QlikView 開発における

ベストプラクティス

ガイドライン

Version: 0.51 - draft
Date: Oct, 2012
Author: BPN, JCA

目次

はじめに

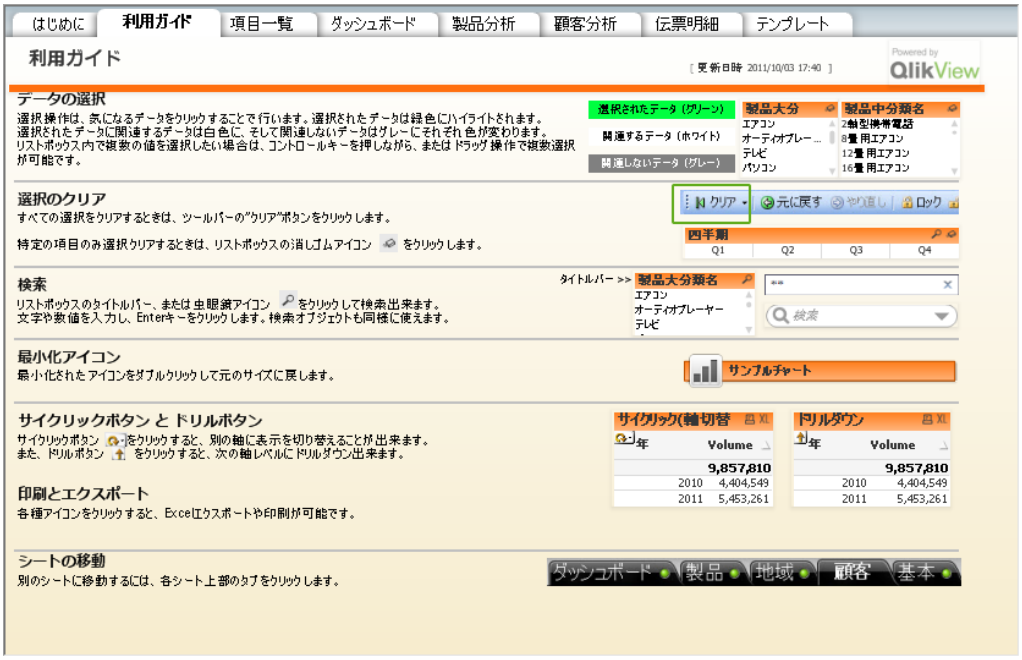
このベストプラクティスガイドは、QlikView 開発者向けのリファレンスマニュアルです。QlikView 開発者とは、QlikView アプリケーションを設計・実装し、専門領域がデータモデリングからスクリプト作成や UI 設計などのタスクを担当します。本ドキュメントは、小規模の部門または大企業のいずれで使用される場合も、使い勝手が良く、高度に最適化され、構成可能な QlikView アプリケーションを作成するための最適な方法とベスト・プラクティスを、理解していただけるように作成されています。

UI 設計

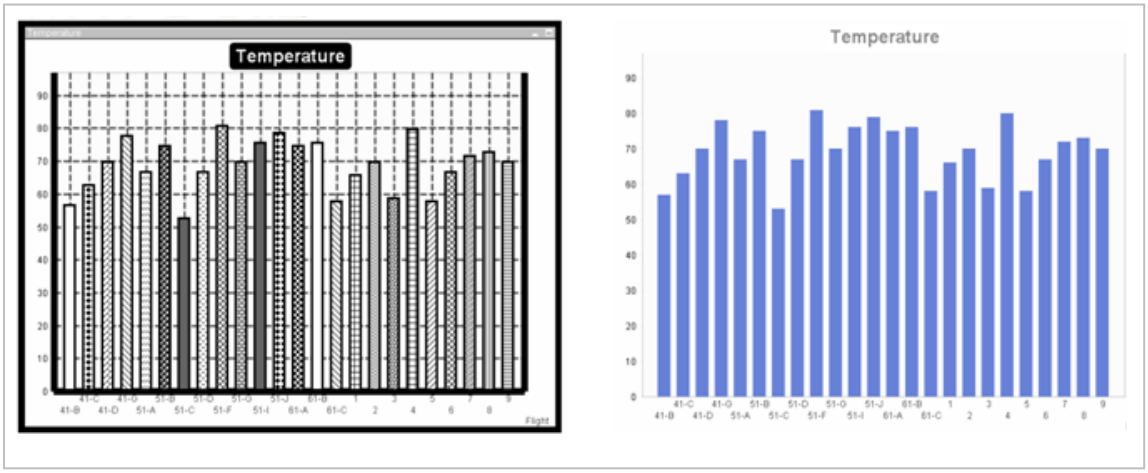
UI 設計は重要なポイントで、ユーザー普及率、利用率、分析速度、および使用パターンに影響します。そしてこれらはすべて、QlikView のドキュメントがいかに効果的なものになるかに影響します。Stephen Few と Edward Tufte によって生み出された優れたインターフェース設計の原則が、QlikView ドキュメントの設計および作成に QlikTech が推奨するベストプラクティスの基盤です。以下の概要は、この原則に基づく優れた設計の一部を（概念レベルで）示したものです。QlikTech は、これらの原則を立証するのに役立つ QlikView 導入例、ドキュメント、スライド資料、その他の資料を多数作成しています。

例

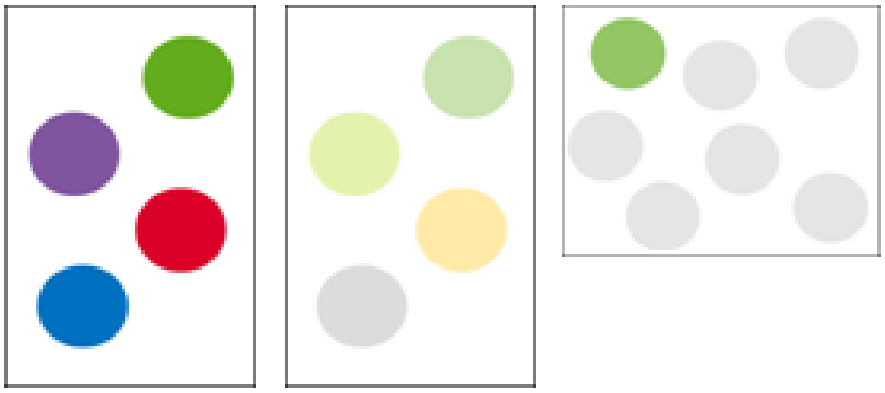
整合性および簡素化のために提供または開発されたテンプレートの使用：



スペースを節約する為の透明な囲いの使用:

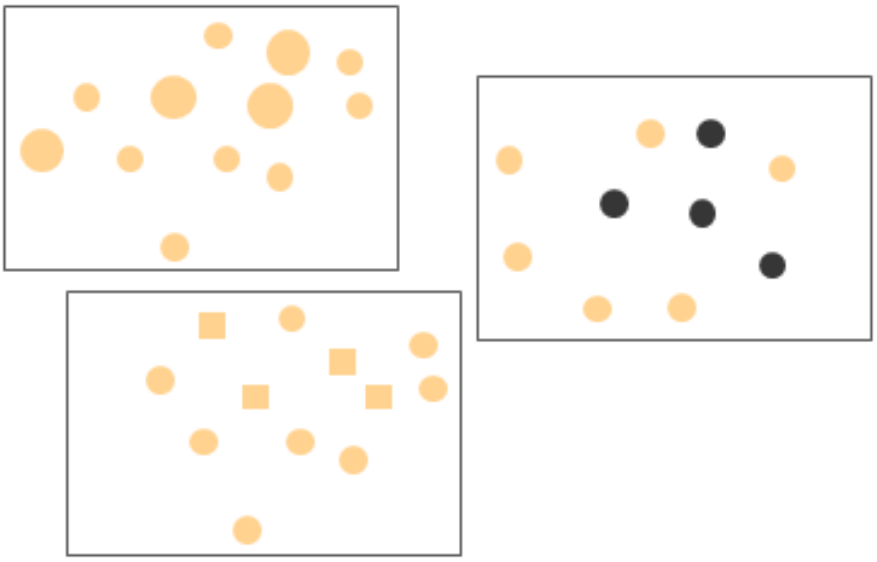


強調色と比較して、中間色と落ち着いた彩色、およびコントラストの使用:



落ち着いた彩色と中間色は目の疲れがはるかに少なく、ユーザー普及率を向上します。コントラストの使用は、関心のあるポイントや例外を素早く目で認識するのに役立ちます。原色でのコントラストの使用は難しいため、これらのコンセプトは両立します。特に、例外または異常値をハイライト表示する場合は、すべてのチャートで落ち着いた彩色とコントラストを組み合わせて使用することを検討してください。

データポイントに注意を促すためのサイズ、形状および明度の使用:



形状は、素早く認識させるためのもう一つのポイントであり、データポイントをグループに分割するために使用できます。色の明度は、値の範囲や異常値を示す場合に有効です。

多数の設計ベストプラクティスが、http://www.demo.qlikview.com のデモアプリケーションで公開されています。さらに、非常に包括的な QlikView の設計技法を紹介したスライド資料集もご用意しています。QlikCommunity にアクセスし、***DataVisualization*** のキーワードを検索してください。

UI 設計のその他のベストプラクティスには、以下が含まれます。

- ✓ 同じ場所の各シートに現在の選択ボックスを配置します。
- ✓ 各シートの同じ場所にリストボックスを表示します。
- ✓ 頻繁に使用する順にリストボックスとマルチボックスを整理します(使用頻度の高いものを上、使用頻度の低いものを下)。次に、リストボックスを階層順でグループに下位分類します(最大グループを上、最小グループを下)。
- ✓ 各ストレート/ピボットテーブルにドロップダウン選択プロパティを配置します。
- ✓ 式を直接式エディタに定義するのではなく、変数を式として使用します。
- ✓ ドリルダウングループを作成する場合に、ドリルグループのフィールドのラベルに式を追加します。例えば、式の結果が、2012 > 四半期 となるように、Only(上位レベルの項目) & '>' & '現在のレベルの項目名' (例:=only(年)&'>'&'四半期')となるようにする必要があります。2012 はドリルダウンした項目、四半期はチャート内に表示される値です。
- ✓ ストレート/ピボットテーブルに異常値を定義するのではなく、異常値であることを視覚的に素早く表示するようなチャートを使用します。
- ✓ 常にヘルプ/利用ガイドタブ、および/またはヘルプサイトへのリンクを含めます。ヘルプ/利用ガイドタブの例は、QlikView の「始めに」のセクションに記載されています。インタラクティブな使い方のヘルプのページを、アプリケーション全体で使用できるテンプレートにコピーすることを検討してください。

- ✓ 分かりやすい見出しで各シートとオブジェクトを命名します。
- ✓ 色覚異常および簡易性を考慮すると、白黒チャートが最適です。
- ✓ 赤と緑 – 視覚的に訴える表現が必要な時は、極力赤と緑は避けるようにします。赤と緑の識別が困難な人がいることを考慮してください。
- ✓ 赤と緑は良し悪しの指標やパフォーマンスとも関連しています。良し悪しを示す場合にのみ、赤と緑を使用します。
- ✓ 組織の標準 PC に適合する解像度に合わせて設計します。(例 1024 x 768)
- ✓ 常に、ソート順、およびリストボックスに頻度(数または%)を表示するかどうかを検討します。(便利な場合もありますが、常にというわけではありません)
- ✓ 各シートの同じ位置に繰り返しオブジェクト(クリアボタン)を配置します。
- ✓ マルチボックスは QV の操作に慣れている人には便利ですが、あまり直観的ではありません。リストボックスにはより多くのスペースが必要ですが、この方が適しています(たとえば、グレーの領域がよく見えます)。
- ✓ 整理されたチャートのレイアウト – 軸タイトル、チャートのタイトル、テキストなどが見やすく整列されるようにします。
- ✓ 階層を持つ軸は順番に配置するようにします。(大分類＞中分類＞小分類)
- ✓ 日付と時間はほとんどのアプリに重要な要素であり、検索および使用のためには極めて直観的であることが求められます。
- ✓ テーブル列は常に検索可能でなければいけません。(必要な場合は常にテーブルに合計を表示)

QlikTech では、すべての QlikView 開発者と設計者に、QlikView の導入を開始する際に設計のベストプラクティスの考え方を組み入れることを強く推奨しています。優れたインターフェース設計により、高い普及率と効果的なインターフェースが可能になります。さらに QlikView の豊富なレイヤにより、すべての QlikView アプリケーションでワールドクラスの可視化と設計を実現できます。

また、QlikView の新規設計者が QlikView を導入する場合は、すべての開発者と設計者が QlikView Designer コースを受講することを強くお勧めします。Designer コースは、優れた設計を強化し、さらにその設計を簡単かつ的確な方法で実現する QlikView 技法を習得できるように構成されています。トレーニングは優れた設計を練習し、その設計を実験的に各自の QlikView アプリケーションに適用できる良い機会でもあります。

UI 設計の参考文献:

QlikTech デモサイト http://www.demo.qlikview.com
QlikCommunity 上で公開している QlikTech の可視化された設計プレゼンテーション、
DataVisualization.ppt
Information Dashboard Design（情報ダッシュボード設計）、（Stephen Few）
Show Me the Numbers（番号表示）、（Stephen Few）
The Visual Display of Quantitative Information（定量的情報の可視化した表示）、（Edward R. Tufte）
Visual Explanations（可視化した説明）、（Edward R. Tufte）

スクリプト作成

概要

スクリプト作成は、QlikView 開発者が QlikView 環境にデータを取り込む際の抽出、変換、ロードプロセスを自動化する環境です。各 QlikView ドキュメント（アプリケーション）には、このプロセスが利用できるスクリプトエディタが搭載されています。

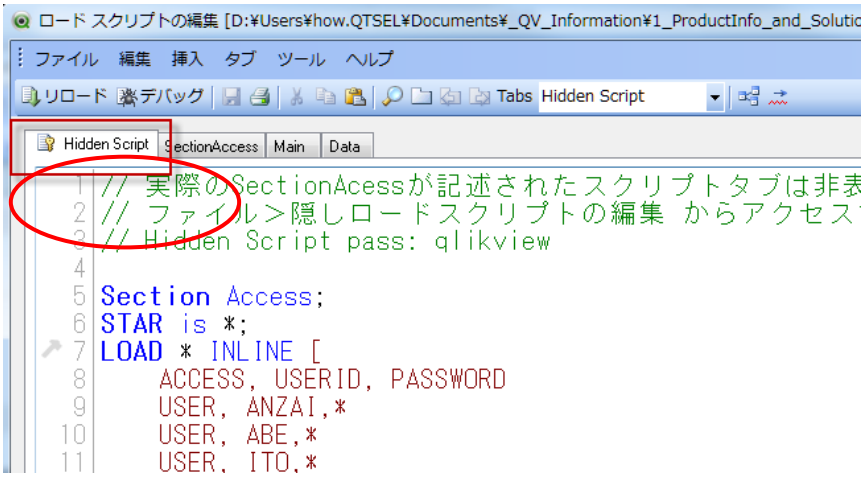
ベストプラクティスでは、スクリプト内で複数のタブを使用するとさまざまな部分が分割され、将来の開発およびサポート情報を簡潔に表示できます。アプリケーションの複雑さによって、さまざまなスクリプトセクションがあります。スクリプトの共通部分は、以下の通りです。

- ✓ セキュリティ（通常は、隠しスクリプト）
- ✓ 日付およびカレンダー情報
- ✓ データソースごとのタブ
- ✓ 主な計数（メジャー）／コアテーブルごとのタブ
- ✓ 検索テーブルごとのタブ

セキュリティタブ（隠しスクリプト）

QlikView では、***ドキュメントプロパティ：セキュリティ***および***シートプロパティ：セキュリティ***タブからドキュメントユーザーの特権を制限できます。ドキュメントユーザーが ADMIN としてログインしている場合、すべての設定を変更可能です。

ユーザー制限されたドキュメントを開く際に必要なユーザーID とパスワードは、ロードスクリプトで指定されます。QlikView にログファイルの生成を許可すると、そのユーザーID とパスワードはログファイルに表示されます。ただし、隠しスクリプトに記述すると、ログファイルにユーザー情報は表示されません。隠しスクリプトを開く [Hidden Script] ボタンは、[ロードスクリプトの編集] メニューにあります。



ロード文を先行

先行 load ステートメントを使用すると、スクリプトが簡素化され、分かりやすくなります。この例として、次のコードをご覧ください。

テーブル 1：
```
LOAD  CustNbr as [Customer Number],
      ProdID  as [Product ID],
      floor(EventTime) as [Event Date],
      month(EventTime) as [Event Month],
      year(EventTime) as [Event Year],
      hour(EventTime) as [Event Hour];

SQL SELECT
      CustNbr,
      ProdID,
      EventTime
FROM MyDB;
```

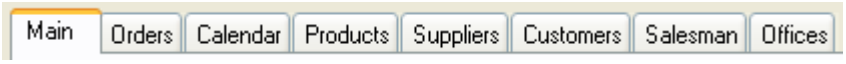
このコードは SQL SELECT ステートメントを簡素化します。開発者は同じ SQL ステートメントに埋め込まれた複雑な QlikView の変換なしで、他のツールを使用してステートメントをテスト／追加することができます。

LOAD 機能の詳細については、QlikView オンラインヘルプ及びリファレンスマニュアルをご参照ください。

その他のスクリプト作成ベストプラクティス：

- ✓ Autonumber は、開発デバッグ完了後にのみ使用します。数字が挿入された値をデバッグする方が、サロゲートのみを使用するより簡単です。Autonumber の使用方法／使用タイミングが不明の場合は、QlikView リファレンスマニュアルをご参照ください。

- ✓ 開発者を混乱させないように、各タブにわかりやすい名前をつけます。



- ✓ Concatenate／join ステートメントを命名します。

- ✓ QVW にスクリプトを追加するには、大きいデータセットにバイナリロードしてから、スクリプトを拡張するのが最善策です。その後、開発が完了に近づいたら、スクリプトを統合します。この操作は機能的には何も変更しませんが、開発時間を短縮できます。

- ✓ HidePrefix=%を使用すると、開発者は設計者がほとんど使うことがないキー項目やその他のフィールドを非表示にできます（これは、共同開発が行われている場合のみ適切です）。

- ✓ Applymap()関数を使用する場合は、'Unknown'&項目 というように、標準的なデフォルト値を入力します。そうすることで、ユーザーはどの値が不明であるかが分かり、管理者をわずらわせることなくソースシステムを調整できます。Applymap()の使用方法／使用タイミングが不明の場合は、QlikView リファレンスマニュアルをご参照ください。

```
StateMapping:
mapping load * inline [
St,State
Tx,TX
Te,TX
Tex,TX];
```

```
LOAD
ApplyMap( 'StateMapping' , St, 'Other')
```

✓ フィールド名にアンダースコアやスラッシュ（つまり「技術的」に使用されるものすべて）は使用せず、代わりに、ユーザーが分かりやすい名前をスペース付きで使用します。
たとえば、「mnth_end_tx_ct」ではなく「Month End Transaction Count」を使用します。

✓ Qualify *は、絶対に必要な場合にのみ使用します。開発者によってはスクリプトの冒頭にQualify *を使用し、キーのみを修飾する場合があります。これにより、Left Join ステートメントなどを使用してスクリプトを作成する場合に多くの問題が発生します。長期的に見ると、作業量の割に価値がありません。Qualify と Unqualify の使用方法／使用タイミングが不明の場合は、QlikView リファレンスマニュアルをご参照ください。

✓ すべての ODBC／OLEDB データベースへの接続には、「Include」ファイルまたは隠しスクリプトを使用します。

✓ パス名はスクリプト全体でハードコードせずに、変数を使用します。こうすることでメンテナンス作業を軽減でき、パスの検索も容易になります（検索しやすいようにパスを最初のタブに入れている場合）。

✓ すべてのファイル参照では、UNC 命名規則を使用する必要があります。C:\MyDocs\...は使用しないでください。

✓ デバッグ用にロード時間情報が必要な場合は、常にログファイル生成オプションをオンにしておきます。

✓ 各タブにスクリプト見出しのコメントを記述します。以下の例をご参照ください。

```
//============================================================================
// App Name:    Wireframe
// Author:      Matt Stephens, QlikTech
// Created:     June, 2010
// Purpose:     This app is a template app demonstrating the use of
//              wireframe backgrounds to organize QlikView screens into
//              logical and effective presentation themes.   There is also
//              a zip file called Wireframe Images.zip that accompanies
//              this QVW.   It holds dozens of pre-built wireframe images
//              in various color schemes.
// Modified:    July 18, 2010 BPN - added Intro tab comments
//============================================================================
```

✓ タブ内のスクリプトセクションに簡単なコメントを記述します。以下の例をご参照ください。

```
// --------------------------------
// Load the Sessions table first
// --------------------------------
Sessions:
LOAD
     MakeDate(LEFT(Timestamp,4), MID(
     Date(Timestamp, 'YYYYMMDD') &'_'
     Time(Timestamp)     as SessionsTi
     Timestamp           as Timestamp,
```

✓ 必要に応じて、変更日付のコメントを追加します。以下の例をご参照ください。

```
Looptable:
LOAD FileName as QVDName
//FROM $(MetaPath)FileList.qvd(qvd)
resident FileList //changed 2010-09-06
WHERE UPPER(Extention) = 'QVD';
```

✓ インデントを使用して、開発者がスクリプトを読みやすいようにします。以下の例をご参照ください。

```
// ---------------------------------------
// Main loop though all QVDs found above
// ---------------------------------------
for X = 1 to fieldvaluecount('QVDName');
    let QVDName = fieldvalue('QVDName',$(X));

    Load *,
        upper('$(QVDName)') & '_' & Date(Today(), 'YYYY-MM-DD') as LoadDateKey,
        lower('$(QVDName)') as FieldQVDFileName,
        Upper('$(QVDName)' &'-'& date(Today(),'YYYY-MM-DD')) as FieldHeaderKey;

    QvdFieldHeader:
    LOAD
        //lower('$(QVDName)') as QVDFileName,
        date(Today(),'YYYY-MM-DD') as FieldHeaderDate,
        FieldName   as QVDFieldName,
```

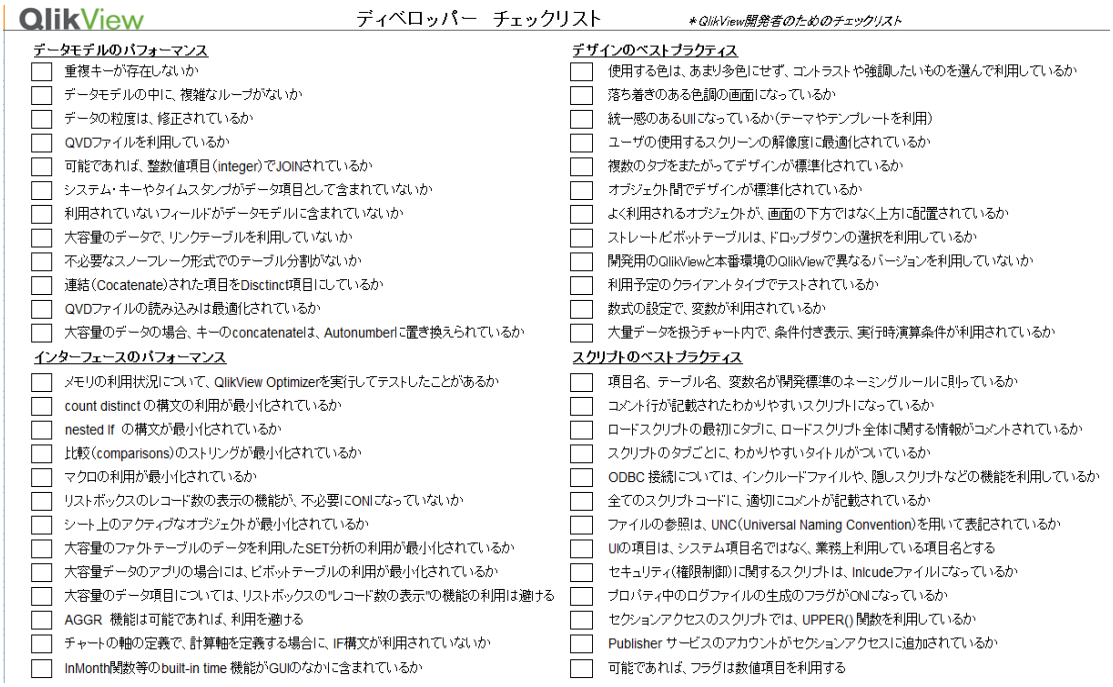
- ✓ load ステートメントで LOAD *は使用しません。ロードする列は明示的に記述します。そうすることで、どのフィールドがロードされるかが分かり、新規列がソーステーブルに追加されたり、ソーステーブルから削除されても影響されません。また、開発者はスクリプト内でロード済みフィールドを容易に識別できるようになります。以下の例をご参照ください。

```
// ============================================================
// Locations Data from a QVD
// ============================================================
Locations:
LOAD LocationNbr        as [Location Nbr],
     AddressLine1       as [Address Line 1],
     AddressLine2       as [Address Line 2],
     City               as City,
     Country            as Country,
     CountryRegionCode  as [Region Code],
     PostalCode         as [Postal Code],
     [State / Province] as [State Code]
FROM [$(QvdPath)Locations.qvd] (qvd);
```

開発チェックリスト

QlikTech は、開発者チェックリストを使用して開発のベストプラクティスを強調および強化することを推奨しています。
多くのお客様では、ベストプラクティスのテンプレートまたはサンプルからこのチェックリストを作成しています。チェックリストの可視性およびプレゼンスを促進する方法の一つは、チェックリストを 1 ページに収め、各開発者に配布することです。チェックリストを掲載および頻繁に参照しやすくなります。一部の顧客は、QVW をテスト環境または本稼働環境にリリースする前にベストプラクティスに準じていることを確認するために、チェックリストをコードの見直しに使用します。

チェックリストのスクリーンショットのサンプルを以下に示します。



QlikView　ディベロッパー　チェックリスト　*QlikView開発者のためのチェックリスト

データモデルのパフォーマンス
- ☐ 重複キーが存在しないか
- ☐ データモデルの中に、複雑なループがないか
- ☐ データの粒度は、修正されているか
- ☐ QVDファイルを利用しているか
- ☐ 可能であれば、整数値項目(integer)でJOINされているか
- ☐ システム・キーやタイムスタンプがデータ項目として含まれていないか
- ☐ 利用されていないフィールドがデータモデルに含まれていないか
- ☐ 大容量のデータで、リンクテーブルを利用していないか
- ☐ 不必要なスノーフレーク形式でのテーブル分割がないか
- ☐ 連結(Cocatenate)された項目をDisctinct項目にしているか
- ☐ QVDファイルの読み込みは最適化されているか
- ☐ 大容量のデータの場合、キーのconcatenateは、Autonumberに置き換えられているか

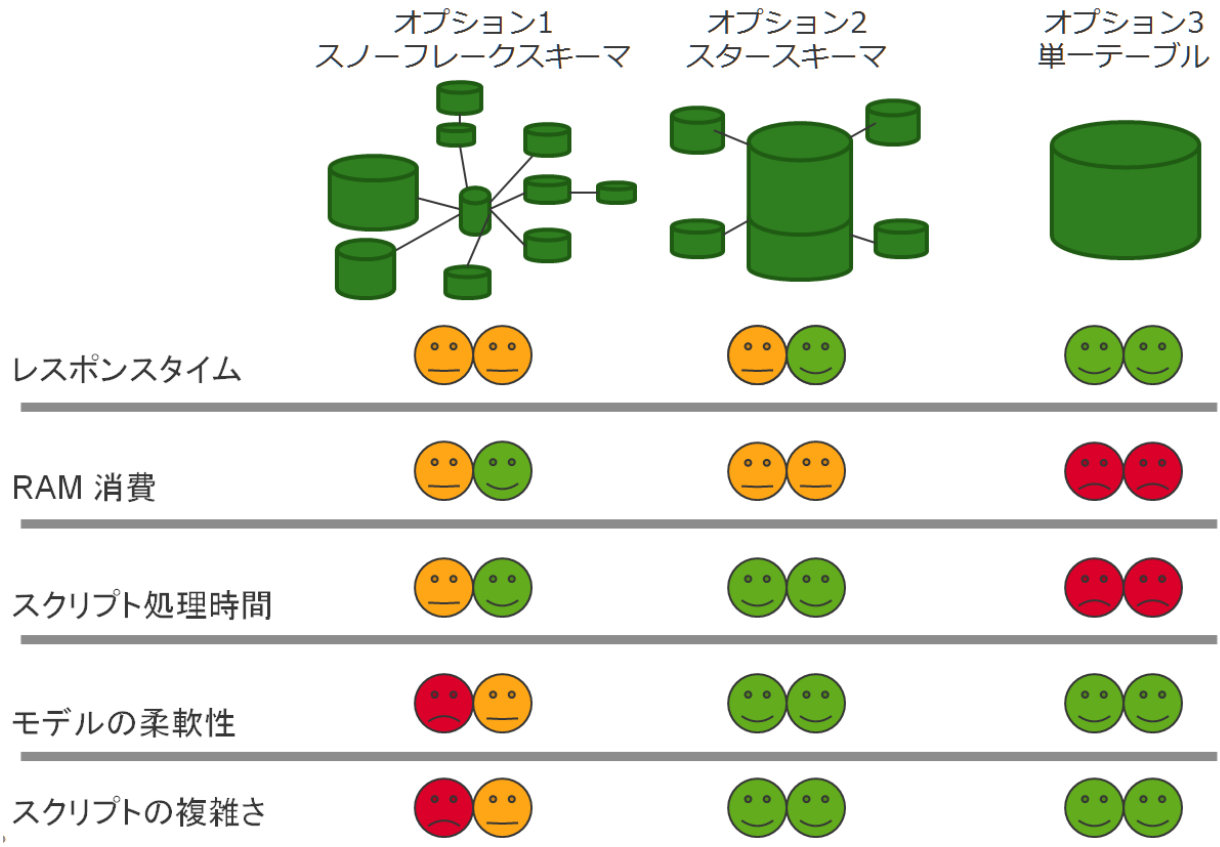
インターフェースのパフォーマンス
- ☐ メモリの利用状況について、QlikView Optimizerを実行してテストしたことがあるか
- ☐ count distinct の構文の利用が最小化されているか
- ☐ nested If の構文が最小化されているか
- ☐ 比較(comparisons)のストリングが最小化されているか
- ☐ マクロの利用が最小化されているか
- ☐ リストボックスのレコード数の表示の機能が、不必要にONになっていないか
- ☐ シート上のアクティブなオブジェクトが最小化されているか
- ☐ 大容量のファクトテーブルのデータを利用したSET分析の利用が最小化されているか
- ☐ 大容量データのアプリの場合には、ピボットテーブルの利用が最小化されているか
- ☐ 大容量のデータ項目については、リストボックスの"レコード数の表示"の機能の利用は避ける
- ☐ AGGR 機能は可能であれば、利用を避ける
- ☐ チャートの軸の定義で、計算軸を定義する場合に、IF構文が利用されていないか
- ☐ InMonth関数等のbuilt-in time 機能がGUIのなかに含まれているか

デザインのベストプラクティス
- ☐ 使用する色は、あまり多色にせず、コントラストや強調したいものを選んで利用しているか
- ☐ 落ち着きのある色調の画面になっているか
- ☐ 統一感のあるUIになっているか(テーマやテンプレートを利用)
- ☐ ユーザの使用するスクリーンの解像度に最適化されているか
- ☐ 複数のタブをまたがってデザインが標準化されているか
- ☐ オブジェクト間でデザインが標準化されているか
- ☐ よく利用されるオブジェクトが、画面の下方ではなく上方に配置されているか
- ☐ ストレートピボットテーブルは、ドロップダウンの選択を利用しているか
- ☐ 開発用のQlikViewと本番環境のQlikViewで異なるバージョンを利用していないか
- ☐ 利用予定のクライアントタイプでテストされているか
- ☐ 数式の設定で、変数が利用されているか
- ☐ 大量データを扱うチャート内で、条件付き表示、実行時演算条件が利用されているか

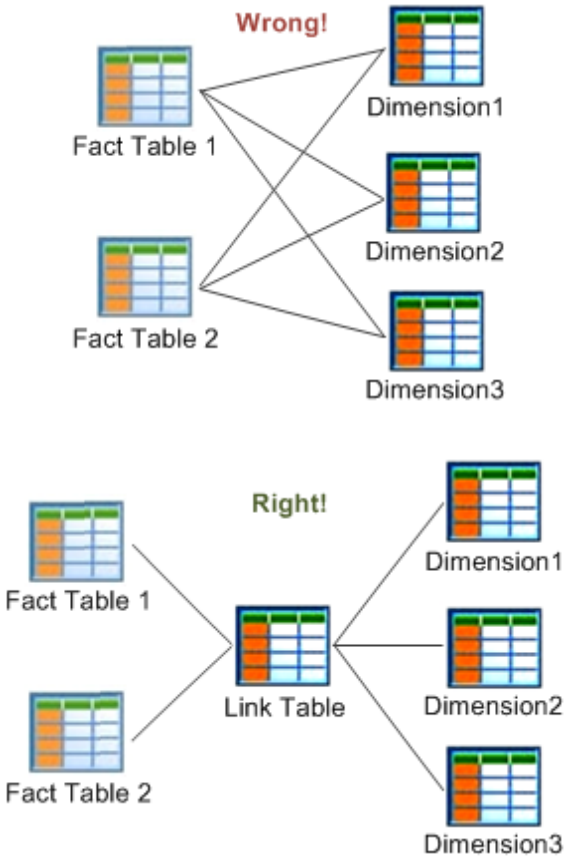
スクリプトのベストプラクティス
- ☐ 項目名、テーブル名、変数名が開発標準のネーミングルールに則っているか
- ☐ コメント行が記載されたわかりやすいスクリプトになっているか
- ☐ ロードスクリプトの最初にタブに、ロードスクリプト全体に関する情報がコメントされているか
- ☐ スクリプトのタブごとに、わかりやすいタイトルがついているか
- ☐ ODBC 接続については、インクルードファイルや、隠しスクリプトなどの機能を利用しているか
- ☐ 全てのスクリプトコードに、適切にコメントが記載されているか
- ☐ ファイルの参照は、UNC(Universal Naming Convention)を用いて表記されているか
- ☐ UIの項目は、システム項目名ではなく、業務上利用している項目名とする
- ☐ セキュリティ(権限制御)に関するスクリプトは、Inlcudeファイルになっているか
- ☐ プロパティ中のログファイルの生成のフラグがONになっているか
- ☐ セクションアクセスのスクリプトでは、UPPER() 関数を利用しているか
- ☐ Publisher サービスのアカウントがセクションアクセスに追加されているか
- ☐ 可能であれば、フラグは数値項目を利用する

データモデル

以下は、QlikView で構築できる 3 つの基本データモデルの図です（他にも多くの組み合わせがあります）。これらの 3 つの例を使用して、それぞれのパフォーマンス、複雑性、柔軟性の違いを説明します。

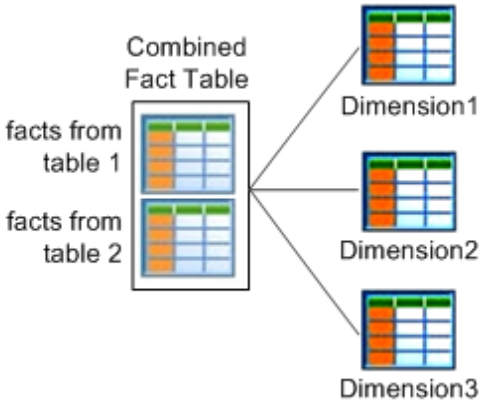


一般的には、スタースキーマが高速で柔軟な QlikView アプリケーションにとって最適なソリューションですが、複数のファクトテーブルが必要な場合があります。以下に、誤った結合方法と正しい結合方法を示します。



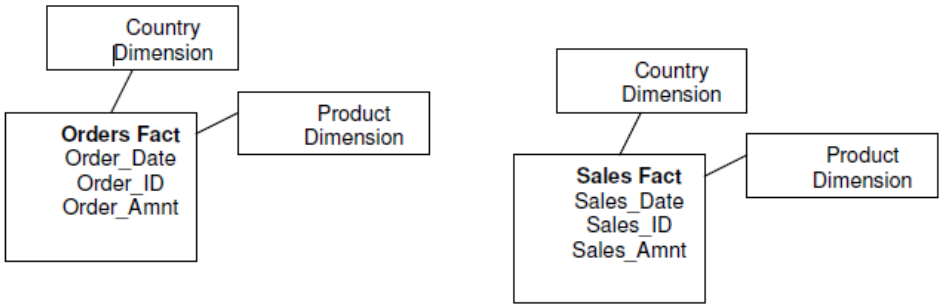
リンクテーブルの作成および使用方法についてのその他の例は、オンラインの QlikCommunity（http://community.qlikview.com/）をご参照ください。

複数のファクトテーブルのモデリング以外に、2 つのファクトテーブルを 1 つのファクトテーブルに結合する方法もあります。以下の図をご覧ください。

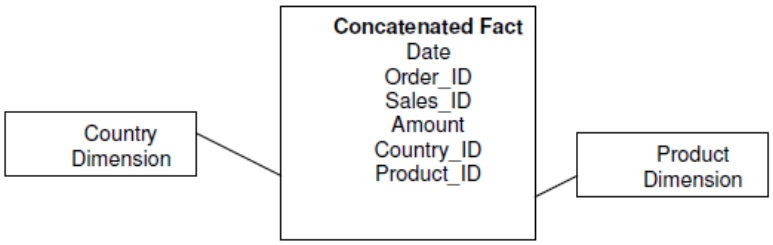


この方法を説明するために、以下のセクションで 2 つのファクトテーブルを 1 つのファクトテーブルに結合するシナリオを示します。

Before



After



Script Example:

```
Load OrdersFact
        Order_Date as Date
        Order_ID
        Order_Amount as Amount
        Country_ID
        Product_ID
        'Order' as TransactionType
CONCATENATE
Load SalesFact
        Sales_Date as Date
        Sales_ID
        Sales_Amount as Amount
        Country_ID
        Product_ID
        'Sale' as TransactionType
```

スクリプトで'Sales' および 'Order' タイプを付与することでトランザクションタイプとして識別できるようになります。

このファクトテーブルの結合の例を以下に示します。

Sales

Region	Product	Date	Sales
RegionA	P1	2009-01-31	100
RegionA	P1	2009-02-28	120
RegionA	P1	2009-03-31	140
RegionA	P2	2009-01-31	500
RegionA	P2	2009-02-28	550
RegionA	P2	2009-03-31	600
RegionB	P1	2009-01-31	50
RegionB	P1	2009-02-28	55
RegionB	P1	2009-03-31	60
RegionB	P2	2009-01-31	200
RegionB	P2	2009-02-28	180
RegionB	P2	2009-03-31	160

Plan Yearly

Region	Date	Plan
RegionA	2009-01-1	8000
RegionB	2009-01-1	10000

Procurement Cost

Product	Date	Cost
P1	2009-01-31	130
P1	2009-02-28	1400
P1	2009-03-31	1600
P2	2009-01-31	500
P2	2009-02-28	650
P2	2009-03-31	600

Concatenated Facts

Region	Product	Date	Sales	Plan	Cost
RegionA	P1	2009-01-31	100		
RegionA	P1	2009-02-28	120		
RegionA	P1	2009-03-31	140		
RegionA	P2	2009-01-31	500		
RegionA	P2	2009-02-28	550		
RegionA	P2	2009-03-31	600		
RegionB	P1	2009-01-31	50		
RegionB	P1	2009-02-28	55		
RegionB	P1	2009-03-31	60		
RegionB	P2	2009-01-31	200		
RegionB	P2	2009-02-28	180		
RegionB	P2	2009-03-31	160		
RegionA		2009-01-1		8000	
RegionB		2009-01-1		10000	
	P1	2009-01-31			130
	P1	2009-02-28			1400
	P1	2009-03-31			1600
	P2	2009-01-31			500
	P2	2009-02-28			650
	P2	2009-03-31			600

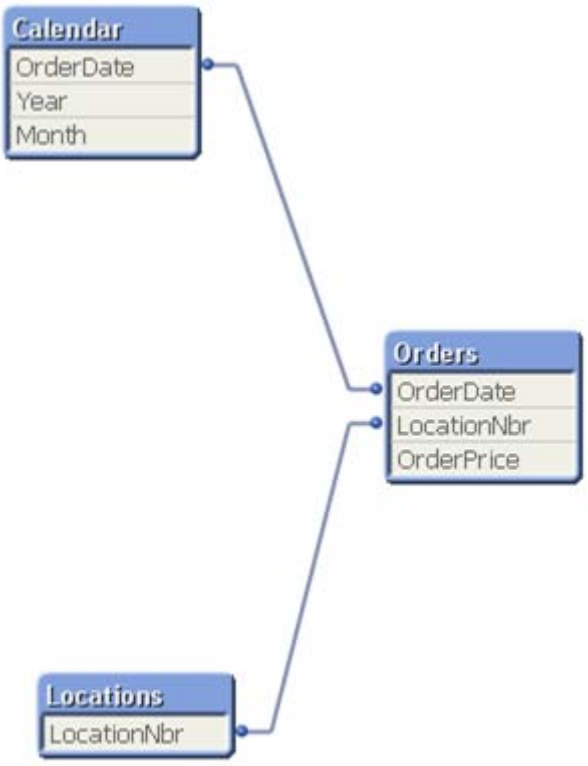
大規模なデータセット

QlikView では、極めて大規模なデータセットを定期的に処理できます。ただし、ユーザー体験と必要なハードウェアを最適化するためには、いくつかの考慮事項があります。

以下のシナリオを考えてみてください。大規模な注文データ（10 億行）があるとします。経営陣にはハイレベルの KPI のサマリー、ビジネスアナリストには傾向分析、受注処理チームには詳細なテーブルと数値を提供する必要があります。QlikView には多くのデータ設計方法がありますが、デモ用に以下の 3 つについて調べてみます。

1. 詳細ファクトテーブルのみ – QlikView が詳細を表示し、明細からサマリーまで、必要な KPI を集計できます。

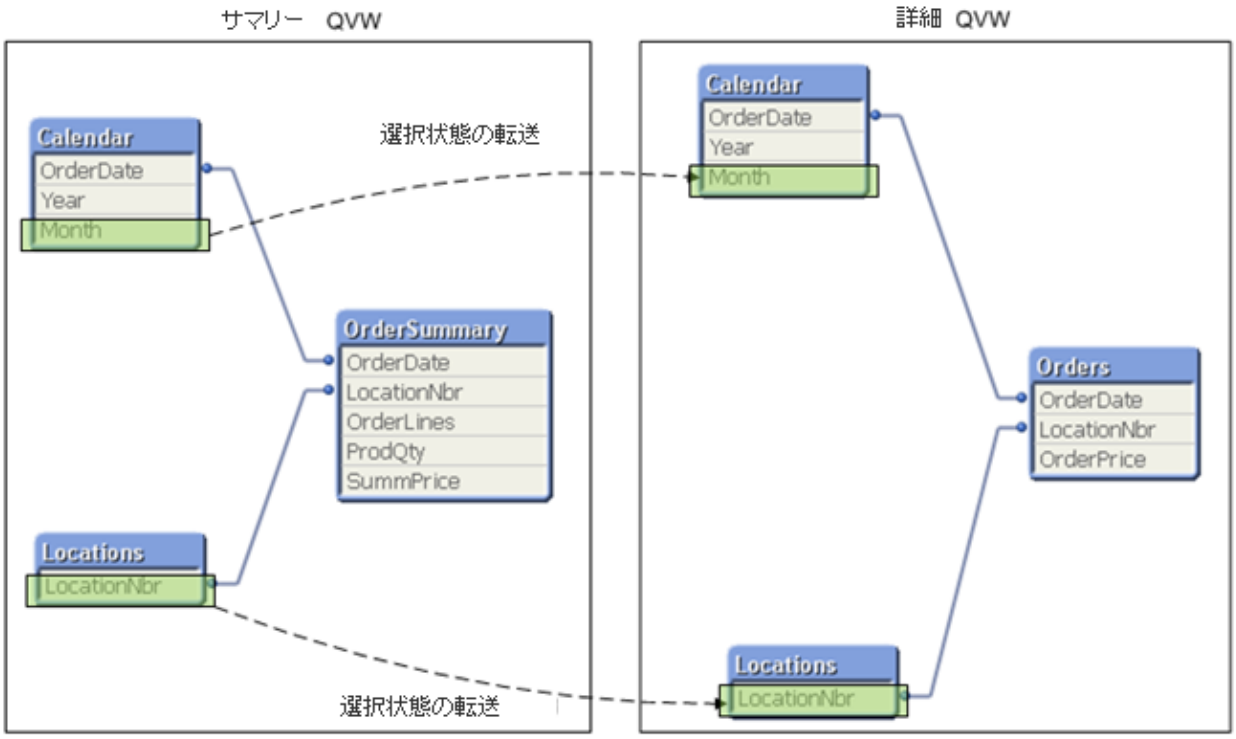
    a. 長所 – シンプル。これは実装するのが最も簡単なソリューションです。オーダーを詳細レベル（おそらく SKU レベル）でデータモデルに結合し、すべての概要レベルの KPI、傾向チャート、詳細テーブル、選択を QVW に設計するだけです。
    b. 短所 – 選択が行われるごとに、最大で 10 億行を演算処理する必要があります。QlikView はおそらく、この作業を許容範囲のパフォーマンスで行える唯一の BI ツールですが、ユーザーの体感速度が必要以上に低くなります。



2. ドキュメント連携 – 2 つ以上の QVW が作成されます。その 1 つは、プライマリファクトテーブルとして注文明細(Orders)テーブルを持ち、もうひとつは、サマリーテーブルとして事前集計された注文テーブルを持ちます。ここでは、QVW は 2 つのみで考えます。以下の図は、「サマリー」QVW からのデータモデルと、「詳細」QVW からのデータモデルを示したものです。2 つのモデルの軸の値はほぼ同じです。主な相違点は、データモデルのファクトテーブルです。ユーザーは、サマリーアプリケーションから開始して、ハイレベルの KPI とチャートを表示できます。

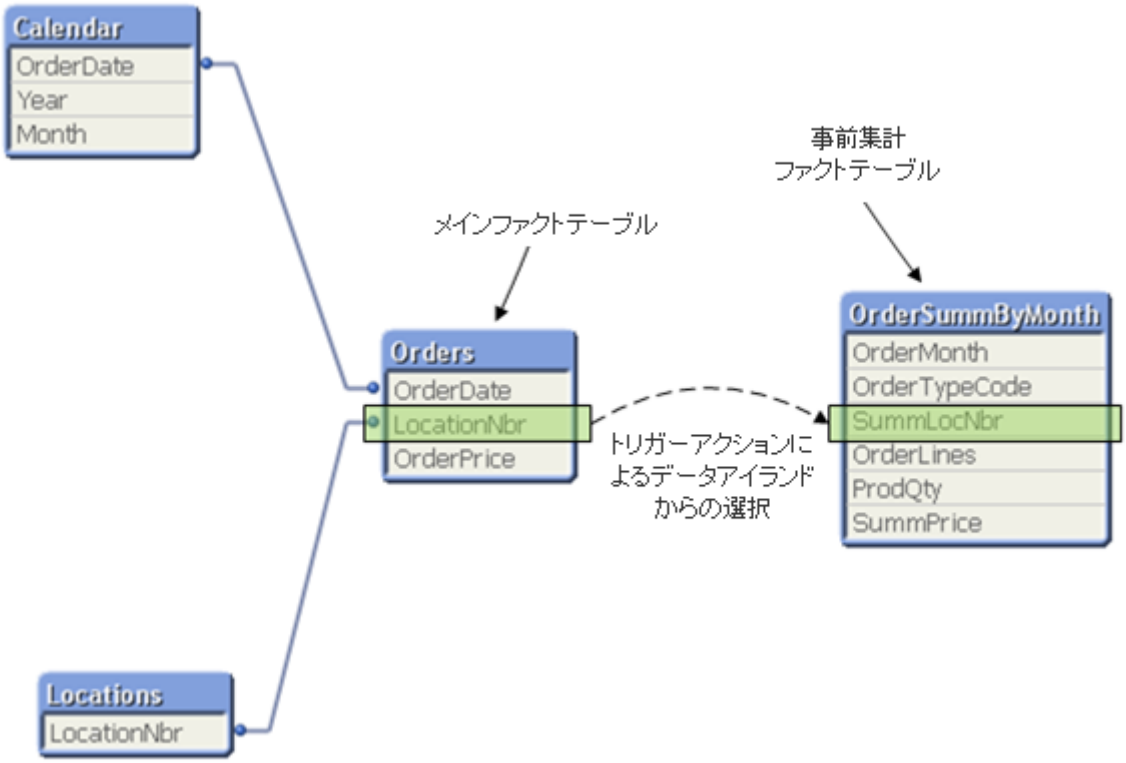
   詳細を掘り下げる場合は、QlikView のドキュメント連携機能を使用して、ある QVW から別の QVW へ選択を転送し、転送先の QVW を開きます。新しいチャートとタブが表示され、(設計に応じて)ユーザーはそれらのチャートとタブが、ある QVW から別の QVW に転送されたことを知る必要はありません。つまり、***ユーザーが必要とする場合にのみ***、10 億行のファクトテーブルを使用します。残りの処理は、1 億行より少ない、たとえば事前集計された注文サマリテーブルで行われます。ドキュメント連携については、QlikView リファレンスマニュアル、および他の QlikView ドキュメントで詳しく説明しています。

   a. 長所 – ハードウェア、および QlikView のナビゲーションとチャート作成の応答速度が最適化されます。ユーザーの選択とナビゲーションは、各自のニーズによって異なります。ユーザーがそのレベルで処理されたものを必要としない場合に、CPU と RAM が 10 億行もの詳細を無駄に処理するのを防ぎます。
   b. 短所 – このアプローチの場合、テーブル(QVD)を事前集計して保持しておく必要があります。これは 1 度きりの開発努力で済むものの、注文テーブルが 1 バージョンのみ必要なオプション 1 よりも多少複雑です。



3. 3 つ目のオプション（最後のオプションというわけではない）は、1 つの QVW データモデルにおいて、詳細テーブルの ***他に***、事前集計されたサマリーテーブルを使用するものです。以下の図は、テーブルの詳細バージョンとして同じデータモデルの事前集計済みテーブルを使用する方法を示したものです。まず、事前集計済みテーブルを、（データモデル内のもう 1 つのテーブルに結合されていない）データアイランドとしてロードします。次に、詳細ファクトテーブルで関連選択がされる際、それらの選択をトリガー済みアクションによって事前集計済みテーブルに転送できます（QlikView バージョン 9 以上）。

   a. 長所 – このオプションでは、大規模なテーブルの詳細バージョンとサマリーバージョンの両方を使用するため、2 つ目の QVW とドキュメントの連携は不要です。
   b. 短所 – このオプションでは、1 つのテーブルから別のテーブルに選択を転送するアクションをトリガーするための設定を QVW で行う必要があります。QVW は時間の経過と共に変化するため、これらのアクションをトリガーする場所とタイミングを追跡する必要があります。



注意：上記シナリオで述べたニーズを満たす方法は他にも多くあります。これらは、極めて大規模なデータセットを管理する QlikView の機能と能力を呼び出す 3 つの例にすぎません。大規模なデータセットおよび QlikView 導入を最適な方法で管理する他の方法の例については、Architecture Best Practices Guide をご参照ください。

データモデルに影響を与える主な要因:

- ✓ 一意の列データ
- ✓ 一意のキー項目情報

どちらも、データモデルのメモリサイズとユーザー体験に影響を及ぼす可能性があります。またテーブルが多くなると、リンクがメモリを多く消費する場合があります。
データ構造を変更することで、メモリ消費量を 50%削減できることが知られており、その結果、UI 応答も高速化されます。

データモデルのサイズと複雑性の削減に関するヒントについては、本ドキュメントの最適化セクションをご参照ください。

変数、マクロ、アクション

変数

QlikView ドキュメント内のさまざまな式を処理する優れたプラクティスを紹介します。最も使用される式は、計算を行うチャートで使用される Sum(Sales)、Sum(Price*Quantity)などの式です。そのような式は、他のオブジェクトやシートで再使用される傾向にあります。チャート属性、色の数式、条件付き表示など、他にも多くの式があり、[設定]>[変数一覧] メニューでそれらすべてを表示できます。

とりわけ高機能なユーザーインターフェースでは、QlikView での式の使用が非常に増える可能性があります。これらの式をより効率的に処理するニーズは拡大しており、この問題は変数を使用することで解決できます。

計算式を変数にする理由:

- ✓ 再利用:通常、販売などの指標の計算式は QlikViewドキュメント全体で同じであるため、各チャートに個別に記述する意味はありません。
- ✓ 計算式の整合性強化:同じ指標を計算する計算式を複数持つというリスクを回避します。
- ✓ 変更管理の一元化:計算式を変更する必要がある場合は、変数を 1 つ変更するだけで、その変数を参照するすべてのチャートとその他のオブジェクトにその変更が適用されます。
- ✓ 必要に応じて、入力ボックスでエンドユーザーに変更を許可:これは、KPI 目標値または汎用パラメータの場合です。

変数は、[変数一覧 / 数式一覧] メニューで、またはスクリプトの SET/LET ステートメントを使用して手動で作成できます。変数は、任意の種類の文字列または数字を保持できる名前と値を持ち、各シートオブジェクトからの参照として使用できます。入力ボックスとは、ユーザーインターフェースに変数を表示できるように設計されたオブジェクトです。

式を変数に設定する場合は、次の手順に従います。

1. いずれかのチャートで [数式] タブに移動し、計算式の 1 つ(たとえば、Sum(SalesValue))をコピーします。
2. [変数一覧] メニューに移動し、「追加」ボタンをクリックして変数を作成します。vFormulaSales などの名前を付けます(フィールドと区別するために、すべての変数名に v から始まる名前を付けるのがベストプラクティスです)。
3. 変数リストから変数を選択し、[定義] テキストボックスのチャートから計算式を貼り付けます。計算式がイコールサイン(=)で始まる場合は、それを削除します。最後に、[OK] をクリックして変更を保存します。
4. チャートプロパティの [数式] タブに戻り、計算式を以下で置き換えます。$(vFormulaSales)

$記号展開は、変数内に含まれる文字列が、計算が必要な式であることを示しています。

最後の手順として、その他のすべてのオブジェクトにおいて、コピーした計算式に置き換えます。すべてが新しい変数を持つ同じ計算式を参照するようになります。Sum(Sales)を表示する必要がある新しいオブジェクトもすべて、変数を参照します。
このプラクティスを適用していない QlikView ドキュメントが相当数ある場合もありますが、今から始めても決して遅くはありません。長期的に見ると、置き換える価値があります。

変数は環境に左右されないため、QVW でハードコーディングせずに環境間でデータベース設定を切り替えるためによく使用されます。そのためのベストプラクティス技法については、下記のサンプルコードをご参照ください。

```
SET vEnvironment= 'PROD';
IF vEnvironment = 'PROD' THEN
…ODBC CONNECT TO MyOracleDBProd (XUserID is *****, Xpassword is ****)
        SET vDBName = 'MyOracleDBProd';
ELSEIF vEnvironment = 'TEST' THEN
…ODBC CONNECT TO MyOracleDBTest (XUserID is *****, Xpassword is ****)
        SET vDBName = 'MyOracleDBTest';
ELSE
…ODBC CONNECT TO MyOracleDBDev (XUserID is *****, Xpassword is ****)
        SET vDBName = 'MyOracleDBDev';
END IF
```

LOAD ステートメントでは、現在は以下のように vDBName を参照しています。

```
SQL SELECT *
FROM $(vDBName).MySchema.MyTable;
```

QVW が移行された際、この変数値を環境ごとに変更する簡単な方法として、以下の 2 つがあります。

1. 開発者または管理者に、スクリプトの変数値を手動で変更するよう依頼します。

2. SET vEnvironment…ステートメントが含まれる Include ファイルを使用します。各環境には、その環境が保持する独自の Include ステートメントのテキストファイルがあります。QVW はそのディレクトリ内にある include ファイルにロードされるため、常に環境に適切な変数セットを取得します。

変数は、共通式（メトリック）ロジックを格納するために使用される場合も、多くの QlikView ドキュメントで使用される場合もあります。式ロジックは、Excel、フラットファイル、またはデータベースに格納できます。

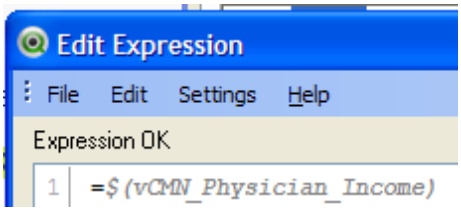
以下のサンプルは、医療関係の指標について Excel ファイルに格納されたいくつかの式を示しています。

Subject	VariableName	VariableValue
Physician	vCMN_Top_Physicians_by_Profit	Max(IncomeChildTopup) + Max(IncomeExcessBedDays) + max(IncomeTariff) + Sum(IncomeTopUp) - Sum(Cost)
Physician	vCMN_Bottom_Physicians_by_Profit	Max(IncomeChildTopup) + Max(IncomeExcessBedDays) + max(IncomeTariff) + Sum(IncomeTopUp) - Sum(Cost)
Physician	vCMN_Physician_Income	Max(IncomeChildTopup) + Max(IncomeExcessBedDays) + max(IncomeTariff) + Sum(IncomeTopUp)
Physician	vCMN_Current_Year_Costs	Sum(Cost * _FULL_TY)
Physician	vCMN_Last_Year_Costs	Sum(Cost * _FULL_LY)
Physician	vCMN_Year_Var_Costs	Sum(Cost * _FULL_LY) - Sum(Cost * _FULL_TY)
Physician	vCMN_Prev_Month_Costs	Sum(Cost * _FULL_PRM)
Physician	vCMN_Current_Month_Costs	Sum(Cost * _FULL_TM_TY)
Physician	vCMN_Month_Var_Costs	Sum(Cost * _FULL_PRM) - Sum(Cost * _FULL_TM_TY)
Patient	vCMN_Visits_Per_Patient	Sum (CountVisits)/Count (PatientCount)
Patient	vCMN_Concept_Cost	Count(PatientCount)*Sum (CONCEPTCOST)
Patient	vCMN_Procedures_Per_Patient	Count (CountProcedureInfoCode)/Count (LASTNAME)

これらの変数は簡単な LOAD ステートメントを使用して QVW ファイルに読み込まれ、以下のロジックを使用して変数に変換されます。

```
// ---------------------------------------------------------------------------
// This code converts the variables from table values to document variables
// ---------------------------------------------------------------------------
Let RowCount = NumMax(NoOfRows('CommonVariables'),0)-1;
For i=0 to '$(RowCount)'
  Let TempVarName = peek('VariableName',$(i),'CommonVariables');
  Let TempVarValue = peek('VariableValue',$(i),'CommonVariables');
  Let $(TempVarName) = '$(TempVarValue)';
Next
```

この変換が完了すると、変数は QVW のいかなる式でも使用できます。変数を使用する式ロジックの例を、以下に示します。



この方法を使用すると、指標のロジックを集中管理できます。スプレッドシートまたはデータベース内のロジック拡張を簡単に変更およびテストでき、次回リロード用にトリガーされると、QVW はロジックをリロードできます。

マクロ

以下は、アプリケーションにマクロステートメントを使用する際に考慮すべき事項です。

マクロを実行すると、すべてのキャッシュ、Undo レイアウトバッファ、Undo 論理操作バッファは自動的に削除されます。これは通常、クライアントが体験しているように、パフォーマンスに多大な悪影響を及ぼします。キャッシュなどが削除される理由は、これらがマクロからの選択であるプロパティを変更する可能性があるためです。その結果として、キャッシュされた状態とマクロから変更された状態間の競合が発生し、これらの競合が事実上、常にクライアントをクラッシュまたはハング（最悪の場合は、サーバーもハングまたはクラッシュ）させます。

マクロ自体は VBS(Visual Basic Script)レベルで実行されますが、QlikView は通常、デフォルトで数千倍高速のアセンブラレベルで実行されます。さらに、非同期で著しくスレッド化されている QlikView とは対照的に、マクロはシングルスレッドの同期であるため、マクロが完了するまで事実上 QlikView の計算をすべて中断してしまいます。QlikView はその後、中断された計算を再開する必要がありますが、これは慎重を要するプロセスであり、（少なくとも、過去の事例を見た限りでは）デッドロックの原因になります（つまり、マクロがまだ実行中で、実行が完了する可能性がないまま、QlikView がフリーズします）。

QlikView はパフォーマンスと安定性において順次最適化されていますが、マクロには常に、パフォーマンスの低下と標準の QlikView 機能との格差という問題があります。マクロは今後も増加し続けると思われますが、パフォーマンスの観点からは望ましいものではなくなっていきます。この事実と、マクロが QlikView で実現されたすべての最適化を弱める傾向にあるという上記の事実とが相まって、マクロが大規模なアプリケーションの重要な部分になるとすぐに、深刻なマイナスのトレードオフが発生します。

マクロは QlikView の機能面では二次的な性質のものです。まず QlikView のすべての内部的な基本機能が実行およびテストされ、続けてマクロが実行およびテストされます。このことは、マクロが事実上、基本となる QlikView 機能と同じステータスまたは優先度になることは決してないことを意味しています。マクロは常に最終手段であり、それ以上のものではないと考えてください。オブジェクトのプロパティなどに関して自動化 API は基本的な QlikView を反映しているため、マクロの内容はバージョンごとに変わる可能性があります。これが移行問題における共通の課題となっています。いったんマクロがアプリケーションに組み込まれると、バー

ジョンが新しくなるたびに、マクロが QlikView の構造的な変更によって影響を受けていないことを確認するために、このアプリケーションを再試験する必要があります。この作業のために、マクロはメンテナンスの点で非常にやっかいなものとなっています。

シンクライアント（Ajax）のサーバー環境ではマクロの一部のみが動作します。ローカルでの操作（クリップボードへのコピー、エクスポート、印刷など）はサポートされていないためです。ローカル操作の一部にはサーバー側と同等の機能（Server-SideExport など）がありますが、各クライアントが事実上サーバーのパフォーマンスに悪影響を及ぼすため、パフォーマンス的には非常に高負荷なものです。

まとめ：我々が尽力しなくてはならないのは、マクロに関する認識を高めることです。マクロを使用して数千レコードを問題なく処理できる場合でも、必ずしも適切にスケールアップできるわけではありません。問題はおのずと明らかになり、大規模なデータセットが関与してくると、その問題はいっそう深刻化します。一方、イベントの中にはマクロを使用しないとキャプチャできないものもあり、その理由でマクロを完全に避けることは困難です。R&D 部門は、マクロの機能を基本的な QlikView の機能としてできるだけ多く組み込むよう常に努力しており、そのため長期的なマクロの使用を制限しています。ただし、前述したように、特定のイベントは外部マクロ以外で捕らえることが困難です。

上記で述べた内容をすべて考慮すると、マクロは推奨される QlikView の設計パターンにはなり得ません。

アクション

アクションは QlikView 9 からの新しい構成要素です。旧ボタンショートカットから生まれたもので、これらボタンショートカットもアクションに置き換えられます。旧ショートカット（シート、シートオブジェクト、フィールドおよび変数に関する最も一般的な操作を含む）より広範囲の操作を提供するだけでなく、一連の操作を 1 つのアクションに定義することもできます。アクションの導入によりマクロの必要性は大幅に減ります。マクロはパフォーマンスの点では決して効率的でないため、これは歓迎すべきことと言えます。

新しいアクションはボタン上で使用できるだけではありません。テキストオブジェクト、線／矢印オブジェクト、ゲージチャートでも使用でき、該当するシートオブジェクト上でクリックすると実行されます。

QlikView 旧バージョンにおけるマクロのトリガーは、アクションのトリガーに置き換えられました。これにより、マクロを使用せずに非常に精巧なトリガーを作成できるようになります。旧バージョンにおけるマクロのトリガーは、QlikView にロードされると自動的に RunMacro アクションに変換されます。

トリガーの詳細については、QlikView リファレンスマニュアルをご参照ください。

プロジェクト管理

概要

このセクションでは、実装フェーズ中ではなくプロジェクトを開始する前に、さまざまなことを検討し決定することを推奨します。

SCRUM 方法論は QlikView でもうまく機能します。その重要な要因は、以下の通りです。

- ✓ QlikView のプロジェクトは非常に迅速であるため、SCRUM のメソッドである頻繁なプロジェクトミーティングは、QlikView 開発にも効果があります。
- ✓ 開発者の一人が共有オブジェクトを変更する場合は、同時に開発を行う開発者の間で、通知のルールを設定する必要があります。
- ✓ 以下のプロセスを定義します。
  - ➢ QA
    - ✧ QA を実行する場合の「エラー」とは？
      - ● 誤ったデータ／合計がエラー
      - ● 誤ったラベル／説明がエラー
    - ✧ 「拡張」とは？
      - ● 項目／シートがエンドユーザーの初回承認に合格済みの場合、レイアウトの変更（項目の追加、変更）は拡張
    - ✧ *実装前にエラーは修正し、拡張（追加開発）は承認を得る必要があるため、エラーと拡張を区別することは重要です。再 QA には多くの作業が必要になるため、QA 中には拡張を避けます。
  - ➢ 変更要求
    - ✧ ユーザーが項目の変更を要求する場合にするべきこととは？許可を求めるタイミングはいつか？
- ✓ コミュニケーションおよび実行計画
  - ➢ 範囲の変更または拡張要求を検討するために、主要関係者がミーティングを開くのはいつか？

DSDM の方法論

- ✓ RAD の方法論に基づく
- ✓ アジャイルメソッドの 1 つで、アジャイルアライアンスに参加
- ✓ プロセスとコンセプトは SCRUM(スクラム:アジャイルソフトウェア開発の手法のひとつ）と類似していますが、以下の点が異なります。
  - ➢ SCRUM より専門用語が少ない
  - ➢ SCRUM の役割およびタイトルの教育なし
  - ➢ 必要なドキュメント数が少ない
- ✓ 世界的に認識されているアジャイル RAD 方法論
- ✓ 反復かつ増分
- ✓ 継続的なユーザー関与を重視
- ✓ 「スケジュール通り、予算通り」で、経過時間と範囲意識を重視
- ✓ スケジュールに組み込まれた要件変更に対する調整
- ✓ 重要な顧客プロジェクトおよび PMO への簡単な組み込み
- ✓ 「分かりやすい言葉」によるプロジェクト努力、役割、文書化
- ✓ 周期的な受注および売上増

RAD／DSDM の方法論

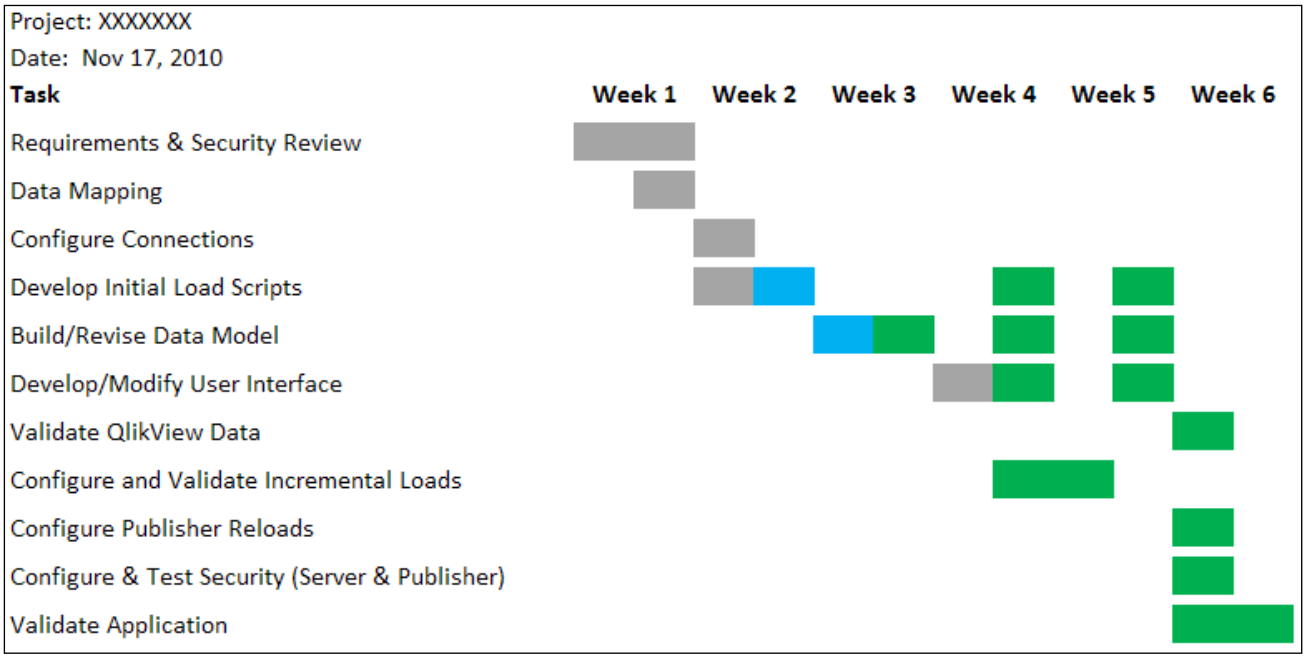
- ✓ RAD = Rapid Application Deployment（高速アプリケーション開発）

- ✓ DSDM = Dynamic Systems Development Method（ダイナミックなシステム開発方法）
  - ➢ RAD および DSDM の方法論は、北米およびヨーロッパではアジャイルプロジェクト管理アライアンスの一部として広く認識される
  - ➢ RAD および DSDM の方法論は、変更を最低限に抑えた典型的な QlikView プロジェクトのプロファイルに適合
  - ➢ RAD および DSDM の方法論は、プロジェクトのガバナンスおよびテンプレートを促進するために参照および調査可能
  - ➢ RAD および DSDM の方法論は、知識移転の基盤を提供
  - ➢ RAD および DSDM の方法論は、無駄のない完全なリソース要件とドキュメンテーションを提供

RAD／DSDM の要素

プロジェクトフェーズ	3 フェーズ: ● プロジェクト準備 ● プロジェクト開発ライフサイクル ● プロジェクトの完了
プロジェクトチームリソースの役割	6 つの役割 ● プロジェクトオーナー／スポンサー ● 技術アナリスト ● プロジェクトマネージャー／ビジネスアナリスト ● 専門的なサービスコンサルタント ● QlikView サービスパートナー開発者 ● 顧客プロジェクトチーム
必須ドキュメント	8 ドキュメント ● スコープ定義を含むプロジェクト憲章 ● 要件:ビジネス、機能、非機能、技術 ● テスト計画およびサマリー ● プロジェクトのスケジュールおよび計画 ● 設計および開発のサマリー ● 引き継ぎおよびサポートのサマリー ● プロジェクトの完了時のチームインタビューのサマリー ● 顧客満足度インタビューのサマリー
定義ドキュメント	「プロジェクト憲章」 目次付きの単一ドキュメント ドキュメント: ● 憲章の目的、エグゼクティブサマリー、プロジェクト概要、目標、目的および成果物を含めた範囲、仮定を含めた条件、コミュニケーション計画、課題管理、リスク管理、制約事項、エスカレーションパス、構造的アプローチ、チーム組織計画、チーム連絡帳 ● 付録ドキュメント:プロジェクトスケジュール（スプレッドシート）、SOW（Scope of work)、変更要求、マイルストーンサマリー（要件、設計および開発、テスト、導入）
プロジェクトスケジュール	MS Project にエクスポート可能な Excel スプレッドシート、およびその他の MS Project 形式のプロジェクト管理ソフトウェア

QlikView プロジェクトのサンプルプロジェクト計画を以下に示します。QlikTech は、QlikView プロジェクトにおいて、プロジェクト計画を作成し、そのプロジェクト計画を順守することを推奨します。また大規模プロジェクトの場合は、適格なプロジェクトマネージャーのサポートを受けることを推奨します。その他多数のテンプレートやサンプルは、QlikTech Expert Services、またはオンラインの QlikCommunity より入手可能です。



セキュリティ（セクションアクセス）

セキュリティに対処するために、QlikView スクリプトにセクションアクセスを設定できます。セクションアクセスは.qvw ファイルに保持されます。つまり、1 つのファイルを作成すると、多数のユーザーやユーザーグループのデータを格納できます。QlikView は、セクションアクセス内の情報を使用することにより、認証と許可を行い、許可されたデータのみが各ユーザーに表示されるよう、データを動的に削減します。

QlikView では、以下のアクセスレベルが提供されます。

- ✓ ADMIN – ドキュメントのすべての項目を変更できます。ADMIN アクセス権を有する者は、[**ドキュメントプロパティ**] および [**シートプロパティ**] ダイアログの [**セキュリティ**] ページを使用して、ユーザーがドキュメントを変更する可能性を制限できます。
- ✓ USER – [セキュリティ] ページにはアクセスできません。
- ✓ NONE – 明確にするためにオプションで使用され、常に「アクセス権なし」と見なされます。

QlikView Publisher は、その「loop and reduce」機能を使用して、リロード中にユーザーまたはグループごとに行単位で QVW を分割できます。また、ドキュメントを開いているときに、セクションアクセスで動的に行うこともできます。どちらの方法も効果的で、それぞれ利点があります。QlikView Publisher の「Loop and reduce」機能は、サーバー上の QVW のメモリ消費量を削減できる一方、セクションアクセスによる方法は、ドキュメントと一緒に移植できます。セクションアクセスを使用するもう一つの理由は、QVW でユーザーID、パスワード、またはその両方を使用してアプリケーションを認証できるという点です。この点は、QVW を AccessPoint からダウンロードするために有効化する場合、またはユーザーに配布する場合は特に重要です。

QlikTech は、QVW を配布するに際は、QVW をパスワードで保護する、または少なくともセクションアクセスを使用してユーザーID で QVW を認証することをお勧めします。

セクションアクセスを使用する際のベストプラクティス:

- ✓ セクションアクセスでは、load ステートメントを利用する場合は常に Upper()関数をすべての列で使用します。（.qvd から読み込みであっても）
- ✓ セキュリティ用の AD グループ
- ✓ include ファイル内のセキュリティ
- ✓ QlikView Publisher 機能を使った QMC 上でのセクションアクセステーブルの一元管理
- ✓ リンクテーブルのないデータモデルで「スタースキーマ」設計を利用します。リンクテーブルはパフォーマンスを大幅に低下させます。
- ✓ 1 つのファクトテーブルにすべての軸が直接結合されている状態が最適です。まれなケースとして、追加の「スノーフレーク」軸を使用します。
- ✓ ファクトテーブルに定義する最大列数は 30～40。（多少の増減は可能ですが、適切なサーバーを使用している場合、ファクトテーブルのレコード数が 1,000 万件未満の場合を除き、150 列を超えるようにはしません）

ロードスクリプトや数式の最適化

概要

アプリケーションの開発フェーズが完了し、導入フェーズが開始された場合、エンドユーザー体験が円滑でシームレスになるよう、アプリケーションのメモリ使用量を最適化するベストプラクティスを検討することが非常に重要です。このセクションでは、データの最適化と式の取り扱いにおけるベストプラクティスについて説明します。

データのロード

ロード後、不要なフィールドはすべて削除します。不要なフィールドとは、チャート、リストボックスなどで現在使用されていないもの、つまり未使用のフィールドです。

- ✓ データが膨大な場合は、複数の時間枠に分割するようにします。例：5 年間（またはそれ以上）のデータから「今年対昨年」の QVW を作成します。これは、今年と昨年のデータを比較したいためです（この場合）。注：これにより、エンドユーザーの操作も速くなります。ユーザーの 80%が過去 13 カ月分のデータのみを見ていることが分かりましたが、QVW には 60 カ月分のデータがあると仮定します。2 つのバージョンの QVW（過去 13 カ月分のデータを持つバージョンと、60 カ月すべてのデータを持つバージョン）を作成すると、ユーザーはまず 13 カ月のバージョンの QVW を分析し、もう 1 つのバージョンは必要なときにのみリンクできます。この結果、エンドユーザーセッションの 80%は、アプリケーションを分割する前と比較して、消費する RAM、CPU、および処理時間はわずかです。エンドユーザーの体験が向上し、ハードウェアの可能性がいっそう広がります。

- ✓ データを正規化しすぎないようにしてください。典型的な QlikView アプリケーションでは、合計で 6～10 のテーブルになるように計画します。これはガイドラインにすぎませんが、QlikView のデータモデルとのバランスがあります。詳細は、本ドキュメントの「データモデル」セクションをご参照ください。

  Mapping Load を使用して小さい「leaf」テーブルを消去し、コード値を他の軸またはファクトテーブルに合わせます。

- ✓ Count(Distinct)は遅いので消去

- ✓ Count(Distinct)と同様に遅いので、Count(数値)または Count(テキスト)を消去

- ✓ 可能なものはすべて文字列ではなく数字で保存

- ✓ フィールド数を減らしてテーブルを非正規化

- ✓ ファクトレコードからのスノーフレーク軸は 1 レベルのみ（ファクト、軸、スノーフレーク）

- ✓ 適用可能な場合は、Autonumber を使用

- ✓ Incremental Load テンプレートを使用して増分的にロードし、履歴の.qvd ファイルを増分時間枠に基づき複数の.qvd に分割

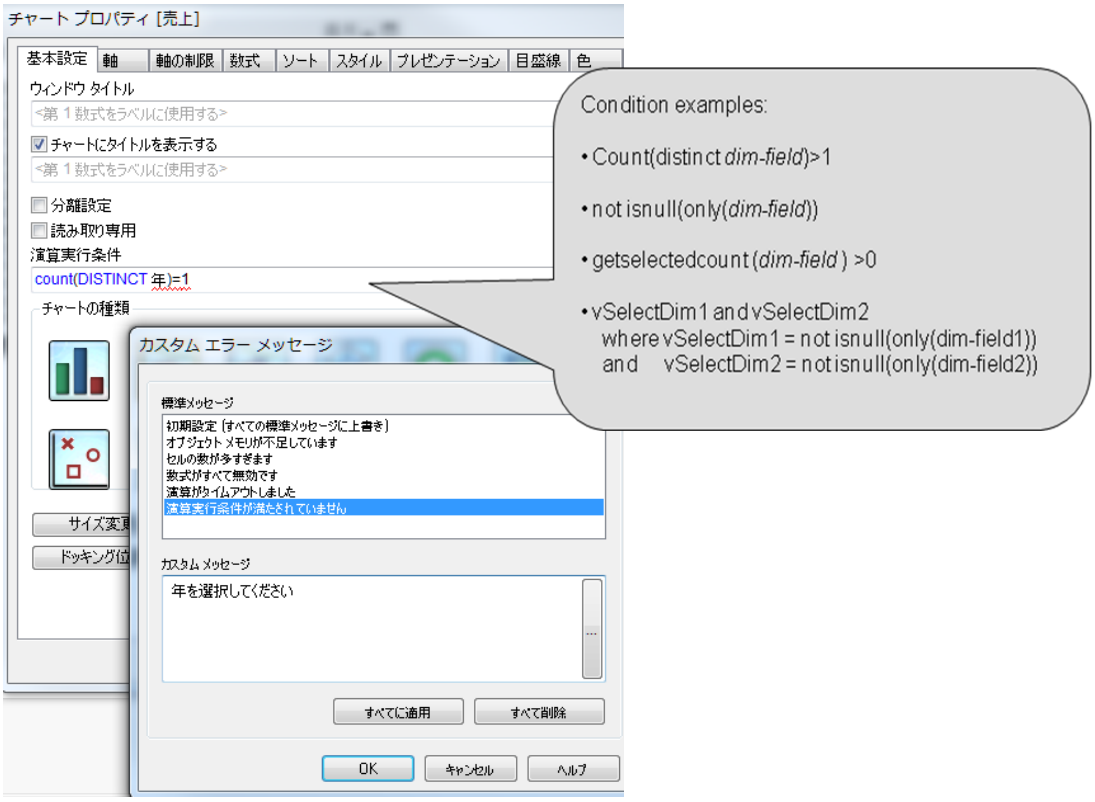
- ✓ ファイルを参照する場合は、常に相対パスを使用

- ✓ UNC 名を使用。自動化タスクはパスを参照できない場合があるため。

- ✓ include ファイル内のローカルユーザー設定
  - ➢ 接続定義用の include ファイル
- ✓ 日付と時刻が必要な場合は、タイムスタンプを日付フィールドと時刻フィールドに分割
- ✓ 時刻が不要の場合は、floor()または date(date#(..))を使用して日付から時刻を削除
- ✓ すべての関連テーブルを 1 つのスクリプトで処理する場合は、autonumber()により広範囲に連結されたキー項目を縮小
  - ➢ （文字列と、結果として表示される数値フィールドの長さが同じ場合は、英数字フィールドを変換する利点はありません）
- ✓ 論理関数（IF）には数値フィールドを使用（文字列の比較は遅い）
- ✓ (a – b) / b よりも(a / b) – 1 が好ましい
- ✓ date(max(SDATE,'DD.MM.YYYY')) は max(date(SDATE,'DD.MM.YYYY'))より高速
- ✓ ソースデータの粒度が本当に分析に必要かを精査する
- ✓ スクリプトで事前計算された数値フラグ（1 または 0）を使用
- ✓ sum(Flag * Amount) 対 sum(if(Flag, Amount))
- ✓ 開いているチャートオブジェクト数を削減
- ✓ なるべくスクリプト内で計数値（メジャー）を計算（モデルサイズ<>オンラインパフォーマンス）
- ✓ チャート／ピボットオブジェクト内の式の数を制限し、複数のオブジェクトに分散（自動最小化を使用）
- ✓ サーバーBIOS 内の Hyperthreading を無効化。Hyperthreading（Intel の CPU のみ）はスクリプト処理を減速させる可能性あり
- ✓ マクロを使用する場合は慎重に！
- ✓ 非常に大きい QVW の場合は、RAM で選択を事前キャッシュすることでさらに最適化できます。QlikView は、チャート内の計算式の計算結果を共有キャッシュメモリに格納します。計算式とフィルターが同じ場合、同じユーザーまたは別のユーザーがキャッシュ結果をフェッチします。つまり、処理せずに結果が即座に配信されます。キャッシュエントリは、QV モデルが更新後などにリロードされるまで、割り当てられたキャッシュメモリに残ります。つまり、キャッシュが空であるため、リロード後最初のユーザーには待機時間があります。この問題は、Visual Basic スクリプト（VBS）で解決できます。VBS はアプリケーションでユーザー選択をシミュレートし、データモデルの更新後に（QlikView Publisher の外部実行タスクによって）自動的に起動されます。
  以下に示すこの VBS サンプルは、アプリケーションの全フォルダで実行され、すべてのチャートを開き、軸 Region 内のすべてのフィールドを選択します。

```
set x = CreateObject("QlikTech.QlikView")
set doc = x.OpenDoc("qvp://BISERVER/xyz/Value_Management_Dashboard.qvw")

loop_through_objects(doc)
doc.CloseDoc
x.Quit

'I run through the application. Called by IOpenTheDocument
Sub loop_through_objects(doc)

  For i = 0 to doc.NoOfSheets - 1

      doc.ActivateSheet i
      ' Selection-loop thru the field Region
      Set val=doc.Fields("Region").GetOptionalValues
      For y=0 to val.Count-1

       INH=val.Item(y).Text
       doc.Fields("Region").select INH

       Objects = doc.ActiveSheet.GetSheetObjects

       For j = lBound(Objects) To uBound(Objects)
          set table = Objects(j)
          select case Objects(j).getObjectType
          case 1,4,10,11,12,13,14,16,20,21,27
           On Error Resume Next
            CellRect = doc.GetApplication().GetEmptyRect()
           CellRect.Top = 0
           CellRect.Left = 0
           CellRect.Width = table.GetColumnCount
           CellRect.Height = table.GetRowCount
           set CellMatrix = table.GetCells( CellRect )
           For RowIter = 0 to CellMatrix.Count-1
           Next
          End Select
       Next
      Next
      doc.Fields("Region").clear
  Next

End Sub
```

✓ 大容量のテーブルデータを扱う場合には、計算条件式を使用する計算を制限します。チャート・オブジェクトの計算は、特に複雑な計算式が多くある場合は、システム負荷を増加させます。そのためフィルターが設定されていないと、待機時間が生じる場合があります。チャート・オブジェクトの計算を開始する前に、年度、製品カテゴリ、地域、またはこれらの軸すべてを強制的にユーザーに選択させるのは理にかなっています。



✓ 複雑なデータ計算式の代わりに、変数、および／または Set Analysis を使用します。式内で時間関数を使用すると（下記 1～3）、待機時間は単純比較（下記 4）または Set Analysis（下記 5）の場合と比較して 3～15 倍になります。

1. sum(if(inmonth (Date,date(max(total Date)),-12), Sales))
2. sum(inmonth (Date,date(max(total Date)),-12) * -1 * Sales)
3. sum(if(inmonth (Date, vPYMonthEnd,0), Sales))
4. sum(if(Date>= vPYMonthStart and Date <= vPYMonthEnd, Sales))
5. sum({$<Date={">=$(vPYMonthStart) <= $(vPYMonthEnd)"}>} Sales)

これらの最適化ベストプラクティスは、QlikView Developer および Designer トレーニングコースで紹介し、演習します。QlikTech は、QlikView の導入を最適化し、QlikView で実現できる ROI を最大化するためにも、このトレーニングに参加されることを強くお勧めします。

コード管理および移行ガイドライン

コード管理

コード管理のベストプラクティスの概要を提供するため、企業における QlikView アプリケーションの変更プロセスを取り扱った架空のケーススタディをご紹介します。架空の会社名は Acme Industries です。

レポート:Acme Industries:変更管理ガイド

QlikView アプリの変更 – リリースプロセス

エンドユーザーが QlikView アプリケーションの拡張を要求したり、不具合を報告することもあります。そのような変更要求は、可能な場合は Acme Industries の標準ソフトウェア変更管理手順を使用して管理します。

複数の変更要求を実装し、新規リリースにロールアップできます。新規リリースの頻度は決定されていませんが、アプリケーションが安定すると、新規リリースは日次的なイベントではなく、四半期または月次のイベントとなります。

変更要求の作成

変更要求は、まず Acme Industries の変更要求ツールまたはバグ追跡システムに入力する必要があります。その後、QlikView プロジェクトマネージャーが変更要求に優先順位を付けます。変更要求が増えてくると、プロジェクトマネージャーは最も高い優先順位の変更要求を QlikView 開発者に割り当て、実装を依頼します。

変更要求の実装

1. QVW ファイルのチェックアウト

   次に、QlikView 開発者は、Acme Industry のリビジョン管理ツールから適切な QVW ファイルをチェックアウトします。(QVW ファイルをチェックアウトする代わりに、QlikView 開発者がスクリプトファイルおよびレイアウトファイルを QVW からエクスポートし、優先するリビジョン管理ツールで管理することもできます)。

2. QlikView アプリケーションの変更

   QlikView アプリケーション開発者は、QlikView Designer および Developer コースで身に付けた技術をもとに、QlikView アプリケーションを拡張、および/または修理します。QlikView アプリケーションは、開発者のデスクトップ上でローカルに、またはテストサーバー上でリモートに変更および拡張できます。次の表は、この 2 つの方法を比較したものです。

開発場所	必要なデスクトップソフトウェア	必要なサーバーソフトウェア	注
ローカルデスクトップ	QlikView Desktop	なし	サーバーベースの開発は大規模データの場合に有効です。
リモートテストサーバー	Windows リモートデスクトップ接続クライアント	QlikView Desktop	

通常、大規模な導入には、**テスト**用と**本稼働**用に別のホストを使用するのが優れたプラクティスです。ただし、小規模な導入の場合は、必要に応じて 1 台のマシンを両方の役割に使用できます。1 台のマシンを両方の役割に使用する場合は、**テスト**用および**本稼働**用ファイルを別のディレクトリツリーに格納します。システム管理者は、Windows NTFS ファイル権限を使用して**本稼働**ディレクトリツリーへのアクセスを制御できます。本ドキュメントの残りの部分では、別のテスト用および本稼働用のホストの使用について説明します。ただし、1 台のマシンで別のテスト用および本稼働用ディレクトリ**ツリー**を使用する場合も、同じ概念を適用できます。

3. テストサーバーへのテスト導入

   QlikView 開発者は、テストサーバーの QlikView サーバーフォルダに更新済み QVW ファイルをコピーし、優先するクライアントタイプ（QlikView の Internet Explorer プラグインなど）からアクセスした際にアプリケーションが予期した通りに動作することを検証します。

4. 本稼働サーバーへのテストリロード

   QlikView 開発者はサーバー管理者と連携し、更新済み QVW ファイルが本稼働サーバーへデータを正常にリロードできることを検証します。QlikView 開発者が更新済み QVW ファイルの新しいデータソースに call を追加した場合、本稼働環境ではこれらのデータソースを使用できない可能性があるため、このテストは特に重要となります。

5. チェックイン前のデータ削減

   ソフトウェア開発におけるリビジョン管理システムの目的は、アプリケーションによって処理されたデータではなく、ソフトウェアアプリケーションの成果物自体の変更を管理することです。そのため、新しいバージョンの QVW ファイルをチェックインする前に、QlikView Desktop の**ファイル> データの削除 > すべての値を削除**により、QVW ファイルからすべてのデータを削除する必要があります。
   これにより、QVW ファイルのサイズがおよそ 2 MB から 200 kb に削減されます。ロードスクリプトが実行されている場合は、本稼働サーバーへの導入後 QVW ファイルにデータが追加されるため、データのない QVW ファイルをチェックインしても問題はありません。

   この手順は、QVW ファイルのリビジョン管理を実施している場合にのみ適用できます。他のアプローチによりスクリプトファイルおよびレイアウトファイルのリビジョン管理を実施している場合は適用できません。

6. チェックインおよび導入

   開発者は、更新済みかつ縮小済みの QVW ファイルをリビジョン管理システムにチェックインし、システム管理者に更新済みファイルが使用可能であることを知らせます。次にシステム管理者は、更新済み QVW ファイルをサーバー上の**本稼働**ディレクトリにコピーします。

   Acme が他のアプローチを使用して QVW ではなくレイアウトファイルとスクリプトファイルのリビジョン管理を実施している場合、開発者は修正済みスクリプトファイルとレイアウトファイルをチェックインし、システム管理者はこれらの更新済みスクリプトファイルとレイアウトファイルを本稼働用の QVW ファイルにインポートする必要があります。

変更要求のクローズ

変更要求は、Acme Industries の変更要求ツールまたはバグ追跡システムで「closed」とマーク付けする必要があります。
以上が、QlikVIew の変更管理ソリューションの実装に関する概要説明です。

移行ガイドライン

このセクションでは、開発者および専門家ユーザーによる本稼働用 QlikView ドキュメントの変更を、本稼働環境へ移行する流れについて説明します。プロセスを正しく進めるためには、数人が連携して取り組む必要があります。

以下は、本セクションで使用するさまざまな役割についての説明です。

役割

役割	責務
管理者	QlikView Publisher の夜間ジョブ（作成、変更、保守） 変更の懸け橋としての役割（ソース管理） 開発者／専門家ユーザーによる新規変更を本稼働へ統合
開発者	通常、データモデルを開発し、ソースシステムと連携
専門家ユーザー	通常、チャート／レポートおよびレイアウトを作成／変更
QA ユーザー	レポート要件を定義し、レポートが正しいことを最終的に検証するエンドユーザー。開発チームのメンバーではない開発者などが担当。

定義

ドキュメント：QlikView QVW ファイル。=「QVW」

本稼働候補：最新の変更がすべて適用されたドキュメント。通常は、管理者の手中にあるドキュメント。どのドキュメントが本稼働候補であるかを把握するのは管理者の責務であり、本稼働候補は 1 つのドキュメントにつき常に 1 つのみです。

変更の定義

変更には、開発者がドキュメントを開く、編集する、および変更を保存する必要がある、すべてのものが含まれます。これにはレイアウトの変更（項目などの移動）、色の変更、あらゆるもの（リストボックス、チャート、グラフィックなど）のプロパティの追加／変更、スクリプト、サイクルグループ、ドリルグループ、マクロコードの変更も含まれます。
ただし、自動プロセス（QlikView Publisher など）による夜間のデータ更新は含まれません。

概要

基本的なプロセスは、開発者がドキュメントのレイアウトをいくつか変更し、その変更を本稼働環境に統合します。開発者は変更が正しいことを確認後、管理者に連絡します。管理者は開発者（開発者は複数の場合もある）による変更を統合し、開発者がレビューできるようにドキュメントを QA に置きます。

開発者とエンドユーザーがドキュメントが正しいことを検証すると、管理者は変更を本稼働に移行できます。
変更が多数の場合は、管理者はすべての関係者と共同でリリーススケジュールを構築し、それらの変更を一括でリリースします。

エンドユーザーコミュニケーションプロセス

特定のアプリケーションで新機能が使用可能になったことをエンドユーザーに通知するプロセスを開発する必要があります。つまり、このプロセスは、開発が終了し、アプリケーションが本稼働に移行されると、エンドユーザーに新機能が使用可能であることを通知するものです。プロセスは変更の規模によって異なります。簡単な新しいレポートであれば、通知は不要でしょう。常に多くの人が使用しているレポートの変更の場合は、すべての人に電子メールで通知するか、または変更のトレーニングセッションを設ける必要があります。

リリース・ステラテジー

同じドキュメントにおいて、複数の開発者による変更がいくつかある場合は、リリース・**ステラテジー**を開発する必要があります。リリース・**ステラテジー**を通じて、本稼働への移行数と、新機能を本稼働へ移行する追加

オーバーヘッドを最小限に抑えるよう、変更を調整します（QA、変更を統合するための管理時間、移行中のエラーの可能性など）。

開発者向け詳細

以下の詳細は、各開発者が新規の変更作業を行い、変更を本稼働環境へ追加する際の一般的な手順です。

開発者が本稼働用ドキュメントの変更の開発要求を受け取った場合の処理方法は、以下の 2 つです。

1. ドキュメントの多くの部分に影響を及ぼす変更が多数ある場合
   a. 開発者または専門家ユーザーは管理者に連絡し、開発者が最新版のドキュメントに独占的にアクセスできるよう求めます。
   b. 開発者は、以下の方法で変更作業を行います。
      ① 管理者から受け取ったドキュメントを各自の作業ディレクトリにコピーします。
      ② ドキュメントを変更します。
      ③ 新規ドキュメントに適切な名前を付けて、e:\QlikView Development Server Directory にコピーし（「Development Processes.doc」を参照）、QA ユーザーが変更を検証できるようにします。
      ④ QA(テスト担当）ユーザーが変更を検証後、ドキュメントを以下とともに管理者に返却します。
         i. 変更の概要リスト。（管理者は開発者からドキュメントを受け取り、一部の基準を検証し（詳細は、本ドキュメントの「管理者向け詳細」セクションを参照）、ドキュメントを本稼働に直接配置するため、すべてを網羅した変更のリストは不要です。）
         ii. ドキュメントが QA ディレクトリに置かれた後、QA 処理を行う QA ユーザーのリスト。管理者は、QA ユーザーの特定のリストに対し、ドキュメントへのアクセス権を付与します。
      ⑤ 管理者は、アプリケーションが本稼働環境で正しく動作し、適切な基準が満たされているかという基本的なチェック（スクリプトが壊れていないか、すべてのレポートが正しく動作するかなど）を実施後、ドキュメントを本稼働に配置します。

2. 変更がわずかである、または変更がドキュメントの特定の部分に限定されている場合
   a. 開発者は E:\QlikView Copy of Production からドキュメントのバージョンを各自の作業ディレクトリにコピーし、変更作業を行います。変更内容をすべてリストに保持しておくことが非常に重要です。
   b. 開発者はドキュメントに適切な名前を付けて、開発サーバー上のディレクトリ e:\QlikView Development Server Directory にコピーし（「Development Processes.doc」を参照）、QA ユーザーが変更を検証できるようにします。
   c. この時点で、QA ユーザーは変更を検証し、開発者は管理者に以下を通知します。
      ① ドキュメントの保存場所
      ② ドキュメント名（正確なファイル名）
      ③ 管理者が本稼働に統合する必要がある変更内容
      ④ ドキュメントが QA に置かれた後に、そのドキュメントを確認する必要があるユーザー

   d. 管理者は開発者による複数の変更を本稼働ビルドに統合します。そのビルドを開発者および QA ユーザーによる QA 用に E:\QlikView QA Server Directory に保存し、開発者および QA ユーザーにドキュメント名と QA の準備が整ったことを通知します（管理者が複数の開発者による変更を統合した可能性があるため、ファイル名は開発者が最初に命名したものとは異なる場合があります）。

e. 開発者（または開発者と QA ユーザー）がドキュメントを QA 処理すると、管理者はそのドキュメントを本稼働に移行します。

管理者向け詳細

管理者の役割には、本稼働環境を整った状態に保ち、新規変更を本稼働環境に統合することが含まれます。これらの変更には、スクリプト、レポート、QlikView Publisher のジョブへの変更が含まれます。この役割を遂行するためには、すべての QlikView 製品をしっかりと理解しておくことが必要です。全 QlikView 製品にわたるデータのフローを管理することになるためです。

すべてのタスクの一般原則

- ✓ 開発 – QA – 本稼働という順序で進行する開発プロセスで変更を本稼働に移行する場合は、伝統的な SDLC（System development life cycle)プラクティスが適用されます。QlikView には 2 つの異なるドキュメント間の差異を表示する機能はないため、変更を比較し本稼働へ移行する作業は手作業となります。

- ✓ 開発者による新規変更を統合する場合は、最新の変更がすべて保持されているドキュメントを把握し、確実に開発者の変更を最新のドキュメントに移行する必要があります。

- ✓ QA に同じドキュメントの複数のバージョンを同時に置かないようにする必要があります。QA に同じドキュメントの複数のバージョンが存在すると、変更を本稼働へ移行する際に、変更を再統合しなければならないためです。つまり、変更には、追加 QA が必要になります（変更が他の変更に影響を及ぼす可能性があるため）。

管理者のタスク

- ✓ 変更を本稼働に調整
- ✓ システム問題の修正
- ✓ 適切なソフトウェア設計原則を用いた作業（バックアップ、品質チェックなど）
- ✓ コーディング規約／プラクティスの採用
- ✓ 優れたシステムを促進する設計原則に関する開発者の指導
- ✓ QlikView Server の監視／調整
- ✓ セキュリティグループへの変更の監視
- ✓ User CAL の管理
- ✓ ナビゲーションに関する問題点の提起（ビジネス上のニーズがある場合は、却下される可能性があります）

管理者のタスクには、以下は含まれません。

- ✓ 数値／データの正確性の検証
- ✓ QlikView プロジェクトの管理
- ✓ 「外観」の良しあしの判断

新規レポートへの統合（レイアウト変更）

開発者が管理者に変更を通知する際は、以下を含める必要があります。

- ✓ 本稼働に移行する変更を含むドキュメントの場所と名前
- ✓ 変更のリスト。これらの変更には、以下が含まれます。
  - ➢ レイアウトオブジェクト（グラフ、レポート、リストボックスなど）
  - ➢ レポートが参照する変数
  - ➢ サイクリック・ドリルダウングループ
- ✓ ドキュメントの QA 担当者のリスト

このプロセスでは、管理者の作業に必要な、ドキュメントの本稼働候補バージョンと、管理者が開発者（開発者は複数の場合あり）から受け取った新たに拡張されたドキュメントが必要です。

- ✓ 管理者は変更リストに従って進め、変更を新しい本稼働候補に移行します。多くの場合は、新規オブジェクトを本稼働候補に単にコピー／貼り付けし、古いオブジェクトを削除するだけです。
- ✓ 新規ドキュメントに変更を移行したら、そのドキュメントが正しいことを検証する必要があります。
- ✓ すべてのフィールドを検査します。チャートやグラフでは、すべてのフィールドが本稼働候補に存在していることを確認する必要があります。通常、存在しないフィールドは赤で表示されるか、またはチャートプロパティでフィールの隣に赤い X が表示されます（チャートの場合）。
- ✓ すべての式を確認します（式タブ内だけでなく、すべての場所の式）。
- ✓ 有効であることを確認します。
- ✓ 効率的であることを確認します（式をさらに効率的にできる場合は、データモデルの変更が追加で必要になる場合があります）。
- ✓ サイクリック・ドリルダウングループ – これらはコピー／ペーストできないため、再構築する必要があります。グループ内のソート順も確認します。ソート順は開発者にとって重要な場合があります。
- ✓ Island Table、新規フィールドなど、その他の補助的なアーキテクチャ。

新規スクリプトの統合

- ✓ 新規スクリプトを特定し、スクリプトが採用しているコーディング規約に沿うことを検証します。
- ✓ 正しく実行されることを確認します。
- ✓ 新規スクリプトを既存のスクリプトに統合します。新規スクリプトをドキュメントに継続的に追加せずに、既存のスクリプトで新規スクリプトを統合して効率性を高めることも可能です。

QlikView Publisher への変更

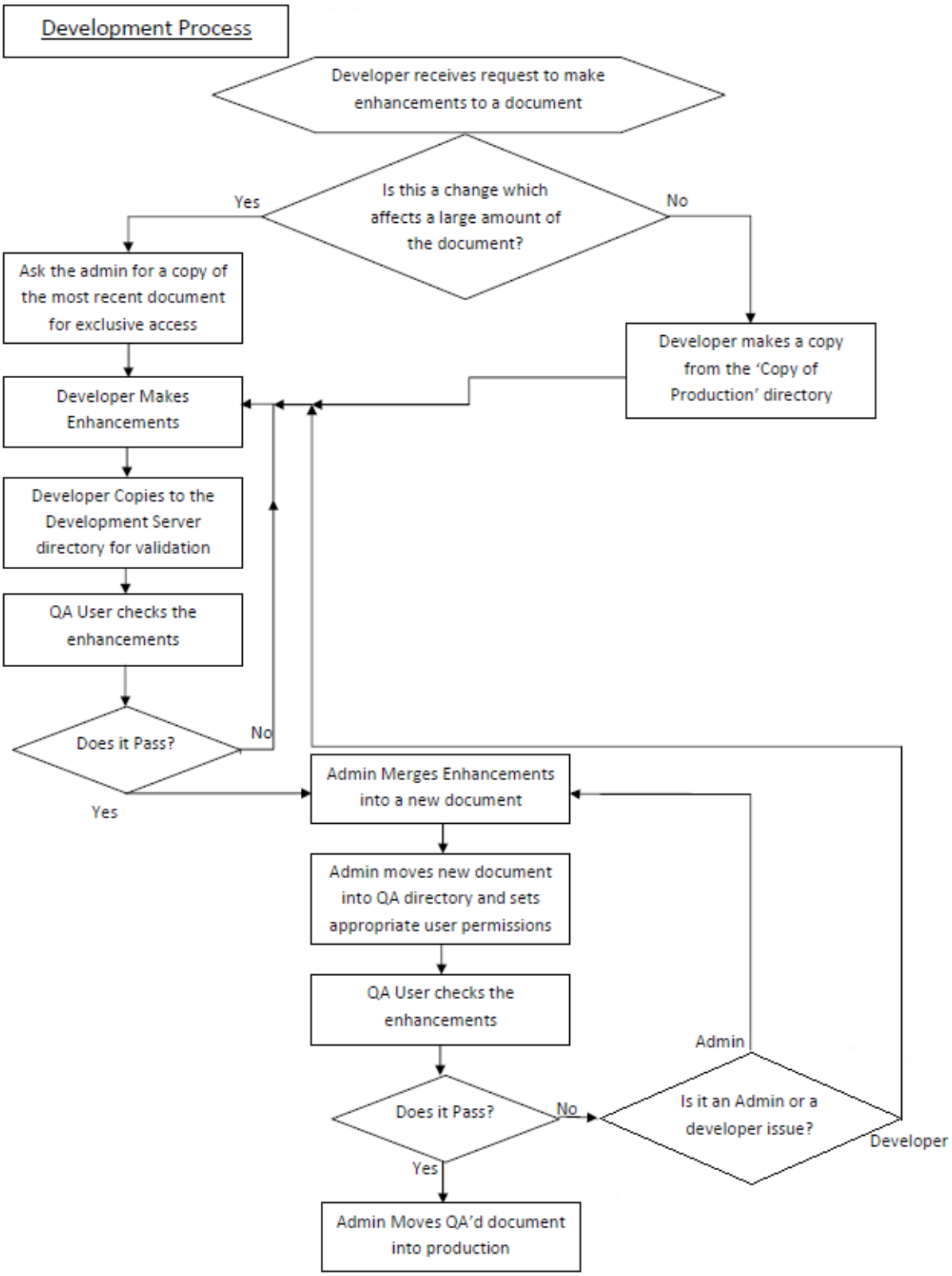
ユーザーは、本稼働へ移行する QlikView Publisher の項目を管理者へ提供しません。代わりに、QlikView Publisher の変更はその他の拡張要件に含まれています。

たとえば、新たなスケジュールでデータをリロードする新規要求が出されると、それは QlikView Publisher の変更として実行される必要があります。追加ドキュメントを使用した新しいスクリプトモジュールが追加されたら、追加ドキュメントを受け入れられるように QlikView Publisher を調整する必要があります。

何をどのタイミングで完了させる必要があるかについては、管理者の裁量で QlikView Publisher を変更します。順守するガイドラインはいくつかあります。

- ✓ ユーザーがシステムを使用中は、QlikView Server のリソースに大容量のドキュメントを「配信」しないでください。そうすると、QlikView Server のパフォーマンスは極端に落ちます。サイズの小さいドキュメント（所要時間が 30 秒以内）は問題ありません。
- ✓ 新規データモデルの変更（スクリプトの変更など）を検証する場合は、QlikView Publisher のジョブを開発し、QlikView Publisher からそのジョブを実行してデータ移行が正しく行われていることを検証するのが最善策です。このようにしないと、QlikView Publisher に適切な権限があり、実際の本稼働環境でジョブを実行するように定義されているかどうかが分かりません。

開発プロセス


Development Process
Developer receives request to make enhancements to a document
Is this a change which affects a large amount of the document?
Yes
No
Ask the admin for a copy of the most recent document for exclusive access
Developer makes a copy from the 'Copy of Production' directory
Developer Makes Enhancements
Developer Copies to the Development Server directory for validation
QA User checks the enhancements
Does it Pass?
No
Yes
Admin Merges Enhancements into a new document
Admin moves new document into QA directory and sets appropriate user permissions
QA User checks the enhancements
Admin
Does it Pass?
No
Is it an Admin or a developer issue?
Developer
Yes
Admin Moves QA'd document into production

命名規則

QlikView Publisher

- ✓ 一般的に、タスク名にアクションの時刻(「Daily」、「Nightly」など)を挿入することは勧められていません。これは、タスクはさまざまなジョブで、別々にスケジュールできるためです。
- ✓ 一般的に、ジョブ名にアクションの時刻を挿入することが勧められています。
- ✓ QA 環境と開発環境が混在している場合は、すべての名前(個々の.qvw を除く)に Dev_または QA_のプレフィックスを付ける必要があります。本稼働の場合は、Prod_のプレフィックスを付けるか、またはプレフィックスを付けません(全員が同じ規則に従う限り、どちらを選択するかは任意です)。
- ✓ 本稼働に移行されたアプリケーションには常にログファイルを使用します。ログファイルが存在しなければ、QlikView Publisher はアプリケーションの実行が成功／失敗したかについての詳細を取得できません。
- ✓ UNC 名を使用します。使用しない場合、自動タスクがパスを参照できない可能性があります。

略語	略語タイプ	意味
All	環境	すべての環境に適用
DEV	環境	開発環境
PRD	環境	本稼働環境
TST	環境	テスト環境
APR	Publisher 項目	Publisher Access Point リソース
JOB	Publisher 項目	Publisher ジョブ
SDF	Publisher 項目	Publisher ソースドキュメントフォルダリソース
TSK	Publisher 項目	Publisher タスク
DSR	Publisher 項目	ディレクトリサービスリソース

スクリプト作成およびレイアウト

命名規則の策定

- ✓ データフィールドにビジネス名を使用します。例:CustNo ではなく Customer Nbr
- ✓ すべての略語は標準タイプです。略語のリストを入手して使用します(つまり、略語リストで指定されているように、Description には常に Desc を使用します)。
- ✓ プレフィックスを利用
  - ➢ 変数 =「v」で開始 例: vCurrentYear
  - ➢ キー項目 =「%」で開始 例: %CustomerKey
  - ➢ フラグ項目 =「_」で開始 例: _YTDFlag
  - ➢ サイクリックグループ =「<」で開始 例: <ProductCycle
  - ➢ ドリルダウングループ =「>」で開始 例: >GeographyDrilldown
  - ➢ キー項目区切り文字 =「_」で区切る 例: キー項目[Company&'_'&Nbr]
  - ➢ 一時フィールド／テーブル =「_tmp」で終了 例: Daily_Trans_tmp

フォルダ構造

概要

本稼働用の QlikView ファイルを格納するフォルダも重要です。フォルダの構造によって、開発者はアクセス権のあるファイルを簡単に配置したり読み込むことができるようにする必要があります。以下に、QlikView ファイルの格納場所に関する方針の一部をご紹介します。

QlikView Folder Breakdown Options

QVW= .qvw files only
QVD= .qvd files only
Data= .xls, .txt, .csv, .mdb, etc.... Used for data loading static data sets
Config= .txt, .ini, etc... used for control of QVW or inputs
** NOTE: each folder below contains each of the 4 component folders*

Deptartment Breakdown	Project Breakdown	Application Breakdown	Mixed Breakdown
Dev	Dev	Dev	Dev
Sales	Sales Dashboards Project	Corporate Dashboard	Sales
QVW	QVW	QVW	Sales Analysis App
QVD	QVD	QVD	QVW
Data	Data	Data	QVD
Config	Config	Config	Data
Finance	Scorecard Rewrite	Sales Analysis App	Config
Marketing	Shared	Supply Chain Analysis	Finance
Shared	QVW	Timesheet Trends	Marketing
Test	QVD	Shared	Shared
Sales	Data	Test	Test
Finance	Config	Corporate Dashboard	Sales
Marketing	Test	Supply Chain Analysis	Finance
Shared	Sales Dashboards Project	Shared	P&L Dashboard
QVW	Shared	Prod	QVW
QVD	Prod	Corporate Dashboard	QVD
Data	Shared	Sales Analysis App	Data
Config		Supply Chain Analysis	Config
Prod		Timesheet Trends	Marketing
Sales		Shared	Shared
Finance			Prod
Marketing			Sales
Shared			Finance
			Marketing
			Shared

部門単位の分割

この方針は、対象領域全体で使用される可能性のあるすべてのファイルを格納する共有フォルダを含め、環境フォルダの下の特定の対象領域のフォルダにファイルを分けるというものです。これは、常に多くの新規開発プロジェクトが進行中ではない、確立した導入に適した戦略です。この戦略は、プロジェクト固有のフォルダを持たないため、多くの新規開発プロジェクトが進行中である場合は好ましくありません。つまり、本稼働で使用するかどうかがはっきりしない新規の開発ファイルは、確立した本稼働ファイルとともに開発／テスト／QA フォルダに格納されます。

プロジェクト単位の分割

この方針は、複数の開発プロジェクトが同時に進行中の場合によく使用されます。プロジェクトの名前が付いたフォルダ内にプロジェクト固有のファイルが格納されます。こうすることで、開発者と管理者はプロジェクト関連の新規ファイルを素早く識別し、テストおよびコード移行用に区分けできます。ファイルが本稼働に移行されると、本稼働には開発プロジェクトはないため、すべてのファイルは「共有」フォルダに格納されます。

アプリケーション単位の分割

この戦略は、同じファイルを相互にそれほど使用しない大規模の QlikView アプリケーションが存在する場合によく使用されます。アプリケーション別にファイルを分けると、拡張またはアプリケーションの追加に必要なファイルを簡単に識別できます。複数のアプリケーションで共通して使用するファイルすべてを格納する共有フォルダもあります。共通して使用するファイルが時間とともに増加する場合は、このアプローチはあまり望ましいものではありません。

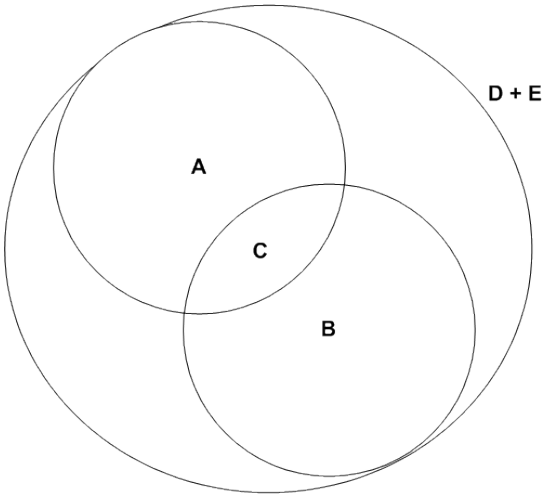
混合方式での分割

この戦略は、部門分割をプロジェクト分割またはアプリケーション分割のいずれかと組み合わせます。上図は、デモ用に部門／アプリケーションの組み合わせを示したものです。このアプローチは、複数の部門が相互にそれほど重複しないアプリケーションを開発している大規模な QlikView 導入によく使用されます。

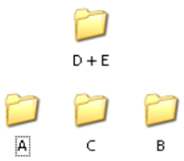
上記のいずれのアプローチ、またはアプローチを組み合わせて使用できますが、整合性がフォルダ作成方針の鍵となることに留意することが重要です。もちろん、QlikView 導入の進行に併せて方針を切り替えることもできますが、コード管理および導入に対する制御とガバナンスを維持するためには、採用中の方針*内*の整合性と慎重性を保つようにします。

フォルダのセキュリティ

この例は、QlikView のソースファイルの安全性を確保する方法を示したものです。その方法は多数ありますが、QlikTech はベストプラクティスとして、各自の開発方針と一致するセキュリティ方針を採用することを推奨します。そうすることで、開発ファイルにアクセスするべきでないユーザーからそれらのファイルを切り離すことができると同時に、共有ファイルにはアクセスできるようにすることで、再利用および整合性を実現できます。



ソースデータのファイル構造が一致するように、異なるグループを使用します。
A) 部門 1／開発グループ 1
B) 部門 2／開発グループ 2
C) 共有会社データ（Shared_Folders）
D) QlikView 管理者はすべてのグループと配布済みドキュメントへのアクセス権あり
E) QlikView サービスのサービスアカウントはグループ D のメンバー
このサービスには、データベース、ファイルシステム、Active Directory の読み込みアクセス権が必要



テストおよび承認

実行するテスト

1. リロードテスト
   ① ローカルでリロード
   ② QVS からのリロード（手動）
   ③ 遠隔の QlikView Publisher タスクからの自動リロード

2. QlikView Publisher のテスト
   ① タスクのリロード、配布、ステータスを個別にテスト
   ② 必要に応じて依存関係を構築し、個別にテスト

3. ユーザーテスト（UI）
   ① 個別ユーザーテスト
   ② プライベートブックマーク
   ③ エクスポート（全フォーマット）
   ④ 同時ユーザーテスト（非構造化）
   ⑤ 同時ユーザーテスト（複合処理での機能テスト）
   ⑥ 整合性・結合テスト（複数のアプリケーション）

4. ユーザーテスト（コラボレーション）
   ① 新規コラボレーションオブジェクトのテスト
   ② 共有コラボレーションオブジェクトのテスト
   ③ サーバーブックマーク

5. パフォーマンステスト
   ① QVW 上で 10 人以上のユーザーによる同時アクセステスト

6. リグレッションテスト
   ① 追加変更の QVW をテストする場合は、QVW 初回導入時のテストケースを使用して、回帰テストを実施します。

7. サイクルテスト
   4～6 の完全なサイクルリロードを毎日実行し、QVW をソースからユーザーインターフェースに更新します。

環境

a. 単一環境 – ローカルテスト
このアプローチはクライアントのサーバーが 1 台（PROD）のみであり、コードが本稼働に移行できるようになるまで、開発者のマシン上でローカルにコードをテストする必要がある場合に使用します。このテストは、開発者のマシン上でローカルユニットテストと一緒に実施できます。テストでは、使用が限定されたバージョンの QVW を PROD に配置して、受け入れテストを実施します。QVW のそのバージョンがテストに合格すると、PROD バージョンとして使用できます。

b. 2 つのサーバー環境
このアプローチでは、すべてのテストが開発／テストサーバー上で実施され、テストに合格すると、コードは PROD に移行されます。ユニットテストは開発者がローカルで実施できますが、正式な受け入れテストは、開発者がユニットテストを完了後、開発／テストサーバー上で実施する必要があります。

c. 3 つのサーバー環境

このアプローチは、DEV、TEST および PROD が使用されること以外、基本的には上記のアプローチと同じものです。DEV はまずユニットテストに使用されます。次に、TEST が受け入れテストに使用され、テストに合格すると、QVW が PROD に移行されます。

エンドユーザーテスト

a. 個別 – 非構造的
これらのテストは、構造化計画や何をどのようにテストするかの指示がなく、エンドユーザーが自由な形式でテストを実施します。各エンドユーザーには v9 のテスト環境へのアクセス権が付与されるだけで、好きなようにテストするよう指示されます。

b. 個別 – 構造的
これらのテストでは、何をどのようにテストするかが指示されますが、テストを実施する時期、およびユーザーグループ内でテストを調整する時期については指示されません。どの QVW をテストし、どの機能を特定の順序でテストするのかが指示されることもあります。結果は集計され、テストを完了したユーザー間で比較されます。

c. グループ – 非構造的
エンドユーザーのグループ全員に v9 テスト環境へのアクセス権が付与され、一定期間、同時にテストを実施します。テストは n 時間または数日間にわたって実施されるか、または連続して 1 つのテストウィンドウを使用します。エンドユーザーは、何をどのようにテストするかはっきり指示はされず、すべてをテストするように指示されます。

d. グループ – 構造的
エンドユーザーのグループ全員に v9 テスト環境へのアクセス権が付与され、一定期間、同時にテストを実施します。テストは n 時間または数日間にわたって実施されるか、または連続して 1 つのテストウィンドウを使用します。エンドユーザーは、何をどのようにテストするか、具体的に指示されます。お互いに連携することで（できるだけ同じ部屋または電話会議で）、同時に機能を開始、もしくはドキュメントにアクセスする並行処理テストを実施できます。結果は集計され、対照のために 8.5 環境で実施された類似のテストと比較されます。

テストの最適化

テストのためにエンドユーザーの時間と労力を確保するのは難しいことです。これらのアプローチのいくつかは、テストを推進し、確実に良い結果を得る上で役立つものです。

1. 事前にユーザーのテストウィンドウを調整します。環境を監視できる「テストウィンドウ」を用意し、質問や要望に対するサポートを行うことをエンドユーザーに知らせます。これにより、エンドユーザーの時間は浪費されず、ヘルプや指示が必要なときの待ち時間は発生しません。

2. 新規バージョンのドキュメントに最大のバグを発見したエンドユーザーまたはエンドユーザーグループに賞を与えます。これにより、エンドユーザーのモチベーションが高まり、QVW のストレステストを徹底的に行うことにもつながります。

3. テストのために、ユーザーを同じ場所に集めます。大きい会議室を探し、エンドユーザーにその会議室でテストを実施してもらいます。ユーザーがラップトップを持ち込めない場合は、トレーニング用ラボが最適です。

4. テスト中は電話や Skype でユーザーを連携させます。テストウィンドウ中は、エンドユーザーが相互に会話や質問ができ、同時テストの指示を受けられるように、会議回線をつないでおきます。

5. ユーザーにテストするドキュメントまたは機能のチェックリストを渡します。ユーザーは、QVW で使用できる機能すべてを覚えることができない場合があります。ブックマーク、レポート、エクスポート、条件付きで見えるオブジェクト、コラボレーションなどを含むチェックリストをユーザーに渡します。

6. 事前にユーザーのテストウィンドウを調整します。環境を監視できる「テストウィンドウ」を用意し、質問や要望に対するサポートを行うことをエンドユーザーに知らせます。これにより、エンドユーザーの時間は浪費されず、ヘルプや指示が必要なときの待ち時間は発生しません。

7. 新規バージョンのドキュメントに最大のバグを発見したエンドユーザーまたはエンドユーザーグループを表彰します。これによりエンドユーザーのモチベーションが高まり、QVW のストレステストを徹底的に行うことにもつながります。

8. テストのために、ユーザーを同じ場所に集めます。大きい会議室を探し、エンドユーザーにその会議室でテストを実施してもらいます。ユーザーがラップトップを持ち込めない場合は、トレーニング用ラボが最適です。

9. テスト中は電話でユーザーを連携させます。テストウィンドウ中は、エンドユーザーが相互に会話や質問ができ、同時テストの指示を受けられるように、電話会議回線をつないでおきます。

10. ユーザーにテストするドキュメントまたは機能のチェックリストを渡します。ユーザーは、QVW で使用できる機能すべてを覚えることができない場合があります。ブックマーク、レポート、エクスポート、条件付きで見えるオブジェクト、コラボレーションなどを含むチェックリストをユーザーに渡します。

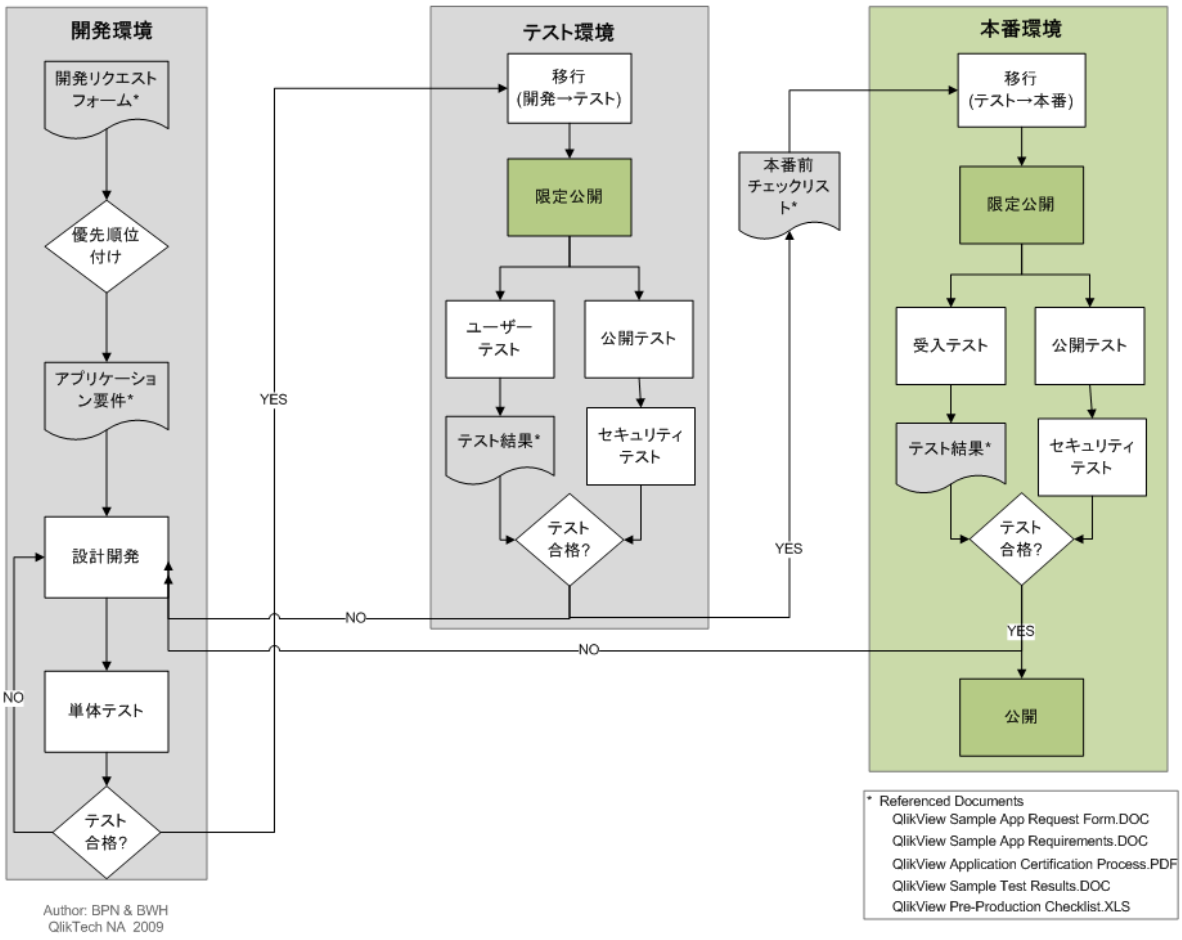
テストする方法は多数あります。効果的なテストを成功させる 2 つの最も重要な要因は以下の通りです。

1. 計画
   テスト項目を分割し、個別（開発、リロード、公開、セキュリティ、使用方法、監視、トラブルシューティングなど）にテストを実施できるようにテストを計画します。次に、これらの責務を結合し、可能な場合はサイクルテストを実施します。テスト計画を文書化し、繰り返せる手順を繰り返します。

2. テスト時間の確保
   QlikView の開発者、管理者、エンドユーザーに、テストのための時間を確保することを約束してもらいます。共通のベンチマークでは、テストには QVW の開発時間の少なくとも 10%に相当する時間を充てるべきであるとしています。つまり、QVW の開発に 160 時間を要した場合は、本稼働に移行する前に、16 時間のテストを実施する必要があります。これは、16 ユーザーが 1 時間ずつ、または任意のユーザー数と時間の組み合わせで合計 16 時間（開発時間の 10%）にすることもできます。エンドユーザーに構造化された時間と構造化されていない時間を与えてその時間数をカウントします。また、エンドユーザーに問題点を明らかにするモチベーションがあることを確認します。

アプリケーション開発ワークフロー

QlikTech は、QlikView 環境全体におけるコード移行に関する手順を図式化した開発ワークフロードキュメントの作成を推奨しています。下図は、開発ワークフローの例を示したものです。

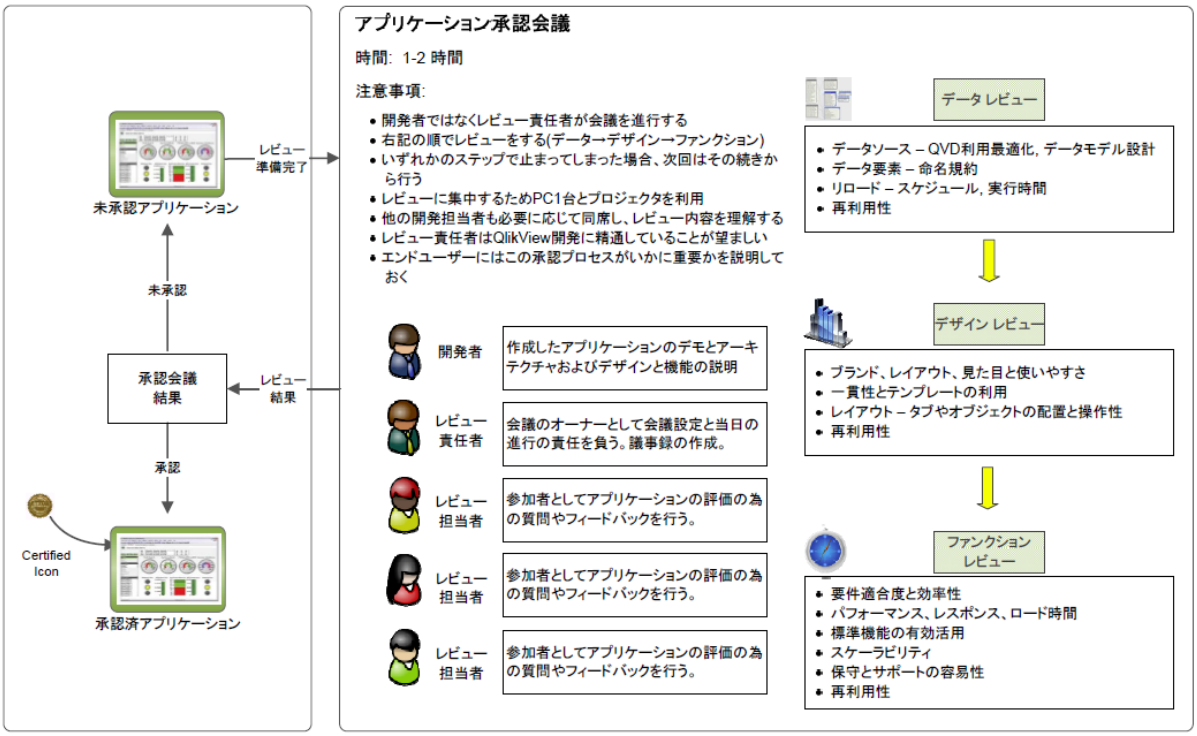


承認

多くの企業は、QlikView 開発のスピードと柔軟性からメリットを得ていますが、非常に重要な、または高品質の QlikView アプリケーションの開発において、より厳格なプロセスを保持したいと考えています。「ハイスピードな開発」のアプリケーションと「高い品質」のアプリケーションを区別するために、多くの QlikView クライアントは承認プロセスを使用しています。

このプロセスは、QlikView アプリケーションの「承認」を受けるためのチェックポイントの役割を果たします。承認とは、アプリケーションがこのプロセスを経て、承認（受入）されたことを意味します。承認を受けると、「承認済み」アイコンがアプリケーションのタイトルセクションに付けられます。ユーザーとサポートチームは、どのアプリケーションが承認され、どのアプリケーションが承認されていないかが分かります。チームは承認されていないアプリケーションに同レベルのサポートを提供しないことで、このプロセスを強調できます。

下図は、一例として、承認のためのミーティングがどのようなものかを示したものです。



トラブルシューティングおよびサポート

サポートタイプ

QlikView のアプリケーションおよび環境のサポートは、さまざまな方法で行うことができます。QlikTech は、ベストプラクティスとして、サポートレベルおよびサービスを以下の分野に分類することを推奨しています。

- ✓ QlikView アプリケーション（QVW）
- ✓ QlikView インターフェース（エンドユーザーサポート）
- ✓ QlikView Server／Publisher
- ✓ QlikView データアーキテクチャ（QVD や QVW）

QlikView クライアントの多くは、重要度の高いアプリケーションのサポートには承認された QVW を使用します。これは、ビジネスチームが独自の QVW を作成中の場合に特に有効です。サポートチームは、コード／インターフェース／データの見直しを行ったことがある承認アプリケーションのサポートのみを担当します。承認プロセスの詳細については、本ドキュメントの「テストおよび承認」セクションをご参照ください。

QlikView 開発チーム

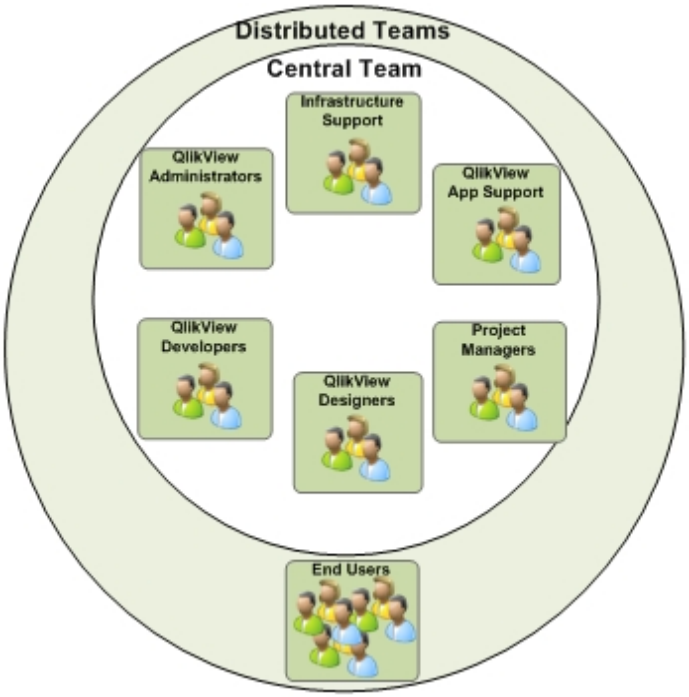
QlikView は非常に柔軟性があり、容易に採用できる BI ツールです。従って、開発チームはサポート、管理、開発、トレーニング、および管理用にさまざまなモデルを作成できます。ここで述べるシナリオを通して、開発環境における可能な QlikView の役割設定を説明します。

クライアントには、開発に対する独自の IT 標準を参照することが推奨されます。この意思決定を後押しできる、または少なくとも選択肢を狭めることができるためです。QlikTech は、これらのシナリオの中の 1 つを明示的に奨励してはいませんが、クライアントには、開発の本質と有するスキルセット考慮した上で、これらの設定のうちのどれが最適であるかを各自で判断することを要望します。

QlikView のチーム構成には、集中管理型から完全分散型まで 5 つのオプションがあります。

集中管理型

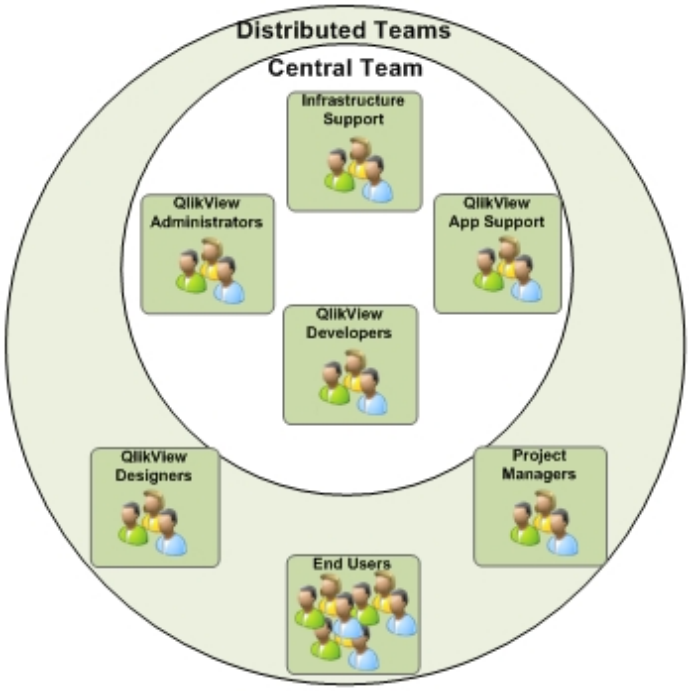
セントラル（IT）・チーム	分散（業務）チーム
インフラサポート	エンドユーザー
QlikView 管理者	
QlikView アプリケーションサポート	
QlikView 開発者	
QlikView デザイナー	
プロジェクトマネージャー	



このオプションでは、各部門は QlikView アプリケーションを使用する開発者、サポート担当者、または管理者を提供する必要はありません。各部門は新しいアプリケーションを要求し、中央の QlikView サービスとともに利用します。このアプローチの長所は、すべてのサービスが 1 つのチームによって提供されることによる統制、スキルセットの共有、整合性、およびガバナンスです。

共同開発（v1）

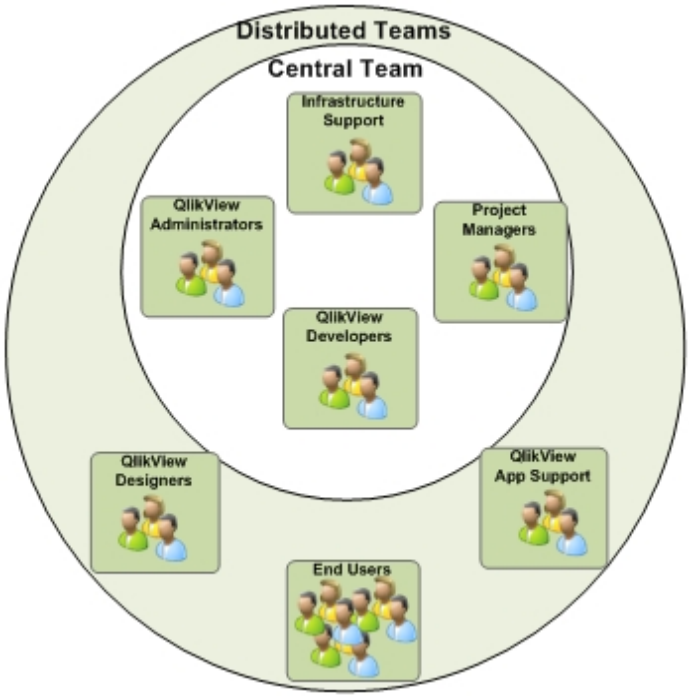
セントラル（IT）・チーム	分散（業務）チーム
インフラサポート	エンドユーザー
QlikView 管理者	QlikView デザイナー
QlikView アプリケーションサポート	プロジェクトマネージャー
QlikView 開発者	



このオプションでは、エンタープライズ開発は中央機能として保持され、スクリプト作成およびデータモデリングは、熟練した QlikView 開発者とデータ専門家が対応します。各部門は、すべてのトレーニング、プロジェクト管理、アプリケーション設計、テスト、サポートを担当します。
このアプローチの長所は、バックオフィスの BI 業務は集中管理されていますが、設計およびプロジェクト管理は業務チームが行うという点です。そのため業務チームは、部分的に IT リソースに依存せずに、迅速に業務を遂行できます。

共同開発（v2）

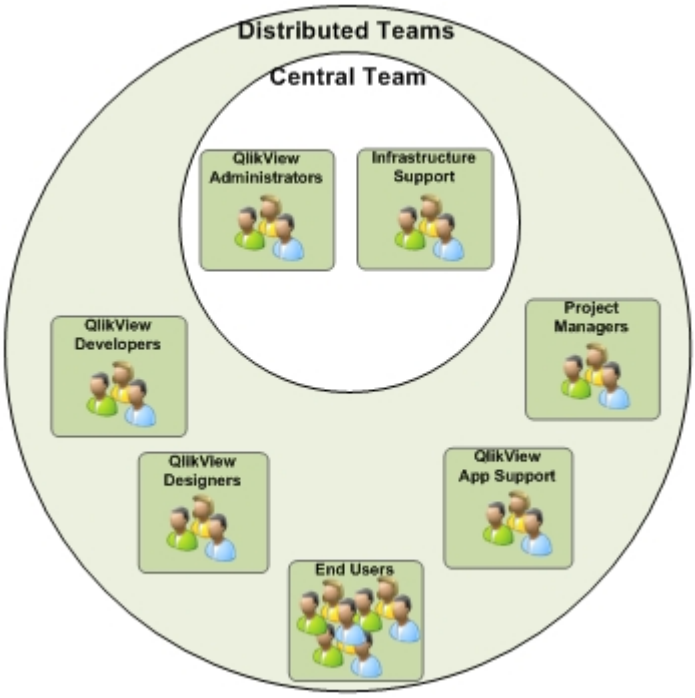
セントラル（IT）・チーム	分散（業務）チーム
インフラサポート	エンドユーザー
QlikView 管理者	QlikView アプリケーションサポート
QlikView 開発者	QlikView デザイナー
	プロジェクトマネージャー



このオプションでは、サポートは各部門に移行されていますが、プロジェクト管理は、QlikView の専門知識や設計管理を生かせるよう、セントラル（IT）・チームが担当します。すべてのトレーニング、アプリケーション設計、サポートは各部門が担当します。このアプローチの長所は、QlikView アプリケーションサポートも業務チームに分散されていること以外は、v1 共同開発モデルと類似しています。このアプローチにより、IT リソースはさらに解放され、エンドユーザーへのサポートやトレーニングなど、タスクの一部が分散（業務）チームに委ねられます。

ほぼ分散型

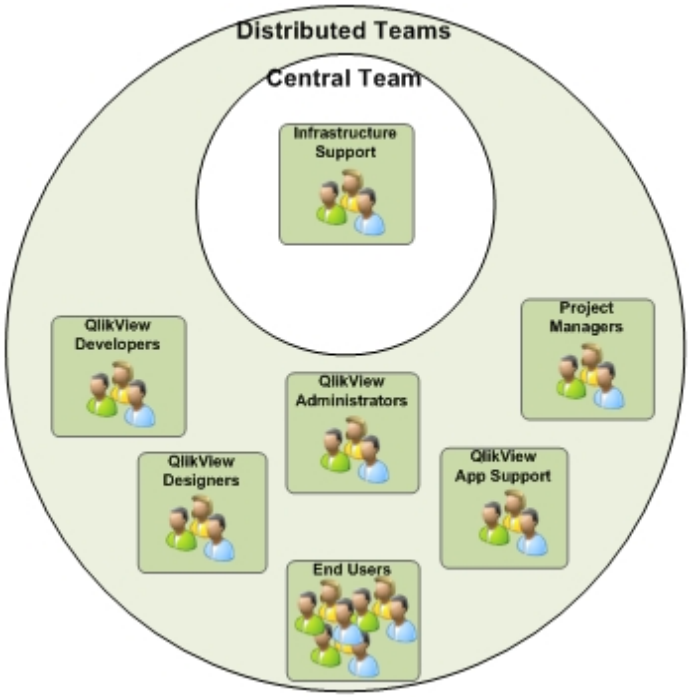
セントラル(IT)・チーム	分散(業務)チーム
インフラサポート	エンドユーザー
QlikView 管理者	QlikView アプリケーションサポート
	QlikView 開発者
	QlikView デザイナー
	プロジェクトマネージャー



このオプションでは、各部門がすべてのトレーニング、プロジェクト管理、アプリケーション設計、スクリプト作成、開発、テスト、サポートを担当します。セントラル(IT)・チームはインフラサポートと QlikView 管理を担当します。こうすることで、小規模の IT チーム(通常、2 名未満)が権限、サーバー設定、バッチ処理(リロード)を管理できます。このアプローチの長所は、中央(IT)チームが非常に小規模なことです。

完全分散型

セントラル(IT)・チーム	分散(業務)チーム
インフラサポート	エンドユーザー
	QlikView 管理者
	QlikView アプリケーションサポート
	QlikView 開発者
	QlikView デザイナー
	プロジェクトマネージャー



このオプションでは、インフラは引き続き中央で管理されますが、QlikView のその他の開発、テスト、サポート、使用方法のトレーニングに関しては、すべて各部門に分散されます。そのため、分散(業務)チームは QlikView の技術をすべて身に付ける必要があります。このオプションの長所は、セントラル(IT)・チームにソフトウェア固有のリソースが不要であるという点です。しかしながら、このアプローチの難点は、複数の分散(業務)チームにソフトウェア固有のリソースがある必要があり、それらは重複する可能性もあることです。

開発チーム構成の選択は、QlikView のエンタープライズ導入においては重要な手順です。現在は共同開発オプション(2 および 3)が最も一般的ですが、ニーズと長所が一致したオプションが、各クライアントに適したオプションです。

QlikView Centers of Excellence

QlikTech は、QlikView のエンタープライズ導入を実装するクライアントに対し、QlikView Centers of Excellence（またはコンピテンシーセンター）の設立を真剣に検討することを推奨しています。この拠点は必要に応じて公式または非公式なものにできますが、Centers of Excellence でサービスとシナジーを共有することで、エンタープライズ導入において大幅なコスト削減実現し、対象範囲が広範となります。

QlikView Centers of Excellence（CoE）

IT とビジネスユーザーの混合チームによって統合された QVCoE は、組織の知識とベストプラクティスを集約し、組織のビジネスインテリジェンスソリューションを標準化、実装、管理します。

目的:

1. QlikView のベストプラクティスの開発および共有
2. QlikView アプリケーションの機能と操作の確保
3. 共通 BI アーキテクチャの設計および実装
4. 組織全体での BI の使用を促進

組織構造

- ✓ 会社の文化、関与するチームのスキルセット、方針に応じて集中型または分散型
- ✓ 専任の組織またはバーチャルな組織
- ✓ 「大きく考える」必要はあるが小規模でスタート
- ✓ QlikView および会社の成長を考慮した計画

利点

- ✓ 企業情報の信用性と信頼性の向上
- ✓ 全社的な BI の使用開始
- ✓ 意思決定プロセスの加速化
- ✓ リソースの最適化とコスト削減
- ✓ BI の洞察力によるビジネスプロセスの革新
- ✓ 変化するビジネス要件をサポートする QlikView BI アプリケーションの継続的な進化

ステップ:

1. エグゼクティブスポンサー
2. COE の使命および目的
3. 資金調達
4. 組織構造／報告網
5. 機能領域
6. 必須の役割
7. COE の KPI

QlikTech は、QlikViewCenters of Excellence を設立するための集中サービス、KPI、構造、およびベストプラクティスを定義するのに役立つ資料とテンプレートを提供できます。これは、QlikView の導入に相応するサイズと範囲にする必要があります。つまり、小規模で開始し、QlikView 導入が拡大するのに併せて拡張できます。

トレーニング

効果的なトレーニングは、すべての QlikView アプリケーションの作成、継続的なアップグレード、または導入に不可欠です。一人のマネージャー用に小規模な販売管理ダッシュボードを導入する場合、または数千ユーザー向けのアプリケーションを世界中に展開する場合にかかわらず、トレーニングはプロジェクトの成功と、業務を効率的に遂行するためのシステムの利用方法に大きく影響します。

QlikView エデュケーションサービスでは、様々な形式でトレーニングサービスを提供しており、受講者は要望に応じたトレーニング形式をお選びいただけます。

トレーニングコースに関する詳細情報および受講申し込みについては、下記 QlikView トレーニングページをご覧ください。

http://www.qlikview.com/jp/services/training

まとめ

QlikView は非常に高速で柔軟性のある BI ソリューションです。開発で管理する流動的部分（QVD および QVW）はわずかですが、適用できるリソース、プロセス、環境、用途、配信プラットフォーム、設計などの組み合わせは無限にあります。ガバナンスと整合性に対するニーズは急速に高まっており、QlikView も例外ではありません。

これらのベストプラクティスは完全に包括的なものではありません。最新のベストプラクティスに精通するための良い方法は、本書をはじめとするドキュメントや他の情報の利用を増やすことです。以下に、その一部を紹介します。

- ✓ QlikCommunity – 10,000 人のユーザーが利用している QlikView のオンラインコミュニティ。
- ✓ QlikView Japan グループ – QlikCommunity 内のグループ。主に日本語での QlikView に関する製品情報、ノウハウを共有するためのグループです。グループの内容を閲覧するには登録が必要です。
- ✓ QlikTech Expert Service – この一連のサービスは、開発プロジェクトにおけるベストプラクティス、アーキテクチャ、管理によって迅速なスピードが得られるよう、プロジェクトメンバーを支援します。利用できる QlikTech サービスの詳細、および世界各国の QlikTech 認定パートナーについては、最寄りの QlikTech オフィスにお問い合わせください。

QlikTech は、クライアントが QlikView 導入を最適化し、導入までの時間を短縮するベストプラクティスと技術を習得することを強くお勧めします。これらのベストプラクティス他の実装を導入に組み入れることをご検討ください。QlikView の成功をさらに促進する素晴らしい方法です。

以上