8 DATA MODEL OPTIMIZATION

Objectives

- Discuss performance tuning
- Explore the impacts of appropriate document design
- Introduce security concepts
- Review best practices

8.1 Performance Tuning

8.1.1 What makes up the size of a QlikView document?

- Number of rows
- Number of columns
- Number of distinct values / column
- Data structure
- Number of sheets / sheet objects

Number of rows

- Question: Is it really necessary to have 10 years of data in the same QlikView document?
- Question: If there are too many detail rows, is it feasible to create a roll-up version at a higher aggregate level?

Number of Columns

Be selective about what fields are included in the QlikView document

- Focus on the larger fact tables first
- But don't ignore the dimension tables

Eliminate fields that are not currently being used

- All fields *might* be used in the future but are not used now
- The goal is to have a Lean QlikView document

Number of Distinct Values / Column

Fields with highly distinct values use the most space

Many of them can be eliminated / truncated altogether

- System keys
- Timestamps with minutes / seconds

Others can be transformed by breaking them up into separate components

- Phone Numbers
- Timestamps

Data Structure

Key Fields

Denormalization vs. Normalization

Normalization" Definition (Wikipedia):

Normalization splits up data to avoid redundancy (duplication) by moving commonly repeating groups of data into a new table. Normalization therefore tends to increase the number of tables that need to be joined in order to perform a given query, but reduces the space required to hold the data and the number of places where it needs to be updated if the data changes.

Number of sheets / sheet objects

Remove hidden sheets and sheet objects that are not being used

Remove unused variables

Remember to keep your QlikView document as trim as possible

8.1.2 Document Design

Avoid "Show Frequency"

Forcing Proper Object Display

- Calculation Conditions
- Show Conditional

Avoid too many active objects on the same sheet

Minimized objects consume no resources, use them whenever appropriate

8.1.3 Security Impacts - Making the Right Choices

QlikView Publisher Security

Section Access

Publisher Security

Breaking up one large QlikView file into multiple smaller files based on row level security

Effective for memory management if security profiles do NOT contain much overlapping data

Section Access

Dynamically reduces the QlikView file at logon based on user authorization

8.2 Best Practices for QlikView File Optimization

8.2.1 Complex Dimensions and Expressions

Many dimensions and expression that are to be placed in charts or tables require some degree of complex scripting such as IF THEN ELSE statements or WHERE [FIELD1] IS NULL.

Many developers build their initial data model using the processes above but then stop amending the script and work solely within the dashboard/GUI. When designing/writing your script you should already be aware of some of the measures that you are looking to create in the end.

Where ever possible you should look to place all complex formulas and statements within the script of the document and not in the actual dashboard/document objects.

If you are to use complex expressions within a dimension or calculation then you should move as much of this as possible into the scripting of the QVW.

8.2.2 Resource Intensive Expressions

Count (Distinct 'FieldName')

Replace the count() with sum() and the distinct qualifier by assigning the value '1' to each distinct occurrence as it is read in the script

If (Condition(Text),…..)

Map Text to numeric e.g. by using autonumber and/or do the test in the script

Sum (If (Condition, 'FieldName'…))

Here the aggregation is independent of the table dimensions and the result is distributed over the dimensions of the table. The problem can be treated either by doing the test in the script and aggregating in the table or by doing the whole operation in the script. There are numerous techniques for this e.g. interval match, group by, peek, if….then…else.

If (Condition, Sum('FieldName')..)

Included here to emphasize the difference to above. This aggregation is completely contextual

If (Condition1, Sum('FieldName'), If (Condition2, Sum('FieldName')

The logic of nested If..then else.. is conceptually easy but can often become troublesome to administer. We have seen cases with hundreds of nesting levels. This will be memory and CPU intensive. Often the "Conditions" can be

replaced by transforming them. A typical example is aggregating quantity*price where price is variable. This can be handled by "extended interval match". If two conditions, e.g. " A AND B " are to be satisfied the test might be replaced by a condition "C".

Sort text

QlikView automatically evaluates if a Field is to be treated as numeric, text or general. Fields evaluated as text will be sorted as text which is the slowest sort operation. This can be replaced manually to sort by load order.

Dynamic captions and text objects

Expressions can be entered almost anywhere that you can enter text. The evaluation of an expression is however dependent on its environment. Expressions in charts and straight-and pivot-tables that are defined in the expressions dialog are embedded and only calculated when the object is active. For instance they are not calculated when the object is minimized.

On the other hand if the object title is calculated this calculation is performed every time any change occurs. We also have numerous ways of defining show conditions, calculation conditions etc. These tests will also be performed at all times. Some expressions are more expensive than others and of course become more expensive the more frequently they have to be evaluated. The introduction of asynchronous calculation has shifted the behavior and these effects may have become more noticeable in your documents. The time functions e.g. **Now()**, **Today()** will be evaluated whenever a recalculation has to be done. Especially the Now() function can become quite costly since it causes a recalculation of the document every second. For example

```
If ( ReloadTime()+3>Now(), 'Old Data', 'New Data')
```

Here one might consider...

```
If ( ReloadTime()+3>Today(), 'Old Data', 'New Data')
```

As a simple test, put the expressions into textboxes. Then try sizing the textbox with Now() in it.

8.2.3 Adding Aggregatable Columns in to Your Script

It is sometimes good to add manual columns in to your script to give your document the ability to sum up over these values. This will mean you can then place the complex statement (**If**) in to your script and have a simple sum in your dashboard object.

To do this you should simply place an IF THEN ELSE statement in to your script that substitutes a database column or a 1 or 0 as a value to enable a summing/count to take place.

Examples:

```
IF(ACTIVE='Y',1,0)
```

```
IF(ACTIVE='Y',sales_amount,0)
```

Forcing through a 0 then takes out all of the unnecessary / unwanted values when you apply your SUM.

8.3 Creating Incremental Loads

An incremental load is a common database task. Incremental loads are used when loading only the new or changed records from a database. QVD files are used to do incremental loads in QlikView. Properly engineered incremental loads can significantly improve performance.