Dynamic Views in Qlik Sense SaaS

written by DaltonRuer | March 12, 2021

I need more data

We've all said that before. More importantly, we have all heard that from other users before. Well let's imagine that your users are using something like the Qlik SAP Orders to Cash Accelerator application. They don't have data for Orders. Or Fulfillment. Or Billing. Or Receivables. They have it **all**. The data for the **entire process.** Clearly those users wouldn't need or want more would they?

Even those users might get greedy. No way you say?

Well, Qlik also provides accelerators for SAP Financials and SAP Inventory Management. So I can easily imagine a user wanting insights from those applications as well even while viewing insights about the entire Orders to Cash process.

How will you provide the access?

The question isn't if they will want more, we will know they will, the question is … "How will you provide the access?"

Document Chaining

One solution might be Document Chaining. That's a phrase that goes all the way back to early QlikView days. It involves having one application invoke another application and pass the current filter state. In our case perhaps while the users are in the Orders to Cash application we simply provide buttons that would allow them to go to the SAP Financials or SAP Inventory Management applications. Our friends at Vizlib recently made life easy by providing that functionality for you. They have also documented how you could code up the passing of current filters. In a Document Chaining situation your end user would then have access to the entire application and the data was already loaded into it. Yeah, we passed a Company Code or a Cost Center or a Profit Center and our user has access to anything in the SAP Financials application, Problem solved.

On Demand Application Generation

Maybe they do want access to "some" of the SAP Financials data, but they want **current** data. Meaning they want you to read the data right as they request it. Yipes!

Years ago Qlik introduced On Demand Application Generation (ODAG for short) as a way to provide a bit more flexibility for responding to our end users wants and whims. Cases just like that.

Use cases where you need access to what could be gajigabytes of data, and you need it right now. Creating an On Demand Application is just like creating any other application. With the exception, that the LOAD script is modified to only read the data based on the values passed to the application.

So our calling application would pass any selections that the

end user had made, and the ODAG application would reload all of the data based on those selections, and the user would then be presented with that new version of the application. Maybe the problem is solved.

If you are an old Qlikkie you might recall Direct Query and be thinking "why wouldn't I just use that?" Direct Query provides for that ability to dimensional values inside of an application, and then it pulls other data fields live for the end user while they are working. There are a number of limitations in working with Direct Query that might prohibit using it that ODAG solves. ODAG also provides the ability to keep the parent application nimble and focused, while still allowing for access to the other data/visuals only when needed. An application built to support ODAG can be called by many different applications. A base application can also call out to as many ODAG built applications as it requires. Providing incredible reuse and flexibility, as well as real time access to those gajigabytes of data you have.

Dynamic Views

Dynamic Views provide a slightly different form of ODAG that I really like. It's similar in that you still build another application, called a template app, but you don't need to build a very robust application. You only need to extract the minimum set of data needed for the end user, and you only need to build the minimum set of charts. That sounds odd, but here is the beauty … your end user will be visualizing the charts in the application they are already in.

Why do I like that? Because as I always say "**Context is King**." If the user has filtered based on a bunch of analysis, Dynamic Views provide the ability to visualize that data, external to the application itself, but right there alongside the charts they are currently looking at. Here is a bonus, if they are looking at a Dynamic View and they change their selections, they will see a warning sign that the data doesn't match their

current filters. They can choose to refresh so that it does match the new filters, or they can click a button to re-apply the old selections that were used when the Dynamic View was last refreshed.

Step 1: Enable it

Enough of the options and background information. Let's jump right in to how to enable Dynamic Views in your SaaS tenant. You will need to go to your Management Console, and choose Settings. Scroll through the settings and you will need to ensure that both "On-demand app generation" and "Enable dynamic views" are enabled.

Step 2: Building an Application that can be called for ODAG/Dynamic Views

For my example, let's imagine that what my end user wants to see in the SAP Orders to Cash application is some very specific real time data from the General Ledger. Oooh, that does sound like it might have gajigabytes of data. In fact, while I'm at it, here is the data model for the SAP Financials application. Can you see why we might not want to just include all of it in our O2C application, and might not want to load that entire application On Demand?

When ODAG was first introduced the coding was admittedly pretty complicated. Sure there were helpful Qlik Community posts out there showing you how to do it. But the reality is that for many, it was a little overwhelming. The good news. Ok I should say "great news" is that it has been rearchitected

and is super simple. Here is a very simple SQL query that pulls the specific fields that are end users need to see in real time from my Data Mart that resides in Snowflake. Nothing fancy, a few fields and a simple WHERE clause.

Notice below I have copied that same command and made 2 very simple changes to the WHERE clause that allows me to dynamically use the ODAG variables that will get set when I call the application from my Orders to Cash accelerator. When I said it was made much easier than it was originally, I meant it. What about that weird "IF '$(odagActive)'=" THEN" line at the top? Well if I try to Load the script when I'm building the application the WHERE clause will fail because those ODAG variables haven't been passed. If I can't load any sample data, I can't build my visuals. This nifty little trick enables me to simply force the ODAG variables that are created when called, and ensure that I can load some sample data to utilize to create my visuals. You can find complete documentation for how to create your ODAG/Dynamic View templates right on the Qlik Help site.

If you are thinking ahead to the fact that perhaps your data is numeric. Or to the fact that maybe you want only Selected values. Or to the fact that you want to load Excluded values. Don't worry, Qlik Help also has you covered with a robust explanation of all versions of the ODAG variables you can call.

Step 3: Creating the Charts to be used

This step is also easy. Just create any type of charts that you want to utilize and then simply make them Master

Visualizations. In my example case here I have created a Table object that contains the General Ledger Details that we loaded, and 2 KPI objects that sum the Credits and the Debits.

Step 4: Create the Dynamic View

In our calling application, SAP Orders to Cash in my case, you simply go to Edit mode, and choose Dynamic Views. Then press Create New.

A dialog box will appear and all you need to need to do is provide a Name, and choose the Template app from the dropdown list. Being super creative I've called mine "General Ledger Details" and the template app I built in steps 2&3 I called "SAP GL Details."

When it comes to Row Limit you can choose from None or Row Limit Expression, which is what I chose. None simply means that the data source you are calling is small and you simply don't care if the end user has filtered anything or not. In my case, you know that gajigabytes of General Ledger Data, I did want to ensure that my end user selected something. In some cases you might want to do a COUNT(SomeFieldID) type expression that counts how many ID's are possible and then set the Maximum Row Count to some number like 50,000. You would determine your limit based on the time your end users will be willing to wait for data to be loaded. Of course that will depend on your network bandwidth and data source.

In my case as you can see I simply said "as long as the end user selects at least 1 Company Code or at least 1 Cost Center then let's go get the data."

Finally press "Create" and you will have created your first Dynamic View. Yeah!

Step 5: Drag and Drop

Hopefully that doesn't sound too intimidating.

But that's it my friends, after you have created the Dynamic View it will then display the Master Visuals you created in Step 3 and all you need to do is Drag and Drop the ones you want to utilize in your app onto your canvas.

Step 6: Take it for a Spin

In this animated GIF you will see that after having selected some values I simply press Refresh and the KPI's and Details are now included.

Summary

A few simple steps and I've now provided the ability for my end users to access SAP General Ledger details in a live query mode. In a manner that honors their need to retain the context of why they selected certain filters in the first place. The best part is, you can as well.

If you were not aware that Qlik provided this simple and elegant form of providing users access to data on the fly, I sure hope you will consider Dynamic Views in the future.

If you were aware of Dynamic Views but had been afraid of the old style of coding that was required for ODAG templates, I hope you will reconsider their use because it really has become so much easier.

If you have been using ODAG and Dynamic Views in your Windows environment, I hope this brief post confirms what you were

hoping to hear. That all this wonderful functionality is available in Qlik Sense SaaS as well.

Why so much hope? Because if there is one thing I know for certain, your end users are going to ask for more data and I want you to succeed in giving it to them.