Buenas Prácticas

Scripting

Month 11, 2018

José Manuel Martínez Mayoral

Director BI

Qlik LEAD WITH DATA

Agenda

- Qlik Deployment Framework
- Arquitectura en 3 capas
- Mejora del Rendimiento
- Facilitar el Mantenimiento
- Incrementar la Velocidad de Desarrollo

Buenas Prácticas

Scripting

Qlik Deployment Framework

Arquitectura 3 capas

Mejora del Rendimiento

Facilitar el Mantenimiento

Velocidad de desarrollo

Qlik Deployment Framework

¿Por Qué usar QDF?

- Provee una estructura de directorios optimizada, racional y común para todos los proyectos.
- Cada directorio tiene una función específica predefinida.
- **Reusabilidad**: Los objetos se reutilizan, no se reinventan
- Facilita la configuración y compartición de datos entre proyectos.
- Los desarrolladores comparten standards y buenas prácticas.
- Es una plataforma robusta para aplicaciones. Acaba con las aplicaciones aisladas.
- Desarrollos rápidos en entornos crecederos.
- La metodología se replica desde un único departamento hasta toda la empresa.
- Ampliamente probado con éxito en miles de instalaciones en todo el mundo.

Qlik Deployment Framework

¿Por Qué usar QDF?

- Sin QDF: Cada Proyecto necesita un tiempo largo de arranque. La falta de standards y buenas practicas hacen que el mantenimiento sea todo un desafío

- Framework propio del cliente: Puede acortar el tiempo de desarrollo pero su implementación consume más tiempo y no sigue ningún standard o recomendación.

- QDF: Alinea los proyectos QlikView con las organizaciones y los procesos para acortar el arranque y los tiempos de desarrollo.

Qlik Deployment Framework

Estructura de un contenedor

99.Shared_Folder
o
1.Project_HR

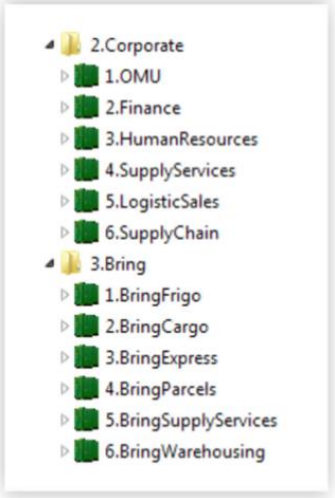
Existe una estructura común que contiene datos, KPIs y accesorios comunes a todos los proyectos de la empresa y la misma estructura replicada para cada uno de los proyectos individuales con datos, Kpis y accesorios específicos.

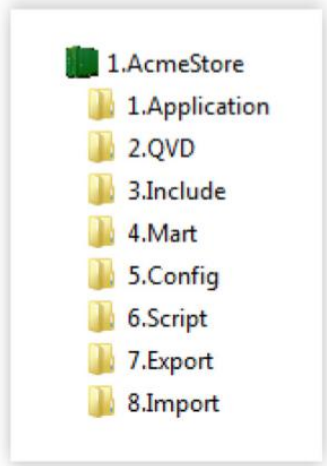
1.Application

Aplicaciones .qvw. A su vez puede dividirse en subcarpetas según la naturaleza de las aplicaciones: 1.Extractor , 2.Dashboard

2.QVD

Ficheros de datos. Se pueden organizar en subcarpetas según su función: 1.Extract 2.Transform 3.Load

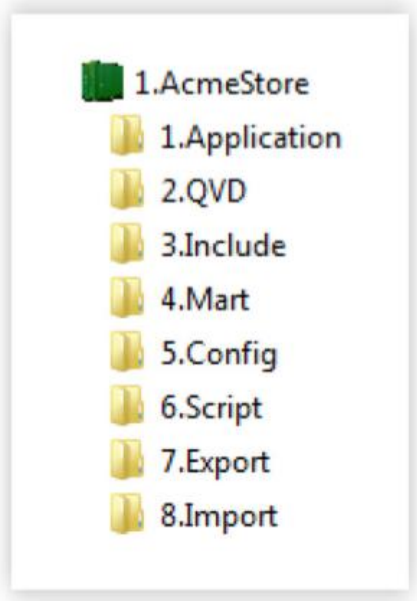




Qlik Deployment Framework

Estructura de un contenedor

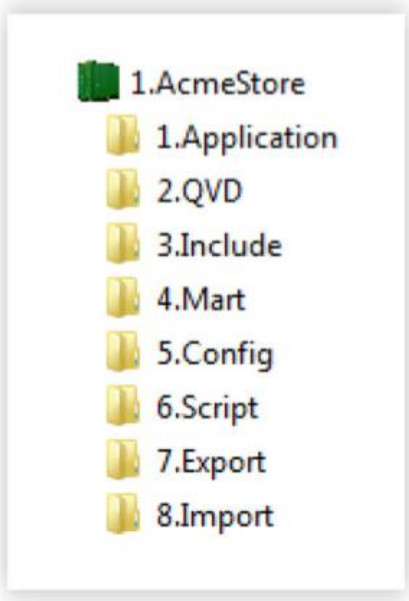
3.Include	Partes de código llamadas desde la pestaña Main del script
1.BaseVariable	Almacena todas las variables que necesita el Framework, como las rutas dentro de los contenedores
2.Locale	Configuración específica para diferentes regiones como separador decimal, formato de fecha, nombres de días y de meses...
3.ConnString	Cadenas de conexión a los orígenes de datos
4.Sub	Almacén para subrutinas. Es el modo de reutilizar código entre aplicaciones.
5.ColorScheme	Definición de los colores para la aplicación
6.Custom	Objetos a incluir en el script que no tienen cabida en las subcarpetas anteriores



Qlik Deployment Framework

Estructura de un contenedor

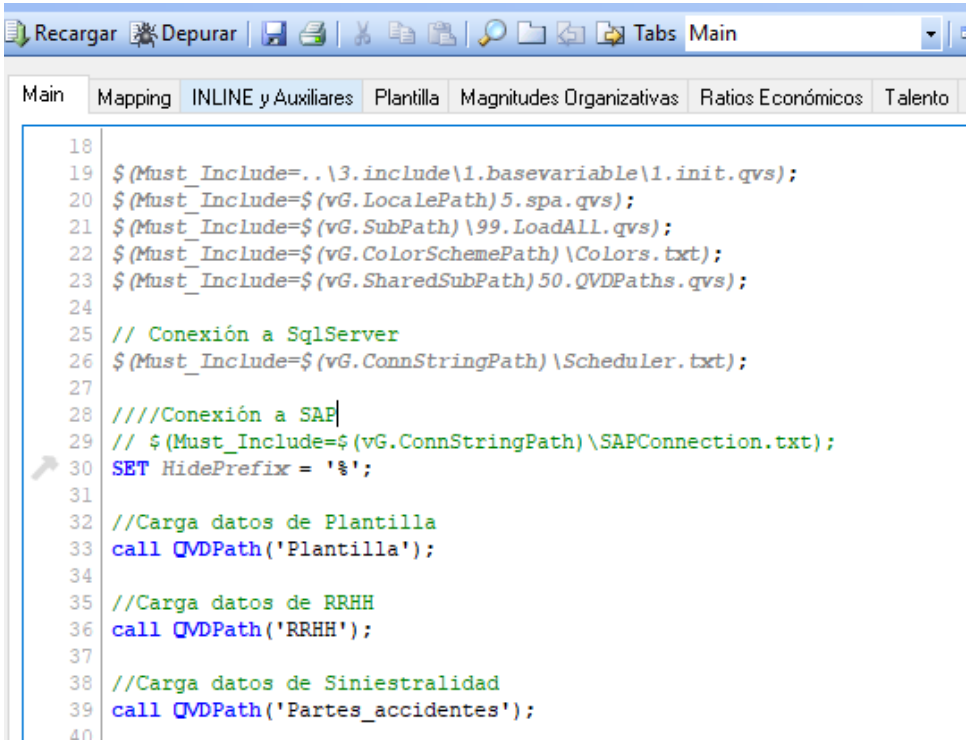
4.Mart	Carga y compartición de datos con desarrolladores y usuarios avanzados. Básicamente son qvw que se cargan con Binary en aplicaciones finales
5.Config	Ficheros de configuración como xls y txt para traducciones de idiomas, Biblioteca de KPIs, dimensiones no contenidas en los origenes de datos… etc.
6.Script	Partes de código a incluir en el script. Típicamente contiene **la sección de acceso.**
7.Export	Directorio usado para almacenar datos exportados desde QlikView. Normalmente .txt o .qvx
8.Import	Datos en ficheros .xls .dat o .txt y datos auxiliares como códigos postales, Paises ISO
9.Misc	Cosas relacionadas con la aplicación
1.Documentation	Documentos, toma de requisitos, actas de reuniones...
2.Extensions	Extensiones usadas en el desarrollo de la aplicación
3.Images	Iconos, fotos, logotipos, baderas de diomas y todas las imágenes utilizadas en la aplicación.



Qlik Deployment Framework

¿Cómo se usa?

- Cada directorio tiene un .txt con la instrucción para pegar en el script.



Buenas Prácticas

Scripting

Qlik Deployment Framework

Arquitectura 3 capas

Mejora del Rendimiento

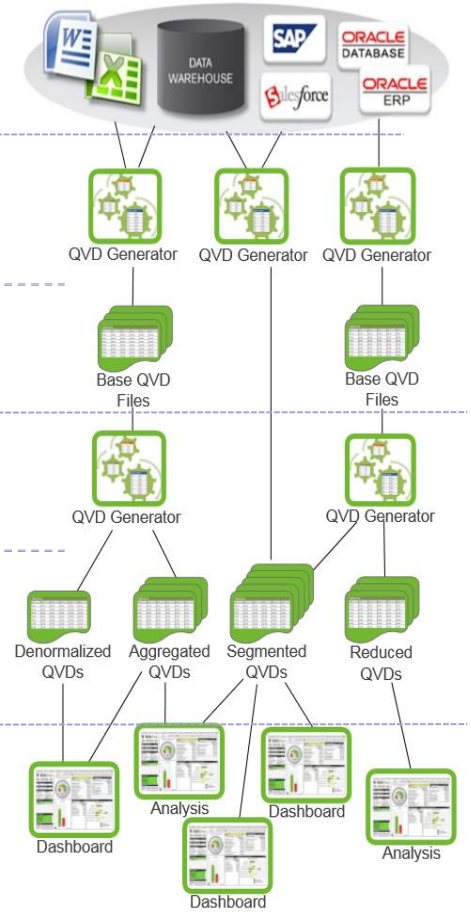
Facilitar el Mantenimiento

Velocidad de desarrollo

Arquitectura de 3 capas

Fuentes de datos: Bases de datos, ficheros… etc

Capa de Extracción	Extractores	Aplicaciones que extraen información (y opcionalmente desnormalizan) de los orígenes de datos en bruto o con mínimas modificaciones
	QVDs Base	Ficheros QVD generados por los extractores. La relación con las tablas originales puede ser una a una o contener vistas desnormalizadas de varias tablas.
Capa de Transformación	Transformadores	Leen los QVDs anteriores para hacer agregaciones, desnormalizaciones y manipulaciones y generan los QVDs finales que se usarán en la capa de presentación
	QVDs Transformados	Estos QVDs están agregados, reducidos, divididos por categorías y depurados para optimizar su reutilización, facilidad de implementación y rendimiento.
Capa de Presentación	Cuadros de Mando	Aplicaciones en producción usando los datos de los QVDs anteriores. Ya no debería ser necesario aplicar buenas prácticas de modelización de datos para optimizar el rendimiento de las aplicaciones



Qlik LEAD WITH DATA

Arquitectura de 3 capas

Ventajas

- **Reutilización**. Los mismos QVDs sirven para muchos cuadros de mando
- **Integridad** de datos entre aplicaciones. Todas las aplicaciones obtienen los datos de los mismos QVDs. Una vez realizada la extracción estos datos están libres de las modificaciones de la BD.
- **Normalización** de datos. Los criterios de transformación son comunes a todas las aplicaciones.
- Mejor **experiencia de usuario**. La extracción se realiza en periodos de baja actividad, lo que le libera durante el horario laboral para atender peticiones de usuario con mayor velocidad.
- Incremento del **rendimiento** de los servidores de BD. Una sola extracción en lugar de una consulta pesada por cada cuadro de mandos.
- Facilidad de **mantenimiento**. Una modificación en los extractores o transformadores afectan a todas las aplicaciones.
- Fácil **ampliación** con nuevos orígenes de datos. Basta con añadir nuevos extractores y transformadores siguiendo los mismos criterios definidos previamente.

Buenas Prácticas

Scripting

Qlik Deployment Framework

Arquitectura 3 capas

Mejora del Rendimiento

Facilitar el Mantenimiento

Velocidad de desarrollo

Mejora del Rendimiento

Recomendaciones

- Los campos clave entre las tablas deben ser números siempre que sea posible.
- Utilizar **Autonumber** para convertir en texto campos numéricos, especialmente los campos clave.
- Modelo de estrella en vez de copo de nieve.
- Eliminar claves sintéticas y referencias circulares.
- Se desarrollará utilizando la sentencia **Qualify** sólo cuando sea imprescindible.
- Mapear todas las tablas con dos campos (código + descripción)
- **timestamp**, separar ese campo en dos: uno para fecha y otro para hora.
- Evitar comparaciones con campos de texto. Son mucho más lentas.

Mejora del Rendimiento

Recomendaciones

- Añadir un campo con ‘1’ a cada registro. SUM() es mucho más rápido que COUNT().
- Toda expresión que no sea dependiente de las selecciones del usuario debería ser calculada en el script incluso si eso implica modificar el modelo de datos.
- Evitar la anidación de LOAD Precedentes

```
VENTAS:
LOAD ID_CLIENTE,
     SUM(IMPORTE) AS IMPORTE
GROUP BY ID_CLIENTE;
LOAD ID_CLIENTE,
     FACTURA,
     IMPORTE
FROM VENTAS.qvd   (qvd);
```

```
AUX_VENTAS:
LOAD ID_CLIENTE,
     FACTURA,
     IMPORTE
FROM VENTAS.qvd   (qvd);

VENTAS:
LOAD ID_CLIENTE,
     SUM(IMPORTE) AS IMPORTE
RESIDENT AUX_VENTAS
GROUP BY ID_CLIENTE;

DROP TABLE AUX_VENTAS;
```

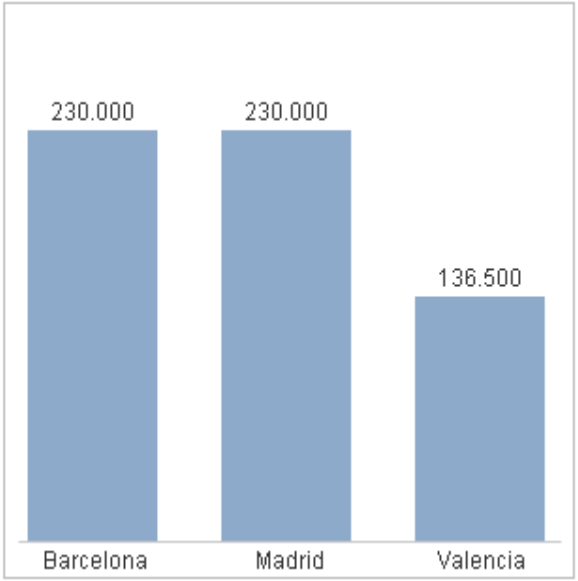
Mejora del Rendimiento

Recomendaciones

- Crear campos Flag en las tablas del modelo para evitar el uso de IFs en las expresiones.

Provincia	Ventas
Avila	187.000
Badajoz	256.000
Barcelona	230.000
Bilbao	227.000
León	49.000
Madrid	230.000
Segovia	90.000
Sevilla	150.000
Valencia	136.500

```
IF(Provincia='Madrid'
   OR Provincia='Barcelona'
   OR Provincia='Valencia',
SUM(Ventas))
```



Provincia	Flag	Ventas
Avila	0	187.000
Badajoz	0	256.000
Barcelona	1	230.000
Bilbao	0	227.000
León	0	49.000
Madrid	1	230.000
Segovia	0	90.000
Sevilla	0	150.000
Valencia	1	136.500

```
SUM(Flag*Ventas)
SUM({<Flag={1}>} Ventas)
```

Buenas Prácticas

Scripting

Qlik Deployment Framework

Arquitectura 3 capas

Mejora del Rendimiento

Facilitar el Mantenimiento

Velocidad de desarrollo

Facilitar el mantenimiento

Organizar el script en pestañas

- Las pestañas son las partes en las que se divide el código del script, por lo que hay que intentar que estén estructuradas por bloques de sentencias relacionadas y que sean fáciles de entender.
- El nombre de las pestañas debe ser descriptivo. Preferiblemente en minúsculas (ocupan menos).
- Cada pestaña tendrá una o varias secciones para separar el código y para poder monitorizar, mediante el log, las sentencias de una forma más granularizada.
- Siempre la misma estructura en todos los documentos:

Facilitar el mantenimiento

Comentarios y sangrado de código

- El uso de comentarios es necesario para que las aplicaciones sean autocomentadas en la medida de lo posible y así facilitar a terceros su lectura y comprensión.
- En cada una de las pestañas tiene que haber una cabecera indicando el autor, una descripción del contenido y de las secciones, y la versión Así como las modificaciones realizadas.

```
//============================================================================
// AUTOR:       Jose Manuel Martínez
// DESCRIPCIÓN: Contiene toda las sentencias que hay que ejecutar al final de la  recarga del documento.
//                      Contiene la llamada al include que elimina las variables que no se necesitan en diseño.
// VERSIÓN:     v00 - Versión Inicial (12/12/2016)
//
// MODIF:  Maria Jesús Ayucar– 03/01/2017  - Añadido  bucle lectura de TODOS los .qvd del
// directorio
// MODIF:  Hector Muñoz– 07/01/2017 – Modificado el bucle de MºJesús para carga incremental
//
//============================================================================
```

Facilitar el mantenimiento

Comentarios y sangrado de código

- Se comentará la carga de cada tabla con el mismo propósito añadiendo comentarios relevantes para futuros desarrolladores.

```
//----------------------------------------------------------------------------------------------------------------
// TABLA:       LOG
// DESCRIPCIÓN: Tabla de log que contiene la información de la ejecución de las
//              sentencias de la aplicación.
// ORIGEN:      Fichero de QVD 'LOG.qvd'
//----------------------------------------------------------------------------------------------------------------
```

- Cada modificación debe estar perfectamente documentada, con el nombre del desarrollador, la fecha, Tipo cambio (modificación, código nuevo o suprimido), el motivo de la modificación y si procede, el usuario que lo solicitó.

```
//MODIF: JTZ 30/10/2016: Se comenta para que no den error los empleados que se han ido de la empresa.
// Si se descomenta, las horas YTD no cuadran con las de Formación

//WHERE EXISTS(%KeyFormacion)
```

Facilitar el mantenimiento

Comentarios y sangrado de código

- Es muy recomendable que cada campo tenga un comentario de línea a la derecha explicando el contenido del mismo, sobre todo en aquellas tablas que no lean de tablas residentes o cuando el renombrado no sea muy descriptivo por motivos de brevedad.
- Los campos, asignaciones, palabras reservadas y comentarios deben de estar perfectamente sangrados para facilitar la lectura.

```
SECCION_ACCESO:
LOAD    [CFG_USUARIOS Acceso]       AS  ACCESS,      // Acceso del usuario
        [CFG_USUARIOS Nombre]       AS  USERID,      // Identificador del usuario
        [CFG_USUARIOS Contraseña]   AS  PASSWORD,  // Contraseña del usuario
        [CFG_USUARIOS Licencia]     AS  SERIAL       // Licencia del usuario
RESIDENT CFG_USUARIOS
WHERE   [CFG_USUARIOS Acceso] = 'USER';
```

- Los sangrados serán de los espacios que quiera el desarrollador, pero constantes a lo largo de todo el código.

Facilitar el mantenimiento

Nombrado y tipado de palabras

- Las funciones de QlikView se escribirán con la primera letra de cada palabra en mayúscula.

  DocumentPath(), Count(), PurgeChar(), SetDateYearMonth()

- Las palabras reservadas siempre van en mayúscula.

LOAD, CONCATENATE, CONNECT, DROP TABLE, FOR..NEXT, IF..THEN..ELSE..END IF,….

Facilitar el mantenimiento

QVDs

- A la hora de almacenar los qvds hay que llamarlos de la misma forma que se llama la tabla a partir de la que se crean.

CLIENTES_INTERNACIONAL.qvd, FACTURAS.qvd, CFG_EJECUCION.qvd, ...

- Para distinguir el origen de los QVDs almacenados en la carpeta 2.QVD y facilitar su clasificación, es recomendable utilizar nomenclaturas distintas para los ficheros, utilizando un prefijo distinto que identifique su tipo.

  - **QL_** para los QVDs generados por el Loader
  - **QH_** para QVDs históricos cuando se usan cargas incrementales… etc.

Facilitar el mantenimiento

Carga de campos

No se recomienda el uso de sentencias **LOAD ***. En su lugar, se listarán todos los campos a cargar. De esta manera se previenen errores en el futuro en caso de añadir o quitar columnas en la tabla origen, se optimiza la lectura de la base de datos y clarifica el propósito de la carga.

```
LOAD PERNR          AS IdEmpleadoPaquete,
//     Date(ENDDA) as ENDDA,
     CURRENCY       AS MonedaPaquete,
     SUBTY          AS Subtipo,
     CPI01,
//     CPI02, CPI03, CPI04, CPI05, CPI06, CPI07,
     AMOUNT01,
//      AMOUNT02, AMOUNT03, AMOUNT04, AMOUNT05, AMOUNT06, AMOUNT07,
     CURRENCY01,
//     CURRENCY02, CURRENCY03, CURRENCY04, CURRENCY05, CURRENCY06, CURRENCY07,
//     EXCHANGE_RATE as "Tipo Cambio",
//     EXCHANGE_DATE as "Fecha Cambio",
//     COL_INDEX     as "Coste de Vida",
//     COLI_DATE     as "Fecha Coste Vida",
     STATUS         AS "Estado"
```

Buenas Prácticas

Scripting

Qlik Deployment Framework

Arquitectura 3 capas

Mejora del Rendimiento

Facilitar el Mantenimiento

Velocidad de desarrollo

Velocidad de desarrollo

Renombrado de campos

- Los nombres deberán ser cortos y descriptivos.
- Renombrarlos siempre con el nombre con el que se van a visualizar en el CdM de modo que al usuario le resulte fácil entenderlos. De esta forma se gana mucho tiempo al no tener que utilizar la etiqueta de visualización de cada campo.
- Los nombres de los campos estarán separados por espacios y la primera letra en mayúscula, salvo acrónimos y casos especiales. Se admiten tildes y caracteres del alfabeto castellano.

Ejemplos: **Cliente, [ID Cliente], [Año Facturación], ...**

- Los nombres de los campos que se utilicen para cruces entre tablas se compondrán de la cadena '**%KEY_**' seguido del nombre de las tablas con la primera letra en Mayuscula y sin espacios. Separamos las dos tablas con '_'.

Ejemplos: **[%KEY_Calendario_Clientes], [%KEY_Clientes_Facturas], ...**

- Estos campos es muy recomendable que se creen como resultado de la función AutonumberHash256 () ya que se optimiza la asociación entre las tablas.
- Los campos índice los renombraremos a **Id_<campo>**. De esta forma aparecerán agrupados en los desplegables de los objetos y serán fácilmente seleccionables.

Velocidad de desarrollo

Renombrado de campos

- Dependiendo de lo que el desarrollador esté familiarizado con los orígenes de datos, existen 2 estrategias:

1.- En el momento de la carga

```
CLIENTES:
LOAD
    ID_CLIENTE       AS "Código Cliente",
    NOMBRE           AS Nombre,
    IDENTIFICACION   AS "N.I.F",
    TIPO_CLI         AS "Tipo Cliente"
FROM CLIENTES.qvd (qvd);
```

2.- Al final del Script

```
CLIENTES:
LOAD
    ID_CLIENTE ,
    NOMBRE,
    IDENTIFICACION,
    TIPO_CLI
FROM CLIENTES.qvd (qvd);
.....
.....
RENAME FIELD  ID_CLIENTE TO "Código Cliente";
RENAME FIELD  NOMBRE     TO Nombre;
RENAME FIELD  IDENTIFICACION TO "N.I.F";
RENAME FIELD  TIPO_CLI   TO "Tipo Cliente";
```

Velocidad de desarrollo

Cómo nombrar las tablas

- El nombre de las tablas será corto, descriptivo y siempre en mayúscula.
- Como separador usaremos el subrayado '_'. Para tablas del mismo tipo se añadirá el sufijo **'_NNN'** (secuencia). Cuando se trate de una tabla de mapeo se añadirá el prefijo **'MAP_'**. Cuando se trate de una tabla auxiliar se añadirá el prefijo **'AUX_'** y el sufijo **'_NNN'** (secuencia) si procede.

```
MAP_PAISES:
MAPPING LOAD
Cod_pais, Nombre
FROM PAISES_ISO.qvd (qvd);
```

```
AUX_VENTAS:
LOAD ID_CLIENTE,
     FACTURA,
     IMPORTE
FROM VENTAS.qvd  (qvd);

VENTAS:
LOAD ID_CLIENTE,
     SUM(IMPORTE) AS IMPORTE
RESIDENT AUX_VENTAS
GROUP BY ID_CLIENTE;

DROP TABLE AUX_VENTAS;
```

Velocidad de desarrollo

Cómo nombrar las variables

- Todas las variables del documento se han de añadir a la lista que hay en el fichero de configuración 3.Include/1.BaseVariables (rutas) o en 5.config (KPIs) para su correcta clasificación y parametrización.
- En casos excepcionales pueden definirse en la pestaña Main, como las usadas para sustituir literales.
- Siguiendo las recomendaciones del Deployment Framework, el nombre de las variables estará formado por 3 caracteres que identifiquen su tipo(Minúscula, Mayúscula y punto) , seguido de una cadena de palabras que la identifiquen, con la primera letra de cada palabra en mayúscula, sin caracteres especiales (No se admiten acentos, subrayados, guiones… etc).

vL.AdministradorProyecto	Variable local utilizada en un entorno local (el propio documento)
vG.Color1	Variable global utilizada en varios documentos dentro de un proyecto
vU.DesarrolladoPor	Variable universal (en todos los entornos QlikView)

Velocidad de desarrollo

Otras recomendaciones

- Utilizar el **HidePrefix=%** para ocultar a los diseñadores campos que sólo se vayan a utilizar durante el proceso de desarrollo.

- Utilizar el tercer parámetro de la función **Applymap()** para poder identificar de manera eficiente campos no correctamente informados o con información segmentada o errónea.

```
MAP_MITABLA:
Mapping LOAD  * INLINE [
Localidad,  Zona
Madrid,        1
Parla,          2
Alcorcón,      2
Alcobendas,  3
Algete,         3
];
 LOAD
ApplyMap ('MAP_MITABLA ', Localidad, 'Resto')
```

Gracias por la atención.

j.martinezmayoral@gmail.com