QlikView

QlikViewにおけるデータフロー

QlikView Technical Brief

September 2013

QlikView 11

qlikview.com

QlikView

もくじ

QlikView

はじめに

QlikViewのBusiness Discoveryは、データ接続、変換、配信、分析という一連のプロセスの上に成り立っています。本書では典型的なQlikViewの構成を例に一連のデータフローの概要を紹介するとともに、個々のコンポーネントの役割を説明します。複数の異なるデータソースからどのようにデータが抽出され、論理的に一貫性を保つ形で処理され、最終的にユーザーが展開されたアプリケーションをどのように扱うかについて説明します。

概要

QlikViewのエンタープライズシステムについて説明するには、まず以下の4つのコンポーネントについて説明する必要があります。

QlikView Desktop, QlikView Server, QlikView Publisher, Client

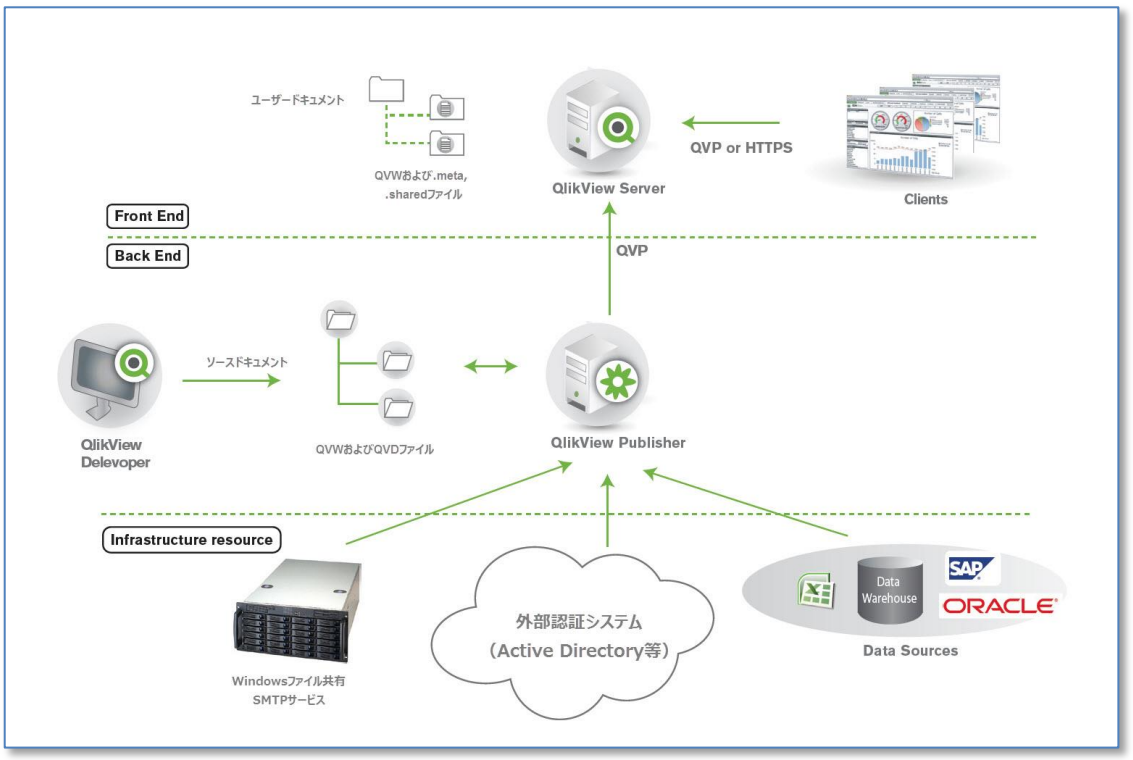


図1:　QlikViewのアーキテクチャ概要

QlikView

QLIKVIEW DESKTOP :

QlikViewアプリケーションを作成するための開発ツール。アプリケーション開発者はこのツールを用いてデータソースからデータを抽出し、必要に応じて加工し、画面上に可視化していきます。アプリケーションの描画はクライアント側で処理されますが、アプリケーションのデータ処理はサーバー側で管理されます。

QLIKVIEW SERVER :

QlikViewアプリケーションをクライアントに提供するためのエンジンコンポーネントで、アプリケーション内の演算処理やセキュリティ管理を司ります。

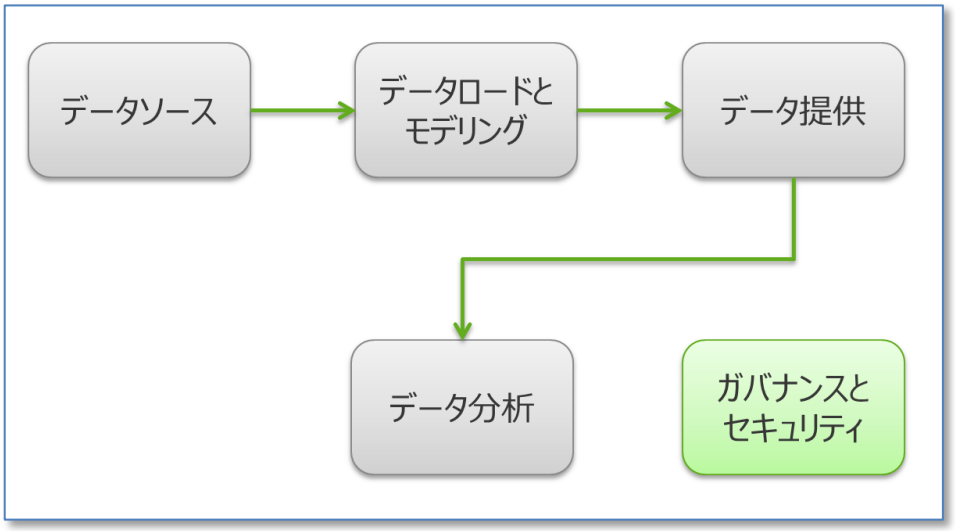
QLIKVIEW PUBLISHER :

QlikViewアプリケーションに取り込まれるデータの更新・分割・配信機能を提供するリロード／配信エンジンです。

CLIENT :

ユーザーがQlikViewアプリケーションを使ってデータを参照したり分析したりするためのフロントツール。デスクトップツールを用いる場合とブラウザベースの場合があります。いずれの場合もアプリケーション稼働には主にサーバー側のリソースを消費する為、クライアント側のコンピュータリソースを消費することはほとんどありません。

本書では、データがどこからどのようにQlikViewに取り込まれ、モデル化され、分析の為に提供されるようになるか順を追って説明していきます。また、補足情報としてデータガバナンスやセキュリティについても述べていきます。



データソース

QlikViewは複数の異なるデータソース（例：データベース、Excelシート、HadoopやGoogle BigQueryなどの”ビッグデータ”）からデータを抽出し、分析と可視化に適した同質なデータセットを生成します。

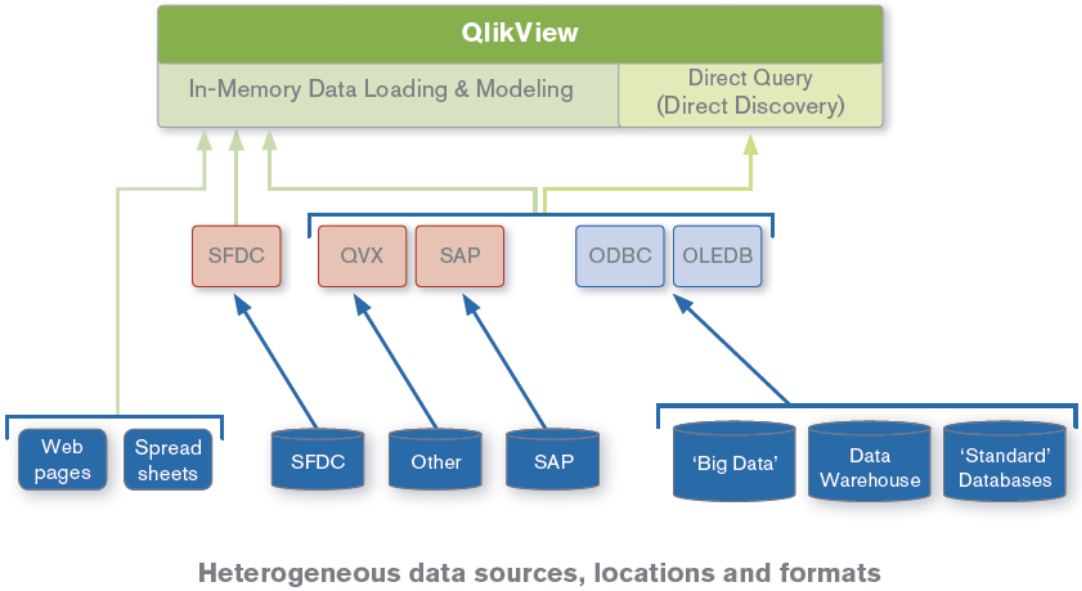


図2:　複数データソースからQlikViewへのデータロード

QlikViewでは多くのシステム、すなわち標準ODBC、OLE DB、JDBCデータストア、スプレッドシートやWebページ（HTML, XMLなど）から（Salesforce.comやSAP、Google BiGQueryなどの）カスタム接続を要するシステムまで様々なソースからデータを抽出します。ほとんどのデータソースでは、ウィザードを用いて接続定義をすることで手順を簡単にし、開発者はデータ抽出の仕方をコントロールできます。例えば、開発者はある項目を抽出から除外したり名称を変更したりすることができます。データウェアハウスは必ずしも必要ではないですが、もしすでに存在する場合は活用することでデータ抽出を効率化できます。Hadoopベースのビッグデータシステムへのアクセスも可能です。ODBCドライバやJDBCドライバが存在しない場合は、QVXと呼ばれるオープンなデータ交換用プロトコルを活用することで標準接続を提供していないデータソースに対するカスタムコネクタを作成することも可能です。

QlikView

データロードとモデリング

QlikViewを用いたデータ分析における第一の特徴はインメモリエンジンにあります。1993年の創業以来、QlikViewはデータ分析に対するインメモリ型のアプローチを踏襲し続けており、20年以上この技術に基づいて業界屈指のインメモリ分析機能を提供し続けています。さらにQlikViewではDirect Discoveryと呼ばれるダイレクトクエリ機能を追加し、ソースシステムへ直接クエリを実行することも可能となっています。

データはロードスクリプトを介して様々なデータソースからQlikViewのインメモリエンジンへロードされます。ロードスクリプトはQlikViewアプリケーションに内包され、SQLに類似した言語を用いてデータソースに接続し、データモデリングを実行します。データはロードスクリプトが実行されて初めてロードされます。QlikView Publisherを使うとデータロードを指定したスケジュールやトリガーに応じて実行できます。

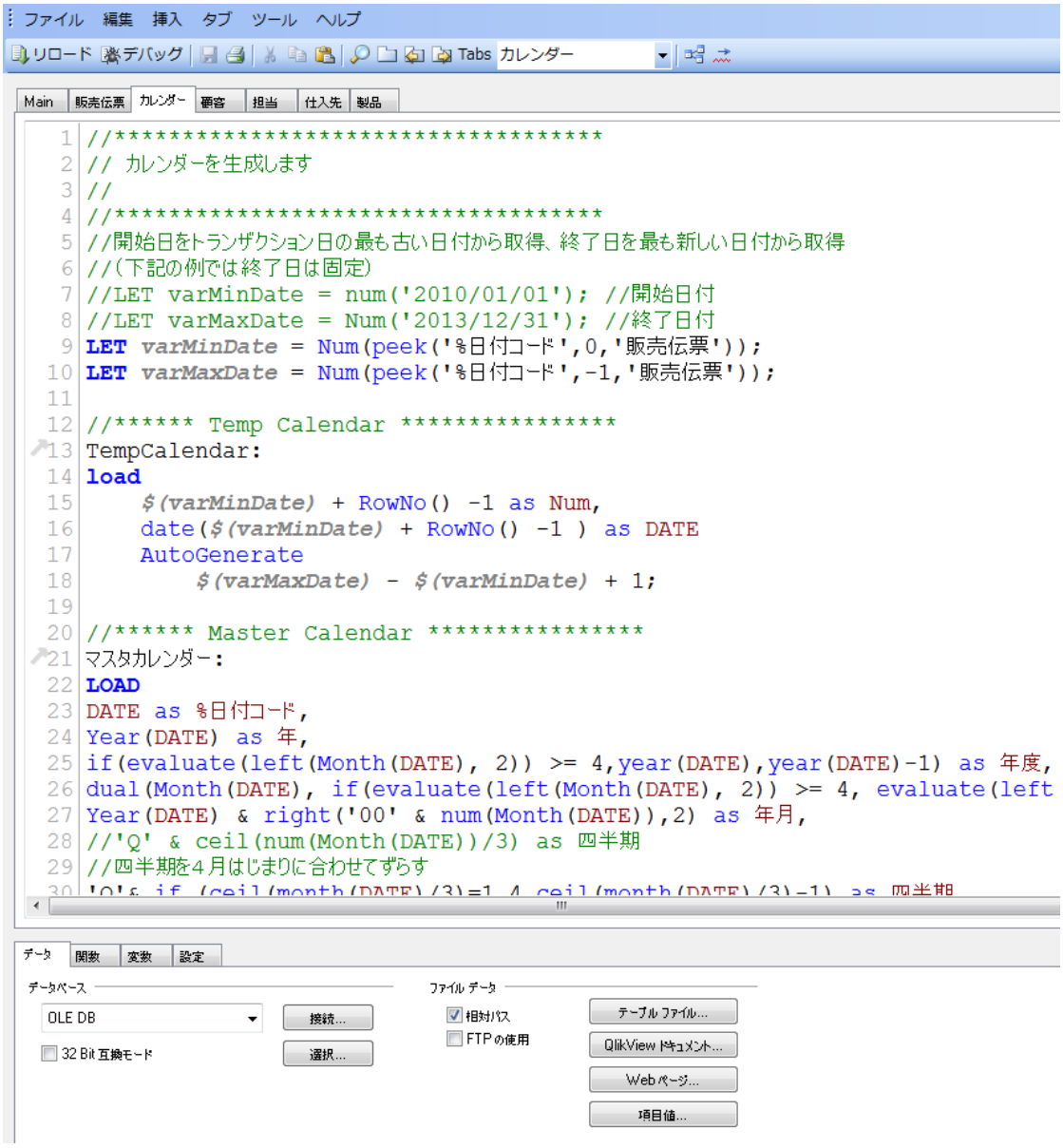


図3: ロードスクリプトの例

一旦QlikViewアプリケーションにロードされると、データはインメモリに保持されます。これはつまり、QlikViewアプリケーションはデータロード時に一度データソースへのアクセスを必要とし、履歴データも含めた全データを取り込むということになります。差分や更新などの新規データについては、それらのデータのみをロードして履歴データに追記することで毎回全データを再ロードすることを防ぐことができます。さらにQlikViewではインメモリでのデータストアを最適にするため、洗練されたアルゴリズムを用いてデータを圧縮します。詳細は下記のブログをご覧ください。

http://community.qlikview.com/blogs/qlikviewdesignblog/2012/11/20/symbol-tables-and-bit-stuffed-pointers

複数データソースからQlikViewに取り込む前に、データをモデリングする目的でもロードスクリプトを使います。実際BIツールは複数のデータソースにわたって重複して使われたり、わかりにくい名称を使われている項目が存在したり、データが不完全な場合にきちんと対処できなくてはなりません。

異なるデータソースからのデータを関連付けるには、キーを用いますが、同じ意味を持つデータが異なる名称としてそれぞれのソースデータで扱われている場合もあります。（例えば図4における“Sales”、“Sales Revenue”、“Sales Numbers”はすべて同じ意味のデータかもしれません。）QlikViewではこれらの類似したデータ項目を容易に統合することができます。（図4にあるように、3つの“Sales xxx”項目を名称変更して一つの“Sales$”項目に統合します。）

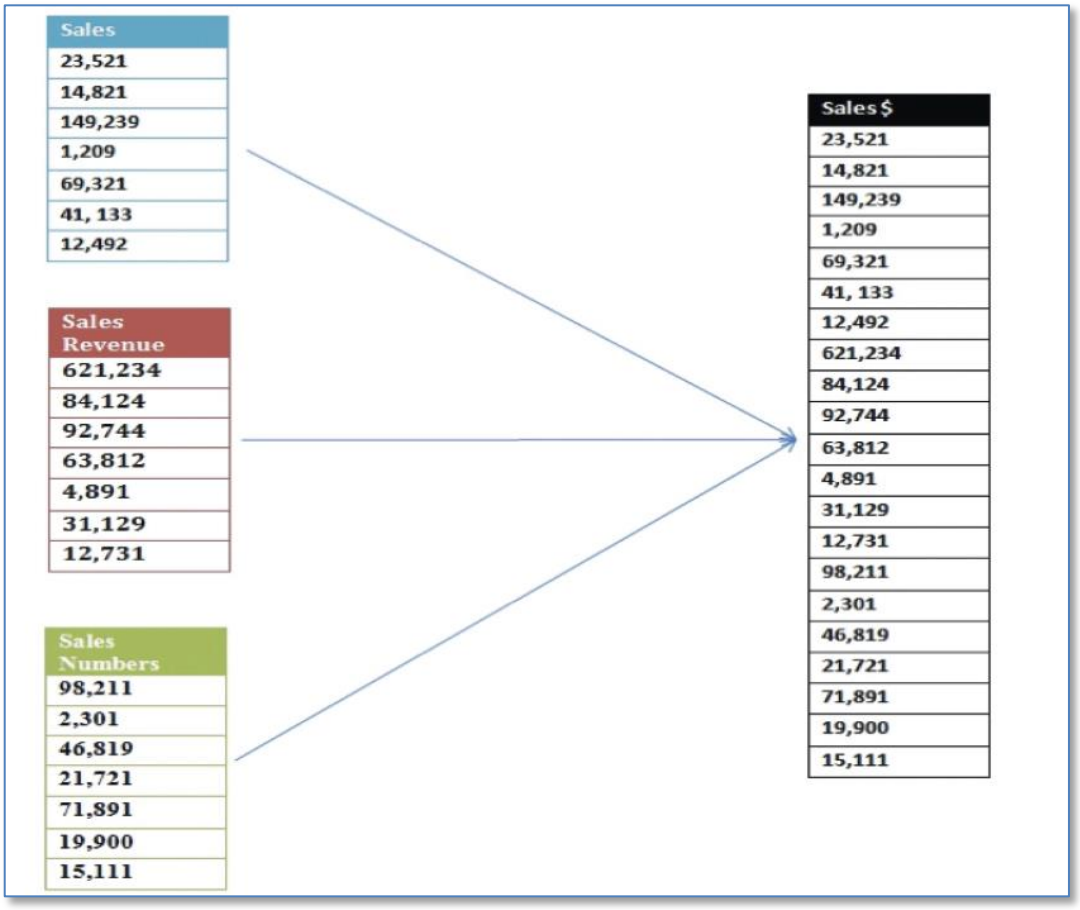


図4:　異なるソースからの項目名の変更

さらに厄介な問題としてデータフォーマットの問題があります。例えばあるソースデータでは“YYYY-MM-DD”形式の単一項目で格納されているデータが、別のソースでは年、月、日、といったようにそれぞれ別の項目として定義されている場合があります。アプリケーション開発者はこれらの項目が統一のビューで表現されるよう考慮する必要があります。

ロードスクリプトを使えば、項目を名称変更したり、分割したり、結合したり、その他さまざまな変換処理を施すことができます。例えば、テーブルをジョインしたり、“姓”と“名”の各項目を結合して“氏名”という項目を新たに作成したりできます。QlikViewはマルチソースからのデータを扱うことができるため、例えば販売データベース上で実績があった従業員のみをHRデータベースから営業担当として取得する、といった条件に応じた処理も可能です。

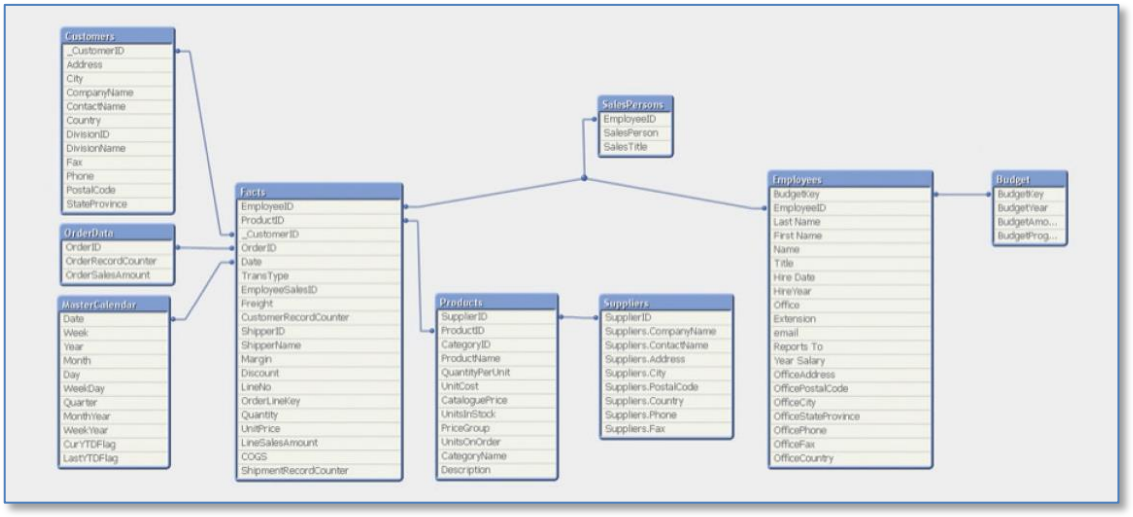


図5:　データモデルビューアー

図5に示されるデータモデルビューアーでデータどうしの関連を確認するだけでなく、情報密度や項目名、テーブル名などの情報も確認できます。スクリプト開発時に直面する様々な問題を解決するのにも役立ちます。

QlikViewはロードされたデータに対し独自の連想技術でデータを関連付けます。複数ソースから取り込まれたデータは、それがどこからロードされたものかに関わらず、分析目的で使うためにエンジン内で単一のデータエンティティとして扱われます。QlikViewでは同じ名称、同じ型の項目は自動的に関連付けられます。これによりエンドユーザーは様々な異なるシステムからのデータを考慮しながら試行錯誤するのではなく、あたかも単一のテーブルを扱っているかのように“解の探索”をすることができます。図5に示される通り、ファクトテーブルと従業員テーブルはEmployeeIDで自動的に関連付けられています。

データ格納方法

QlikViewではファイル単位でデータの一貫性を保持しており、実際には（“.qvw”ファイルと呼ばれる）すべてのQlikViewアプリケーション自体に必要なすべてのデータが格納されています。QVWファイル内のデータには、前回のロードスクリプト実行により抽出されたデータが含まれ、ディスク上にバイナリデータに変換された形で保持されます。ロードスクリプト自体もまた、プレゼンテーションレイヤの一部としてQVWファイルに内包されます。

中規模以上の環境では、データステージングレイヤが使われることが多々あります。その目的は、a) 個別の分析要件に最適化されたデータの塊（例えばさまざまな会計システムからのデータを会計データパッケージとしてまとめておく）を提供すること、b) QlikView用に最適化されたデータローディング環境を提供すること、にあります。QlikViewアプリケーション開発者は“.qvd”ファイルと呼ばれるQlikView専用のデータファイルを作成することで“.qvw”ファイルに高速にデータロードすることが可能になります。

一般的なQlikView環境では、“QVDレイヤー”と呼ばれる.qvdファイルの集合が使われます。（例：会計データQVD、販売データQVD、月別QVDなど）。開発者はQVDレイヤーの棚からファイルを取り出すかのように容易にデータを取り出し、QlikViewアプリケーションを作成でき、さらに多くのQlikViewアプリケーションで一貫したデータの再利用を促進することができます。図6を参照ください。

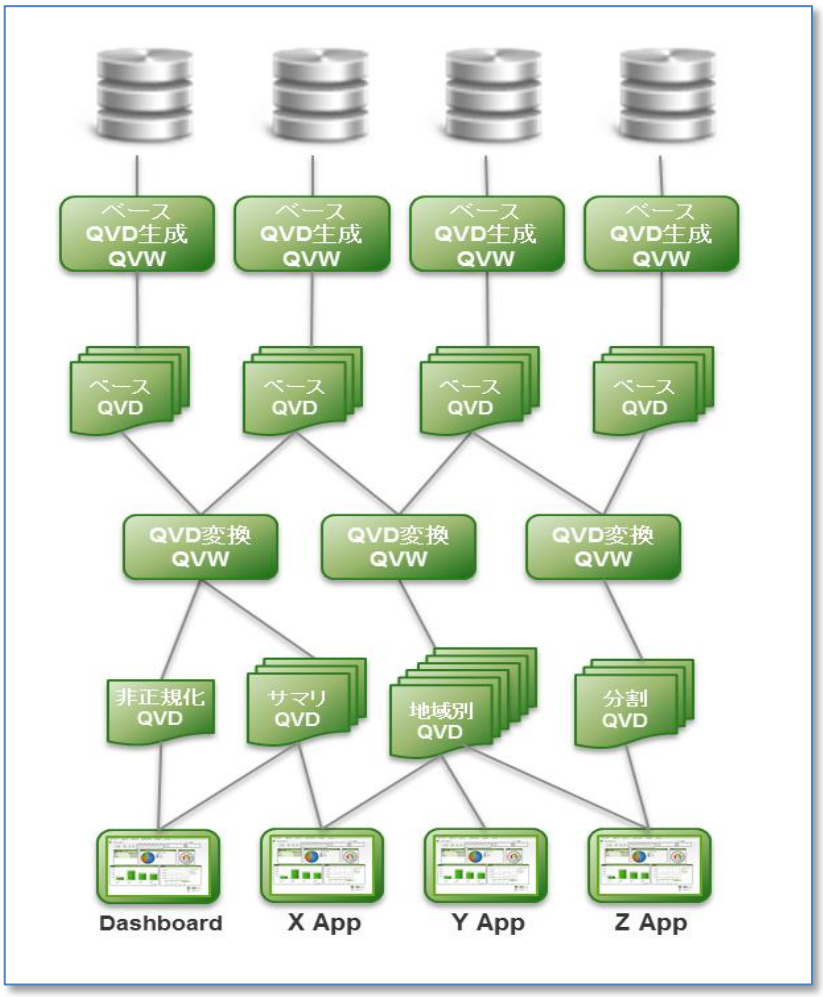
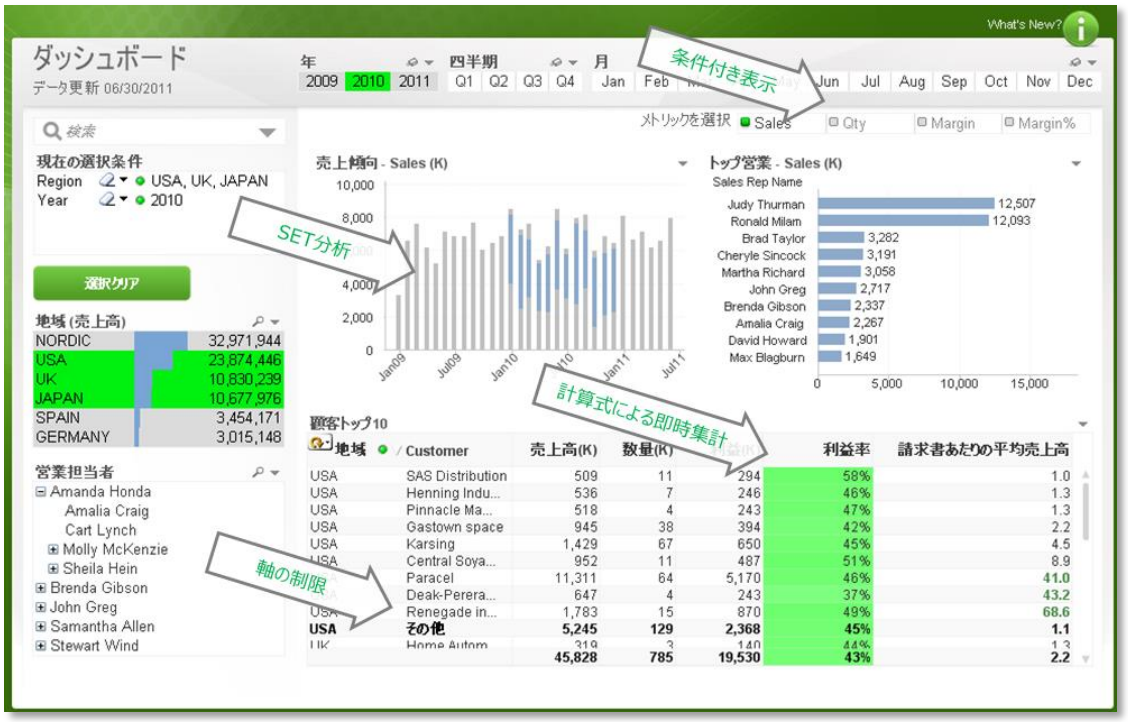


図6:　QVDレイヤーの例

QlikView

データの分析

データがQlikViewのインメモリ連想エンジンにロードされると、多岐にわたる強力な分析機能が利用できます。これは高速かつ非常に柔軟なQlikViewのインメモリテクノロジの特性に起因します。開発者は洗練された分析アプリケーションを作成することができ、それによりビジネスユーザーは豊富な分析機能を使って様々な疑問に対して“解の探索”ができるようになります。QlikViewの各オブジェクトからアクセスできる数式を使えば、インメモリデータを動的に集計、操作、比較することができます。データモデルには存在しない軸をその場で作成することも可能です。新たな階層を定義したり、比較分析用に異なるデータセット（グループ）を作成したりすることもできます。



今日、BI業界ではインメモリに関する議論がしばしば行われています。インメモリという表現だけで、分析に必要な機能すべてを語ることはできません。インメモリソリューションを選択するときにはそれを使ってどんなことができるようになるのかを明確にイメージすることが肝要です。QlikViewでは、インメモリ技術により複数テーブルを含むデータモデルから、データを連携させた形で、即時集計（つまり、事前計算や事前集計が不要）を行うことができます。この点でQlikViewは大変ユニークであると言えます。

QlikViewの内部的な処理の詳細については、下記のブログを参照ください。

http://community.qlikview.com/blogs/qlikviewdesignblog/2013/07/15/logical-inference-and-aggregations

数式の中で数百にも上る関数を利用して、開発者は動的かつ高度に連携したアプリケーションを作成できます。これらの関数は図７のように集計関数、財務関数、統計関数、マッピング関数や数値関数といった具合にグループ化されています。

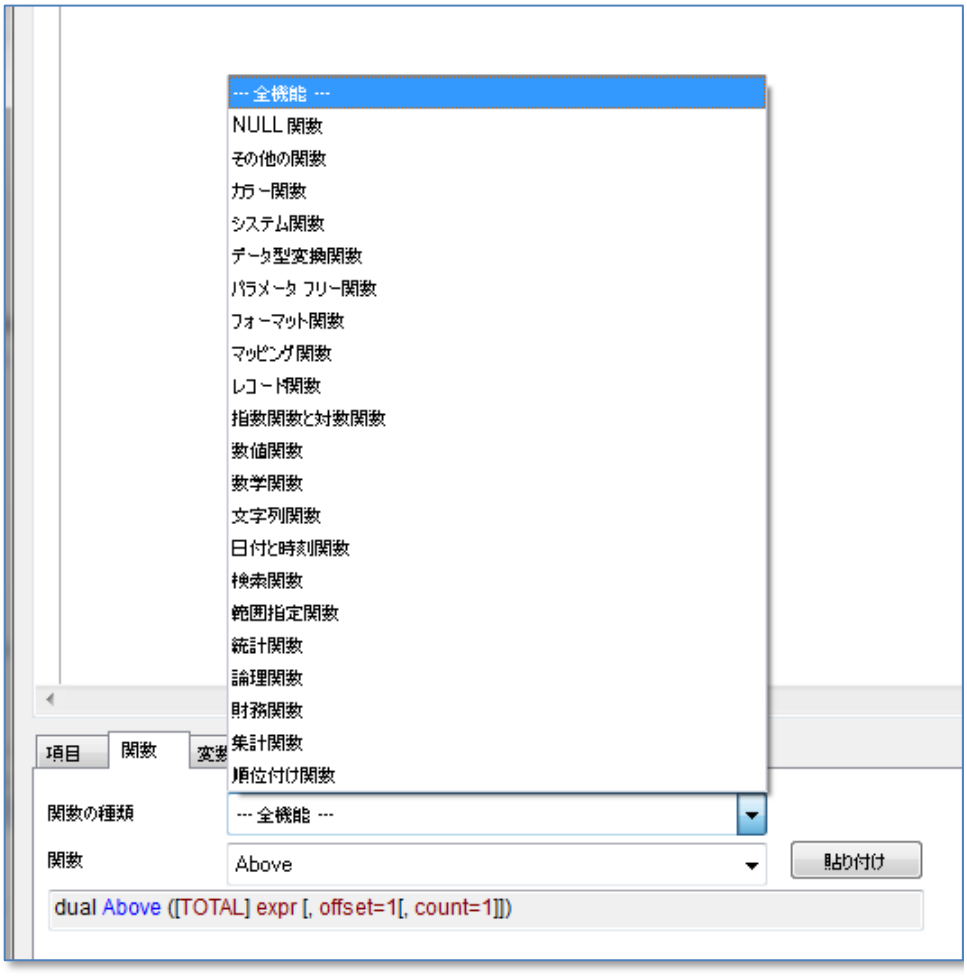


図7:　数式として使用可能な関数の種類

QlikViewのインメモリ分析エンジンが提供する特徴的な機能

- 計算軸
- 明細からの即時集計
- 階層構造の柔軟な定義
- SET分析
- 比較分析
- 条件付き表示

QlikView

データガバナンスとセキュリティ

経理部門で使われている「売上」と営業部門で使われている「売上」は同じでしょうか？それをどうやって確認したらよいでしょうか？また、別のアプリケーションで計算されたそれぞれの数値は同じ値となっているでしょうか？これらは当局からトレーサビリティが求められるような法定帳票などでしばしば緊急かつ重要な問題となります。一貫性と信頼性を確保することはデータガバナンスにおいて重要な要素となります。

QlikView Governance Dashboard と QlikView Expressorは、各QlikViewアプリケーションに対するデータガバナンスと集中管理されたデータプロビジョニング（事前準備）機能を提供します。Governance Dashboardは、QlikViewに流れるデータフローの包括的なビューを提供することで、データがどのように処理され、誰がいつ何を参照したかを可視化します。一方QlikView Expressorにより、売上高や従業員コスト、収益といったビジネス指標を一貫性を保ちながら追跡可能な形で管理することが可能となります。データ管理者はQlikView Expressorを使ってQlikView環境全体にわたる共通のビジネスルールを定義できます。

セキュリティとは、誰がどのデータにアクセスできるかを管理することを意味します。通常すべてのQlikView環境において、統合Windows認証やその他サードパーティ製品によるシングルサインオンなどの認証が要求されます。ユーザー認証が確立されると、今度はデータセットへのアクセス認可の問題が浮上します。認可はアプリケーションレベル、アプリケーション内のオブジェクトレベル、データの項目や行レベルで設定されます。QlikViewは様々な業界標準のテクノロジーを採用し、ユーザーのデータに対するアクセスコントロールを実現します。QlikViewシステムでは、クライアントサーバー間のすべての通信はHTTP(S)あるいはQlikView独自のQVPプロトコルを介して行われ、必要なポート以外は開く必要はありません。詳細については、[参考]セクションに記載の「QlikViewセキュリティ概要」をご覧ください。

QlikView

参考

QlikView アーキテクチャおよびシステムリソースの利用について

QlikView ガバナンス概要

QlikView セキュリティ概要

QlikView Design Blog Post: Logical Inference and Aggregations

QlikView Design Blog Post: Symbol Tables and Bit Stuffed Pointers