**Building a Dynamic Views Solution**

Introduction

The dynamic view approach expands the potential use cases for Business Discovery, enabling business users to conduct associative analysis on larger data sources and allows customer choice in when to push down compute to source and when to use the in-memory engine.

This workshop is designed to deliver an introduction into the steps required to build a dynamic view solution in Qlik Sense using the built-in navigation options. The source data itself is a cut down version of sales and inventory data stored as qvd files, the process will be the same for a real-world dynamic view deployment.

## Scenario

Company X has a series of requirements they wish to deliver with Qlik Sense, they want to allow a section of users to access all their data without consuming all the data into a Qlik Sense app. The following metrics are required to help understand the product sales mix and relevant stock level details of the products Company X sells:

* Product sales over time.
* Product analysis with sales amounts and order quantities
* Customer demographic analysis by occupation and gender
* A detailed list of sales orders with associated attributes
* A detailed list of stock movement data by product and product sub categories

## Goals and Content

The workshop will cover the scripts required to create the hybrid application which will be an aggregated view of the total dataset with associated dimensions as well as the dynamic views which can be refreshed on an ad-hoc basis.

Within the DynamicViews\_Course.zip file you will find the following assets:

Source data qvd files

* InternetSales\_Fact.qvd
* StockMovementDetails.qvd
* Customers.qvd
* Geography.qvd
* ProdSubCategory.qvd
* Products.qvd

Source Qlik Sense application which will be used as the template for the detail app dynamic view script

* DetailsTemplate.qvf .

## Outcome

The result of the workshop will be three applications, two of which will act as the detail apps which form the basis of the dynamic views. Below are diagrams which show the process flow:



1. User logs into the Hybrid App containing two dynamic views, source data containing dimensional data qvd files and an aggregated view of the InternetSales\_Fact qvd. Selections are made to hit the governed limit based on distinct ProductNames.
2. Once the governed limit is reached the dynamic views can be refreshed.
3. The ProductKey associated values are passed into the dynamic view which is refreshed and applies a WHERE clause to filter the data.

  
Details of the dynamic view process below:

1. Data items are selected from charts/filter boxes in the aggregated hybrid app containing in memory analytics and dynamic views.
2. Once a governed limit is reached, based on a data limit - count(distinct ProductName) the dynamic views are available for refresh.
3. The selections have generated a filtered list of associated ProductKey’s and these are dynamically passed into one of the detail apps which contain charts and the reload is initiated.
4. The detail dynamic view source app contains SET statements which indicate which fields are used for filtering (ProductKey) and creates an INLINE LOAD statement dynamically with these values.
5. The detail dynamic view source app contains script to convert these into a WHERE clause variable and the app is reloaded with the where clause and published to a stream with an optional timeout.

Sample chart outcomes are shown below:

Sales and Inventory Hybrid:



Sales Details:



Stock Movement Detail Sheet:



Exercise 1

1. In this first short exercise, we will build enable the dynamic views by activating it in the management console.
2. Open the Qlik Sense Management console.
3. Navigate to the Configuration\settings tab and enable dynamic views:   
     
   

We have now completed the first short exercise.

Exercise 2

In this exercise, we will build a simple aggregated Qlik Sense application based on the dimensional qvd files and an aggregated view of the **InternetSalesFact qvd** file and produce a few simple visuals. This will also act as the initial hybrid application in the dynamic view process.

## Create a New Qlik Sense Application

1. Unzip the **ODAG\_Course.zip** file to a location we will use to create a folder connection later.
2. Open the Qlik Sense Hub and click *Create new app* and provide a name such as Sales and Inventory Hybrid.
3. Go into the *Data Load Editor* and create a new folder connection to the location where we unzipped the file from step 1, for example:  
     
     
   \*
4. Create the load script for the dimensional qvd files (Customers.qvd, Geography.qvd, ProdSubCategory.qvd, Products.qvd.

## Create the summary load script and charts

1. To create a summary view of the fact data we will use the **InternetSalesFact.qvd** file and only use the following fields:

* ProductKey
* CustomerKey
* OrderQuantity
* DiscountAmount
* SalesAmount
* TaxAmt
* Freight
* OrderDate
* ShipDate

The measure columns will be aggregated using a sum() function and the date fields will be transformed to only go down to Year, Month granularity.

1. Create the aggregated load script which should look something like this:  
     
   
2. Load the data into the application.
3. Create a new sheet and add three filter boxes for ProductSubCategoryName, Color and CountryRegionName
4. Create three measures in the Master Items:  
   # Products - count(distinct ProductName), # Customers - count(distinct EmailAddress) and  
   Avg Sales Amount - avg(SalesAmount)
5. As the # Products measure will be used in the dynamic view expression later add conditional colors to further indicate to the user that they have reached the required number of products to allow the dynamic views to be refreshed:



The KPI’s should look as follows:



1. Create two new measures:  
   Sales Amt - sum(SalesAmount) and Order Qty - sum(OrderQuantity)
2. Create an area line chart using Order Year, Order Month as dimensions and Sales Amt as measure with Order Qty as the alternative measure:



We have now completed the exercise.

# Exercise 4

In this exercise, we will build the dynamic view details app to allow users at Customer X to access the lowest level of detail in their dataset namely the sales orders and lines. We will use the template app provided and copy the script to a new app and modify some variables/build some sample charts, finally we will create the dynamic view link in the hybrid app.

1. Upload the template app (**DetailsTemplate.qvf)** into your content space in the hub.
2. Create a new application called **OrderDetails.qvf**.
3. Open the *Script Editor* and add a new section called Dynamic Data in the newly created **OrderDetails.qvf**.
4. Open the **DetailsTemplate** app navigate to the Dynamic Data section and copy then entire script in that section (Lines 1 – 114).
5. Paste previously copied script into the newly created **OrderDetails.qvf** Dynamic Data section.

 The dynamic view script contains several different sections to generate the where clause, in the example we have three subroutines which will not require modifying for the purposes of this exercise. For reference, they are explained below:

|  |
| --- |
| The first part of the process specifies the dimensional values which are to be passed dynamically from the hybrid app into the detail dynamic view source app via an INLINE LOAD:    Options exist to specify how the data in the fields will be selected by prefixing the column name as follows:   * ods = Selected values * odo = Associated values * odso = Selected/associated values   Quoting and delimiters can also be specified (default being single quotes with comma delimited).  This will then call the INLINE Load subroutine.  The next subroutine dynamically constructs an INLINE[] Load statement which will contain all of the data items selected/associated from the hybrid app passed into the detail dynamic view source app:    The final subroutine in the process constructs the $(WHERE\_PART) variable which populates the where clause executed on the fact table select statement. It is specifically designed to cater for multiple values in a QVD file where clause (which does not support IN Clauses) so the mixmatch() function is used:    An alternative subroutine (not called in this exercise) is present in thr example to act as a reference showing the changes requried if the data sources are SQL compliant databases:    The $(WHERE\_PART) variable now constructs a where clause using the **IN** function. |

## Modifications to the dynamic view app load script

1. Replace the “YourField” references (lines 64-71) with the first field we will use in the where clause generation: ProductKey and enable the values to be selected by association (odo):



1. Create a new field reference which will also be used in the WHERE clause generation: CustomerKey and enable the values to be selected by association (odo):



Next, we need to add the list of field names to the script section which invokes the WHERE clause creation subroutines:



1. Change the ‘YourField’ reference to ProductKey and add CustomerKey:



Next, we need to add the folder connection details by populating the $(FOLDER) and $(FACT\_QVD) variables:



1. Add the folder connection details and the qvd file: **InternetSales\_Fact.qvd**



1. Add the load script for the **InternetSales\_Fact.qvd** file and change the FROM clause to use the $(FACT\_QVD) variable and add the $(WHERE\_PART) variable to the end of the script.

**NOTE** In a real-world deployment the fact table could contain billions of rows so for the purposes of construction of the detail app you will need to limit the amount of data in the detail app to build the charts. For this exercise, you could add a temporary hard-coded where clause to limit the data and reload e.g. WHERE mixmatch([ProductKey],'477','478','479') which can then be replaced with the $(WHERE\_PART) variable once the chart building is complete:



1. The final stage of the process is to add the dimensional load statements for the following:

* Customers.qvd
* Products.qvd
* ProdSubCategory.qvd
* Geography.qvd

1. We also need to restrict these loads to those items which have corresponding keys in the fact table add where exists clauses as required, an example of this for the customers below:



## Create the dynamic view charts in the detail app

The order details app now contains the relevant script, from here we need to build the charts which will be used as dynamic views:

1. Add a table chart containing:

* SalesOrderNumber
* SalesOrderLineNumber
* OrderDate
* sum(SalesAmount) renamed to Sales Amount
* sum(OrderQuantity) renamed to Order Qty
* sum(TaxAmt) renamed to Tax Amt

Add the chart to the Master items called Sales Detail

1. Add two KPI objects for

* # Products - count(distinct ProductName)
* # Customers - count(distinct EmailAddress)

Add both KPI objects to master items

The final sheet should look as follows:



## Creating the Dynamic Views

The final part of the process will create the dynamic view in the hybrid app using the charts from detail app.

1. Open the hybrid app created in exercise 2 and navigate to the edit sheet view.
2. Open the Dynamic views console from the assets panel:   
     
     
     
   and click *Create new*.
3. The dynamic view should be setup to point to the detail app created previously. The expression to allow the charts to be refreshed on no more than 50 distinct ProductNames being selected for example:  
     
   
4. Add the charts from the dynamic view panel to the sheet



The intial rendering of the charts will show the constraints to execute the charts have not been met:



1. Select a ProductSubcategoryName (or any other filter(s)) until the contraints have been met and the charts reflect:  
     
     
     
   And refresh the chart.  
     
   We have now completed thre exercise.

## Bonus Exercise

Customer X has a further requirement to allow users to drill from the same hybrid app we have developed into a lower level of detail in a dynamic view containing stock movements to allow analysis of the following:

* #Products. #Inward Movements. #Outward Movements
* Detail list of MovementDate, ProductKey, ProductName, Balances and inward/outward movements in a table
* Filters by ProductSubCategoryName, ProductName and Color.

To facilitate this **StockMovementDetails.qvd** can be used. The additional requirements are to pass through the ProductKey from the hybrid app to a dynamic view with a limit of no more than ten distinct ProductNames.



© 2018 QlikTech International AB. All rights reserved. Qlik®, Qlik Sense®, QlikView®, QlikTech®, Qlik Cloud®, Qlik DataMarket®, Qlik Analytics Platform®, Qlik NPrinting®, Qlik Connectors®, Qlik GeoAnalytics®, Qlik Core®, Associative Difference®, Lead with Data™, Qlik Data Catalyst™, Qlik Associative Big Data Index™ and the QlikTech logos are trademarks of QlikTech International AB that have been registered in one or more countries. Other marks and logos mentioned herein are trademarks or registered trademarks of their respective owners.

BIGDATAWP092618\_MD

**About Qlik**

Qlik is on a mission to create a data-literate world, where everyone can use data to solve their most challenging problems. Only Qlik’s end-to-end data management and analytics platform brings together all of an organization’s data from any source, enabling people at any skill level to use their curiosity to uncover new insights. Companies use Qlik products to see more deeply into customer behavior, reinvent business processes, discover new revenue streams, and balance risk and reward. Qlik does business in more than 100 countries and serves over 48,000 customers around the world.

**qlik.com**