、Module名やLine Item名を’で括ってください。

- 名称に数字は入っている。
- 名称に特殊な記号(四則演算等)が含まれている。

‘が不要な場合、付けてFormulaを書いても保存されます。

‘をつけてFormulaを書くことで不都合はありません。

Chart作成ボタン

PivotでRowsを2つ以上にすると、Chart作成ボタンは無効化されます。
Rowsは1つ以下にしてから、Chartを作成してください。

Sortボタン

▽Sortできない条件

Sort対象がPivotでNestしていると、Sortボタンを押しても警告ができます。

▽Sort

Sortのダイアログ上で、下部のOKボタンがでなくなることがあります。

これは条件によって非表示になっているのではなく、ブラウザサイズが合わないことで表示できなくなっています。

OKボタンがでてくるまで、ブラウザの表示のサイズを縮小してください。

HideとShowの違い

ShowとHideの違いは、新規項目が追加された際の動きで、
Showは選択したものだけを表示するので、新規追加項目は非表示になり、
Hideは選択したものだけを非表示にするので、新規追加項目は自動的に表示される。

また、Showは、選択したItemの順番で並び替えもできる。
※)ShowでTimeを並び替えると、FYが変わるタイミングでShowし直しになるので、非推奨。

Tips

知っていると便利なテクニック集です。

本当はもっとありますが、効果的で簡易なもののみここで紹介します。

もっと知りたい場合は、逆引きAnapediaまたはAnaplan CommunityのBest Practicesを見てください!

Subsetに属しているかどうか

Subsetに属しているかどうかを条件判定に使いたいことがあります。
ListのPropertiesとSubsetをダブルメンテナンスするのは避けたいです。
そんなときには、以下のFormulaでPropertiesからSubsetの中身を判定してください。

ISNOTBLANK(ITEM(Subset))

Importの警告を表示したくない。

重複取込の警告にはISFIRSTOCCURNECE関数を使いましょう。

No Valueの警告にはISNOTBLANK関数を使いましょう。

ModuleからのImportであれば、これらを利用することで警告がでなくなります。

1つ前のList Itemの値を取得したい

ListのPropertyに、1つ前のList Itemを登録しておき、それを条件にLOOKUPすることで、値が取得できます。

これで、Custom TimeなどでもPREVIOUS関数などを再現できます。

※)参照元と参照先が同じLine ItemでLOOKUPすることはできません。これはPREVIOSUやNEXTでないとできません。

Listに紐づくデータを一括コピーしたい

Settings > VersionsにあるBulk Copyのボタンから、Listを選択して、コピー元とコピー先のListのItemを選択し、すべてのModuleのデータ(青字)をコピーすることができます。

※)Versionsのメニューに入っているので誤解を受けることが多いですが、Versionだけではなく、すべてのListで利用可能です。

※)このボタンをActionにすることはできません。下のRFEのKudoをクリックして応援してください。

https://community.anaplan.com/t5/Idea-Exchange/Add-Bulk-Copy-to-action-list-to-be-added-in-process-chain/idi-p/40525

User別Filter

Line Itemで条件を組んでFilterを作る際に、そのまま作ると全ユーザーでFilterの条件が同じになってしまいます。

しかし、Filterの条件は自動でCurrent Userが指定される機能があるので、User別Filterを組むことができます。

実装例は、逆引きAnapediaまたは以下のリンクを参考にしてください。

https://help.anaplan.com/anapedia/Content/Modeling/Working_with_Data/Current%20User%20Filter.html

Anaplan

ソートしたい(1)

3種類の方法があります。都合の良いものを選択して実装してください。

方法1: 並び替え用のLine Itemを使用して並び替える。

1. Moduleに、ソート用のLine itemを作成する。
2. 1で作成したLine itemに、並び変えたい順に数字を入力する。
3. Line itemをクリックで選択状態にし、Sortボタンをクリックし、並び替え方法を指定する。
4. 並び変わったことを確認する。

メリット: ModuleでSortの定義を管理することができる。

ソートしたい(2)

方法2: Listに作成したSort用のPropertiesを使用して並び替える。

1. Listに、Sort用のPropertyを作成する。
2. Moduleに、ソート用のLine itemを作成し、作成したPropertyを参照する。
3. 作成した列に、並び変えたい順に数字を入力する。
4. Line itemを選択状態にし、Sortボタンをクリックし、並び替え方法を指定する。
5. 並び変わったことを確認する。

メリット: ListでSortの定義を管理することができる。

ソートしたい(3)

方法3: ListのMember順を並び替える。(Module のSort 機能とは別案。)

1. Listの“Tree View”または“Grid View”で、並び順を変更したメンバーを選択し、Moveボタンをクリックする。
2. 2. 移動先を指定する。
3. 3. 並び変わったことを確認する。

メリット: Member の並びを設定すれば、全Moduleに並び順が反映される。
ただし、Moduleごとに並び順を変えることはできない。

Time範囲外のPERIOD

他システムのデータなどを取り込んだ際、Line Itemの値にTimeの範囲外の値を保持したい場合があります。

この場合、Time Rangeで長期間を指定したRangeをひとつ作ってください。
DimensionとしてApplyされていなくても値を保持できるようになります。

チェックボックスの一括更新

カーソルを合わせて、Shiftキーを押すとチェックボックスのチェックを反転できます。

範囲指定、または特定のヘッダーを選択してShiftキーを押すことで、一括更新が行えます。

一括更新のImportを組むよりもとても楽なので、エンドユーザーに上記の方法をご案内することをオススメします。

Tooltip

Dashboard上にPublishされているModule等には、マウスカーソルを合わせることで解説を表示できる機能があります。

https://help.anaplan.com/anapedia/Content/Dashboards_and_Visualization/Build%20Dashboards/Add_Tooltips_Grids_Buttons_Line_Items.htm

以下のようなTooltipを書くと、ユーザービリティが向上します。

青字:編集可能/黒字:編集不可能
【改行】Ctrl＋Alt＋Enter
【Undo】Ctrl＋Z

Grid Style/Global Grid Style

Dashboard上のModuleは、Grid Styleとして枠組みの色を変更できます。

また、DashboardにPublishするたびにStyleを変更するのも面倒なので、DefaultのStyleを設定することが可能です。

以下のGlobal Grid Styleを参照してください。

https://help.anaplan.com/anapedia/Content/Dashboards_and_Visualization/Build%20Dashboards/Change_a%20Grid_Style.html

Contents

レフトナビにどのDashboard/Moduleを表示するのかを制御する機能ですが、Moduleを表示することはあまりないので、最初からModuleに対してはShow New ContentsをOffにしておくことで設定の手間を減らすことができます。

グレーな製品仕様

それは不具合ではありません。仕様です。
※回避策付き

Column Headerが自動マッピングされない

File上のColumn Headerに1つでも空欄のCellが混ざっている場合と Column Headerに重複文字列がある場合、ModuleへのImportで、自動マッピングされなくなります。

Fileなどのテンプレートでは、 Column Headerに空欄のないようにして、自動マッピングするか、手動マッピングをするようにしましょう。

BLANKにはUpdateできない

- No DataやBLANKをSourceにしてModuleのCellをUpdateすることはできません。
- BooleanとNumberのFormatであれば、初期値(FALSEや0)をSourceにしてUpdateを行うことができますが、Text, List, Dateなどはデータの上書きでBLANKにする方法はありません。
- 特に"Only update imported cells"のオプションでは、初期化処理無しでUpdateのみを行うので、Text, List, Dateなどのデータは更新不可能です。
- Text, List, DateなどのCellを初期化するには、ModuleへのImportを行う際に利用可能な"Clear target prior to import"を利用して、データの上書きをする前にClearする(初期化する)という方法しかありません。
- 回避策の実装方法は、逆引きAnapediaを参照してください。

親と同じCODEを持つ子はImportで無視される

▽回避策

前提:子ListをNumbered Listにした上で、子ListのPropertiesに、子ListのNAMEとCODEに該当するText FormatのPropertiesを追加する。

1.親ListのImport

2.子ListのImport(CODEは取り込まず、NAMEとCODEのcombinationで取り込む)

3.子List→子ListでNAME(#)一致で、CODEを取り込み

TimeでSUM

TimeをDimensionに持つModuleに対して、Timeを条件にSUMすることはできません。

MOVINGSUMまたはCUMULATEで合計値を出した後に、LOOKUPしましょう。

EとROUND(1)

Numberは、極端に大きい数字や小さい数字の場合、1.0E-5や1.00000000000666E12というような指数としてサーバー内で扱われています。

NumberをTextにする際に、TEXT()関数を使いますが、このTEXT関数はセルに表示される値ではなく、セルの内部で持っている値をそのままテキストにします。

EとROUND(2)

ROUND関数を利用して、小数点下2桁で切り捨てる処理を行いたい場合、
ROUND関数の返り値に対して、稀に.00000000001 などといったかなり小さい数値が加えられる場合があります。
この不具合を回避するためには、最終出力を行うFormatをTextにすることで、強制的に小数点以下を切り捨てます。

▽指数表示もROUNDの仕様にも対応した回避策

(IF Value < 0 THEN "-" ELSE "") & TEXT(ROUND(ABS(Value), 0, DOWN)) & "." & TEXT(ROUND(MOD(ABS(Value), 1) * 100, 0, DOWN))

MAILTO関数の文字数上限

MAILTO関数の本文の文字数上限は2000文字です。

しかし、日本語はブラウザ上でサニタイジングされるので、本来の文字数よりも多くカウントされます。
サニタイジングされた上で、2000文字に収まるようにしてください。

▽サニタイジングの例
キ →%E3%82%AD


Anaplan
Driving a new age of
connected planning