Anaplan

2019/08/21
Anaplan Japan株式会社

Anaplan

INDEX

Anaplanの Core Technology

Anaplan Hyperblock™

Anaplan Hyperblock™ technology – Patent #8151056

Calculation engine

In-memory data storage

Hyperblock™ connectors

Intelligent aggregation and calculation

Anaplan

技術背景

Anaplan

計画業務に使われるTechnology

業績管理、市場分析、計画作成等の計画業務向けによく使われる技術として、OLAP分析ツールがあります。

OLAPとは、簡単に言えば「データベースからデータを取り出し、多次元的な分析を行う処理のこと」です。

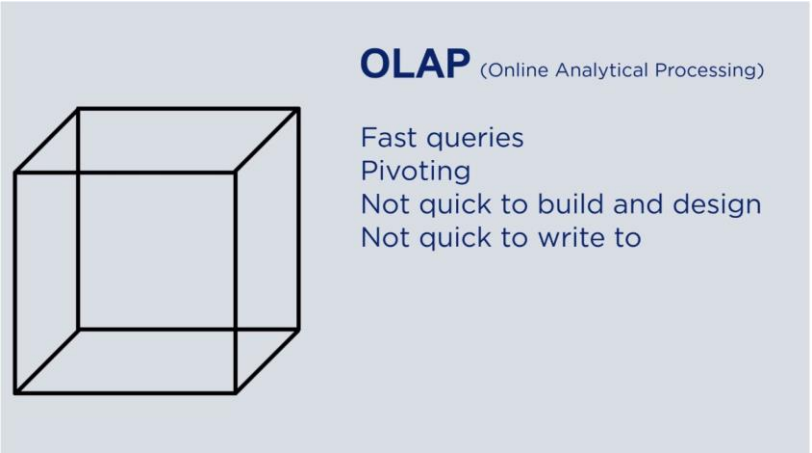
しかし、このOLAPは一般的なDatabaseの得意とする「小さなデータを頻繁に読み書きすること」は苦手です。

Anaplan

OLAP

▽特徴

- 大きなデータに対しての集計などの計算処理(Calculation)が得意。
- 高頻度の読み書き(Read/Write)には不向き。
- Database(RDB)で蓄積したデータをOLAPへ連携して利用することが多い。



▽用途

- 業績管理
- 市場分析
- 計画作成

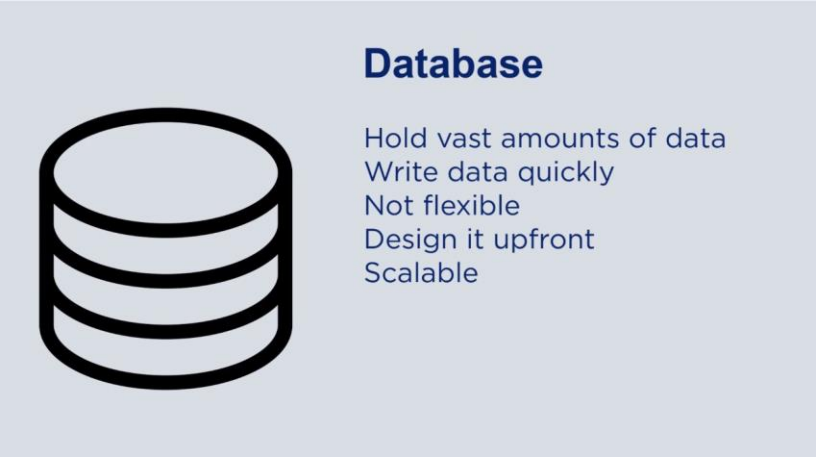
etc.

Anaplan

Database (RDB)

▽特徴

- 日々増えるような小さいデータの高頻度の読み書き(Read/Write)が得意。
- データベース内で計算処理(Calculation)まで行うことはせず、アプリケーションでデータを呼び出して計算処理を行うことが一般的。
- 世間のシステムの9割はこのデータベースを利用している。



▽用途

- ERP
- 銀行のATM
- ECサイト
- ソーシャルゲーム

etc.

Anaplan

Database vs OLAP

	Read/Write 頻度	Read/Write 量	計算処理	設計/実装
OLAP	△	◎	○	△
Database	◎	△	×	○

- 1つのデータベースでOLTP機能とOLAP機能を両立するのは難しい。
- 得意分野をそれぞれ分担して、データ連携を前提としたシステムを構築することが一般的。

Anaplan

Why Anaplan?

Hyperblock Technology

Anaplan's product is a cloud computing, multi-tenant data architecture SaaS platform with a patented, in-memory calculation engine (the Hyperblock).

Anaplan's Hyperblock architecture is a hybrid of relational, vertical, and OLAP databases with an in-memory data store multi-threaded calculation engine.[16] The Hyperblock automatically records updates at a granular level by amending only the affected cells. As volumes scale, users can instantaneously update or change models.[17] A patent application for the technology was filed on November 19, 2009, and US Patent 8151056B2 was awarded on April 3, 2012.[18]

引用元：https://en.wikipedia.org/wiki/Anaplan

Anaplan

Hyperblock vs Database vs OLAP

	Read/Write 頻度	Read/Write 量	計算処理	設計/実装難度
Hyperblock	○	◎	◎	○
OLAP	△	◎	○	△
Database	◎	△	×	○

- 先程難しいと言ったばかりですが、DatabaseとOLAPの機能を備えたHyperblock(特許取得済)がAnaplanを支える技術です。
- 業績管理、市場分析、計画作成などの用途に対して、OLAPよりも高速な計算処理が可能です。
- かつ、これらの用途の前提になるデータ収集も1つのPlatformで行えるので、データ連携を行うことなくシームレスな計画業務を実現します。

Anaplan

さらに?

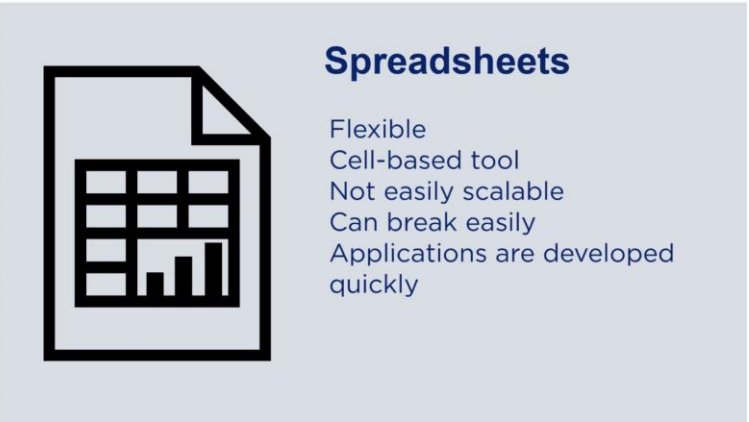
Anaplan

Spreadsheet

業績管理、市場分析、計画作成などの業務では、多くの場面で表計算ソフトも使われています。

簡単に安価で使い始められることから利用者も多く、このユーザビリティに慣れ親しんでいる人も多くいます。

関数の書き方さえ覚えてしまえば、エンジニアの手も必要なく、仕様変更が容易に行えます。



しかし、多人数が同時アクセス可能なデータベースではありません。

Anaplan

Anaplan

Spreadsheetが持つ

- 一般的な表計算ソフトの挙動に近いユーザビリティ
- 一般的な表計算ソフトに近いクイックな仕様変更が可能

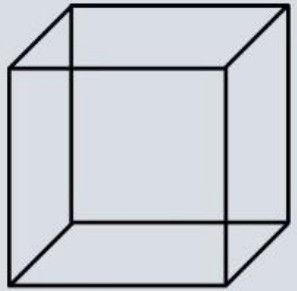
という特徴もAnaplanは兼ね備えています。

これにより、データベースでありながら、業績管理、市場分析、計画作成などの手法が変わりうる業務用途にも対応し続けられるPlatformをして強みを持っています。

Anaplan

Spreadsheet

Database



OLAP

Flexibility

Scalability

Fast Analytics

Hyperblock

設計観点

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

Expectation

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

Anaplan

Moduleの設計

Data, Input, System, Calculation, Output

Anaplan

Moduleの単位?

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

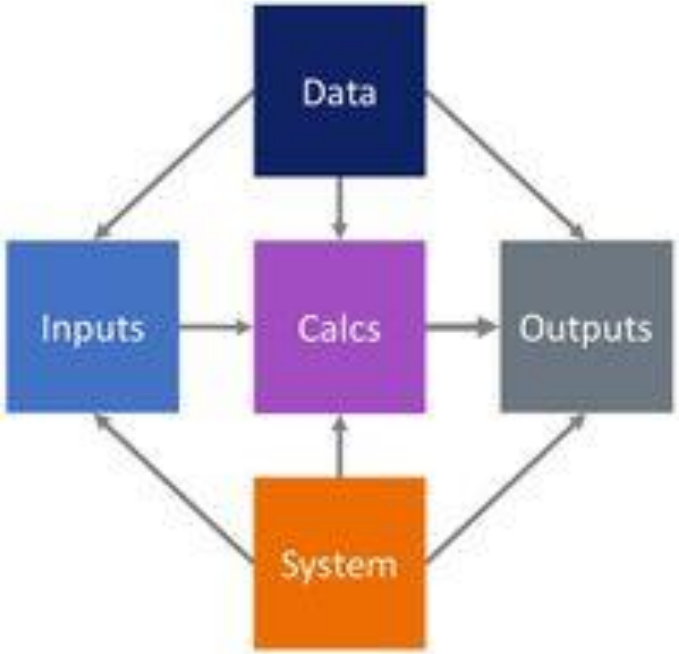
1つにするデメリット

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



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

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

▽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

- 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

「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)

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)

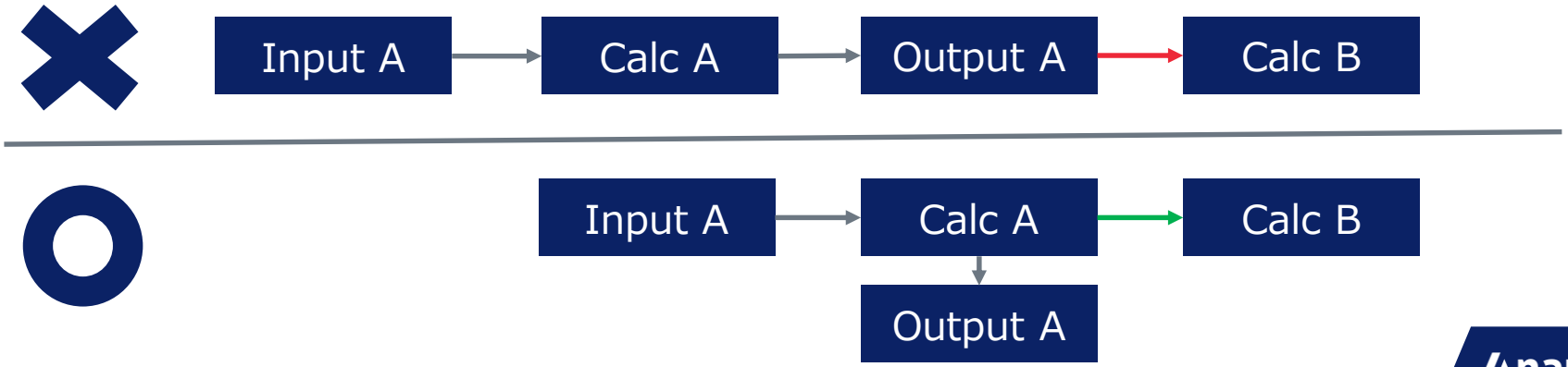
計算中の数値も見たいと言われた場合、それはCalculationを参照するOutputのように実装しましょう。CalculationはDashboardに載せないものという基本は守ったほうが、レビューや引き継ぎが楽になります。

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。

回答(3/3)

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。



Listの設計

健全なModelを育てる第一歩

Anaplan

Listを作る観点

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)
- 業務フローの構造を作る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

「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。

Listを設計する基本(1/5)

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。)

Anaplan

Listを設計する基本(2/5)

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)

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?

Listを設計する基本(4/5)

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。

Listを設計する基本(5/5)

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

Selective Access

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CODE

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。

Dashboardの設計

Anaplan

Landing Dashboard

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

Anaplan

ModuleとChartの数

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。

Importの設計

Importとは

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

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

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)

- 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。

Modelの分け方

Connected Planningのはじまり

Anaplan

Modelの技術的制約

▽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

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

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

容量とは?

Anaplan

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

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。

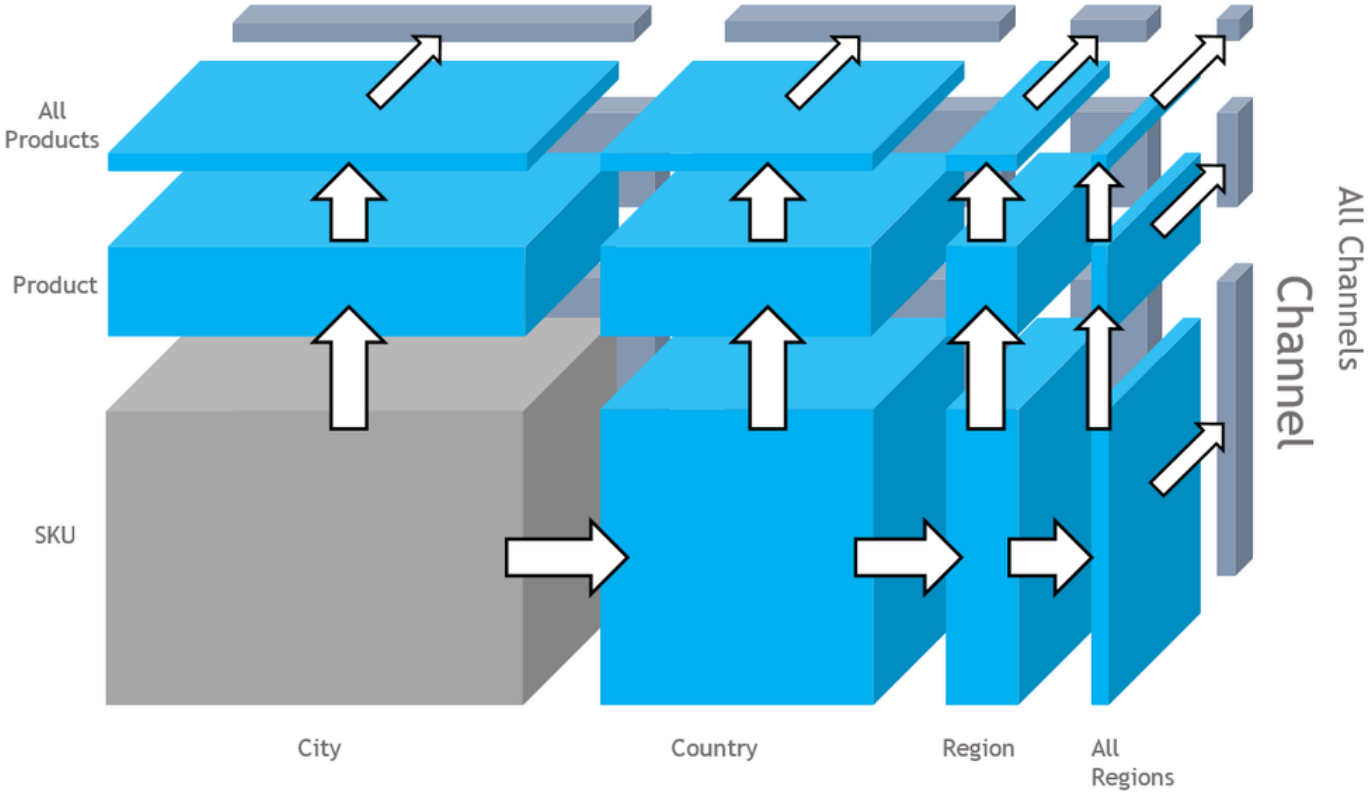
Properties追加による容量は、ModuleのCellと同じ計算になります。
Listは1次元なので、100万件のListにText FormatのPropertyを1つ追加しても8MBにしかなりません。

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

ParentがCellの数に与える影響



環境構成と容量

本番Modelで利用する容量の試算ができたら、次に考慮すべきは、環境構成です。

本番、検証、開発のどの環境を利用するのか?

また、検証はシステムテスト、外部連携テスト、UATなどの、目的別に環境をわけるのか?を考慮しましょう。

また、保守時に本番をコピーして調査を行うような一時的な用途向けの環境の要否も検討しましょう。

上記を含め、必要な環境数と本番で必要なそれぞれの容量を検討しましょう。

失敗事例から学ぶ

Anaplan

ケース1：後先考えずに画面だけ作ってしまった。

データ設計や計算を組めるかの考慮を行わずに、画面仕様だけをユーザーと詰めて進めてしまった場合、以下のような状態になり後からプロジェクトを仕切り直して再構築しなければならない場合があります。

- 計算式が書けない。または計算式が複雑になりすぎてメンテナンスできない。故にやりたいことができないし、拡張性もない。
- 計算式が書けないのでImportをたくさん実装することになり、結果メンテナンスできないものができあがる。
- 本来不要な容量が発生してしまい、容量逼迫問題が発生する。
- ShowやHideを多用しており、メンテナンスできないものができあがる。

ケース1：どうやって防ぐか?

ユーザーとの要件定義や仕様確定には、InputやOutputの画面イメージを見せながら進める必要がありますが、一方で、設計者や実装者は並行してデータ設計や計算ロジックの組み方を意識しながら、画面を構築しましょう。

※設計に必要な観点は、入門資料に書いてあります。

ケース2: 乱雑な実装で生産性が低下してしまった。

Anaplanは生産性の高いPlatformですが、データ保持部分と計算部分などを乱雑に実装すると、参照関係が複雑になってしまいます。

こうなると、「引き継いだ人が理解できなくなる。」どころか、「先週の自分が作ったものすら理解できなくなる。」という状態になってしまいます。

DISCOと規約を守って、高い生産性を維持しましょう。

Anaplan

Anaplanでの要件定義

計画業務のAnaplan化?

計画業務の目的は多岐に渡りますが、抽象化すると以下のようなものがあります。

- 経済的リターンの試算
- 組織体制、アクション、スケジュールなどを決めるためのKPIの見える化

これらの計画立案や計画見直しをAnaplan化するには、どのような要件ヒアリングやプロジェクトスコーピングが適切でしょうか?

データの大分類

▽計画業務で扱うデータは大きく3種類に分類できます。
実績: 実際の結果の過去の数字
計画: 計画立案で策定する将来の数字
見込: 実績に応じて計画を見直して作る直近の将来の数字

Anaplan

ワークフローの大分類

▽Anaplanが得意とする業務は大きくわけて3つに分類できます。

- いわゆる「たくさんの人にExcelで入力テンプレートを配って、提出してもらう」という配布・収集業務
- 予実比較や組織別対比、統計解析等の分析業務や、シナリオを用いたシミュレーション業務
- 最終的なレポーティング業務

計画プロセスの大分類

▽トップダウン・アプローチ

- 全社的な目標やガイドラインの策定。

↓

- 現場レベルへ展開し、妥当性の確認と、具体的なタスクやアクションへつなげる。

▽ボトムアップ・アプローチ

- 現場レベルから計画や見込を申請。

↓

- それぞれの申請への承認、俯瞰的な目線での妥当性の確認。

Anaplan

風呂敷を広げる

まず、3つの大分類の組み合わせでマトリックスを作ってみましょう。
次に、売上/数量/原価/コスト/人員数などの計画分析を行いたい対象のデータそれぞれに対して、対象の業務があるのかないのかを判断しましょう。

	実績	トップダウン計画	ボトムアップ計画	見込
テンプレ配布	?	?	?	?
収集/集計	?	?	?	?
分析	?	?	?	?
シミュレーション	?	?	?	?
レポーティング	?	?	?	?

Anaplan

事例:全国営業計画

本社で作成した製品別の営業目標のガイドラインを日本全国の営業支店へ配布し、各支店から取引先別でどのようにその目標に達する計画なのか、を提出してもらう事例です。

	実績	トップダウン計画	ボトムアップ計画	見込
テンプレ配布	○	○	○	○
収集/集計	○	○	○	○
分析	○	○	○	○
シミュレーション	-	○	○	-
レポーティング	○	○	○	○

Anaplan

事例:需要供給バランス

週次で需要見込を集め、生産計画と比較し、在庫の過不足を検知する事例です。
毎週やっている業務なので、年間計画などの作成はしていません。

	実績	トップダウン計画	ボトムアップ計画	見込
テンプレ配布	○	-	-	○
収集/集計	○	-	-	○
分析	○	-	-	○
シミュレーション	-	-	-	-
レポーティング	○	-	-	○

Anaplan

事例:グローバル予算策定

世界各地の支店から、予算の年間計画を月別で立てて、毎月実績を提出して予実比較を行っている事例です。予算の見直し業務のAnaplan化はまだできていません。

	実績	トップダウン計画	ボトムアップ計画	見込
テンプレ配布	○	-	○	△
収集/集計	○	-	○	△
分析	○	△	○	△
シミュレーション	-	-	△	-
レポーティング	○	-	○	△

プロジェクトスコーピング

広げた風呂敷を一気に畳むのではなく、段階的にリリースを行うプロジェクトスコーピングがおすすめです。

例えば、トップダウン計画と実績だけ初回リリースし、第2フェーズでボトムアップ計画、第3フェーズで見直しをリリースとするようなリリースです。

こうすることで、プロジェクトメンバーの習熟度が上がりつつ、本当に必要な機能の取捨選択が行いやすくなり、後戻りも少ないプロジェクトを進めることができます。

詳細ヒアリング①(配布・収集)

実績、計画、見込それぞれで以下のような観点を確認しましょう。

- 業務サイクルは?年次?月次?週次?
- 時間単位での粒度は?月単位?週単位?日単位?
- どれだけ過去から未来のデータが必要?
- データのキーになる情報はなにか?
- 組織別や製品別にデータのキーが異なることはないか?
- 多通貨で集めるか?

詳細ヒアリング②(配布・収集)

- データ取得元は?
- 提出する人の人数規模は?
- 承認フローはあるか?何段階あるか?差し戻しはあるか?
- 権限やアクセス制御はどうなるか?ReadとWriteはどの単位か?
- 入力時の参考データや比較データはどのようなものがあるか?

詳細ヒアリング③(分析/シミュレーション)

- どの分析軸でデータを集計したいか?
- データの比較対象はなにか?予実?前年?組織別?製品別?
- 比較するデータ間の粒度は合っているか?
- 比較対象のデータ粒度が異なる場合のOutputはどうするか?

マスタメンテ

組織や顧客や製品などのデータのキーになるマスタをどこから取得するのかも確認しましょう。
実績、計画、見込でマスタの粒度が同じなのか異なっているかも確認しましょう。

マスタをAnaplan内で作成して管理することもできますが、外部システムのデータを利用する前提があるのであれば、データの粒度を合わせなければいけないので、取得元のシステムに合わせましょう。

失敗事例から学ぶ

Anaplan

ケース3：Agileを誤解して、要件定義と開発/評価が並行し続ける。

Anaplanは、WaterfallもAgileもどちらの導入手法も可能なPlatformです。
特に、クイックに構築ができることからAgileとの親和性が高いです。

しかし、「Agileは要件をどのタイミングで追加変更しても大丈夫な手法」と誤解することで、以下のような状態に陥ります。

- 開発完了間際まで要件変更による仕様変更が続き、開発が終わらない。
- 品質が安定しない。
- 設計時の考慮が少なく、拡張性がないModelになる。

ケース3：どうやって防ぐか?

業務要件とシステム仕様を混ぜて考えてしまってはいけません。

Agileとはいえ、業務要件は開発着手前に洗い出しましょう。

その上で、拡張性のある全体設計を行い、画面仕様などは作りながら認識合わせをしましょう。

ケース4 :要件すべてを最初のスコープにいれてしまった。

最初のプロジェクトフェーズに要件すべてを詰め込んでしまうことで、以下のような状態に陥ることがあります。

- 適切な権限設定が不明で、汎用的すぎるものを実装せざるを得ず、実装コストが増える。
- 「必要かもしれない。」というレベルの機能に対しての要不要の判断が付きづらく、優先順位のコントロールが難しくなる。

実際に利用を開始しながら、本当に必要な機能を随時リリースすることで、実装負荷を減らすようにしましょう。

ケース5：計画系業務ではなく、実行系業務を対象にしてしまった。

Anaplanは、リアルタイム計算が得意な多次元データベースのPlatformですが、RDBが得意な高頻度高負荷なトランザクション処理は苦手です。

実際の受注に対して在庫引当を行うなどの処理はできません。

入力頻度が中くらいの業務を対象にして、計画・分析・シミュレーション等の用途でAnaplanを活用してください。


Anaplan
Driving a new age of
connected planning