



Tutorial - Pasos siguientes en la elaboración de scripts

Qlik Sense®

November 2024

Copyright © 1993-aaaa} QlikTech International AB. Reservados todos los derechos.

1 Bienvenido a este tutorial	5
1.1 Lo que aprenderá	5
1.2 Destinatarios del curso	5
1.3 Contenido del paquete	5
1.4 Lecciones de este tutorial	6
1.5 Más información y recursos	6
2 Las sentencias LOAD y SELECT	7
3 Transformar datos	8
3.1 Usar el prefijo Crosstable	8
El prefijo Crosstable	8
Borrado de la caché de la memoria	12
3.2 Combinar tablas con Join y Keep	12
Join	13
Uso de Join	13
Keep	16
Inner	17
Left	18
Right	19
3.3 Uso de funciones inter-registro: Peek, Previous y Exists	21
Peek()	21
Previous()	21
Exists()	22
Uso de Peek() y Previous()	22
Uso de Exists()	25
3.4 Correspondencia de intervalos y carga iterativa	29
Uso del prefijo IntervalMatch()	29
Usar un bucle a While y carga iterativa IterNo()	31
Intervalos abiertos y cerrados	32
4 Limpieza de datos	34
4.1 Tablas de correspondencia	34
Reglas:	34
4.2 Funciones y sentencias Mapping	34
4.3 El prefijo mapping	35
4.4 La función ApplyMap()	35
4.5 La función MapSubstring()	38
4.6 Map ... Using	39
5 Manejo de datos jerárquicos	41
5.1 El prefijo Hierarchy	41
5.2 El prefijo HierarchyBelongsTo	42
Autorización	43
6 Archivos QVD.	47
6.1 Crear archivos QVD	48
Store	48
6.2 Leer datos desde archivos QVD	49
Buffer	51

6.3 ¡Muchas gracias!	53
----------------------------	----

1 Bienvenido a este tutorial

Bienvenido a este tutorial, que ofrece una introducción a la creación de scripts avanzados en Qlik Sense.

Una vez que esté familiarizado con los conceptos básicos de las secuencias de comandos, puede comenzar a realizar operaciones más sofisticadas en sus datos a medida que los carga en Qlik Sense. Esto puede incluir, por ejemplo, la transformación de datos mediante tablas cruzadas, la limpieza de datos y la creación y carga de datos desde archivos de datos de Qlik, conocidos como archivos QVD.

1.1 Lo que aprenderá

After completing this tutorial, you should be comfortable with loading data using some of the more advanced scripting functions in Qlik Sense.

1.2 Destinatarios del curso

Debe estar familiarizado con los conceptos básicos de elaboración de scripts en Qlik Sense. Es decir, debe haber cargado datos y manipulado datos utilizando scripts.

Si aún no lo ha hecho, le recomendamos realizar el tutorial de Script para principiantes.

Necesitará acceso al editor de carga de datos y deberá poder cargar datos en Qlik Sense Enterprise on Windows.

Las instrucciones también sirven en general para Qlik Sense Cloud Business.

1.3 Contenido del paquete

El paquete zip que ha descargado contiene los siguientes archivos de datos que necesitará para completar el tutorial:

- *Cutlery.xlsx*
- *Data.xlsx*
- *Events.txt*
- *Employees.xlsx*
- *Intervals.txt*
- *Product.xlsx*
- *Salesman.xlsx*
- *Transactions.csv*
- *Winedistricts.txt*

El paquete también contiene una copia de la app *Tutorial de script avanzado*. Secciones de script adicionales en la app contienen las secuencias de comandos para las otras apps que cree en este tutorial. Puede cargar la app a su centro de control.

Le recomendamos construir la app usted mismo como se describe en el tutorial para maximizar su aprendizaje. Además, tendría que cargar y conectarse a sus archivos de datos como se describe en el tutorial para que las cargas de datos funcionen.




Sin embargo, si tiene problemas, la app puede ayudarlo a solucionarlos. Hemos indicado qué segmentos de script están asociados con cada tema.

1.4 Lecciones de este tutorial

Dependiendo de su experiencia con Qlik Sense, completar este tutorial le llevará entre 3 y 4 horas. Los temas están diseñados para realizarse secuencialmente. No obstante, puede dejarlo y continuar en cualquier momento. Afortunadamente no contiene exámenes.

- Transformar datos
- Uso del prefijo Crosstable
- Combinar tablas con Join y Keep
- Uso de funciones inter-registro: Peek, Previous y Exists
- Correspondencia de intervalos y carga iterativa
- Limpieza de datos
- Manejo de datos jerárquicos
- Archivos QVD

1.5 Más información y recursos

-  [Qlik](#) ofrece una amplia variedad de recursos si desea más información.
- [La ayuda online de Qlik](#) está disponible.
- Hay disponible formación, incluidos cursos gratuitos online, en  [Qlik Continuous Classroom](#).
- Puede encontrar foros de discusión, blogs y más en  [Qlik Community](#).

2 Las sentencias LOAD y SELECT

Puede cargar datos en Qlik Sense usando las sentencias LOAD y SELECT. Cada una de estas sentencias genera una tabla interna. LOAD se utiliza para cargar datos de archivos, mientras que SELECT se utiliza para cargar datos de bases de datos.

En este tutorial usará datos de archivos, por lo que utilizará sentencias LOAD.

También puede usar un LOAD precedente para poder manipular el contenido de los datos cargados. Por ejemplo, renombrar campos debe hacerse en una sentencia LOAD, mientras que la sentencia SELECT no permite ningún cambio en los nombres de los campos.

Tenga en cuenta las siguientes reglas cuando vaya a cargar datos en Qlik Sense:

- Qlik Sense no distingue entre las tablas generadas por una sentencia LOAD o SELECT. Esto significa que, al cargar varias tablas, no importa si se cargan mediante sentencias LOAD o SELECT o una combinación de ambas.
- El orden de los campos en la sentencia o en la tabla original de la base de datos es indiferente para la lógica de Qlik Sense.
- Los nombres de campo distinguen entre mayúsculas y minúsculas y se utilizan para establecer asociaciones entre tablas de datos. Debido a esto, a veces es necesario cambiar el nombre de los campos en el script de carga para lograr el modelo de datos deseado.

3 Transformar datos

Puede transformar y manipular datos en el Editor de carga de datos antes de usar los datos en su app.

Una de las ventajas de la manipulación de datos es que puede elegir cargar solo un subconjunto de los datos de un archivo (por ejemplo, determinadas columnas de una tabla) para que los datos se manipulen de forma más eficaz. También puede cargar los datos más de una vez a fin de dividir los datos sin procesar en varias tablas lógicas nuevas. También es posible cargar los datos de más de una fuente y combinarlos en una sola tabla en Qlik Sense.

Los siguientes ejercicios le mostrarán cómo cargar datos usando el prefijo Crosstable. También aprenderá cómo unir tablas, usar funciones inter-registro, tales como Peek y Previous, y cargar la misma fila varias veces usando While Load.

3.1 Usar el prefijo Crosstable

Las tablas cruzadas son un tipo habitual de tabla, que ofrece una matriz de valores entre dos listas ortogonales de datos de cabecera. Cuando tenga una tabla cruzada de datos, puede usar el prefijo Crosstable para transformar los datos y crear los campos deseados.

El prefijo Crosstable

En la tabla *Product* siguiente tenemos una columna por mes y una fila por producto.

Tabla Product						
Producto	Ene 2014	Feb 2014	Mar 2014	Abr 2014	May 2014	Jun 2014
A	100	98	100	83	103	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

Cuando cargue la tabla, el resultado será una tabla con un campo para *Product* y un campo para cada uno de los meses.

La tabla Product con el campo Product y un campo por cada uno de los meses

Product
Product
Jan 2014
Feb 2014
Mar 2014
Apr 2014
May 2014
Jun 2014

Si desea analizar estos datos, es mucho más fácil tener todos los números en un campo y todos los meses en otro. En este caso, es una tabla de tres columnas con una columna para cada categoría. (*Product, Month, Sales*).

La tabla Product con los campos Product, Month y Sales

Product
Product
Month
Sales

El prefijo Crosstable convierte los datos en una tabla con una columna para *Month* y otra para *Sales*. Otra forma de expresarlo es decir que toma los nombres de los campos y los convierte en valores de campo.

Haga lo siguiente:

1. Cree una nueva app y denomínela *Tutorial de script avanzado*.
2. Agregue una nueva sección de script en el **Editor de carga de datos**.
3. Denomine a la sección: *Product*.
4. En **Archivos adjuntos**, en el menú a la derecha, haga clic en **Seleccionar datos**.
5. Cargue y después seleccione *Product.xlsx*.
6. Seleccione la tabla *Product* en la ventana **Seleccionar datos desde**.



En **Nombres de campo**, asegúrese de que **Nombres de campo incluidos** está seleccionado para que se incluyan los nombres de los campos de tabla cuando cargue los datos.

7. Haga clic en **Insertar script**.

Su script debería tener el aspecto siguiente:

```
LOAD
    Product,
    "Jan 2014",
    "Feb 2014",
    "Mar 2014",
    "Apr 2014",
    "May 2014",
    "Jun 2014"
FROM [lib://AttachedFiles/Product.xlsx]
(ooxml, embedded labels, table is Product);
```

8. Haga clic en **Cargar datos**.
9. Abra el **Visor del modelo de datos**. El modelo de datos tendrá el siguiente aspecto:
La tabla Product con el campo Product y un campo por cada uno de los meses

Product
Product
Jan 2014
Feb 2014
Mar 2014
Apr 2014
May 2014
Jun 2014

10. Haga clic en la pestaña *Product* en el **Editor de carga de datos**.
11. Inserte lo siguiente encima de la sentencia LOAD:
`CrossTable(Month, Sales)`
12. Haga clic en **Cargar datos**.
13. Abra el **Visor del modelo de datos**. El modelo de datos tendrá el siguiente aspecto:
La tabla Product con los campos Product, Month y Sales

Product
Product
Month
Sales

Observe que, por lo general, los datos de entrada solo tienen una columna como campo calificador; como una clave interna (*Product* en el ejemplo anterior). No obstante, puede tener varias. De ser así, todos los campos calificadores deben enumerarse antes que los

3 Transformar datos

campos de atributo en la sentencia LOAD y se debe usar el tercer parámetro del prefijo Crosstable para definir el número de campos calificadores. No puede tener un LOAD precedente o un prefijo delante de la palabra clave Crosstable. Pero sí se puede usar la autoconcatenación.

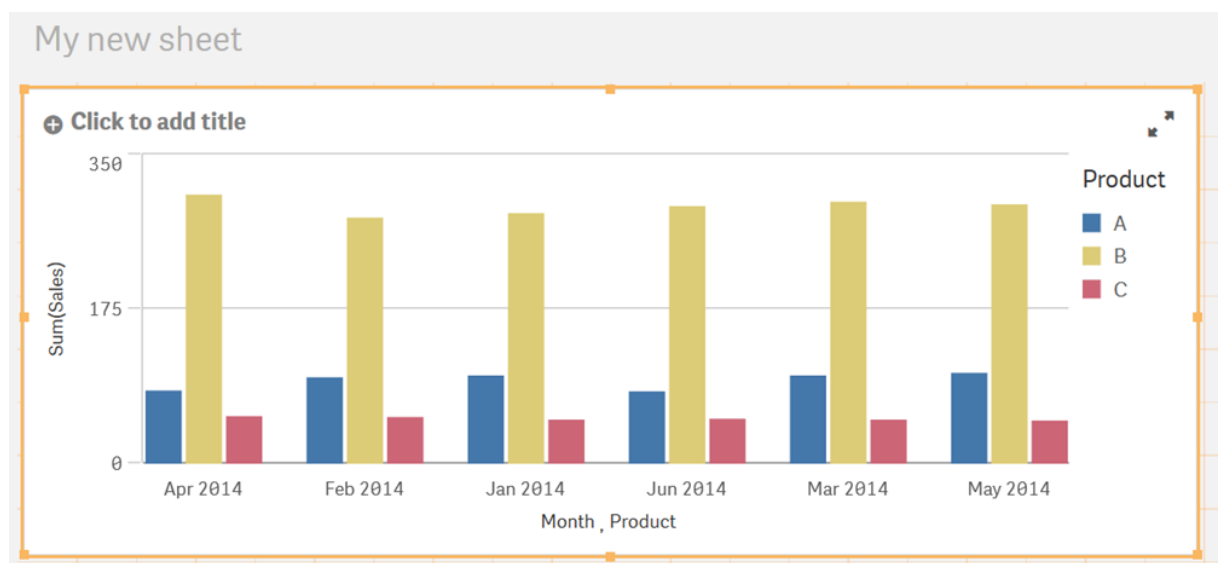
En una tabla en Qlik Sense, sus datos presentarán el siguiente aspecto:

Tabla que muestra unos datos cargados usando el prefijo Crosstable

Product	Month	Sales
A	Apr 2014	83
A	Feb 2014	98
A	Jan 2014	100
A	Jun 2014	82
A	Mar 2014	100
A	May 2014	103
B	Apr 2014	305
B	Feb 2014	279
B	Jan 2014	284
B	Jun 2014	292
B	Mar 2014	297
B	May 2014	294
C	Apr 2014	54
C	Feb 2014	53
C	Jan 2014	50

Ahora puede, por ejemplo, crear un gráfico de barras utilizando los datos:

Gráfico de barras que muestra unos datos cargados usando el prefijo Crosstable





Si desea más información sobre Crosstable, vea esta publicación de blog en Qlik Community: [La carga mediante crosstable](#). Los comportamientos se discuten en el contexto de QlikView. No obstante, la lógica se aplica igualmente a Qlik Sense.

La interpretación numérica no funcionará para los campos de atributos. Esto significa que, si tiene meses como cabeceras de columna, estos no se interpretarán automáticamente. El procedimiento es utilizar el prefijo Crosstable para crear una tabla temporal y ejecutar una segunda pasada para realizar las interpretaciones como se muestra en el siguiente ejemplo:

Tenga en cuenta que esto es solo un ejemplo. No hay ejercicios de acompañamiento que completar en Qlik Sense.

```
tmpData:
Crosstable (MonthText, Sales)
LOAD Product, [Jan 2014], [Feb 2014], [Mar 2014], [Apr 2014], [May 2014], [Jun 2014]
FROM ...

Final:
LOAD Product,
Date(Date#(MonthText, 'MMM YYYY'), 'MMM YYYY') as Month,
Sales
Resident tmpData;
Drop Table tmpData;
```

Borrado de la caché de la memoria

Puede eliminar las tablas que cree para borrar la memoria caché. Cuando cargue en una tabla temporal, como la de la sección anterior, elimínala cuando ya no la necesite más. Por ejemplo:

```
DROP TABLE Table1, Table2, Table3, Table4;
DROP TABLES Table1, Table2, Table3, Table4;
```

También puede eliminar campos. Por ejemplo:

```
DROP FIELD Field1, Field2, Field3, Field4;
DROP FIELDS Field1, Field2, Field3, Field4;
DROP FIELD Field1 from Table1;
DROP FIELDS Field1 from Table1;
```

Como puede ver, las palabras clave TABLE y FIELD pueden ir en singular o en plural.

3.2 Combinar tablas con Join y Keep

Un join es una operación que usa dos tablas y las combina en una sola. Los registros de la tabla resultante son combinaciones de registros de las tablas originales, normalmente de manera que los dos registros contribuyen a que cualquier combinación en la tabla resultante tenga un valor común para uno o varios campos comunes, lo que se conoce como un natural join. En Qlik Sense se pueden efectuar joins en el script, produciendo tablas lógicas.

Es posible unir tablas ya en el script. La lógica de Qlik Sense no percibirá entonces dichas tablas como separadas, sino como el resultado de la unión (join), es decir, como si se tratara de una única tabla lógica. En algunas situaciones esto puede ser necesario, pero tiene sus inconvenientes:

- Las tablas cargadas suelen aumentar de tamaño, lo cual hace que Qlik Sense funcione a menor velocidad.
- Parte de la información podría perderse: la frecuencia (el número de registros) de la tabla original podría no estar ya disponible.

La funcionalidad de Keep, que tiene el efecto de reducir una o las dos tablas a la intersección de los datos de la tabla antes de que las tablas se almacenen en Qlik Sense, se ha diseñado para reducir el número de casos en los que es necesario usar uniones explícitas.



En esta documentación se utiliza el término unir (join) generalmente para referirse a las uniones efectuadas antes de crear las tablas internas. Sin embargo, la asociación que se realiza una vez creadas las tablas lógicas, también es un join en esencia.

Join

La manera más fácil de hacer un join es mediante la inclusión del prefijo Join en el script, el cual une una tabla interna con otra tabla designada o con la última tabla previamente creada. La unión será una unión externa, creando todas las combinaciones posibles de valores de ambas tablas.

Ejemplo:

```
LOAD a, b, c from table1.csv;  
join LOAD a, d from table2.csv;
```

La tabla interna resultante tiene los campos a, b, c y d. El número de registros difiere dependiendo de los valores de campo de las dos tablas.



Los nombres de los campos que se desea unir deberán ser exactamente iguales. El número de campos que se van a unir es arbitrario. Normalmente, las tablas deberían tener uno o varios campos en común. Si no tienen ningún campo en común, se devuelve el producto cartesiano de las tablas. También es posible tener todos los campos en común, pero en general no tiene sentido. A menos que se haya especificado el nombre de una tabla previamente cargada en la sentencia Join el prefijo Join utiliza la última tabla previamente creada. Por lo tanto, el orden de las dos sentencias no es arbitrario.

Uso de Join

El prefijo explícito Join en el lenguaje de script de Qlik Sense realiza una unión completa (full join) de las dos tablas. El resultado es una sola tabla. Tales uniones a menudo pueden dar como resultado tablas muy grandes.

Haga lo siguiente:

1. Abra la app *Tutorial de script avanzado*.
2. Agregue una nueva sección de script en el **Editor de carga de datos**.
3. Denomine a la sección *Transactions*.
4. En **Archivos adjuntos**, en el menú a la derecha, haga clic en **Seleccionar datos**.
5. Cargue y después seleccione *Transactions.csv*.



En **Nombres de campo**, asegúrese de que **Nombres de campo incluidos** está seleccionado para que se incluyan los nombres de los campos de tabla cuando cargue los datos.

6. En la ventana **Seleccionar datos de**, haga clic en **Insertar script**.
7. Cargue y a continuación seleccione *Salesman.xlsx*.
8. En la ventana **Seleccionar datos de**, haga clic en **Insertar script**.

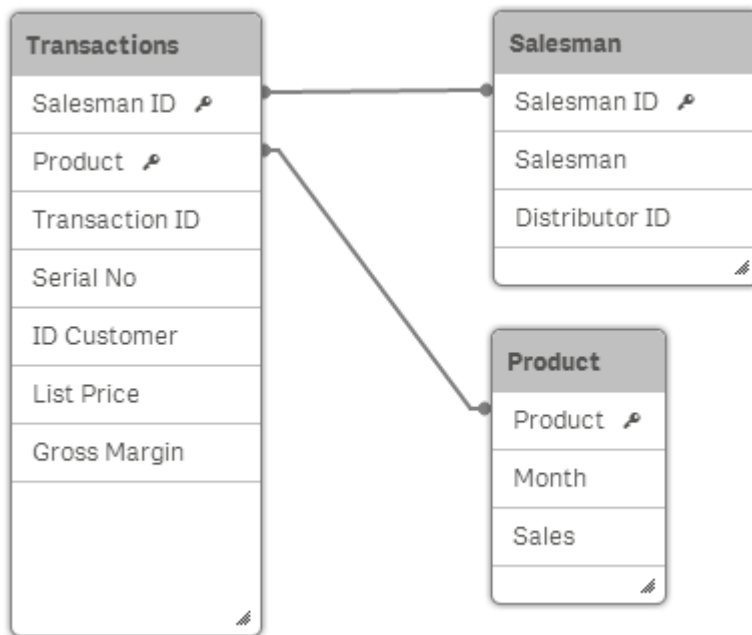
Su script debería tener el aspecto siguiente:

```
LOAD
    "Transaction ID",
    "Salesman ID",
    Product,
    "Serial No",
    "ID Customer",
    "List Price",
    "Gross Margin"
FROM [lib://AttachedFiles/Transactions.csv]
(txt, codepage is 28591, embedded labels, delimiter is ',', msq);

LOAD
    "Salesman ID",
    Salesman,
    "Distributor ID"
FROM [lib://AttachedFiles/Salesman.xlsx]
(ooxml, embedded labels, table is Salesman);
```

9. Haga clic en **Cargar datos**.
10. Abra el **Visor del modelo de datos**. El modelo de datos tendrá el siguiente aspecto:

Modelo de datos: Tablas *Transactions*, *Salesman* y *Product*



No obstante, tener las tablas *Transactions* y *Salesman* separadas puede no ser el resultado deseable. Puede que desee unir ambas tablas.

Haga lo siguiente:

1. Para dar un nombre a la tabla unida, agregue la siguiente línea sobre la primera sentencia LOAD:
Transactions:
2. Para unir las tablas *Transactions* y *Salesman*, agregue la siguiente línea encima de la segunda sentencia LOAD:
Join(Transactions)

Su script debería tener el aspecto siguiente:

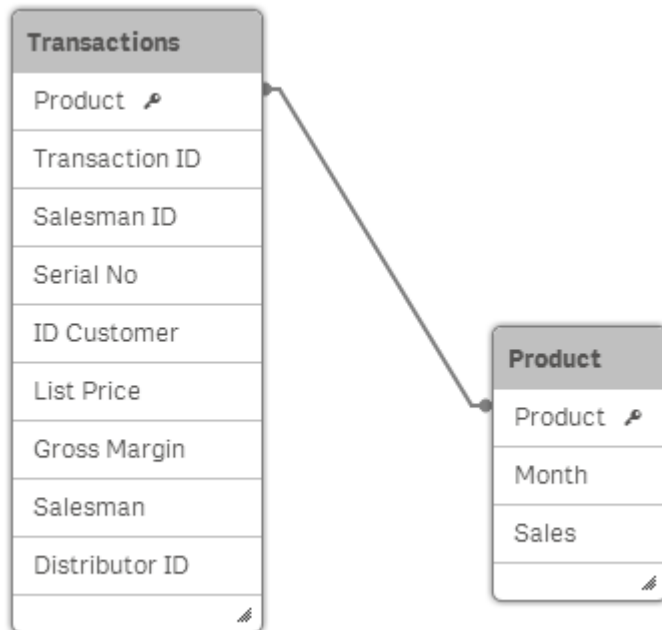
```
Transactions:
LOAD
    "Transaction ID",
    "Salesman ID",
    Product,
    "Serial No",
    "ID Customer",
    "List Price",
    "Gross Margin"
FROM [lib://AttachedFiles/Transactions.csv]
(txt, codepage is 28591, embedded labels, delimiter is ',', msq);

Join(Transactions)
LOAD
```

```
"Salesman ID",  
Salesman,  
"Distributor ID"  
FROM [lib://AttachedFiles/Salesman.xlsx]  
(ooxml, embedded labels, table is Salesman);
```

3. Haga clic en **Cargar datos**.
4. Abra el **Visor del modelo de datos**. El modelo de datos tendrá el siguiente aspecto:

Modelo de datos: Tablas Transactions y Product



Todos los campos de las tablas *Transactions* y *Salesman* están combinados ahora en una sola tabla *Transactions*.



Si desea más información sobre cuándo usar Join, vea estos blogs en Qlik Community: [To Join or not to Join](#), [Mapping as an Alternative to Joining](#). Los comportamientos se discuten en el contexto de QlikView. No obstante, la lógica se aplica igualmente a Qlik Sense.

Keep

Una de las principales características de Qlik Sense es su capacidad de hacer asociaciones entre tablas en lugar de unir las (mediante join). Esto reduce mucho espacio en memoria e incrementa la velocidad, lo que se traduce en una flexibilidad enorme. La funcionalidad de Keep se ha diseñado para reducir el número de casos en los que se tengan que usar joins explícitos.

El prefijo Keep entre dos sentencias LOAD o SELECT reduce una o las dos tablas a la intersección de los datos de la tabla antes de que se almacenen en Qlik Sense. El prefijo Keep debe ir precedido siempre de una de las palabras clave Inner, Left o Right. La selección de los registros desde las tablas se hace de la misma forma que en un join correspondiente. No obstante, las dos tablas no se unen y se almacenan en Qlik Sense como dos tablas independientes y con distintos nombres.

Inner

Los prefijos Join y Keep en el script de carga de datos pueden ir precedidos por el prefijo Inner.

Si se usa antes de Join, especifica que el join entre las dos tablas debería ser un inner join. La tabla resultante contiene solo combinaciones entre las dos tablas, con un conjunto completo de datos de ambas partes.

Si se usa antes de Keep, especifica que las dos tablas deberían reducirse a su intersección común antes de ser almacenadas en Qlik Sense.

Ejemplo:

En estos ejemplos utilizamos las tablas fuente *Table1* y *Table2*.

Tenga en cuenta que esto son únicamente ejemplos. No hay ejercicios de acompañamiento que completar en Qlik Sense.

Table 1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Inner Join

Primero hacemos un Inner Join en las tablas, lo que da como resultado a *VTable*, que contiene solo una fila, el único registro que existe en ambas tablas, con datos combinados de ambas tablas.

VTable:

```
SELECT * from Table1;  
inner join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx

Inner Keep

Si en cambio realizamos un Inner Keep, todavía tendremos dos tablas. Las dos tablas están asociadas por el campo común A.

VTab1:

```
SELECT * from Table1;
```

VTab2:

```
inner keep SELECT * from Table2;
```

VTab1

A	B
1	aa

VTab2

A	C
1	xx

Left

Los prefijos Join y Keep en el script de carga de datos pueden ir precedidos por el prefijo left.

Si se usa antes de Join, especifica que el join entre las dos tablas debería ser un left join. La tabla resultante contiene solo combinaciones entre las dos tablas, con el conjunto de datos completo de la primera tabla.

Si se usa antes de Keep, especifica que la segunda tabla debe reducirse a su intersección común con la primera tabla antes de ser almacenada en Qlik Sense.

Ejemplo:

En estos ejemplos utilizamos las tablas fuente *Table1* y *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Primero realizamos un Left Join en las tablas, dando como resultado *VTable*, que contiene todas las filas de *Table1*, combinadas con campos de las correspondientes filas en *Table2*.

VTable:

```
SELECT * from Table1;  
left join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx
2	cc	-
3	ee	-

Si en cambio realizamos un Left Keep, todavía tendremos dos tablas. Las dos tablas están asociadas por el campo común A.

VTab1:

```
SELECT * from Table1;
```

VTab2:

```
left keep SELECT * from Table2;
```

VTab1

A	B
1	aa
2	cc
3	ee

VTab2

A	C
1	xx

Right

Los prefijos Join y Keep en el lenguaje de script de Qlik Sense pueden ir precedidos por el prefijo right.

Si se usa antes de Join, especifica que el join entre las dos tablas debería ser un right join. La tabla resultante solo contiene combinaciones entre las dos tablas, con un conjunto completo de datos de la segunda tabla.

Si se usa antes de Keep, especifica que la primera tabla debe reducirse a su intersección común con la segunda tabla antes de ser almacenada en Qlik Sense.

Ejemplo:

En estos ejemplos utilizamos las tablas fuente *Table1* y *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Primero realizamos un Right Join en las tablas, dando como resultado *VTable*, que contiene todas las filas de *Table2*, combinadas con campos de las correspondientes filas en *Table1*.

VTable:

```
SELECT * from Table1;  
right join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx
4	-	yy

Si en cambio realizamos un Right Keep, todavía tendremos dos tablas. Las dos tablas están asociadas por el campo común A.

VTab1:

```
SELECT * from Table1;
```

VTab2:

```
right keep SELECT * from Table2;
```

VTab1

A	B
1	aa

VTab2

A	C
1	xx
4	yy

3.3 Uso de funciones inter-registro: Peek, Previous y Exists

Estas funciones se utilizan cuando se necesita un valor de registros de datos previamente cargados para la evaluación del registro actual.

En esta parte del tutorial, examinaremos las funciones Peek(), Previous() y Exists().

Peek()

Peek() devuelve el valor de un campo en una tabla para una fila que ya se ha cargado o que existe en la memoria interna. El número de fila se puede especificar, así como la tabla. Si no se especifica un número de fila, se utilizará el último registro cargado anteriormente.

Sintaxis:

Peek(fieldname [, row [, tablename]])

Fila debe ser un entero. 0 denota el primer registro, 1 el segundo y así sucesivamente. Los números negativos indican el orden desde el final de la tabla. -1 denota el último registro leído.

Si no se establece fila alguna, se presupone -1.

Tablename es una etiqueta de tabla que no finaliza en dos puntos. Si no se especifica *tablename*, se presupone la tabla actual. Si se utiliza fuera de la sentencia **LOAD** o se refiere a otra tabla, debe incluirse *tablename*.

Previous()

Previous() halla el valor de la expresión **expr** utilizando datos del registro de entrada anterior que no se han descartado debido a una cláusula **where**. En el primer registro de una tabla interna, la función devolverá NULL.

Sintaxis:

Previous(expression)

La función `Previous()` puede anidarse para acceder a registros anteriores. Los datos se recuperan directamente de la fuente de entrada; esto también hace posible consultar los campos que no se hayan cargado en Qlik Sense, es decir, aunque no se hayan almacenado en la base de datos asociativa.

Exists()

Exists() determina si un valor de campo específico ya se ha cargado en el campo en el script de carga de datos. La función devuelve TRUE o FALSE, así que se puede utilizar en la cláusula **where** de una sentencia **LOAD** o **IF**.

Sintaxis:

`Exists(field [, expression])`

El campo debe existir en los datos cargados hasta ahora por el script. *Expression* es una expresión que evalúa el valor del campo que buscar en el campo especificado. Si se omite, se asume el valor del registro actual en el campo especificado.

Uso de Peek() y Previous()

En su forma más simple, `Peek()` y `Previous()` sirven para identificar valores específicos en una tabla. Aquí tiene una muestra de los datos de la tabla *Employees* que cargará en este ejercicio.

Muestra de datos de la tabla *Employees*

Fecha	Contratación	Rescisión
1/1/2011	6	0
2/1/2011	4	2
3/1/2011	6	1
4/1/2011	5	2

Actualmente esto solo recopila datos de meses, contrataciones y despidos, por lo que vamos a agregar campos para *Employee Count* y *Employee Var*, usando las funciones `Peek()` y `Previous()`, para ver la diferencia por mes en el total de empleados.

Haga lo siguiente:

1. Abra la app *Tutorial de script avanzado*.
2. Agregue una nueva sección de script en el **Editor de carga de datos**.
3. Denomine a la sección *Employees*.
4. En **Archivos adjuntos**, en el menú a la derecha, haga clic en **Seleccionar datos**.
5. Cargue y después seleccione *Employees.xlsx*.



En *Field names*, asegúrese de que *Embedded field names* esté seleccionado para que se incluyan los nombres de los campos de tabla cuando cargue los datos.

6. En la ventana **Seleccionar datos de**, haga clic en **Insertar script**.

Su script debería tener el aspecto siguiente:

```
LOAD
    "Date",
    Hired,
    Terminated
FROM [lib://AttachedFiles/Employees.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

7. Modifique el script para que su aspecto sea como este:

```
[Employees Init]:
LOAD
    rowno() as Row,
    Date(Date) as Date,
    Hired,
    Terminated,
    If(rowno()=1, Hired-Terminated, peek([Employee Count], -1)+(Hired-Terminated)) as
[Employee Count]
FROM [lib://AttachedFiles/Employees.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

Las fechas en el campo *Date* de la hoja de Excel están en el formato MM/DD/YYYY. Para asegurarse de que las fechas se interpreten correctamente utilizando el formato de las variables del sistema, la función *Date* se aplica al campo *Date*.

La función *Peek()* le permite identificar cualquier valor cargado para un campo definido. En la expresión, primero veremos si *rowno()* es igual a 1. Si es igual a 1, no existirá ninguna *Employee Count*, por lo que simplemente rellenaremos el campo con la diferencia de *Hired* menos *Terminated*.

Si *rowno()* es mayor que 1, observamos el número de empleados (*Employee Count*) del mes pasado y usamos ese número para sumarlo a la diferencia de los empleados contratados (*Hired*) menos los empleados cesados (*Terminated*) de ese mes.

Tenga en cuenta también que en la función *Peek()* usamos un valor (-1). Esto indica a Qlik Sense que busque en el registro anterior al registro actual. Si el (-1) no está especificado, Qlik Sense supondrá que desea buscar en el registro anterior.

8. Agregue lo siguiente al final de su script:

```
[Employee Count]:
LOAD
    Row,
    Date,
    Hired,
    Terminated,
    [Employee Count],
    If(rowno()=1,0,[Employee Count]-Previous([Employee Count])) as [Employee Var]
Resident [Employees Init] Order By Row asc;
Drop Table [Employees Init];
```

La función `Previous()` le permite identificar el último valor cargado para un campo definido. En la expresión primero observamos si `rowno()` es igual a 1. Si es igual a 1, sabemos que no habrá ningún `Employee Var` porque no hay registro para el *Employee Count* del mes anterior. Así que simplemente introducimos 0 para el valor.

Si `rowno()` es mayor que 1, sabemos que habrá un *Employee Var*, por lo que analizaremos el *Employee Count* del mes pasado y restaremos ese número del *Employee Count* del mes actual para crear el valor en el campo *Employee Var*.

Su script debería tener el aspecto siguiente:

```
[Employees Init]:
LOAD
    rowno() as Row,
    Date(Date) as Date,
    Hired,
    Terminated,
    If(rowno()=1, Hired-Terminated, peek([Employee Count], -1)+(Hired-Terminated)) as
[Employee Count]
FROM [lib://AttachedFiles/Employees.xlsx]
(ooxml, embedded labels, table is Sheet1);

[Employee Count]:
LOAD
    Row,
    Date,
    Hired,
    Terminated,
    [Employee Count],
    If(rowno()=1,0,[Employee Count]-Previous([Employee Count])) as [Employee Var]
Resident [Employees Init] Order By Row asc;
Drop Table [Employees Init];
```

9. Haga clic en **Cargar datos**.

En una nueva hoja en la vista general de app cree una tabla usando *Date*, *Hired*, *Terminated*, *Employee Count* y *Employee Var* como columnas de la tabla. La tabla resultante deberá tener el siguiente aspecto:

Tabla que sigue el uso de Peek y Previous en el script

My new sheet

Click to add title					
Date	Sum(Hired)	Sum(Terminated)	Sum([Employee Var])	Employee Count	
Totals	77	31	40		
1/1/2011	6	0	0	6	
2/1/2011	4	2	2	8	
3/1/2011	6	1	5	13	
4/1/2011	5	2	3	16	
5/1/2011	3	2	1	17	
6/1/2011	4	1	3	20	
7/1/2011	6	2	4	24	
8/1/2011	4	1	3	27	
9/1/2011	4	0	4	31	
10/1/2011	4	0	4	35	

Peek() y Previous() le permiten apuntar a filas definidas de una tabla. La diferencia más notable entre las dos funciones es que la función Peek() permite al usuario buscar en un campo que no estaba cargado anteriormente en el script, mientras que la función Previous() solo puede buscar en un campo previamente cargado. Previous() opera en la entrada de la sentencia LOAD, mientras que Peek() opera en el resultado de la sentencia LOAD. (Igual que la diferencia entre RecNo() y RowNo().) Esto significa que las dos funciones se comportarán de manera diferente si tenemos una cláusula Where.

Por tanto, la función Previous() sería más adecuada para cuando necesite mostrar el valor actual en vez del anterior. En el ejemplo, hemos calculado la varianza entre meses.

La función Peek() sería más adecuada cuando utilice un campo que no se ha cargado previamente en la tabla o cuando necesite definir como objetivo una fila específica. Esto se mostró en el ejemplo en el que calculamos *Employee Count* analizando el *Employee Count* del mes anterior y agregando luego la diferencia entre los empleados contratados y despedidos del mes actual. Recuerde que *Employee Count* no era un campo del archivo original.



Si desea más información sobre cuándo usar Peek() y Previous(), vea esta publicación de blog en Qlik Community: [Peek\(\) vs Previous\(\) – When to Use Each](#). Los comportamientos se discuten en el contexto de QlikView. No obstante, la lógica se aplica igualmente a Qlik Sense.

Uso de Exists()

La función Exists() se utiliza a menudo con la cláusula Where en el script para cargar datos en caso de que los datos relacionados ya se hayan cargado en el modelo de datos.

En el ejemplo siguiente también usaremos la función `Dual()` para asignar valores numéricos a las cadenas.

Haga lo siguiente:

1. Cree una nueva app y asígnele un nombre.
2. Agregue una nueva sección de script en el **Editor de carga de datos**.
3. Denomine a la sección *People*.
4. Introduzca el siguiente script:

```
//Add dummy people data
PeopleTemp:
LOAD * INLINE [
  PersonID, Person
  1, Jane
  2, Joe
  3, Shawn
  4, Sue
  5, Frank
  6, Mike
  7, Gloria
  8, Mary
  9, Steven,
  10, Bill
];

//Add dummy age data
AgeTemp:
LOAD * INLINE [
  PersonID, Age
  1, 23
  2, 45
  3, 43
  4, 30
  5, 40
  6, 32
  7, 45
  8, 54
  9,
  10, 61
  11, 21
  12, 39
];

//LOAD new table with people
People:
NoConcatenate LOAD
  PersonID,
  Person
Resident PeopleTemp;

Drop Table PeopleTemp;
```

```
//Add age and age bucket fields to the People table
Left Join (People)
LOAD
    PersonID,
    Age,
    If(IsNull(Age) or Age='', Dual('No age', 5),
    If(Age<25, Dual('Under 25', 1),
    If(Age>=25 and Age <35, Dual('25-34', 2),
    If(Age>=35 and Age<50, Dual('35-49' , 3),
    If(Age>=50, Dual('50 or over', 4)
    )))) as AgeBucket
Resident AgeTemp
Where Exists(PersonID);

DROP Table AgeTemp;
```

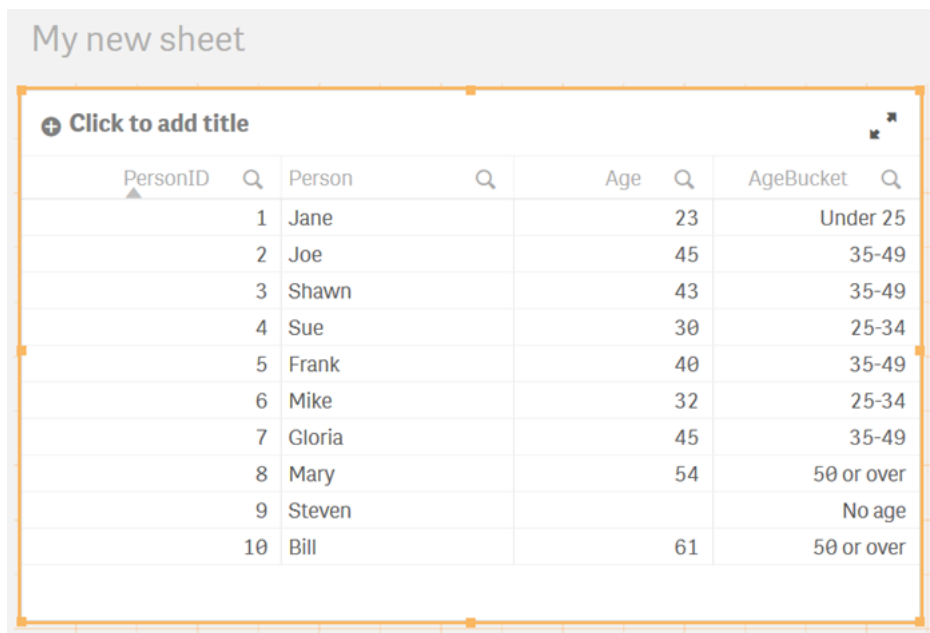
5. Haga clic en **Cargar datos**.

En el script, los campos *Age* y *AgeBucket* se cargan solo si *PersonID* ya se ha cargado en el modelo de datos.

Observe que en la tabla *AgeTemp* hay edades enumeradas para *PersonID* 11 y 12 pero como esos ID no se cargaron en el modelo de datos (en la tabla *People*), son excluidos por la cláusula *Where Exists(PersonID)*. Esta cláusula también puede escribirse así: *Where Exists (PersonID, PersonID)*.

El resultado del script presentará el siguiente aspecto:

Tabla que sigue el uso de Exists en el script



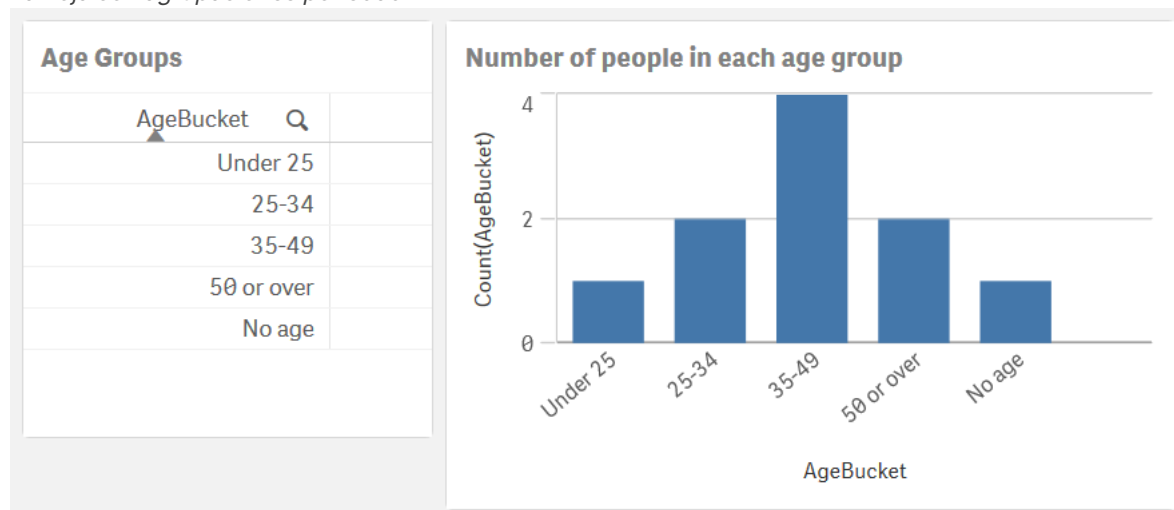
PersonID	Person	Age	AgeBucket
1	Jane	23	Under 25
2	Joe	45	35-49
3	Shawn	43	35-49
4	Sue	30	25-34
5	Frank	40	35-49
6	Mike	32	25-34
7	Gloria	45	35-49
8	Mary	54	50 or over
9	Steven		No age
10	Bill	61	50 or over

Si ninguno de los *PersonID* de la tabla *AgeTemp* se hubiera cargado en el modelo de datos, entonces los campos *Age* y *AgeBucket* no se habrían unido a la tabla *People*. Usar la función *Exists()* puede ayudar a prevenir registros/datos huérfanos en el modelo de datos, es decir, campos *Age* y *AgeBucket* que no tengan personas asociadas.

6. Cree una nueva hoja y asígnele un nombre.
7. Abra la nueva hoja y haga clic en **Editar hoja**.
8. Agregue una tabla estándar a la hoja con la dimensión *AgeBucket* y denomine a la app *Grupos de edad*.
9. Agregue un gráfico de barras a la hoja con la dimensión *AgeBucket* y la medida *Count* (*[AgeBucket]*). Denomine a la visualización *Number of people in each age group*.
10. Ajuste las propiedades de la tabla y el gráfico de barras a sus preferencias y después haga clic en **Hecho**.

Ahora debería tener una hoja similar a esta:

La hoja con agrupaciones por edad



La función *Dual()* es muy útil en el script, o en una expresión de gráfico, cuando se necesita asignar un valor numérico a una cadena.

En el script superior, tiene una aplicación que carga edades, y usted ha decidido colocarlas en rangos para poder crear visualizaciones en función de los rangos de edades frente a las edades reales. Hay un rango para personas de menos de 25, entre 25 y 35, y así sucesivamente. Utilizando la función *Dual()*, los rangos de edad se pueden asignar a un valor numérico que, posteriormente, se puede usar para ordenar los rangos en un cuadro de lista o un gráfico. Por ello, como en la hoja de la app, la ordenación dice "Sin edad" al final de la lista.



Si desea más información sobre *Exists()* y *Dual()*, vea esta publicación de blog en Qlik Community: [Dual & Exists – Useful Functions](#) (*Dual y Exists: Funciones útiles*)

3.4 Correspondencia de intervalos y carga iterativa

El prefijo `Intervalmatch` delante de una sentencia `LOAD` o `SELECT` se utiliza para enlazar valores numéricos discretos con uno o más intervalos numéricos. Es una utilidad muy potente que se puede utilizar, por ejemplo, en entornos de producción.

Uso del prefijo `IntervalMatch()`

La correspondencia de intervalos más básica es cuando tenemos una lista de números o fechas (eventos) en una tabla y una lista de intervalos en una segunda tabla. El objetivo es vincular las dos tablas. En general, es una relación de muchos a muchos, es decir, un intervalo puede tener muchas fechas que pertenecen al mismo y una fecha puede pertenecer a muchos intervalos. Para solucionar esto, debe crear una tabla puente entre las dos tablas originales. Hay varias formas de hacerlo.

La forma más sencilla de resolver este problema en Qlik Sense es usar el prefijo `IntervalMatch()` frente a una sentencia `LOAD` o `SELECT`. La sentencia `LOAD/SELECT` necesita contener dos campos únicamente, los campos "From" y "To" que definen los intervalos. El prefijo `IntervalMatch()` generará a continuación todas las combinaciones entre los intervalos cargados y un campo numérico previamente cargado, especificado como parámetro para el prefijo.

Haga lo siguiente:

1. Cree una nueva app y asígnele un nombre.
2. Agregue una nueva sección de script en el **Editor de carga de datos**.
3. Llame a las secciones *Events*.
4. En **Archivos adjuntos**, en el menú a la derecha, haga clic en **Seleccionar datos**.
5. Cargue y después seleccione *Events.txt*.
6. En la ventana **Seleccionar datos de**, haga clic en **Insertar script**.
7. Cargue y después seleccione *Intervals.txt*.
8. En la ventana **Seleccionar datos de**, haga clic en **Insertar script**.
9. En el script, denomine *Eventos* a la primera tabla y *Intervals* a la segunda tabla.
10. Al final del script agregue un `IntervalMatch` para crear una tercera tabla que una las dos primeras tablas:

```
BridgeTable:
IntervalMatch (EventDate)
LOAD distinct IntervalBegin, IntervalEnd
Resident Intervals;
```

11. Su script debería tener el aspecto siguiente:

```
Events:
LOAD
    EventID,
    EventDate,
    EventAttribute
FROM [lib://AttachedFiles/Events.txt]
```

```
(txt, utf8, embedded labels, delimiter is '\t', msq);
```

Intervals:

LOAD

```
IntervalID,  
IntervalAttribute,  
IntervalBegin,  
IntervalEnd
```

FROM [lib://AttachedFiles/Intervals.txt]

```
(txt, utf8, embedded labels, delimiter is '\t', msq);
```

BridgeTable:

IntervalMatch (EventDate)

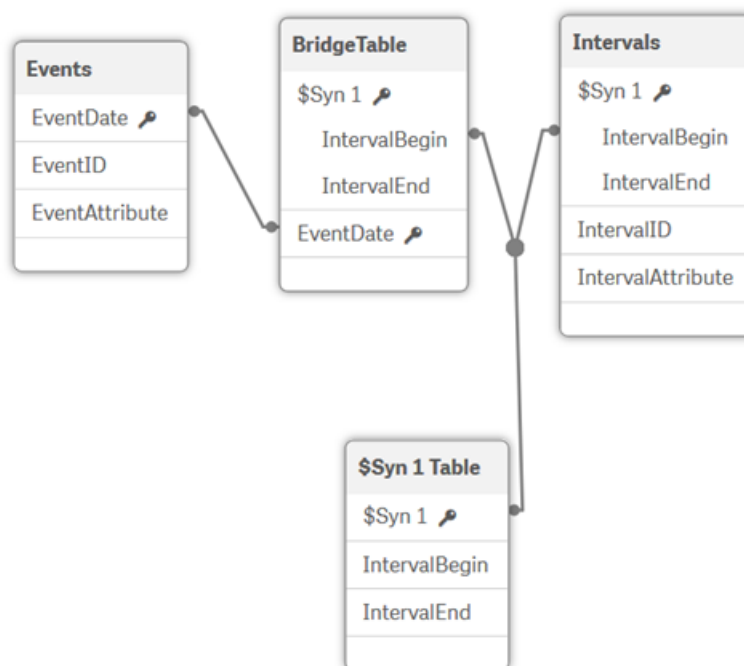
LOAD distinct IntervalBegin, IntervalEnd

Resident Intervals;

12. Haga clic en **Cargar datos**.

13. Abra el **Visor del modelo de datos**. El modelo de datos tendrá el siguiente aspecto:

Modelo de datos: Las tablas Events, BridgeTable, Intervals y \$Syn1



El modelo de datos contiene una clave compuesta (los campos *IntervalBegin* y *IntervalEnd*) que se manifestará como una clave sintética de Qlik Sense.

Las tablas básicas son:

- La tabla *Events* que contiene exactamente un registro por evento.
- La tabla *Intervals* que contiene exactamente un registro por intervalo.

- La tabla puente, que contiene exactamente un registro por combinación de evento e intervalo, y que vincula las dos tablas anteriores.

Tenga en cuenta que un evento puede pertenecer a varios intervalos si estos se solapan. Y un intervalo, por supuesto, puede tener varios eventos que pertenecen a él.

Este modelo de datos es óptimo, en el sentido de que está normalizado y es compacto. La tabla *Events* y la tabla *Intervals* se mantienen ambas sin cambios y contienen el número original de registros. Todos los cálculos de Qlik Sense que operen en estas tablas, por ejemplo, Count(EventID), funcionarán y se evaluarán de manera correcta.



Si desea más información sobre IntervalMatch(), vea este blog en Qlik Community: [Usar IntervalMatch\(\)](#)

Usar un bucle a While y carga iterativa IterNo()

Puede lograr casi la misma tabla puente utilizando un bucle While y IterNo() que crea valores enumerables entre los límites inferior y superior del intervalo.

Se puede crear un bucle dentro de la sentencia LOAD usando la cláusula While. Por ejemplo:

```
LOAD Date, IterNo() as Iteration From ... while IterNo() <= 4;
```

Dicha sentencia LOAD recorrerá en bucle cada registro de entrada y cargará este una y otra vez mientras la expresión de la cláusula While sea verdadera. La función IterNo() devuelve "1" en la primera iteración, "2" en la segunda y así sucesivamente.

Tiene una clave primaria para los intervalos, IntervalID, por lo que la única diferencia en el script será cómo se crea la tabla puente:

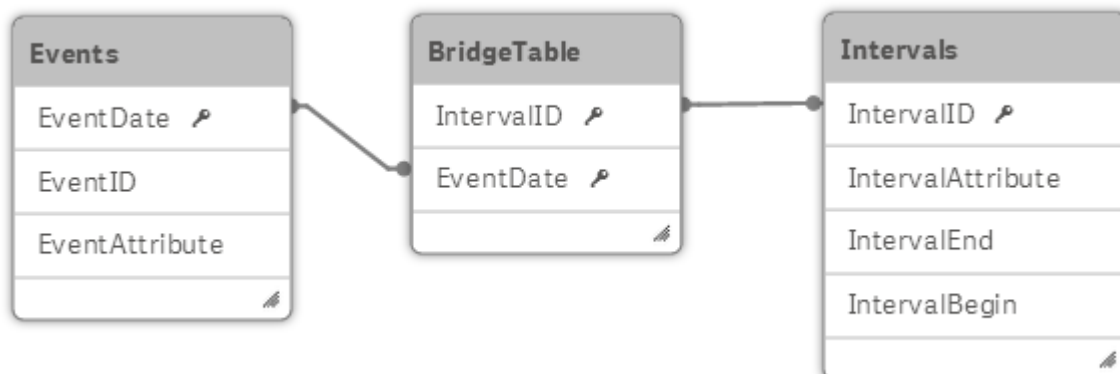
Haga lo siguiente:

1. Reemplace las sentencias Bridgetable existentes por el siguiente script:

```
BridgeTable:
LOAD distinct * where Exists(EventDate);
LOAD IntervalBegin + IterNo() - 1 as EventDate, IntervalID
  Resident Intervals
  while IntervalBegin + IterNo() - 1 <= IntervalEnd;
```

2. Haga clic en **Cargar datos**.
3. Abra el **Visor del modelo de datos**. El modelo de datos tendrá el siguiente aspecto:

Modelo de datos: Tablas Events, BridgeTable y Intervals



Generalmente la solución con tres tablas es la mejor, porque permite muchas relaciones entre intervalos y eventos. Pero una situación muy común es que sepa que un evento solo puede pertenecer a un único intervalo. En tal caso, la tabla puente no es realmente necesaria: El *IntervalID* se puede almacenar directamente en la tabla de eventos. Hay varias formas de lograrlo, pero la más útil es unir BridgeTable con la tabla *Events*.

4. Agregue el siguiente script a la parte final de su script:

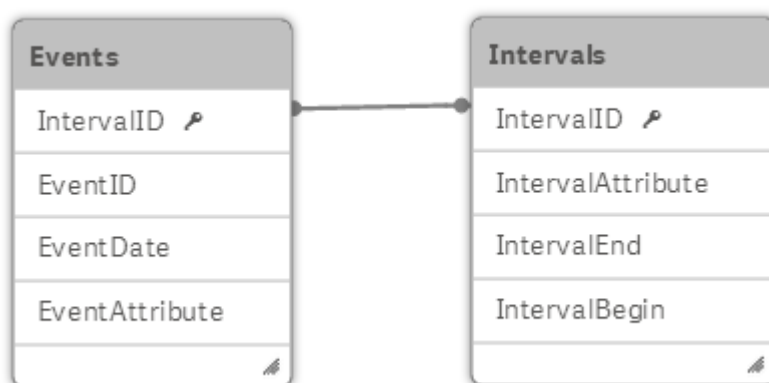
```

Join (Events)
LOAD EventDate, IntervalID
Resident BridgeTable;

Drop Table BridgeTable;
  
```

5. Haga clic en **Cargar datos**.
6. Abra el **Visor del modelo de datos**. El modelo de datos tendrá el siguiente aspecto:

Modelo de datos: Tablas Events y Intervals



Intervalos abiertos y cerrados

El hecho de estar abierto o cerrado un intervalo viene determinado por los extremos, según si están incluidos en el intervalo o no.

- Si los extremos están incluidos, se trata de un intervalo cerrado:
 $[a,b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$
- Si los extremos no están incluidos, se trata de un intervalo abierto:
 $]a,b[= \{x \in \mathbb{R} \mid a < x < b\}$
- Si un extremo está incluido, se trata de un intervalo semiabierto:
 $[a,b[= \{x \in \mathbb{R} \mid a \leq x < b\}$

Si tiene un caso en el que los intervalos se solapan y un número puede pertenecer a más de un intervalo, normalmente necesitará usar intervalos cerrados.

No obstante, en algunos casos no querrá intervalos que se solapen, sino un número que pertenezca solo a un intervalo. Por lo tanto, será un problema si un punto es el extremo de un intervalo y, a la vez, el principio del siguiente. Un número con este valor se atribuirá a ambos intervalos. Por lo tanto, desea unos intervalos semiabiertos.

Una solución práctica es restar una cantidad muy pequeña del valor final de todos los intervalos, creando de esta forma intervalos cerrados pero que no se solapen. Si sus números son fechas, la forma más sencilla de hacerlo es utilizar la función `DayEnd()` que devuelve el último milisegundo del día:

```
Intervals:  
LOAD..., DayEnd(IntervalEnd - 1) as IntervalEnd From Intervals;
```

También puede restar una pequeña cantidad de forma manual. Si lo hace, asegúrese de que la cantidad restada no sea demasiado pequeña, ya que la operación se redondeará en 52 dígitos binarios significativos (14 dígitos decimales). Si usa una cantidad demasiado pequeña, la diferencia no será significativa y volverá a usar el número original.

4 Limpieza de datos

En ocasiones los datos fuente que cargamos en Qlik Sense no tienen exactamente el formato que deseamos en nuestra app de Qlik Sense. Qlik Sense ofrece una serie de funciones y sentencias que permiten transformar nuestros datos en un formato que nos vaya bien.

La sentencia mapping se puede utilizar en un script de Qlik Sense para reemplazar o modificar valores de campo o nombres cuando se ejecuta el script, así pues mapping puede servir para limpiar los datos y hacer que sean más sistematizados o para reemplazar la totalidad o partes de un valor de campo.

Cuando cargamos datos de distintas tablas, los nombres de los valores de campo no se han asignado siempre de forma sistemática. Esta falta de sistematicidad, aparte de ser molesta, impide también las asociaciones, es decir, que se hace imprescindible resolver el problema. La forma elegante de solucionar esto es creando una tabla de correspondencias que compare los valores de campo.

4.1 Tablas de correspondencia

Las tablas cargadas mediante Mapping load o Mapping select se tratan de manera diferente a otras tablas. Se almacenan en un área aparte de la memoria, y se usan solo como tablas de enlace durante la ejecución del script. Tras ejecutarse el script estas tablas se eliminan automáticamente.

Reglas:

- Una tabla de correspondencia debe tener dos columnas, la primera con los valores de la comparación y la segunda con los valores que se desea enlazar.
- Las dos columnas deben tener nombre, pero los nombres no tienen importancia por sí mismos. Los nombres de las columnas no tienen conexión con los nombres de campo en las tablas internas regulares.

4.2 Funciones y sentencias Mapping

Este tutorial aborda el uso de las siguientes funciones/sentencias de asignación de correspondencias:

- El prefijo Mapping
- ApplyMap()
- MapSubstring()
- La sentencia Map ... Using
- La sentencia Unmap

4.3 El prefijo mapping

El prefijo Mapping se usa en un script para crear una tabla de correspondencia. La tabla de correspondencia se puede utilizar después con la función `ApplyMap()`, la función `MapSubstring()` o la sentencia `Map ... Using`.

Haga lo siguiente:

1. Cree una nueva app y asígnele un nombre.
2. Agregue una nueva sección de script en el **Editor de carga de datos**.
3. Denomine a la sección *Countries*.
4. Introduzca el siguiente script:

```
CountryMap:
MAPPING LOAD * INLINE [
Country, NewCountry
U.S.A., US
U.S., US
United States, US
United States of America, US
];
```

La tabla *CountryMap* almacena dos columnas: *Country* y *NewCountry*. La columna *Country* almacena las diversas formas en que se ha introducido country/país en el campo *Country*. La columna *NewCountry* almacena cómo se asignarán los valores. Esta tabla de correspondencia se utilizará para almacenar valores sistemáticos del país *US* en el campo *Country*. Por ejemplo, si *U.S.A.* se almacena en el campo *Country*, asígnelo como de *US*.

4.4 La función ApplyMap()

Use `ApplyMap()` para reemplazar datos de un campo basándose en una tabla de correspondencia previamente creada. La tabla de correspondencia debe cargarse previamente para poder utilizar la función `ApplyMap()`. Los datos en la tabla *Data.xlsx* que cargará se verán así:

Tabla de datos

ID	Nombre	País:	Código
1	John Black	Estados Unidos	SDFGBS1DI
2	Steve Johnson	EE.UU.	2ABC
3	Mary White	USA	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	US	KSD111DKFJ1

Observe que el país se introduce de diversas maneras. Para hacer que el campo país sea consistente, se carga la tabla de correspondencia y después se utiliza la función **`ApplyMap()`**.

Haga lo siguiente:

1. Debajo del script que introdujo arriba, seleccione y cargue *Data.xlsx*, y luego inserte el script.
2. Inserte lo siguiente sobre la sentencia LOAD recién creada:

Data:

Su script debería tener el aspecto siguiente:

```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

Data:
LOAD
    ID,
    Name,
    Country,
    Code
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

3. Modifique la línea que contiene country, de la siguiente manera:

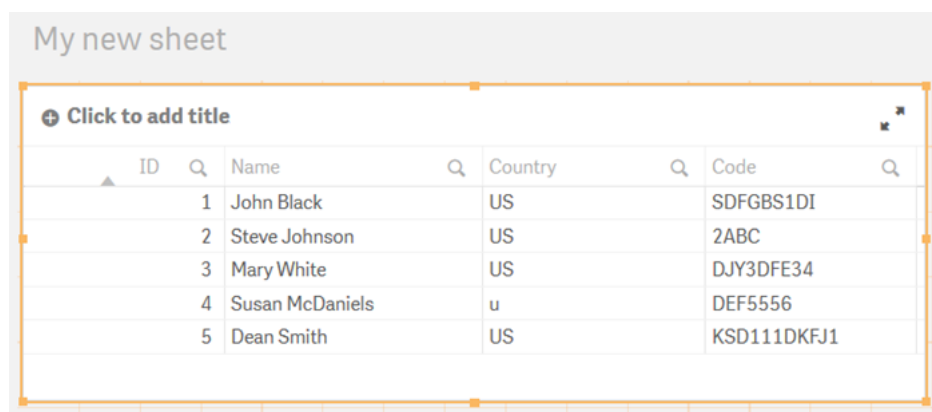
```
ApplyMap('CountryMap', Country) as Country,
```

El primer parámetro de la función ApplyMap() tiene el nombre de enlace entre paréntesis simples. El segundo parámetro es el campo que tiene los datos que se han de reemplazar.

4. Haga clic en **Cargar datos**.

La tabla resultante tiene el siguiente aspecto:

Tabla que muestra los datos cargados usando la función ApplyMap()



ID	Name	Country	Code
1	John Black	US	SDFGBS1DI
2	Steve Johnson	US	2ABC
3	Mary White	US	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	US	KSD111DKFJ1

Las diversas gráficas de *United States* se han modificado por *US*. Hay un registro que no se escribió correctamente, por lo que la función `ApplyMap()` no cambió ese valor del campo. Usando la función `ApplyMap()`, puede usar el tercer parámetro para agregar una expresión predeterminada si la tabla de correspondencia no tiene un valor coincidente.

5. Añada 'us' como el tercer parámetro de la función `ApplyMap()`, para manejar tales casos cuando el país se haya introducido incorrectamente:

`ApplyMap('CountryMap', Country, 'US') as Country,`

Su script debería tener el aspecto siguiente:

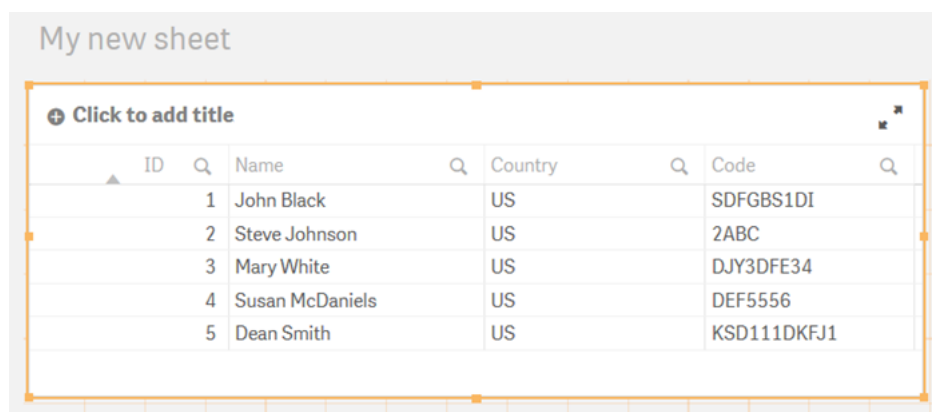
```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

Data:
LOAD
    ID,
    Name,
    ApplyMap('CountryMap', Country, 'US') as Country,
    Code
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

6. Haga clic en **Cargar datos**.

La tabla resultante tiene el siguiente aspecto:

Tabla que muestra los datos cargados usando la función `ApplyMap`



ID	Name	Country	Code
1	John Black	US	SDFGBS1DI
2	Steve Johnson	US	2ABC
3	Mary White	US	DJY3DFE34
4	Susan McDaniels	US	DEF5556
5	Dean Smith	US	KSD111DKFJ1



Si desea más información sobre `ApplyMap()`, vea este blog en Qlik Community: [No use join - use Applymap en su lugar](#)

4.5 La función MapSubstring()

La función MapSubstring() le permite asignar partes de un campo.

En la tabla creada por ApplyMap() ahora queremos que los números se escriban como texto, por lo que la función MapSubstring() se utilizará para reemplazar los datos numéricos por texto.

Para hacer esto debemos crear primero una tabla de correspondencia.

Haga lo siguiente:

1. Añada las siguientes líneas de script al final de la sección *CountryMap*, pero antes de la sección *Data*.

```
CodeMap:
MAPPING LOAD * INLINE [
F1, F2
1, one
2, two
3, three
4, four
5, five
11, eleven
];
```

En la tabla *CodeMap*, los números del 1 al 5, y 11 están enlazados.

2. En la sección *Data* del script modifique la sentencia code de la siguiente manera:
mapsubstring('CodeMap', code) as code

Su script debería tener el aspecto siguiente:

```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];
```

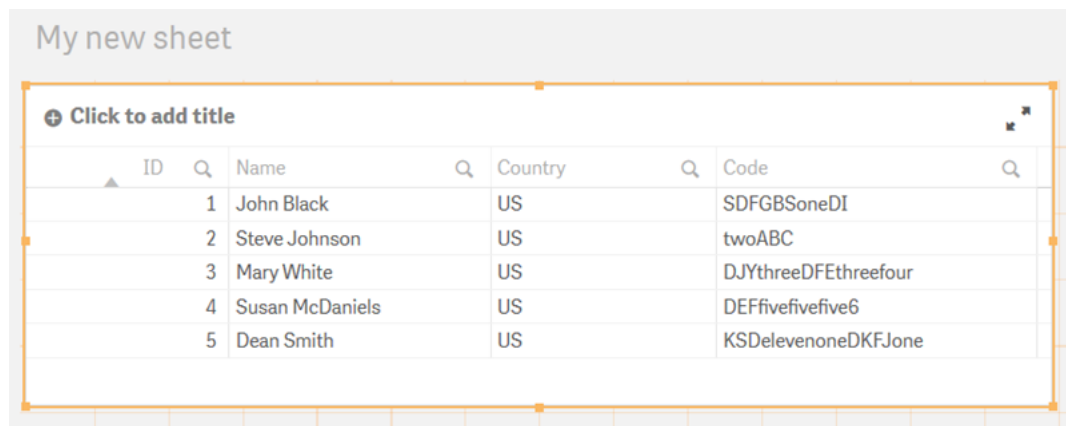
```
CodeMap:
MAPPING LOAD * INLINE [
F1, F2
1, one
2, two
3, three
4, four
5, five
11, eleven
];
```

```
Data:
LOAD
    ID,
    Name,
    ApplyMap('CountryMap', Country, 'US') as Country,
    MapSubString('CodeMap', Code) as Code
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

3. Haga clic en **Cargar datos**.

La tabla resultante tiene el siguiente aspecto:

Tabla que muestra los datos cargados usando la función MapSubString



ID	Name	Country	Code
1	John Black	US	SDFGBSoneDI
2	Steve Johnson	US	twoABC
3	Mary White	US	DJYthreeDFEthreefour
4	Susan McDaniels	US	DEffivefivefive6
5	Dean Smith	US	KSDelevenoneDKFJone

Los caracteres numéricos fueron reemplazados por texto en el campo *Code*. Si un número aparece más de una vez como es el caso de ID=3 y ID=4, el texto también se repite. ID=4. *Susan McDaniels* tenía un 6 en su código. Puesto que 6 no estaba enlazado en la tabla *CodeMap*, permanece sin modificar. ID=5, *Dean Smith*, tenía 111 en su código. Esto se ha enlazado como 'elevenone'.



Si desea más información sobre MapSubstring(), vea este blog en Qlik Community: [Mapping ... and not the geographical kind](#)

4.6 Map ... Using

La sentencia Map ... Using también se puede utilizar para aplicar un mapa a un campo. No obstante, funciona de una manera un poco diferente a ApplyMap(). Mientras que ApplyMap() gestiona la asignación cada vez que se encuentra el nombre del campo, Map ... Using trata la asignación cuando el valor se almacena bajo el nombre del campo en la tabla interna.

Echemos un vistazo a un ejemplo. Supongamos que estábamos cargando el campo *Country* varias veces en el script y queríamos aplicar un mapa cada vez que se cargaba el campo. La función ApplyMap() se puede usar tal como se ilustra anteriormente en este tutorial o se puede usar Map ... Using.

Si usamos Map ... Using entonces el mapa se aplica al campo cuando el campo se almacena en la tabla interna. Así pues, en el ejemplo a continuación, el mapa se aplica al campo *Country* de la tabla *Data1* pero no se aplicaría al campo *Country2* de la tabla *Data2*. Esto se debe a que la sentencia Map ... Using únicamente se aplica a campos denominados *Country*. Cuando el campo *Country2* se almacena en la tabla interna deja de llamarse *Country*. Si desea que el mapa se aplique a la tabla *Country2*, debe utilizar la función `ApplyMap()`.

La sentencia `Unmap` pone fin a la sentencia `Map ... Using`, de modo que si se fuera a cargar *Country* detrás de la sentencia `Unmap`, *CountryMap* no se aplicaría.

Haga lo siguiente:

1. Reemplace el script de la tabla *Data* con lo siguiente:

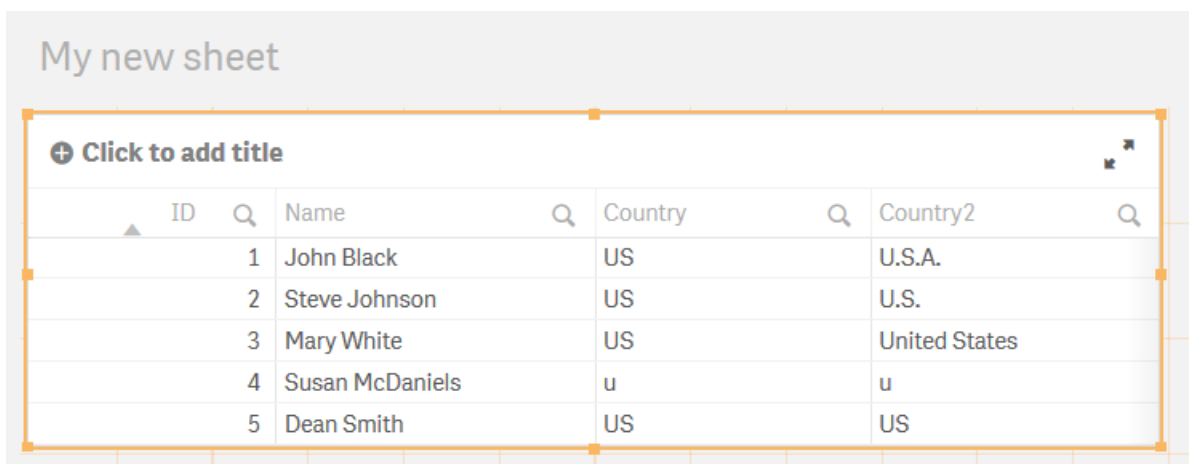
```
Map Country Using CountryMap;
Data1:
    LOAD
        ID,
        Name,
        Country
    FROM [lib://AttachedFiles/Data.xlsx]
    (ooxml, embedded labels, table is Sheet1);

Data2:
    LOAD
        ID,
        Country as Country2
    FROM [lib://AttachedFiles/Data.xlsx]
    (ooxml, embedded labels, table is Sheet1);
UNMAP;
```

2. Haga clic en **Cargar datos**.

La tabla resultante tiene el siguiente aspecto:

Tabla que muestra los datos cargados usando la función Map ... Using



ID	Name	Country	Country2
1	John Black	US	U.S.A.
2	Steve Johnson	US	U.S.
3	Mary White	US	United States
4	Susan McDaniels	u	u
5	Dean Smith	US	US

5 Manejo de datos jerárquicos

Las jerarquías son una parte importante de cualquier solución de business intelligence; se utilizan para describir dimensiones que contienen de forma natural diferentes niveles de granularidad. Algunas son simples e intuitivas, mientras que otras son complejas y exigen pensar mucho hasta poder modelarlas correctamente.

Desde la parte superior de una jerarquía hasta la parte inferior, los miembros son cada vez más detallados y específicos. Por ejemplo, en una dimensión que contenga los niveles Mercado, País, Estado y Ciudad, el miembro América aparece en la parte superior de la jerarquía, el miembro U.S.A. aparece en el segundo nivel, el miembro California aparece en el tercer nivel y San Francisco en el nivel más inferior. California es más específico que U.S.A. y San Francisco es más específico que California.

Almacenar las jerarquías en un modelo relacional es un desafío común con múltiples soluciones. Hay distintos enfoques:

- La jerarquía horizontal
- El modelo de lista adyacente
- El método de enumeración de rutas
- El modelo de conjuntos anidados
- El listado de antepasados

Para lo que nos interesa en este tutorial, crearemos un listado de antepasados ya que presenta la jerarquía de una forma que es directamente utilizable en una consulta. Puede encontrar más información sobre los demás enfoques en Qlik Community.

5.1 El prefijo Hierarchy

El prefijo Hierarchy es un comando de script que se sitúa frente a una sentencia LOAD o SELECT que carga una tabla de nodos adyacentes. La sentencia LOAD necesita tener al menos tres campos: Un ID que es una clave única para el nodo, una referencia al padre y un nombre.

El prefijo transformará una tabla cargada en una tabla de nodos expandidos; una tabla que tenga varias columnas añadidas, una por cada nivel de la jerarquía.

Haga lo siguiente:

1. Cree una nueva app y asígnele un nombre.
2. Agregue una nueva sección de script en el **Editor de carga de datos**.
3. Denomine a la sección *Wine*.
4. En **Archivos adjuntos**, en el menú a la derecha, haga clic en **Seleccionar datos**.
5. Cargue y después seleccione *Winedistricts.txt*.
6. En la ventana **Seleccionar datos desde**, desmarque los campos *Lbound* y *RBound* para que no se carguen.

7. Haga clic en **Insertar script**.
8. Inserte lo siguiente encima de la sentencia LOAD:
Hierarchy (NodeID, ParentID, NodeName)

Su script debería tener el aspecto siguiente:

```
Hierarchy (NodeID, ParentID, NodeName)
LOAD
    NodeID,
    ParentID,
    NodeName
FROM [lib://AttachedFiles/winedistricts.txt]
(txt, utf8, embedded labels, delimiter is '\t', msq);
```

9. Haga clic en **Cargar datos**.
10. Utilice la sección **Vista previa** del **Visor del modelo de datos** para ver la tabla resultante.
La tabla de nodos expandidos resultante tiene exactamente el mismo número de registros que su tabla de origen: Uno por nodo. La tabla de nodos expandidos es muy práctica puesto que cumple diversos requisitos para analizar una jerarquía en un modelo relacional:
 - Todos los nombres de nodos existen en una misma columna, de modo que esto puede utilizarse para hacer búsquedas.
 - Además, los distintos niveles de nodos se han expandido en un campo cada uno; los campos que se pueden utilizar en grupos jerárquicos o como dimensiones en tablas pivotantes.
 - Además, los distintos niveles de nodos se han expandido en un campo cada uno; los campos que se pueden utilizar en grupos jerárquicos.
 - Se puede hacer que contenga una ruta única al nodo, enumerando todos los antepasados en el orden correcto.
 - Se puede hacer que contenga la profundidad del nodo, es decir, la distancia a la base.

La tabla resultante tiene el siguiente aspecto:

Tabla que muestra una muestra de datos cargados usando el prefijo Hierarchy

My new sheet										
NodeID	ParentID	NodeName	NodeName1	NodeName2	NodeName3	NodeName4	NodeName5	NodeName6		
289	288	Bas-Médoc	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc		
290	289	Listrac	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc		
291	289	Paulliac	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc		
292	289	Saint-Estèphe	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc		
293	289	Saint-Julien	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc		
294	288	Haut-Médoc	The World	Europe	France	Bordeaux	Médoc	Haut-Médoc		
295	294	Margaux	The World	Europe	France	Bordeaux	Médoc	Haut-Médoc		

5.2 El prefijo HierarchyBelongsTo

Al igual que el prefijo Hierarchy, el prefijo HierarchyBelongsTo es un comando de script que se coloca frente a una sentencia LOAD o SELECT que carga una tabla de nodos adyacentes.

También aquí la sentencia LOAD necesita tener al menos tres campos: Un ID que es una clave única para el nodo, una referencia al padre y un nombre. El prefijo transformará la tabla cargada en una tabla de antepasados; una tabla que tiene cada combinación de un antepasado y un descendiente listados como registros aparte. Esto hace que sea muy fácil hallar todos los antepasados o todos los descendientes de un nodo específico.

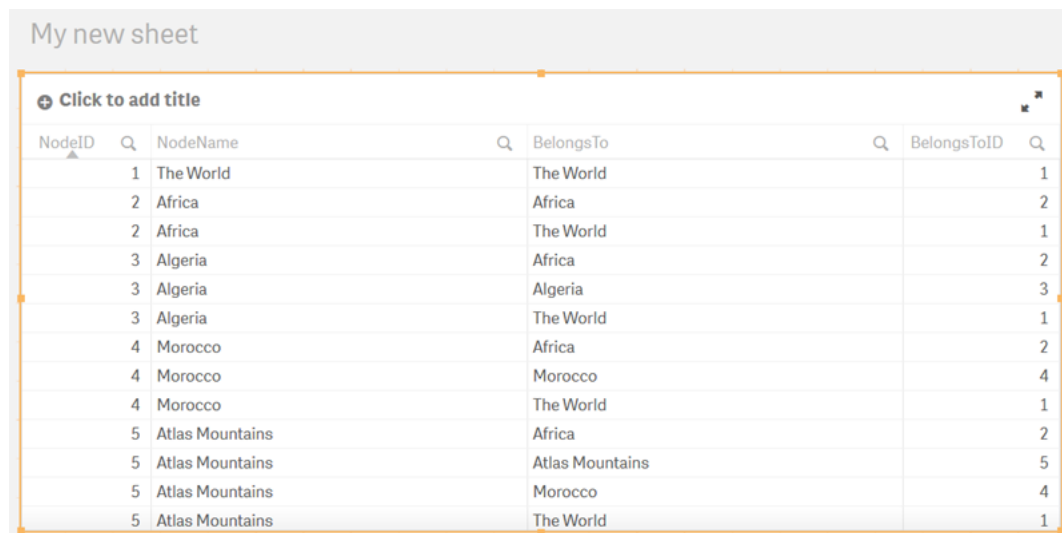
Haga lo siguiente:

1. Modifique la sentencia Hierarchy en el **Editor de carga de datos** de modo que se lea lo siguiente:
`HierarchyBelongsTo (NodeID, ParentID, NodeName, BelongsToID, BelongsTo)`
2. Haga clic en **Cargar datos**.
3. Utilice la sección **Vista previa** del **Visor del modelo de datos** para ver la tabla resultante.
La tabla de antepasados cumple diversos requisitos para analizar una jerarquía en un modelo relacional:

- Si el ID de nodo representa los nodos únicos, el ID de antepasado representa los árboles y subárboles completos de la jerarquía.
- Todos los nombres de nodos existen tanto en su rol de nodos, como en su rol de árboles, y ambos pueden utilizarse para hacer búsquedas.
- Se puede hacer que contengan la diferencia de profundidad entre la profundidad de nodo y la profundidad de antepasado, esto es, la distancia desde la raíz del subárbol.

La tabla resultante tiene el siguiente aspecto:

Tabla que muestra unos datos cargados usando el prefijo HierarchyBelongsTo



NodeID	NodeName	BelongsTo	BelongsToID	
1	The World	The World	1	
2	Africa	Africa	2	
2	Africa	The World	1	
3	Algeria	Africa	2	
3	Algeria	Algeria	3	
3	Algeria	The World	1	
4	Morocco	Africa	2	
4	Morocco	Morocco	4	
4	Morocco	The World	1	
5	Atlas Mountains	Africa	2	
5	Atlas Mountains	Atlas Mountains	5	
5	Atlas Mountains	Morocco	4	
5	Atlas Mountains	The World	1	

Autorización

Es bastante frecuente que las jerarquías se utilicen para dar autorizaciones. Un ejemplo lo tendríamos en la estructura laboral jerárquica de una organización. Cada director tiene derecho a ver todo lo que pertenece a su propio departamento, incluidos todos sus subdepartamentos. Pero no

necesariamente tendrían derecho a ver otros departamentos.

Ejemplo de jerarquía de una organización



Esto significa que a diferentes personas se les permitirá ver diferentes subárboles de la organización. La tabla de autorización tendrá el siguiente aspecto:

Tabla de autorización

ACCESO	NTNAME	PERSONA	PUESTO	PERMISOS
USUARIO	ACME\JRL	John	Director de producto	RRHH
USUARIO	ACME\CAH	Carol	Director general	Director general
USUARIO	ACME\JER	James	Director de ingeniería	Ingeniería
USUARIO	ACME\DBK	Diana	Director financiero	Financiero
USUARIO	ACME\RNL	Bob	Director de operaciones	Ventas
USUARIO	ACME\LFD	Larry	Director de sistemas informáticos	Producto

En este caso, a *Carol* se le permite ver todo lo que pertenece al *CEO* y por debajo; a *Larry* se le permite ver la organización de *Product*; y a *James* se le permite ver solo la organización de *Engineering*.

Ejemplo:

A menudo la jerarquía se almacena en una tabla de nodos adyacentes. En este ejemplo, para resolver esto cargue la tabla de nodos adyacentes utilizando un `HierarchyBelongsTo` y denomine al campo del ancestro *Tree*.

Si desea usar Section Access, cargue una copia en mayúsculas de *Tree* y llame a este nuevo campo *PERMISSIONS*. Por último, necesita cargar la tabla de autorizaciones. Estos dos últimos pasos se pueden hacer usando las siguientes líneas de script. Observe que la tabla `TempTrees` es la tabla creada por la sentencia `HierarchyBelongsTo`.

Tenga en cuenta que esto es solo un ejemplo. No hay ejercicios de acompañamiento que completar en Qlik Sense.

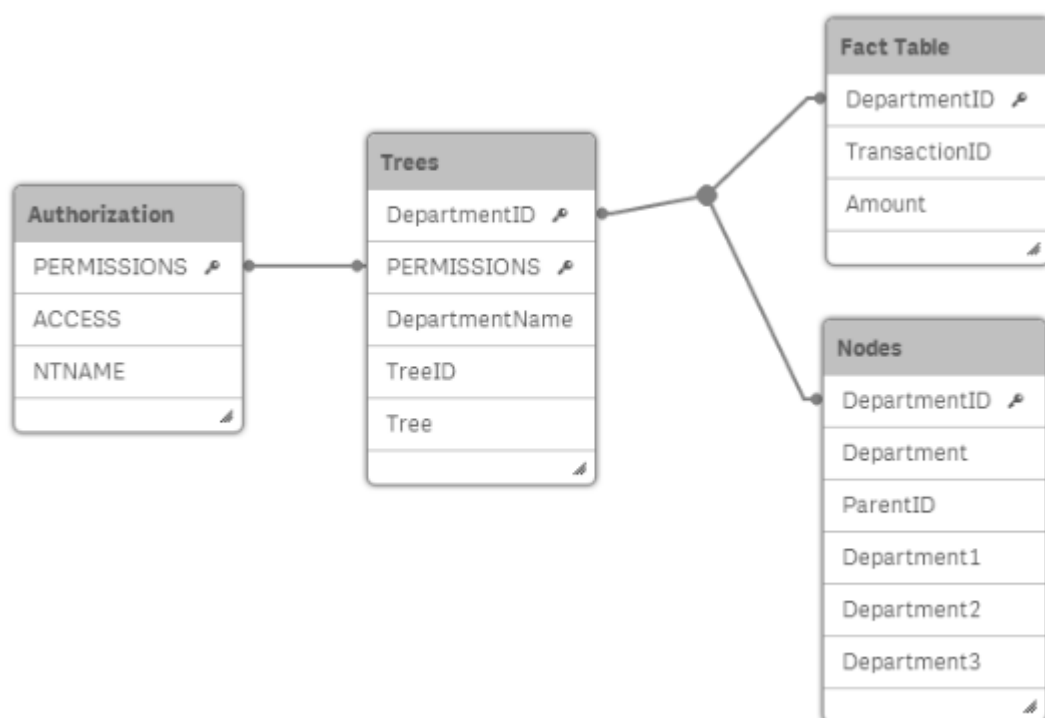
```
Trees:
LOAD *,
    Upper(Tree) as PERMISSIONS
    Resident TempTrees;
Drop Table TempTrees;

Section Access;
Authorization:
LOAD  ACCESS,
      NTNAME,
      UPPER(Permissions) as PERMISSIONS
From Organization;
Section Application;
```

Este ejemplo produciría el siguiente modelo de datos:

5 Manejo de datos jerárquicos

Modelo de datos: Las tablas Authorization, Trees, Fact y Nodes



6 Archivos QVD.

Un archivo QVD (QlikView Data) es un archivo que contiene una tabla de datos exportados desde Qlik Sense o QlikView. QVD es un formato nativo de Qlik y solo pueden leerlo y escribirlo Qlik Sense o QlikView. El formato de archivo está optimizado para mejorar la velocidad de lectura de datos desde un script de Qlik Sense, siendo al mismo tiempo muy compacto. Leer datos desde un archivo QVD es por lo general 10-100 veces más rápido que leer de otras fuentes de datos.

Los archivos QVD se puede leer en dos modos: estándar (rápido) y optimizado (más rápido). El modo seleccionado lo determina de forma automática el motor de script de Qlik Sense. El modo optimizado puede emplearse solo cuando todos los campos cargados o un subconjunto de ellos sean leídos sin ninguna transformación (sin fórmulas que actúen sobre los campos), aunque sí se permite el renombrado de campos. Una cláusula Where que hace que Qlik Sense descomprima los registros también desactivará la carga optimizada.

Un archivo QVD contiene exactamente una tabla de datos y consta de tres partes:

- Una cabecera XML (con juego de caracteres UTF-8) que describe los campos de la tabla, el diseño de la información subsiguiente y algunos otros metadatos.
- Tablas de símbolos en un formato de bytes.
- Datos reales de la tabla en formato de bits.

Los archivos QVD se pueden usar con muchos fines. Podemos identificar claramente al menos cuatro usos fundamentales. En muchos casos se pueden aplicar dos o más de ellos al mismo tiempo.

- Incremento de la velocidad de carga de datos
Al almacenar en búfer los bloques de datos de entrada que no cambian o que cambian lentamente en los archivos QVD, la ejecución de script se vuelve considerablemente más rápida para los grandes conjuntos de datos.
- Reducción de la carga en los servidores de las bases de datos
Mediante el envío a buffers de las partes que no cambian, o cambian muy lentamente, de los datos de entrada de archivos QVD, podemos reducir enormemente la cantidad de datos obtenidos de fuentes de datos externas. Esto alivia la carga de las bases de datos externas y reduce el tráfico de la red. Además, cuando varios scripts de Qlik Sense comparten los mismos datos, solo es necesario cargarlo una vez desde la base de datos de origen en un archivo QVD. Las otras apps pueden hacer uso de los mismos datos a través de este archivo QVD.
- Consolidar datos de múltiples aplicaciones Qlik Sense
Con la sentencia de script Binary, se pueden cargar datos desde una sola aplicación Qlik Sense a otra, pero con los archivos QVD, un script de Qlik Sense puede combinar datos de cualquier cantidad de aplicaciones Qlik Sense. Esto abre todo un mundo de posibilidades, por ejemplo para aplicaciones que deseen consolidar datos similares procedentes de diversas unidades de negocio, etc.

- Carga incremental

En muchos casos habituales, la funcionalidad del QVD puede usarse para la carga incremental, de modo que solo se carguen los registros nuevos de una base de datos en aumento.



Para ver cómo utiliza Automatización de aplicaciones de Qlik la comunidad de Qlik para mejorar los tiempos de carga de los QVD, vea [Cómo dividir archivos QVD usando una automatización para mejorar las recargas](#).

6.1 Crear archivos QVD

Un archivo QVD se puede crear de dos maneras:

- Mediante la creación explícita y su nombrado mediante el comando Store en el script de Qlik Sense.

Indique simplemente en el script que desea exportar una tabla anteriormente leída, o parte de ella, a un nombre de archivo explícito, en una ubicación de su elección.

- Mediante creación y mantenimiento automáticos desde el script.

Mediante la colocación del prefijo Buffer delante de una sentencia de carga o selección, Qlik Sense creará automáticamente un archivo QVD, que, bajo ciertas condiciones, se puede utilizar en lugar de la fuente de datos original al volver a cargar los datos.

No hay diferencia entre los archivos QVD resultantes en cuanto a la velocidad de lectura.

Store

Esta sentencia de script crea un nombre explícitamente denominado QVD, CSV o archivo txt.

Sintaxis:

```
store[ *fieldlist from] table into filename [ format-spec ];
```

La sentencia solo puede exportar campos desde una tabla de datos. Si tuviéramos que exportar campos de varias tablas, debemos hacer previamente un join explícito en el script para crear la tabla de datos que se ha de exportar.

Los valores de texto se exportan al archivo CSV en formato UTF-8. Se puede especificar un delimitador, vea **LOAD**. La sentencia de almacenamiento a un archivo CSV no admite exportación BIFF.

Ejemplos:

```
store mytable into [lib://AttachedFiles/xyz.qvd];
store * from mytable into [lib://FolderConnection/xyz.qvd];
store myfield from mytable into 'lib://FolderConnection/xyz.qvd';
store myfield as renamedfield, myfield2 as renamedfield2 from mytable into
[lib://AttachedFiles/xyz.qvd];
```



```
Store mytable into 'lib://FolderConnection/myfile.txt';
Store * from mytable into 'lib://FolderConnection/myfile.csv';
```

Haga lo siguiente:

1. Abra la app *Tutorial de script avanzado*.
2. Haga clic en la sección de script *Product*.
3. Agregue lo siguiente al final del script:

```
Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
```

Su script debería tener el aspecto siguiente:

```
CrossTable(Month, Sales)
LOAD
    Product,
    "Jan 2014",
    "Feb 2014",
    "Mar 2014",
    "Apr 2014",
    "May 2014"
FROM [lib://AttachedFiles/Product.xlsx]
(ooxml, embedded labels, table is Product);

Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
```

4. Haga clic en **Cargar datos**.

Ahora, el archivo *Product.qvd* debería estar en la lista de archivos.

Este archivo de datos, que es el resultado del script de **Crosstable**, es una tabla de tres columnas, una por cada categoría (Product, Month, Sales). Ahora, este archivo de datos se puede utilizar para sustituir toda la sección de script *Product*.

6.2 Leer datos desde archivos QVD

Un archivo QVD se puede leer o está accesible a Qlik Sense mediante los siguientes métodos:

- Cargar un archivo QVD como una fuente de datos explícita. Los archivos QVD se pueden referenciar mediante una sentencia load en el script de Qlik Sense, igual que cualquier otro tipo de archivos de texto (csv, fix, dif, biff, etc.).

Ejemplos:

```
LOAD * from 'lib://FolderConnection/xyz.qvd' (qvd);
LOAD fieldname1, fieldname2 from [lib://FolderConnection/xyz.qvd] (qvd);
LOAD fieldname1 as newfieldname1, fieldname2 as newfieldname2 from
[lib://AttachedFiles/xyz.qvd] (qvd);
```

- Carga automática de archivos QVD almacenados en el búfer. Cuando se utiliza el prefijo buffer en sentencias load o select, no se necesitan sentencias explícitas para la lectura. Qlik

Sense determinará en qué medida utilizará los datos del archivo QVD en lugar de adquirir los datos utilizando la sentencia LOAD o SELECT original.

- Acceder a archivos QVD desde el script. Se pueden usar varias funciones de script (todas ellas comenzando por QVD) para recuperar información diversa sobre los datos encontrados en la cabecera XML de un archivo QVD.

Haga lo siguiente:

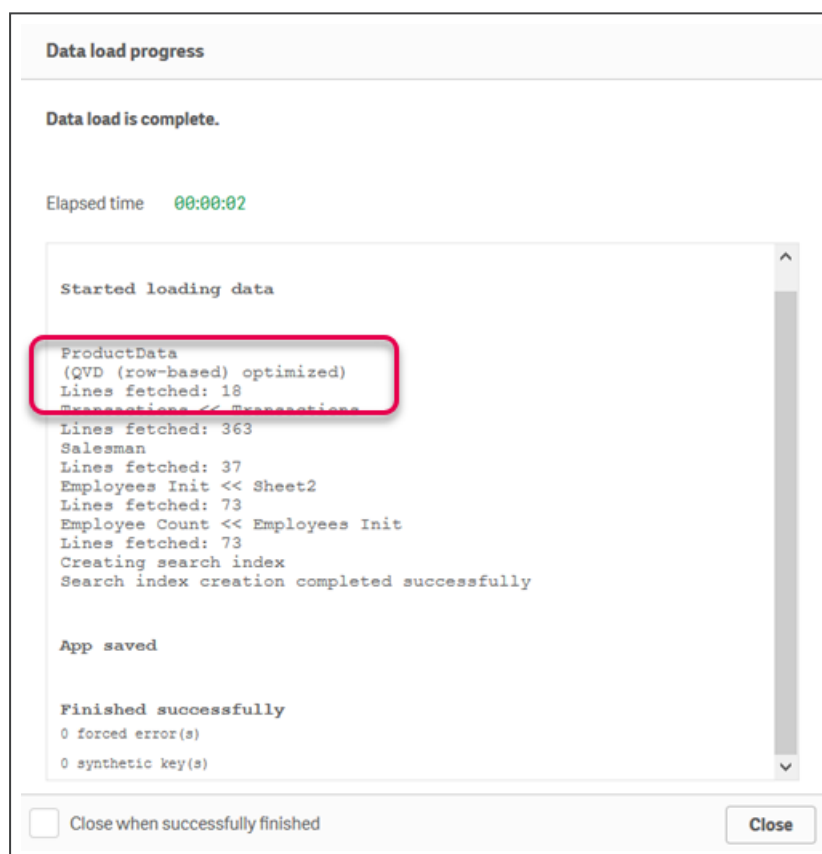
1. Comente todo el script completo en la sección de script *Product*.
2. Introduzca el siguiente script:

```
Load * from [lib://AttachedFiles/ProductData.qvd](qvd);
```

3. Haga clic en **Cargar datos**.

Los datos se cargan desde el archivo QVD.

Ventana de progreso de la carga de datos



Para obtener información sobre el uso de archivos QVD para cargas incrementales, vea esta publicación de blog en Qlik Community: [Vista general de la carga incremental de Qlik](#)

Buffer

Los archivos QVD se pueden crear y mantener automáticamente a través del prefijo Buffer. Este prefijo se puede utilizar en la mayoría de las sentencias LOAD y SELECT de scripts. Indica que los archivos QVD se utilizan para almacenar en caché/búfer el resultado de la sentencia.

Sintaxis:

```
Buffer [ (option [ , option])] ( loadstatement | selectstatement )  
      option::= incremental | stale [after] amount [(days | hours)]
```

Si no se utiliza ninguna opción, el búfer de QVD que crea la primera ejecución del script se utilizará de forma indefinida.

Ejemplo:

```
Buffer load * from MyTable;
```

stale [after] amount [(days | hours)]

Amount es un número que especifica el período de tiempo. Se pueden utilizar Decimals. Si se omite la unidad se interpreta como days.

La opción de stale after se utiliza normalmente con fuentes de bases de datos donde no hay ninguna fecha-hora sencilla en los datos originales. La cláusula stale after simplemente establece un período de tiempo a partir de la hora de creación del búfer de QVD, tras el cual se dejará de considerar válido. Antes de esa hora, el búfer de QVD se utilizará como origen de los datos, tras lo cual se utilizará el origen de datos. El archivo del búfer de QVD se actualizará automáticamente y se iniciará un nuevo período.

Ejemplo:

```
Buffer (stale after 7 days) load * from MyTable;
```

Incremental

La opción de incremental activa la capacidad de leer solo parte de un archivo subyacente. El tamaño anterior del archivo se almacena en el encabezado XML en el archivo QVD. Esto resulta de especial utilidad con archivos de registro. Todos los registros cargados en alguna ocasión anterior se leen desde el archivo de QVD, mientras que los registros nuevos siguientes se leen desde el origen original y, por último, se crea un archivo de QVD actualizado.

Tenga en cuenta que la opción de carga incremental solo se puede usar con sentencias LOAD y archivos de texto y que la carga incremental no se puede usar allí donde los datos antiguos se cambien o se eliminen.

Ejemplo:

```
Buffer (incremental) load * from MyLog.log;
```

Los búferes de QVD se quitarán normalmente cuando ya no se mencionen en ninguna parte a lo largo de una ejecución completa del script de la app que lo ha creado o cuando la app que lo ha creado ya no exista. Debe utilizar la sentencia Store si desea conservar el contenido del búfer como un QVD o un archivo CSV.

Haga lo siguiente:

1. Cree una nueva app y asígnele un nombre.
2. Agregue una nueva sección de script en el **Editor de carga de datos**.
3. En **Archivos adjuntos**, en el menú a la derecha, haga clic en **Seleccionar datos**.
4. Cargue y después seleccione *Cutlery.xlsx*.
5. En la ventana **Seleccionar datos de**, haga clic en **Insertar script**.
6. Comente los campos de la sentencia load y cambie la sentencia load a lo siguiente:

Buffer LOAD *

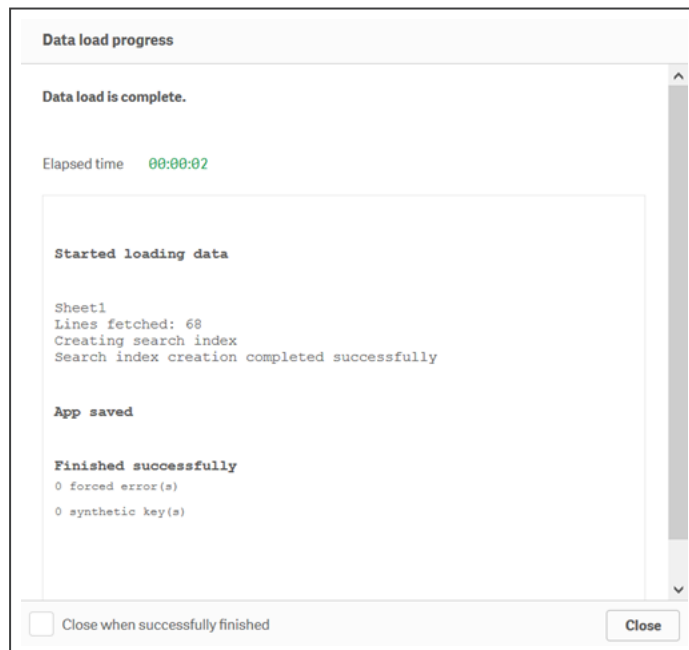
Su script debería tener el aspecto siguiente:

```
Buffer LOAD *
//      "date",
//      item,
//      quantity
FROM [lib://AttachedFiles/Cutlery.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

7. Haga clic en **Cargar datos**.

La primera vez que cargue datos, estos se cargan desde *Cutlery.xlsx*.

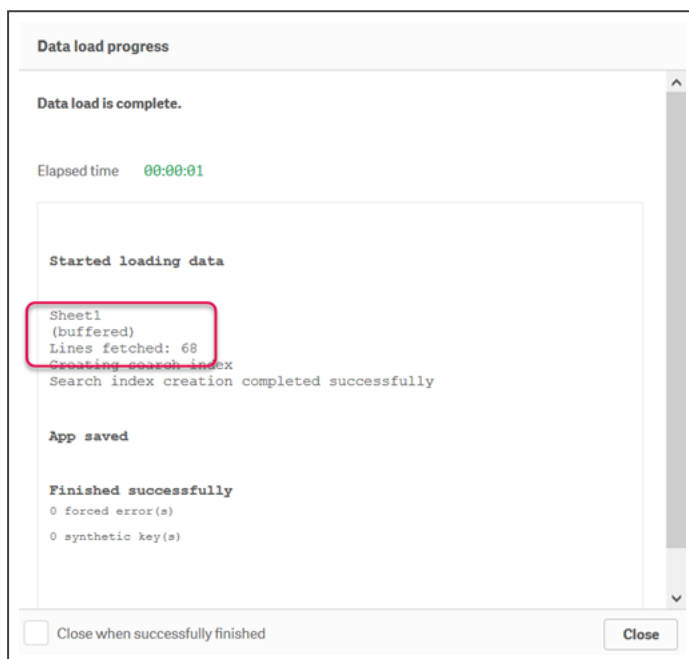
Ventana de progreso de la carga de datos



La sentencia Buffer también crea un archivo QVD y lo almacena en Qlik Sense. En una implementación de Qlik Sense Enterprise on Windows se almacena en un directorio en el servidor de Qlik Sense.

8. Haga clic en **Cargar datos** de nuevo.
9. Esta vez los datos se cargan desde el archivo QVD creado por la sentencia Buffer cuando cargó los datos por primera vez.

Ventana de progreso de la carga de datos



6.3 ¡Muchas gracias!

Ha completado este tutorial, ahora ya posee los conocimientos básicos para crear scripts en Qlik Sense. Por favor, visite nuestro sitio web si desea más información sobre otra formación disponible.