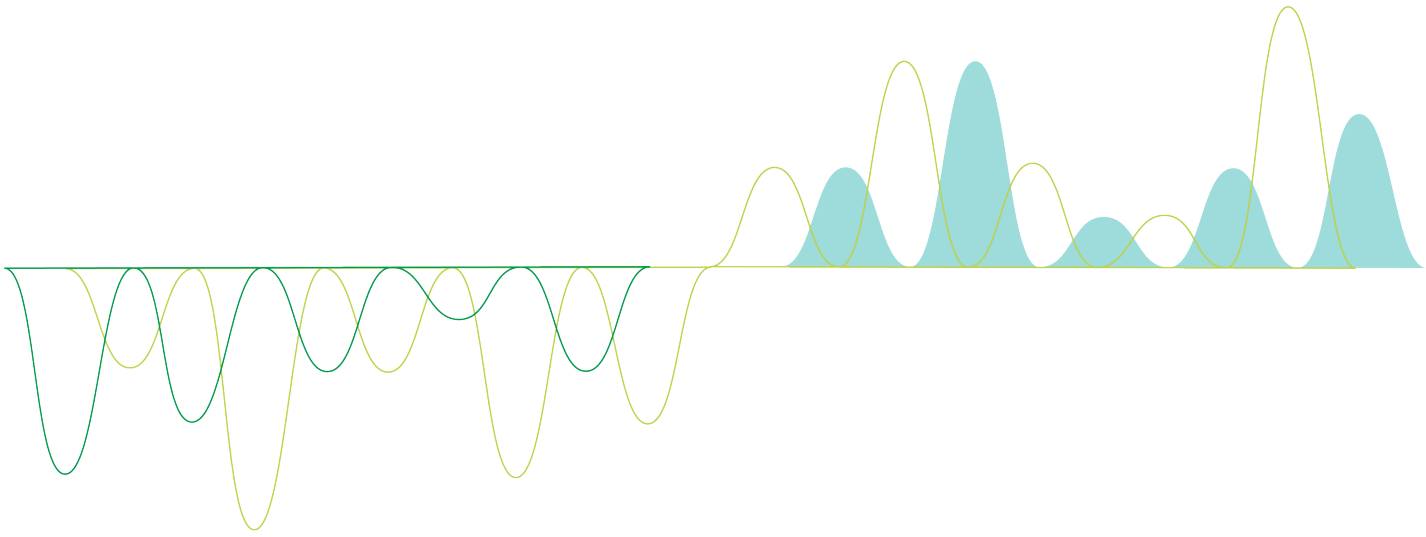
Tutorial – próximas etapas em script

Qlik Sense®
February 2023



Contents

1 Bem-vindo a este tutorial!

Bem-vindo a este tutorial, que apresenta a criação avançada de scripts no Qlik Sense.

Depois de se familiarizar com os conceitos básicos da criação de scripts, você pode começar a executar operações mais sofisticadas em seus dados enquanto os carrega no Qlik Sense. Isso pode incluir, por exemplo, transformar dados usando tabelas cruzadas, limpar dados e criar e carregar dados de arquivos de dados do Qlik conhecidos como arquivos QVD.

1.1 O que você 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 Quem deve realizar este curso

Você deve estar familiarizado com os conceitos básicos da criação de scripts no Qlik Sense. Ou seja, você carregou e manipulou dados usando scripts.

Se você ainda não tiver feito isso, recomendamos concluir o tutorial Criação de script para iniciantes.

Você precisa acessar o editor de carregamento de dados e deve poder carregar dados no Qlik Sense Enterprise on Windows.

As instruções também se aplicam geralmente para o Qlik Sense Cloud Business.

1.3 Conteúdo do pacote

O pacote zip que você baixou contém os seguintes arquivos de dados necessários para concluir o tutorial:

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

O pacote também contém uma cópia do aplicativo *Tutorial - Criação avançada de scripts*. Seções de script adicionais no aplicativo contêm os scripts para os outros aplicativos que você cria neste tutorial. Você pode carregar o aplicativo no seu hub.

Recomendamos a criação do aplicativo, conforme descrito no tutorial para maximizar seu aprendizado. Além disso, você teria que fazer upload e conectar-se aos seus arquivos de dados, conforme descrito no tutorial, para que os carregamentos de dados funcionassem.

No entanto, se você tiver problemas, o aplicativo poderá ajudá-lo a solucionar problemas. Indicamos quais segmentos de script estão associados a cada lição.

1.4 Lições neste tutorial

Dependendo da sua experiência com o Qlik Sense, este tutorial deve levar de 3 a 4 horas para ser concluído. Os tópicos foram pensados para serem concluídos em sequência. No entanto, você pode sair e retornar a qualquer momento. Felizmente, não há testes.

Transformando dados
Usando o prefixo Crosstable
Combinando tabelas com Join e Keep
Usando funções de inter-registro: Peek, Previous e Exists
Correspondências de intervalos e carregamento iterativo
Limpeza de dados
Manipulando dados hierárquicos
Arquivos QVD

1.5 Leituras e recursos adicionais

- O Qlik oferece uma ampla variedade de recursos quando você quiser aprender mais.
- A ajuda online do Qlik está disponível.
- Treinamentos, incluindo cursos online gratuitos, estão disponíveis no Qlik Continuous Classroom.
- Fóruns de discussão, blogs e muitos outros recursos podem ser encontrados na Qlik Community.

2 Comandos LOAD e SELECT do

Você pode carregar dados no Qlik Sense usando os comandoLOAD e SELECT. Cada um desses comandos gera uma tabela interna. LOAD é usado para carregar dados de arquivos, e SELECT é usado para carregar dados de bancos de dados.

Neste tutorial, você usará dados de arquivos e usará comandos LOAD.

Você também pode usar um LOAD precedente para poder manipular o conteúdo dos dados carregados. Por exemplo, a renomeação de campos deve ser feita em um comando LOAD, enquanto o comando SELECT não permite nenhuma alteração nos nomes dos campos.

As seguintes regras se aplicam durante o carregamento dos dados no Qlik Sense:

- O Qlik Sense não diferencia as tabelas geradas por um comando LOAD ou SELECT. Dessa forma, se várias tabelas forem carregadas, não importa se as tabelas são carregadas pelos comandos LOAD ou SELECT ou por uma mistura de ambos.
- A ordem dos campos no comando ou na tabela original do banco de dados não é importante para a lógica do Qlik Sense.
- Os nomes de campo diferenciam maiúsculas de minúsculas e são usados para estabelecer associações entre tabelas de dados. Por isso, às vezes é necessário renomear os campos no script de carregamento para obter um modelo de dados desejado.

3 Transformando dados

Você pode transformar e manipular dados no editor de carregamento de dados antes de usar os dados em seu aplicativo.

Uma das vantagens da manipulação de dados é que você pode escolher carregar apenas um subconjunto dos dados de um arquivo, como algumas colunas escolhidas de uma tabela, para tornar o manuseio de dados mais eficiente. Você também pode carregar os dados mais de uma vez para dividir os dados brutos em diversas novas tabelas lógicas. Também é possível carregar dados de mais de uma fonte e mesclá-los em uma tabela no Qlik Sense.

Os exercícios a seguir mostrarão como carregar dados usando o prefixo Crosstable. Você também aprenderá a unir tabelas, usar as funções inter-registro comoPeek e Previous, e carregar a mesma linha várias vezes usandoWhile Load.

3.1 Usando o prefixo Crosstable

Tabelas cruzadas são um tipo comum de tabela que apresentam uma matriz de valores entre duas listas retangulares de dados de cabeçalho. Sempre que existir uma tabela cruzada de dados, você poderá usar o prefixo Crosstable para transformar os dados e criar os campos desejados.

Prefixo Crosstable

Na tabela a seguir *Product*, há uma coluna por mês e uma linha por produto.

Tabela de produtos

Produto	Jan 2014	Fev 2014	Mar 2014	Apr 2014	Mai 2014	Jun 2014
A	100	98	100	83	103	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

Quando você carrega a tabela, a saída é uma tabela com um campo para o *Product* e um campo para cada um dos meses.

Tabela Product com campo Product, e um campo para cada um dos meses

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

Se desejar analisar esses dados, é muito mais fácil ter todos os números em um campo e todos os meses em outro. Nesse caso, é uma tabela de três colunas com uma coluna para cada categoria. (*Product, Month, Sales*).

Tabela Product com os campos Product, Month e Sales

Product
Product
Month
Sales

O prefixo Crosstable converte os dados para uma tabela com uma coluna para o mês *Month* e outra para *Sales*. Outra forma de expressar isso é dizer que ele converte os nomes de campo para valores de campos.

Faça o seguinte:

1. Crie um novo aplicativo e chame-o *Tutorial de uso de scripts avançado*.
2. Adicione uma nova seção de script no **Editor de carregamento de dados**.
3. Nomeie a seção *Product*.
4. Em **AttachedFiles** no menu direito, clique em **Selecionar dados**.
5. Carregue e, em seguida, selecione *Product.xlsx*.
6. Selecione a tabela *Product* na janela **Selecionar dados de**.

   > *Em **Nomes de campos**, certifique-se de que **Nomes de campos incorporados** esteja selecionado para incluir os nomes de campos da tabela ao carregar os dados.*

7. Clique em **Inserir script**.

Seu script deve ter a seguinte aparência:

```
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. Clique em **Carregar dados**.
9. Abra o **Visualizador do modelo de dados**. O modelo de dados tem a seguinte aparência:

   Tabela Product com campo Product, e um campo para cada um dos meses

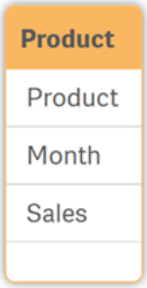
   

10. Clique na guia *Product* no **Editor de carregamento de dados**.
11. Digite o seguinte acima do comando LOAD:

    ```
    CrossTable(Month, Sales)
    ```

12. Clique em **Carregar dados**.
13. Abra o **Visualizador do modelo de dados**. O modelo de dados tem a seguinte aparência:

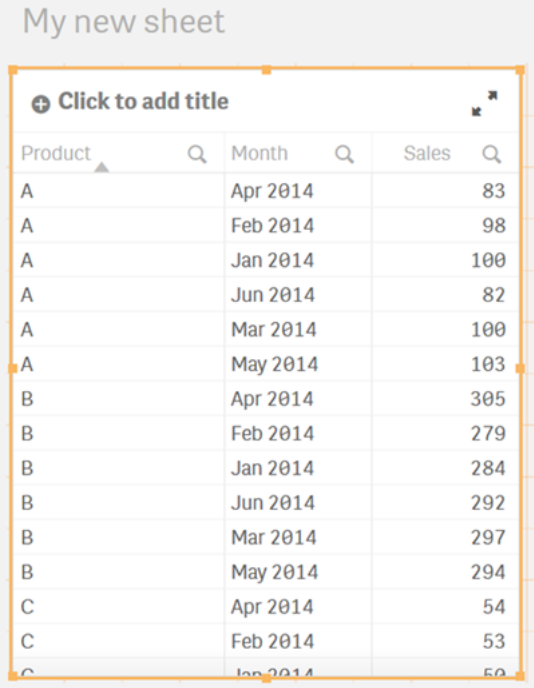
    Tabela Product com os campos Product, Month e Sales

Observe que geralmente os dados de entrada têm apenas uma coluna como um campo qualificador; como uma chave interna (*Product* no exemplo acima). Mas é possível ter várias. Nesse caso, todos os campos qualificadores devem ser listados antes dos campos de atributo no comando LOAD e o terceiro parâmetro do prefixo Crosstable deve ser usado para definir o número de campos qualificadores. Não é possível ter um LOAD anterior ou um prefixo na frente da palavra-chave Crosstable. No entanto, você pode usar a concatenação automática.

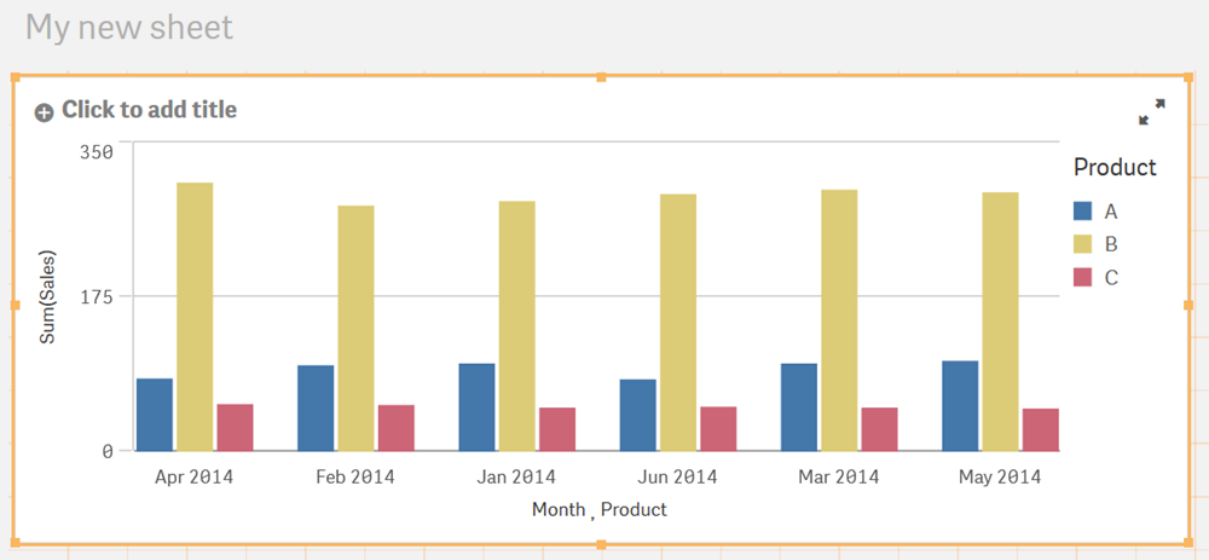
Em uma tabela no Qlik Sense, seus dados têm a seguinte aparência:

Tabela mostrando dados carregados usando o prefixo Crosstable



Agora você pode, por exemplo, criar um gráfico de barras usando os dados:

Gráfico de barras mostrando dados carregados usando o prefixo Crosstable



Para saber mais sobre Crosstable, consulte esta postagem de blog no Qlik Community: The Crosstable Load (A carga de crosstable). Os comportamentos são discutidos no contexto do QlikView. No entanto, a lógica se aplica igualmente ao Qlik Sense.

A interpretação numérica não funcionará para os campos de atributo. Isto significa que se existirem meses como cabeçalhos de coluna, eles não serão interpretados automaticamente. A solução alternativa é usar o prefixo Crosstable para criar uma tabela temporária e executar uma segunda passagem por meio dela para fazer as interpretações como mostrado no exemplo a seguir.

Observe que este é apenas um exemplo. Não há exercícios a serem concluídos no 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;
```

Limpando o cache de memória

Você pode excluir as tabelas criadas para limpar o cache da memória. Ao carregar em uma tabela temporária, como na seção anterior, você deve descartá-la quando ela não for mais necessária. Por exemplo:

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

Você também pode soltar campos. Por exemplo:

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

Como você pode ver, as palavras-chave TABLE e FIELD podem ser singulares ou plurais.

3.2 Combinando tabelas com Join e Keep

A junção é uma operação que usa duas tabelas para combiná-las em uma. Os registros da tabela resultante são combinações dos registros das tabelas originais, de forma que, geralmente, os dois registros que contribuem para qualquer combinação na tabela resultante tenham um valor comum para um ou vários campos comuns, a assim chamada junção natural. No Qlik Sense, as junções podem ser feitas no script, produzindo tabelas lógicas.

É possível unir tabelas que já estão no script. Assim, a lógica do Qlik Sense não verá as tabelas separadas, e sim o resultado da junção, que é uma única tabela interna. Em algumas situações isso é necessário, mas existem desvantagens:

- Normalmente, as tabelas carregadas ficam maiores e o Qlik Sense trabalha mais lentamente.
- Algumas informações podem ser perdidas: a frequência (número de registros) na tabela original pode não mais estar disponível.

A funcionalidade Keep, que tem o efeito de reduzir uma ou ambas as tabelas à interseção dos dados da tabela antes de serem armazenadas no Qlik Sense, foi projetada para reduzir o número de casos que exigem o uso de junções explícitas.

Nesta documentação, o termo junção é geralmente utilizado para junções feitas antes da criação das tabelas internas. Entretanto, a associação, feita após a criação das tabelas internas, também é basicamente uma junção.

Join

A forma mais simples de fazer uma junção é usar o prefixo Join no script, que une a tabela interna a outra tabela nomeada ou à última tabela criada anteriormente. A junção será externa, criando todas as combinações possíveis de valores das duas tabelas.

Exemplo:

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

A tabela interna resultante tem os campos a, b, c e d. O número de registros é diferente dependendo dos valores de campo das duas tabelas.

Os nomes dos campos a serem unidos devem ser exatamente os mesmos. O número de campos a serem unidos é arbitrário. Normalmente, as tabelas devem ter um ou alguns campos em comum. Nenhum campo em comum gerará o produto cartesiano das tabelas. Também é possível ter todos os campos em comum, mas isso normalmente não faz sentido. A menos que o nome de uma tabela carregada anteriormente seja especificado no comando Join o prefixo Join utilizará a última tabela criada anteriormente. Dessa forma, a ordem dos dois comandos não é arbitrária.

Usando Join

O prefixo Join explícito na linguagem de script do Qlik Sense executa uma junção completa das duas tabelas. O resultado é uma tabela. Tais uniões geralmente podem resultar em tabelas muito grandes.

Faça o seguinte:

1. Abra o aplicativo *Tutorial de uso de scripts avançado*.
2. Adicione uma nova seção de script no **Editor de carregamento de dados**.
3. Nomeie a seção como *Transactions*.
4. Em **AttachedFiles** no menu direito, clique em **Selecionar dados**.
5. Carregue e, em seguida, selecione *Transactions.csv*.

   *Em **Nomes de campos**, certifique-se de que **Nomes de campos incorporados** esteja selecionado para incluir os nomes de campos da tabela ao carregar os dados.*
6. Na janela **Selecionar dados de**, clique em **Inserir script**.
7. Carregue e, em seguida, selecione *Salesman.xlsx*.
8. Na janela **Selecionar dados de**, clique em **Inserir script**.

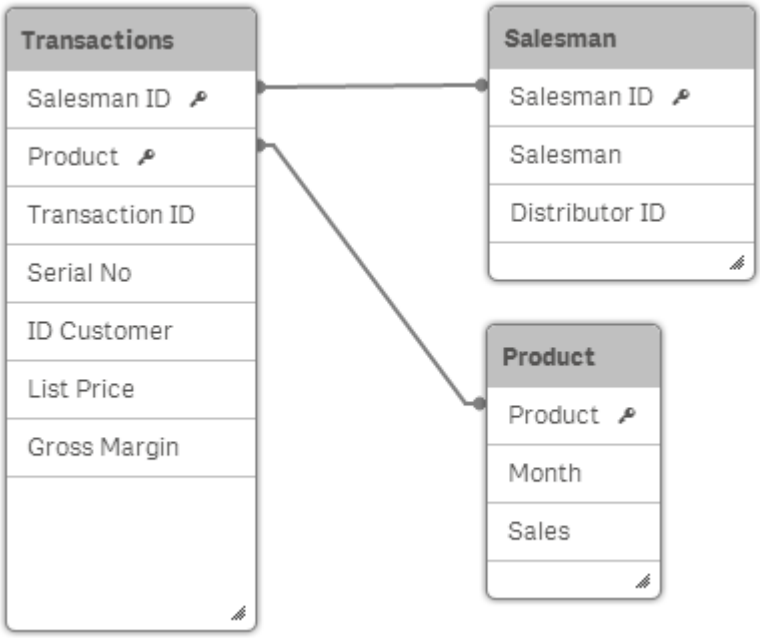
   Seu script deve ter a seguinte aparência:

   ```
   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. Clique em **Carregar dados**.
10. Abra o **Visualizador do modelo de dados**. O modelo de dados tem a seguinte aparência:

    Modelo de dados: Tabelas Transactions, Salesman e Product



No entanto, ter as tabelas *Transactions* e *Salesman* separadas pode não ser o resultado necessário. Pode ser melhor juntar as duas tabelas.

Faça o seguinte:

1. Para definir um nome para a tabela unida, adicione a seguinte linha acima do primeiro comando LOAD:

   ```
   Transactions:
   ```

2. Para unir as tabelas *Transactions* e *Salesman*, adicione a seguinte linha acima do segundo comando LOAD:

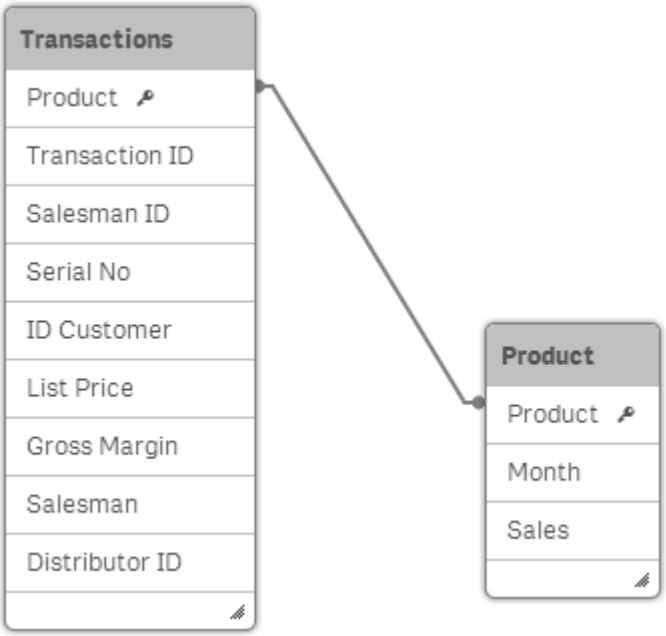
   ```
   Join(Transactions)
   ```

   Seu script deve ter a seguinte aparência:

   ```
   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. Clique em **Carregar dados**.
4. Abra o **Visualizador do modelo de dados**. O modelo de dados tem a seguinte aparência:

Modelo de dados: Tabelas Transactions e Product



Todos os campos das tabelas *Transactions* e *Salesman* agora estão combinados em uma única tabela *Transactions*.

Para saber mais sobre quando usar Join, consulte essas postagens de blog no Qlik Community: To Join or not to Join *(Unir ou não unir),* Mapping as an Alternative to Joining *(Mapeamento como uma alternativa para a união). Os comportamentos são discutidos no contexto do QlikView. No entanto, a lógica se aplica igualmente ao Qlik Sense.*

Keep

Uma das principais características do Qlik Sense é sua capacidade de fazer associações entre as tabelas, em vez de uni-las, reduzindo bastante o espaço usado na memória, aumentando a velocidade e oferecendo enorme flexibilidade. A funcionalidade Keep foi projetada para reduzir o número de casos em que as junções explícitas precisam ser usadas.

O prefixo Keep entre dois comandos LOAD ou SELECT reduz uma ou ambas as tabelas à interseção dos dados dos dados da tabela antes de serem armazenados no Qlik Sense. O prefixo Keep sempre deve ser precedido de uma das palavras-chave Inner, Left ou Right. A seleção de registros das tabelas é feita da mesma maneira que em uma junção correspondente. Entretanto, as duas tabelas não são unidas e serão armazenadas no Qlik Sense como duas tabelas nomeadas separadas.

Inner

Os prefixos Join e Keep no script de carga de dados podem ser precedidos pelo prefixo Inner.

Se for usado antes de Join, especificará que a junção das duas tabelas deve ser interna. A tabela resultante contém apenas combinações entre as duas tabelas com um conjunto de dados completo de ambos os lados.

Se usado antes de Keep, especificará que as duas tabelas deverão ser reduzidas à sua interseção comum antes de serem armazenadas no Qlik Sense.

Exemplo:

Nestes exemplos, usamos as tabelas de origem *Table1* e *Table2*.

Observe que esses são apenas exemplos. Não há exercícios a serem concluídos no Qlik Sense.

Table 1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Inner Join

Primeiro, realizamos um Inner Join nas tabelas, resultando em um *VTable*, contendo apenas uma linha, o único registro existente em ambas as tabelas, com dados combinados de ambas as tabelas.

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

VTable

A	B	C
1	aa	xx

Inner Keep

Se realizarmos um Inner Keep, ainda teremos duas tabelas. As duas tabelas serão associadas através do campo comum *A*.

```
VTab1: SELECT * from Table1; VTab2: inner keep SELECT * from Table2;
```

VTab1

A	B
1	aa

VTab2

A	C
1	xx

Left

Os prefixos Join e Keep no script de carga de dados podem ser precedidos pelo prefixo left.

Se for usado antes de Join, ele especificará que a junção das duas tabelas deve ser à esquerda. A tabela resultante contém apenas combinações entre as duas tabelas com um conjunto de dados completo da primeira tabela.

Se usado antes de Keep, especificará que a segunda tabela deverá ser reduzida à sua interseção comum com a primeira tabela antes de ser armazenada no Qlik Sense.

Exemplo:

Nestes exemplos, usamos as tabelas de origem *Table1* e *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Primeiro, um Left Join é realizado nas tabelas, resultando em um *VTable*, contendo todas as linhas da *Table1*, combinadas com os campos das linhas correspondentes na *Table2*.

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

VTable

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

Se realizarmos um Left Keep, ainda teremos duas tabelas. As duas tabelas serão associadas através do campo comum *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

Os prefixos Join e Keep na linguagem de script do Qlik Sense podem ser precedidos pelo prefixo right.

Se for usado antes de Join, ele especificará que a junção das duas tabelas deve ser à direita. A tabela resultante contém apenas combinações entre as duas tabelas com um conjunto de dados completo da segunda tabela.

Se usado antes de Keep, especificará que a primeira tabela deverá ser reduzida à sua interseção comum com a segunda tabela antes de ser armazenada no Qlik Sense.

Exemplo:

Nestes exemplos, usamos as tabelas de origem *Table1* e *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Primeiro, um Right Join é realizado nas tabelas, resultando em um *VTable*, contendo todas as linhas da *Table2*, combinadas com os campos das linhas correspondentes na *Table1*.

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

VTable

A	B	C
1	aa	xx
4	-	yy

Se realizarmos um Right Keep, ainda teremos duas tabelas. As duas tabelas serão associadas através do campo comum *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 Usando funções de inter-registro: Peek, Previous e Exists

Estas funções são usadas quando um valor dos registros de dados anteriormente carregados é necessário para a avaliação do registro atual.

Nesta parte do tutorial, analisaremos as funções Peek(), Previous() e Exists().

Peek()

Peek() retorna o valor de um campo em uma tabela para uma linha que já foi carregada. O número da linha pode ser especificado, assim como a tabela. Se nenhum número de linha for especificado, o último registro carregado anteriormente será usado.

Sintaxe:

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

A linha deve ser um número inteiro. 0 representa o primeiro registro, 1 o segundo e assim por diante. Números negativos indicam a ordem no final da tabela. -1 representa o último registro lido.

Se nenhuma linha for definida, -1 será exibido.

Tablename é um rótulo de tabela sem os dois-pontos finais. Se nenhum *tablename*for definido, a tabela atual será assumida. Se usado fora do comando **LOAD** ou em referência a outra tabela, o *tablename* deve ser incluído.

Previous()

Previous() encontra o valor da expressão **expr** usando dados do registro de entrada anterior que não foi descartado devido a uma cláusula **where**. No primeiro registro de uma tabela interna, a função retornará NULL.

Sintaxe:

```
Previous(expression)
```

A função Previous() pode ser inserida para acessar os registros anteriores. Os dados são buscados diretamente da fonte de entrada, o que permite fazer referência também aos campos não carregados no Qlik Sense, ou seja, mesmo que não tenham sido armazenados no banco de dados associado.

Exists()

Exists() determina se um valor de campo específico já foi carregado no campo no script de carga de dados. A função retorna TRUE ou FALSE, de forma que pode ser usada na cláusula **where** de um comando **LOAD** ou um comando **IF**.

Sintaxe:

```
Exists(field [, expression ] )
```

O campo deve existir nos dados carregados até o momento pelo script. *Expression* é uma expressão avaliada para o valor de campo a ser pesquisado no campo especificado. Se for omitida, será assumido o valor do registro atual no campo especificado.

Usando o Peek() e Previous()

Na sua forma mais simples, Peek() e Previous() são usados para identificar os valores específicos em uma tabela. Eis uma amostra dos dados na tabela *Employees* que você carregará neste exercício.

Amostra de dados da tabela Funcionários

Date	Contratado	Desligado
1/1/2011	6	0
2/1/2011	4	2
3/1/2011	6	1
4/1/2011	5	2

Atualmente, ela só coleta dados para o mês, contratações e demissões; por isso, adicionaremos campos para *Employee Count* e *Employee Var* usando as funções Peek() e Previous() para ver a diferença mensal no total de funcionários.

Faça o seguinte:

1. Abra o aplicativo *Tutorial de uso de scripts avançado*.
2. Adicione uma nova seção de script no **Editor de carregamento de dados**.
3. Nomeie a seção como *Employees*.

4. Em **AttachedFiles** no menu direito, clique em **Selecionar dados**.
5. Carregue e, em seguida, selecione *Employees.xlsx*.

   Em Field names, certifique-se de que Embedded field names esteja selecionado para incluir os nomes de campos da tabela ao carregar os dados.

6. Na janela **Selecionar dados de**, clique em **Inserir script**.

   Seu script deve ter a seguinte aparência:

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

7. Modifique o script para que ele agora tenha a seguinte aparência:

   ```
   [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]
   embedded labels, table is Sheet1);
   ```

   As datas do campo *Date* na pasta do Excel estão no formato MM/DD/AAAA. Para assegurar que as datas sejam interpretadas corretamente usando o formato das variáveis do sistema, a função Date é aplicada ao campo *Date*.

   A função Peek() permite identificar qualquer valor carregado para um campo definido. Na expressão, examinamos primeiro para ver se rowno() é igual a 1. Se for igual a 1, nenhum *Employee Count* existirá e, portanto, preencheremos o campo com a diferença de *Hired* menos *Terminated*.

   Se rowno() for maior do que 1, observaremos a *Employee Count* do último mês e usaremos esse número para somá-lo à diferença dos funcionários*Hired* menos funcionários *Terminated*.

   Observe também que, na função Peek(), usamos um (-1). Isto informa o Qlik Sense para examinar o registro acima do registro atual. Se (-1) não for especificado, o Qlik Sense considerará que você deseja examinar o registro anterior.

8. Adicione o seguinte ao final do 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];
   ```

   A função Previous() permite identificar o último valor carregado para um campo definido. Na expressão, primeiro examinamos para ver se rowno() é igual a 1. Se for igual a 1, sabemos que não haverá Employee Var porque não há nenhum registro para o mês anterior *Employee Count*. Portanto, basta digitar 0 para o valor.

   Se rowno() for maior que 1, saberemos que haverá um *Employee Var*; portanto, examinaremos a *Employee Count* do último mês e subtrairemos esse número da *Employee Count* do mês atual para criar o valor no campo *Employee Var*.

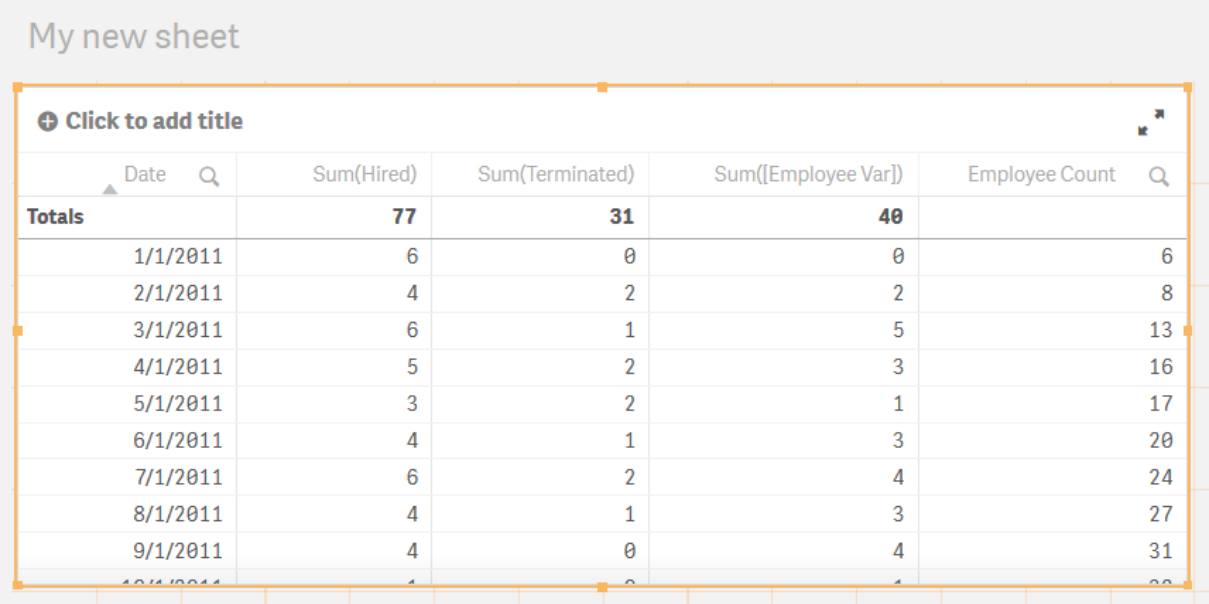
   Seu script deve ter a seguinte aparência:

```
[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. Clique em **Carregar dados**.

   Em uma nova pasta na visão geral do aplicativo, crie uma tabela usando *Date, Hired, Terminated, Employee Count* e *Employee Var* como as colunas da tabela. A tabela resultante deve ficar assim:

   Tabela seguindo o uso de Peek e Previous no script



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

Peek() e Previous() permitem visar linhas definidas em uma tabela. A maior diferença entre as duas funções é que a função Peek() permite que o usuário examine um campo que não tenha sido anteriormente carregado no script, enquanto que a função Previous() só permite examinar um campo anteriormente carregado. Previous() opera na entrada do comando LOAD, enquanto que Peek() opera na saída do comando LOAD. (O mesmo que a diferença entre RecNo() e RowNo()). Isso significa que as duas funções se comportarão de forma diferente se você tiver uma cláusula Where.

Portanto, a função Previous() seria mais adequada quando você precisar mostrar o valor atual em relação ao valor anterior. No exemplo, foi calculado a variação de funcionários em cada mês.

A função Peek() seria mais adequada para quando você estiver visando um campo que não tenha sido anteriormente carregado na tabela ou quando você precisar visar uma linha específica. Isso foi mostrado no exemplo em que calculamos a *Employee Count* examinando a *Employee Count* do mês anterior e somando a diferença entre os funcionários contratados e desligados no mês atual. Lembre-se de que *Employee Count* não era um campo do arquivo original

Para saber mais sobre quando usar Peek() e Previous(), consulte esta postagem de blog no Qlik Community: Peek() vs Previous() - When to Use Each*. Os comportamentos são discutidos no contexto do QlikView. No entanto, a lógica se aplica igualmente ao Qlik Sense.*

Usando Exists()

A função Exists() é geralmente usada com a cláusula Where no script para carregar dados se os dados relacionados já tiverem sido carregados no modelo de dados.

No exemplo a seguir, também usamos a função Dual() para atribuir valores numéricos a cadeias.

Faça o seguinte:

1. Crie um novo aplicativo e nomeie-o.
2. Adicione uma nova seção de script no **Editor de carregamento de dados**.
3. Nomeie a seção como *People*.
4. Digite o seguinte 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;
```

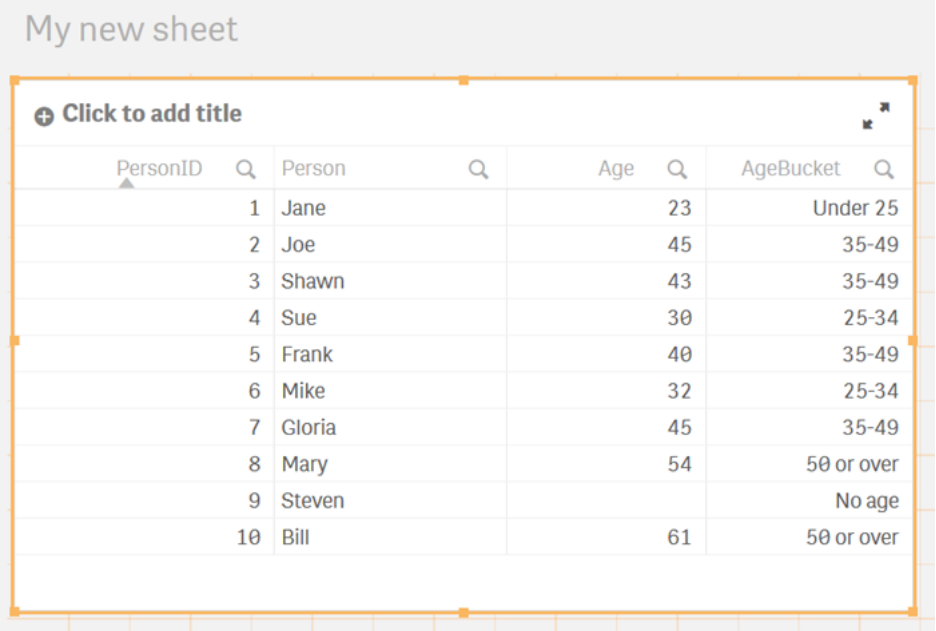
5. Clique em **Carregar dados**.

   No script, os campos *Age* e *AgeBucket* são carregados apenas se *PersonID* já tiver sido carregado no modelo de dados.

   Observe na tabela *AgeTemp* que existem idades listadas para *PersonID* 11 e 12, mas já que estes IDs não foram carregados no modelo de dados (na tabela *People*), eles são excluídos pela cláusula Where Exists(PersonID). Esta cláusula também pode ser escrita da seguinte forma: Where Exists (PersonID, PersonID).

   A saída do script tem a seguinte aparência:

Tabela seguindo o uso de Exists no script

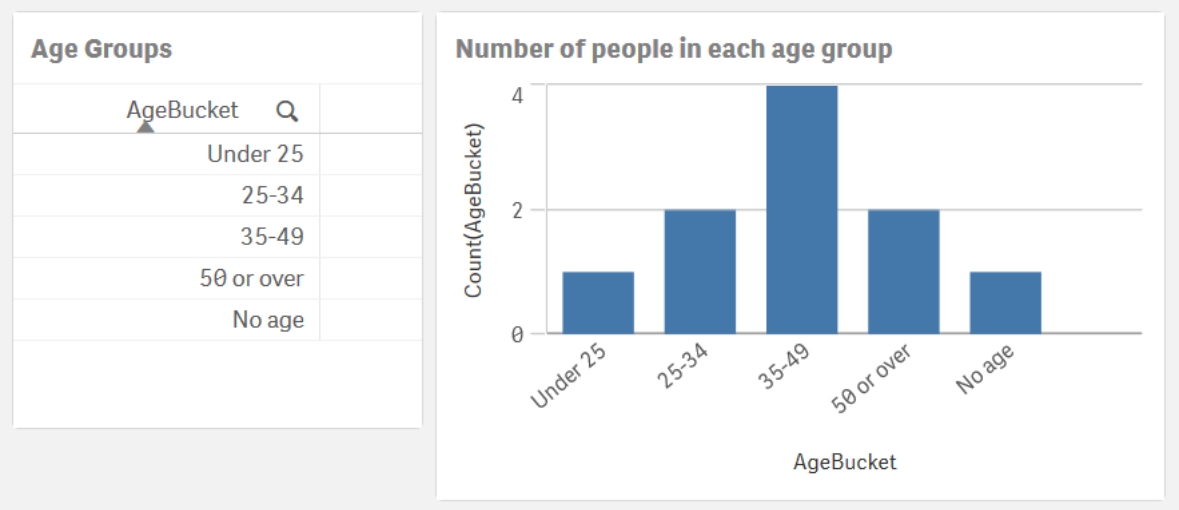


PersonID	Person	Age	AgeBucket
1	Jane	23	Under 25
2	Joe	45	35-49
3	Shawn	43	35-49
4	Sue	30	25-34
5	Frank	40	35-49
6	Mike	32	25-34
7	Gloria	45	35-49
8	Mary	54	50 or over
9	Steven		No age
10	Bill	61	50 or over

Se nenhum *PersonID* da tabela *AgeTemp* tiver sido carregado no modelo de dados, os campos *Age* e *AgeBucket* não teriam sido unidos à tabela *People*. Usar a função Exists() pode ajudar a prevenir registros/dados órfãos no modelo de dados, ou seja, os campos *Age* e *AgeBucket* que não têm nenhuma pessoa associada.

6. Crie uma nova pasta e nomeie-a.
7. Abra a nova pasta e clique em **Editar pasta**.
8. Adicione uma tabela padrão à pasta com a dimensão *AgeBucket* e nomeie a visualização como *Grupos de idade*.
9. Adicione um gráfico de barras à pasta com a dimensão *AgeBucket* e a medida *Count([AgeBucket])*. Nomeie a visualização como *Number of people in each age group*.
10. Ajuste as propriedades da tabela e do gráfico de barras de acordo com suas preferências e clique em **Concluído**.
    Sua pasta deve ter a seguinte aparência:

Pasta com agrupamentos por idade



A função Dual() é útil no script ou em uma expressão do gráfico quando há a necessidade de atribuir um valor numérico a uma cadeia de caracteres.

No script acima, existe um aplicativo que carrega as idades, e você decidiu colocar essas idades em compartimentos para que você possa criar visualizações com base nos compartimentos de idades em relação às idades reais. Existe um compartimento para as pessoas com menos de 25, entre 25 e 35 e assim por diante. Ao usar a função Dual(), um valor numérico pode ser atribuído aos compartimentos de idades, que posteriormente pode ser usado para classificar os compartimentos de idades em uma caixa de lista ou em um gráfico. Portanto, assim como ocorre na pasta do aplicativo, a classificação insere "Nenhuma idade" no fim da lista.

Para saber mais sobre Exists() e Dual(), consulte esta postagem de blog no Qlik Community: Dual e Exists - Funções úteis

3.4 Correspondências de intervalos e carregamento iterativo

O prefixo Intervalmatch de um comando LOAD ou SELECT é usado para vincular valores numéricos discretos a um ou mais intervalos numéricos. Esse recurso é muito útil e pode ser usado, por exemplo, em ambientes de produção.

Usando o prefixo IntervalMatch()

A correspondência de intervalo mais simples é quando você tem uma lista de números ou datas (eventos) em uma tabela e uma lista de intervalos em uma segunda tabela. O objetivo é vincular as duas tabelas. Em geral, este é um relacionamento de muitos para muitos, isto é, um intervalo pode ter muitas datas pertencentes a ele, e uma data pode pertencer a vários intervalos. Para resolver isso, é necessário criar uma tabela de ponte entre as duas tabelas originais. Existem duas formas de fazer isso.

A maneira mais simples de resolver este problema no Qlik Sense é usar o prefixo IntervalMatch() na frente de um comando LOAD ou SELECT. O comando LOAD/SELECT precisa conter apenas dois campos, os campos From e To, que definem os intervalos. Em seguida, o prefixo IntervalMatch() gerará todas as combinações entre os intervalos carregados e um campo numérico anteriormente carregado, especificado como o parâmetro do prefixo.

Faça o seguinte:

1. Crie um novo aplicativo e nomeie-o.
2. Adicione uma nova seção de script no **Editor de carregamento de dados**.
3. Nomeie as seções como *Events*.
4. Em **AttachedFiles** no menu direito, clique em **Selecionar dados**.
5. Carregue e, em seguida, selecione *Events.txt*.
6. Na janela **Selecionar dados de**, clique em **Inserir script**.
7. Carregue e, em seguida, selecione *Intervals.txt*.
8. Na janela **Selecionar dados de**, clique em **Inserir script**.
9. No script, nomeie a primeira tabela como *Eventos* e a segunda tabela como *Intervals*.
10. No final do script, adicione um IntervalMatch para criar uma terceira tabela que vinculará as duas primeiras tabelas:

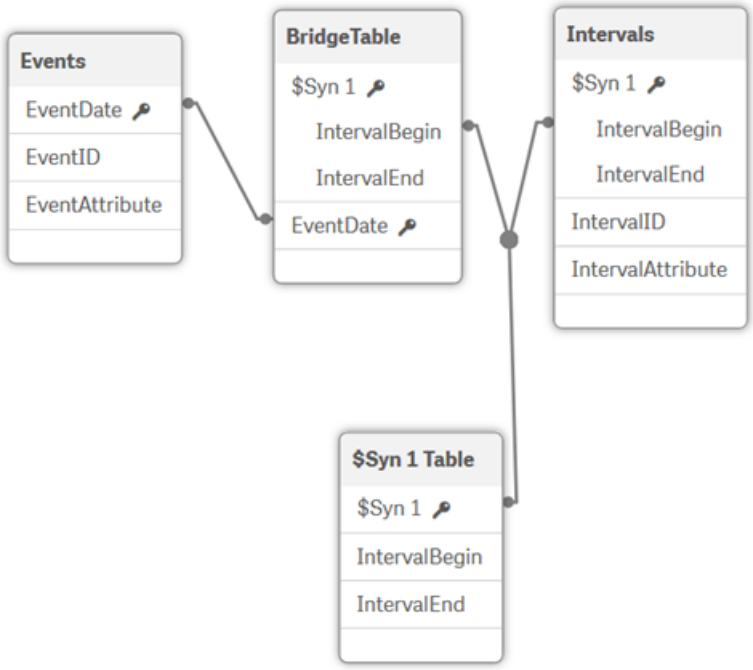
    ```
    BridgeTable: IntervalMatch (EventDate) LOAD distinct IntervalBegin, IntervalEnd Resident Intervals;
    ```

11. Seu script deve ter a seguinte aparência:

    ```
    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. Clique em **Carregar dados**.
13. Abra o **Visualizador do modelo de dados**. O modelo de dados tem a seguinte aparência:

Modelo de dados: Tabelas Events, BridgeTable, Intervals e $Syn1



O modelo de dados contém uma chave composta (os campos *IntervalBegin* e *IntervalEnd*), que se manifestarão como uma chave sintética do Qlik Sense:

As tabelas básicas são:

- A tabela *Events*, que contém exatamente um registro por evento.
- A tabela *Intervals*, que contém exatamente um registro por intervalo.
- A tabela de ponte, que contém exatamente um registro por combinação de evento e intervalo e que vincula as duas tabelas anteriores.

Observe que um evento pode pertencer a vários intervalos se os intervalos forem sobrepostos. E um intervalo pode, obviamente, ter vários eventos pertencentes a ele.

Este modelo de dados é o melhor, no sentido de que ele é normalizado e compacto. As tabelas *Events* e *Intervals* não são alteradas e contêm o número original de registros. Todos os cálculos do Qlik Sense que são operados nessas tabelas, por exemplo, Count(EventID), funcionarão e serão avaliados corretamente.

Para saber mais sobre IntervalMatch(), consulte esta postagem de blog no Qlik Community: Usando IntervalMatch()

Usando um loop While e um carregamento iterativo IterNo()

É possível obter quase a mesma tabela de ponte usando um loop While e IterNo() que cria valores enumeráveis entre os limites inferior e superior do intervalo.

Um loop dentro do comando LOAD pode ser criado com a cláusula While. Por exemplo:

```
LOAD Date, IterNo() as Iteration From … While IterNo() <= 4;
```

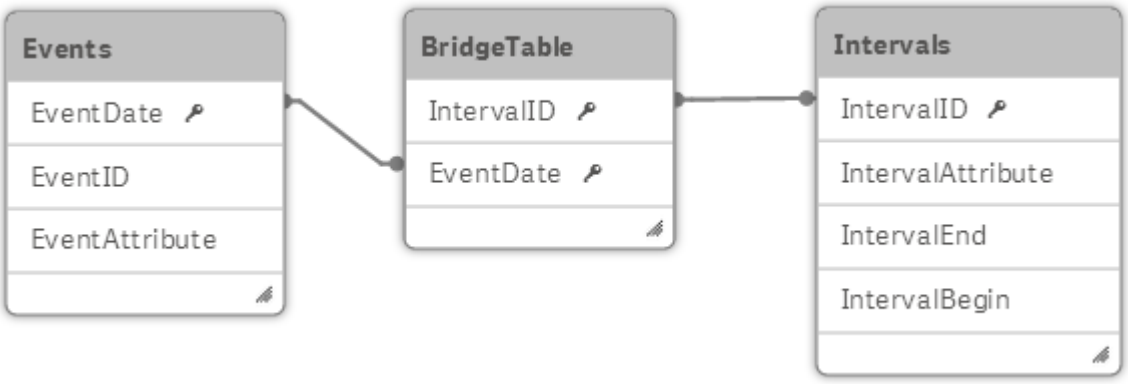
Esse comando LOAD executará um loop em cada registro de entrada, carregando-os continuamente, enquanto a expressão na cláusula While for "true". A função IterNo() retorna “1” na primeira iteração, “2” na segunda e assim por diante.

Como você tem uma chave primária para os intervalos, IntervalID, a única diferença no script será o modo de criação da tabela de ponte:

Faça o seguinte:

1. Substitua os comandos Bridgetable existentes pelo seguinte script:
   ```
   BridgeTable: LOAD distinct * Where Exists(EventDate); LOAD IntervalBegin + IterNo() - 1
   as EventDate, IntervalID     Resident Intervals     While IntervalBegin + IterNo() - 1
   <= IntervalEnd;
   ```
2. Clique em **Carregar dados**.
3. Abra o **Visualizador do modelo de dados**. O modelo de dados tem a seguinte aparência:

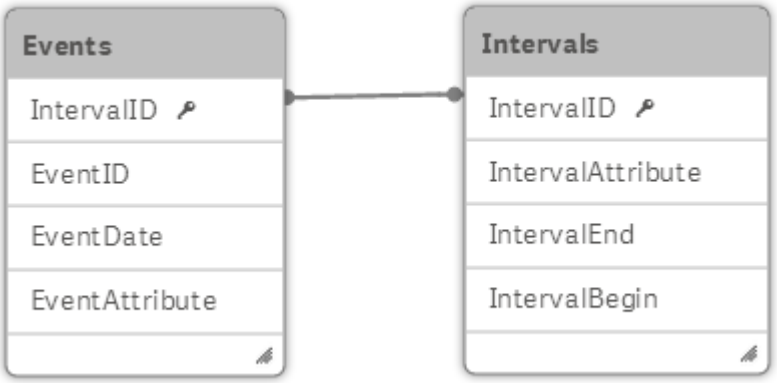
   Modelo de dados: Tabelas Events, BridgeTable e Intervals

   Em geral, a solução com três tabelas é a melhor, pois permite um relacionamento de muitos para muitos entre os intervalos e eventos. Mas uma situação comum é saber que um evento só pode pertencer a um único intervalo. Nesse caso, a tabela de ponte não é realmente necessária: O *IntervalID* pode ser armazenado diretamente na tabela de eventos. Existem várias maneiras de obter isso, mas a mais útil é unir Bridgetable com a tabela *Events*.

4. Adicione o script a seguir ao final do script:
   ```
   Join (Events) LOAD EventDate, IntervalID Resident BridgeTable;   Drop Table BridgeTable;
   ```
5. Clique em **Carregar dados**.
6. Abra o **Visualizador do modelo de dados**. O modelo de dados tem a seguinte aparência:

Modelo de dados: Tabelas Events e Intervals



Intervalos abertos e fechados

Um intervalo aberto ou fechado é determinado pelos pontos de extremidade, estejam eles incluídos no intervalo ou não.

- Se os pontos de extremidade estiverem incluídos, ele será um intervalo fechado:
  $[a,b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$
- Se os pontos de extremidade não estiverem incluídos, ele será um intervalo aberto:
  $]a,b[= \{x \in \mathbb{R} \mid a < x < b\}$
- Se um único ponto de extremidade estiver incluído, ele será um intervalo semi-aberto:
  $[a,b[= \{x \in \mathbb{R} \mid a \leq x < b\}$

Nos casos em que os intervalos sejam sobrepostos e um número possa pertencer a mais de um intervalo, geralmente será necessário usar intervalos fechados.

No entanto, em alguns casos em que você não desejar ter intervalos sobrepostos, você desejará que um número pertença a apenas um intervalo. Dessa forma, ocorrerá um problema se um ponto for o fim de um intervalo e, ao mesmo tempo, o início do próximo. Um número com este valor será atribuído aos dois intervalos. Dessa forma, você desejará ter intervalos semi-abertos.

Uma solução prática para este problema é subtrair uma quantidade muito pequena do valor final de todos os intervalos, criando, assim, intervalos fechados, mas que não sejam sobrepostos. Se os números forem datas, a maneira mais simples de fazer isso é usar a função DayEnd(), que retorna o último milissegundo do dia:

```
Intervals: LOAD…, DayEnd(IntervalEnd – 1) as IntervalEnd From Intervals;
```

Também é possível subtrair uma pequena quantidade manualmente. Se você fizer isso, certifique-se de que o valor subtraído não é muito pequeno, já que a operação será arredondada para 52 dígitos binários significativos (14 dígitos decimais). Se você usar uma quantidade muito pequena, a diferença não será significativa e você usará novamente o número original.

4 Limpeza de dados

Existem situações em que a fonte de dados carregada no Qlik Sense não está necessariamente na forma como desejamos no aplicativo do Qlik Sense. O Qlik Sense oferece uma variedade de funções e comandos que permitem transformar os dados em um formato de acordo com nossas necessidades.

O mapeamento pode ser usado em um script do Qlik Sense para substituir ou modificar os valores ou nomes de campo durante a execução do script; portanto, o mapeamento pode ser usado para limpar os dados e torná-los mais consistentes ou para substituir partes ou a totalidade de um valor de campo.

Ao carregar dados de diferentes tabelas, os valores de campo que indicam a mesma coisa nem sempre são nomeados de forma consistente. Já que esta falta de consistência dificulta as associações, o problema precisa ser resolvido. Isso pode ser feito de uma maneira organizada e simples, criando uma tabela de mapeamento para a comparação dos valores de campo.

4.1 Tabelas de mapeamento

As tabelas carregadas usando carregamento de Mapping ou seleção de Mapping são tratadas de forma diferente de outras tabelas. Eles serão armazenados em uma área separada da memória e usados apenas como tabelas de mapeamento durante a execução do script. Depois da execução do script, essas tabelas são descartadas automaticamente.

Regras:

- A tabela de mapeamento deve ter duas colunas: a primeira contendo valores de comparação e a segunda, os valores de mapeamento desejados.
- As duas colunas devem ser nomeadas, mas os nomes não têm relevância neles mesmos. Os nomes de coluna não têm conexão com os nomes de campo nas tabelas internas normais.

4.2 Funções e comandos do Mapping

As seguintes funções/comandos de mapeamento serão abordadas neste tutorial:

• Prefixo Mapping

• ApplyMap()

• MapSubstring()

• Comando Map … Using

• Comando Unmap

4.3 Prefixo Mapping

O prefixo Mapping é usado em um script para criar uma tabela de mapeamento. Em seguida, a tabela de mapeamento pode ser usada com a função ApplyMap(), a função MapSubstring() ou o comando Map … Using.

Faça o seguinte:

1. Crie um novo aplicativo e nomeie-o.
2. Adicione uma nova seção de script no **Editor de carregamento de dados**.
3. Nomeie a seção como *Countries*.
4. Digite o seguinte script:

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

   A tabela *CountryMap* armazena duas colunas: *Country* e *NewCountry*. A coluna *Country* armazena as várias formas nas quais o país foi inserido no campo *Country*. A coluna *NewCountry* armazena a forma como os valores serão mapeados. Esta tabela de mapeamento será usada para armazenar valores de país *US* consistentes no campo *Country*. Por exemplo, se *U.S.A.* estiver armazenado no campo *Country*, mapeie-o para ser *US*.

4.4 ApplyMap() função

Use ApplyMap() para substituir os dados de um campo com base em uma tabela de mapeamento criada anteriormente. A tabela de mapeamento precisa ser carregada antes que a função ApplyMap() possa ser usada. Os dados na tabela *Data.xlsx* que você carregará são semelhantes aos seguintes:

Tabela de dados

ID	Nome	País	Código
1	John Black	U.S.A.	SDFGBS1DI
2	Steve Johnson	U.S.	2ABC
3	Mary White	Estados Unidos	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	US	KSD111DKFJ1

Observe que o país é inserido de várias maneiras. Para tornar o campo Country consistente, a tabela de mapeamento é carregada e, em seguida, a função **ApplyMap()** é usada.

Faça o seguinte:

1. Abaixo do script digitado acima, selecione e carregue *Data.xlsx* e insira o script.
2. Digite o seguinte acima do comando LOAD recém-criado:

   ```
   Data:
   ```

   Seu script deve ter a seguinte aparência:

   ```
   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
   ```

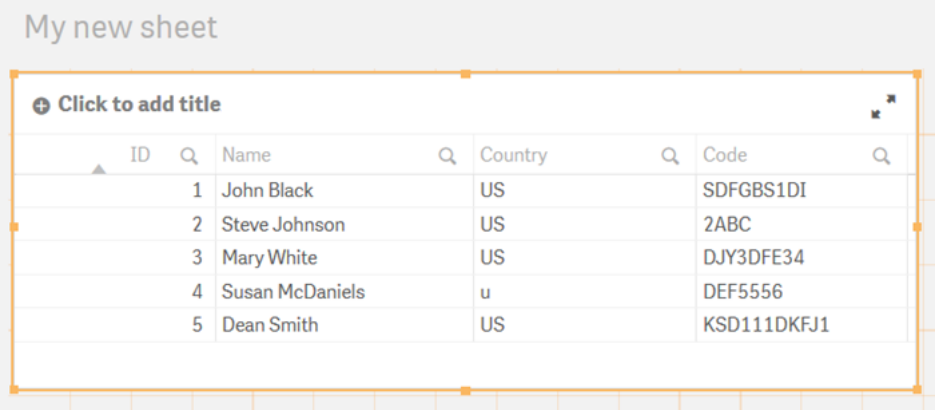
```
is Sheet1);
```

3. Modifique a linha que contém `Country`, da seguinte maneira:
   ```
   ApplyMap('CountryMap', Country) as Country,
   ```

   O primeiro parâmetro da função ApplyMap() contém o nome do mapa entre aspas simples. O segundo parâmetro é o campo que contém os dados que serão substituídos.

4. Clique em **Carregar dados**.
   A tabela resultante tem a seguinte aparência:

   Tabela mostrando dados carregados usando a função 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

   As várias grafias de *United States* foram alteradas para *US*. Existe um registro que não foi escrito corretamente; portanto, a função ApplyMap() não alterou esse valor de campo. Ao usar a função ApplyMap(), é possível usar o terceiro parâmetro para adicionar uma expressão padrão se a tabela de mapeamento não contiver um valor correspondente.

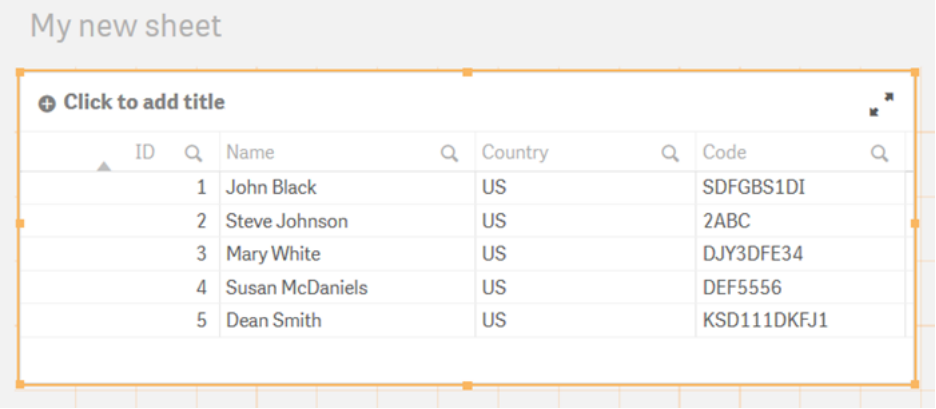
5. Adicione `'US'` como o terceiro parâmetro da função ApplyMap(), para lidar com os casos em que o país possa ter sido digitado incorretamente:
   ```
   ApplyMap('CountryMap', Country, 'US') as Country,
   ```

   Seu script deve ter a seguinte aparência:

   ```
   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. Clique em **Carregar dados**.
   A tabela resultante tem a seguinte aparência:

Tabela mostrando dados carregados usando a função 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

Para saber mais sobre ApplyMap(), consulte esta postagem de blog no Qlik Community: Don't join - use Applymap instead *(Não unir - usar Applymap em vez disso)*

4.5 MapSubstring() função

A função MapSubstring() permite mapear partes de um campo.

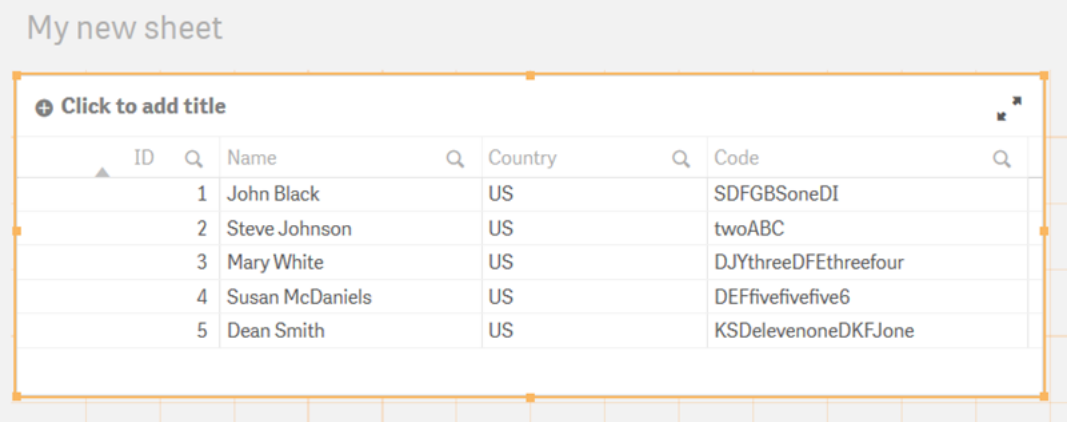
Na tabela criada por ApplyMap(), agora desejamos que os números serão escritos como texto; portanto, a função MapSubstring() será usada para substituir os dados numéricos pelo texto.

Para fazer isso, primeiro precisamos criar uma tabela de mapeamento.

Faça o seguinte:

1. Adicione as seguintes linhas de script à seção *CountryMap*, porém, antes da seção *Data*.
   ```
   CodeMap: MAPPING LOAD * INLINE [ F1, F2 1, one 2, two 3, three 4, four 5, five 11, eleven ];
   ```
   Na tabela *CodeMap*, os números 1 a 5 e 11 serão mapeados.
2. Na seção *Data* do script, modifique o comando `Code` da seguinte forma:
   ```
   MapSubString('CodeMap', Code) as Code
   ```
   Seu script deve ter a seguinte aparência:
   ```
   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. Clique em **Carregar dados**.
   A tabela resultante tem a seguinte aparência:

Tabela mostrando dados carregados usando a função 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

Os caracteres numéricos foram substituídos por texto no campo *Code*. Se um número aparecer mais de uma vez como ocorre com ID=3 e ID=4, o texto também será repetido. ID=4. *Susan McDaniels* tinha um 6 em seu código. Já que 6 não foi mapeado na tabela *CodeMap*, ele permanecerá inalterado. ID=5, *Dean Smith*, tinha 111 em seu código. Ele foi mapeado como 'elevenone'.

Para saber mais sobre MapSubstring(), consulte esta postagem de blog no Qlik Community: Mapping ... e não o tipo geográfico

4.6 Map ... Using

O comando Map ... Using também pode ser usado para aplicar um mapa a um campo. No entanto, ele funciona um pouco diferente de ApplyMap(). Enquanto ApplyMap() lida com o mapeamento todas as vezes que o nome do campo é encontrado, Map ... Using lida como o mapeamento quando o valor é armazenado embaixo do nome do campo na tabela interna.

Vamos dar uma olhada em um exemplo. Suponha que estivéssemos carregando o campo *Country* várias vezes no script e quiséssemos aplicar um mapa todas as vezes que o campo fosse carregado. A função ApplyMap() pode ser usada como ilustrada anteriormente neste tutorial ou Map ... Using pode ser usada.

Se Map ... Using for usada, o mapa será aplicada ao campo quando o campo estiver armazenado na tabela interna. Portanto, no exemplo descrito abaixo, o mapa é aplicado ao campo *Country* da tabela Data1, mas ele não seria aplicado ao campo *Country2* da tabela *Data2*. Isso ocorre porque o comando Map ... Using é aplicado somente a campos chamados *Country*. Quando o campo *Country2* é armazenado na tabela interna, ele não é mais chamado *Country*. Se desejar que o mapa seja aplicado à tabela *Country2*, será necessário usar a função ApplyMap().

O comando Unmap termina o comando Map ... Using, para que se *Country* fosse carregado após o comando Unmap, *CountryMap* não seria aplicado.

Faça o seguinte:

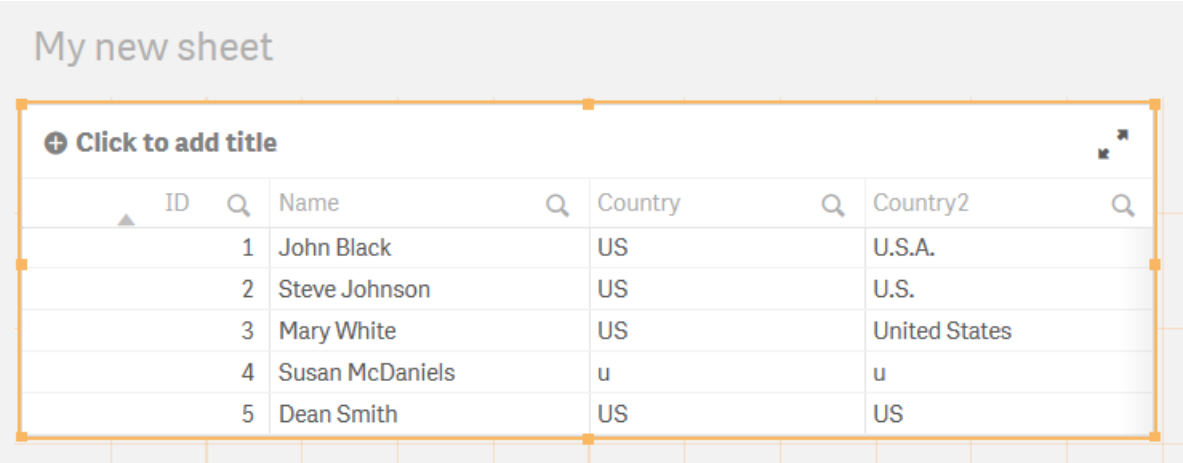
1. Substitua o script da tabela *Data* pelo seguinte:

```
Map Country Using CountryMap; Data1:     LOAD     ID,     Name,     Country FROM
[lib://AttachedFiles/Data.xlsx] (ooxml, embedded labels, table is Sheet1);
LOAD     ID,     Country as Country2 FROM [lib://AttachedFiles/Data.xlsx] (ooxml,
embedded labels, table is Sheet1); UNMAP;
```

2. Clique em **Carregar dados**.

   A tabela resultante tem a seguinte aparência:

   Tabela mostrando dados carregados usando a função 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 Manipulando dados hierárquicos

Hierarquias são uma parte importante de todas as soluções de inteligência empresarial, utilizadas para descrever dimensões que obviamente contêm diferentes níveis de granularidade. Algumas são simples e intuitivas, enquanto outras são complexas e exigem bastante planejamento para serem corretamente modeladas.

Da parte superior de uma hierarquia até a parte inferior, os membros são progressivamente mais detalhados. Por exemplo, em uma dimensão que tenha os níveis Mercado, País, Estado e Cidade, o membro Américas aparece no nível mais alto da hierarquia, o membro EUA aparece no segundo nível, o membro Califórnia aparece no terceiro nível e São Francisco no nível inferior. Califórnia é mais específico do que EUA e São Francisco é mais específico do que Califórnia.

Armazenar hierarquias em um modelo relacional é um desafio comum com várias soluções. Há várias abordagens:

- A hierarquia Horizontal
- O modelo de lista de Adjacência
- O método de enumeração do Caminho
- O modelo de conjuntos Aninhados
- A lista de Ancestrais

Para os fins deste tutorial, criaremos uma lista de Ancestrais, já que ela apresenta a hierarquia de uma forma que pode ser utilizada diretamente em uma consulta. Mais informações sobre as outras abordagens podem ser encontradas no Qlik Community.

5.1 Prefixo Hierarchy

O prefixo Hierarchy é um comando de script que é colocado na frente de um comandoLOAD ou SELECT que carrega uma tabela de nós adjacentes. Também aqui, o comando LOAD precisa ter pelo menos três campos: Um ID que seja uma chave exclusiva do nó, uma referência ao pai e um nome.

O prefixo transformará uma tabela carregada em uma tabela de nós expandidos; uma tabela que tenha um número de colunas adicionais; uma para cada nível da hierarquia.

Faça o seguinte:

1. Crie um novo aplicativo e nomeie-o.
2. Adicione uma nova seção de script no **Editor de carregamento de dados**.
3. Nomeie a seção como *Wine*.
4. Em **AttachedFiles** no menu direito, clique em **Selecionar dados**.
5. Carregue e, em seguida, selecione *Winedistricts.txt*.
6. Na janela **Selecionar dados de**, desmarque os campos *Lbound* e *RBound* para que não sejam carregados.
7. Clique em **Inserir script**.

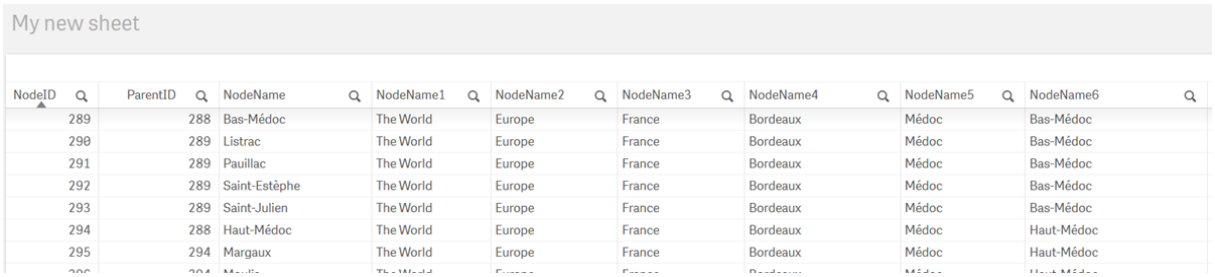
8. Digite o seguinte acima do comando LOAD:
   ```
   Hierarchy (NodeID, ParentID, NodeName)
   ```
   Seu script deve ter a seguinte aparência:
   ```
   Hierarchy (NodeID, ParentID, NodeName) LOAD     NodeID,     ParentID,     NodeName FROM
   [lib://AttachedFiles/Winedistricts.txt]  (txt, utf8, embedded labels, delimiter is '\t',
   msq);
   ```
9. Clique em **Carregar dados**.
10. Use a seção **Pré-visualização** do **visualizador do modelo de dados** para visualizar a tabela resultante.

    A tabela de nós expandidos resultante tem exatamente o mesmo número de registros que a sua tabela de origem: um por nó. A tabela de nós expandidos é muito prática, já que ela atende a alguns requisitos para a análise de uma hierarquia em um modelo relacional:

    - Todos os nomes de nós existem em uma e na mesma coluna; portanto, ela pode ser utilizada para pesquisas.
    - Além disso, os diferentes níveis de nós foram expandidos em um campo cada; campos que podem ser usados em grupos de hierarquia ou como dimensões em tabelas dinâmicas.
    - Além disso, os diferentes níveis de nós foram expandidos em um campo cada; campos que podem ser usados em grupos detalhados.
    - Ela pode ser criada para conter um caminho exclusivo para o nó, listando todos os ancestrais na ordem correta.
    - Ela pode ser criada para conter a profundidade do nó, isto é, a distância da raiz.

    A tabela resultante tem a seguinte aparência:

    Tabela mostrando amostra de dados carregados usando o prefixo 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	Pauillac	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

Para saber mais sobre hierarquias, consulte esta postagem de blog no Qlik Community: Hierarchies *(Hierarquias)*

5.2 Prefixo HierarchyBelongsTo

Assim como o prefixo Hierarchy, o prefixo HierarchyBelongsTo é um comando de script que é colocado na frente de um comando LOAD ou SELECT que carrega uma tabela de nós adjacentes.

Também aqui, o comando LOAD precisa ter pelo menos três campos: Um ID que seja uma chave exclusiva do nó, uma referência ao pai e um nome. O prefixo transformará a tabela carregada em uma tabela de ancestrais, uma tabela que tenha cada combinação de um ancestral e um descendente listado como um registro separado. Por isso, é muito fácil localizar todos os ancestrais ou todos os descendentes de um determinado nó.

Faça o seguinte:

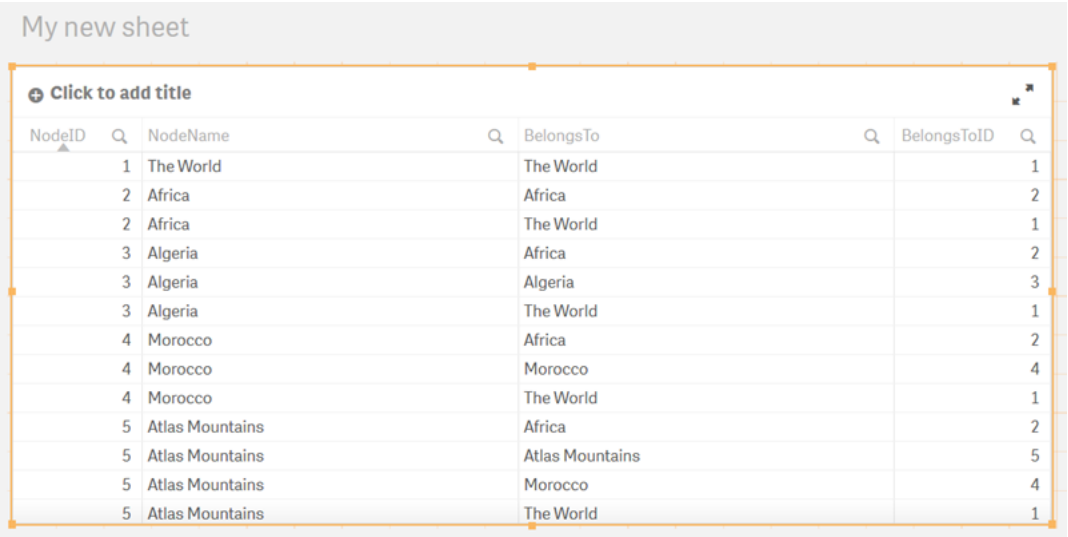
1. No **Editor de carregamento de dados**, modifique o comando Hierarchy para o seguinte:
   ```
   HierarchyBelongsTo (NodeID, ParentID, NodeName, BelongsToID, BelongsTo)
   ```
2. Clique em **Carregar dados**.
3. Use a seção **Pré-visualização** do **visualizador do modelo de dados** para visualizar a tabela resultante.

   A tabela de ancestrais atende a vários requisitos para a análise de uma hierarquia em um modelo relacional:

   - Se o ID do nó representar os nós individuais, o ID do ancestral representará as árvores inteiras e sub-árvores da hierarquia.
   - Todos os nomes de nós existem na função como nós e na função como árvores e as duas podem ser utilizadas para pesquisas.
   - Isso pode ser feito para conter a diferença de profundidade entre a profundidade do nó e a profundidade do ancestral, isto é, a distância entre a raiz da sub-árvore.

   A tabela resultante tem a seguinte aparência:

   Tabela mostrando dados carregados usando o prefixo 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

Autorização

Não é raro que uma hierarquia seja usada para autorização. Um exemplo é uma hierarquia organizacional. Cada gerente deve ter o direito de visualizar tudo o que pertence ao seu próprio departamento, incluindo todos os seus sub-departamentos. Mas eles não devem, necessariamente, ter o

direito de visualizar os outros departamentos.

Exemplo de hierarquia organizacional

Isso significa que diferentes pessoas poderão visualizar diferentes sub-árvores da organização. A tabela de autorização pode ter a seguinte aparência:

Tabela de autorização

ACCESS	NTNAME	PERSON	POSITION	PERMISSIONS
USER	ACME\JRL	John	CPO	HR
USER	ACME\CAH	Carol	CEO	CEO
USER	ACME\JER	James	Engenharia do Diretor	Engenharia
USER	ACME\DBK	Diana	CFO	Financeiro
USER	ACME\RNL	Bob	COO	Vendas.
USER	ACME\LFD	Larry	CTO	Produto

Neste caso, *Carol* pode visualizar tudo o que pertence ao *CEO* e outros; *Larry* pode visualizar a organização do *Product*; e *James* pode visualizar somente a organização *Engineering*.

Exemplo:

Muitas vezes, a hierarquia é armazenada em uma tabela de nós adjacentes. Neste exemplo, para resolver isso, você pode carregar a tabela de nós adjacentes usando um HierarchyBelongsTo e chamar o campo ancestral de *Tree*.

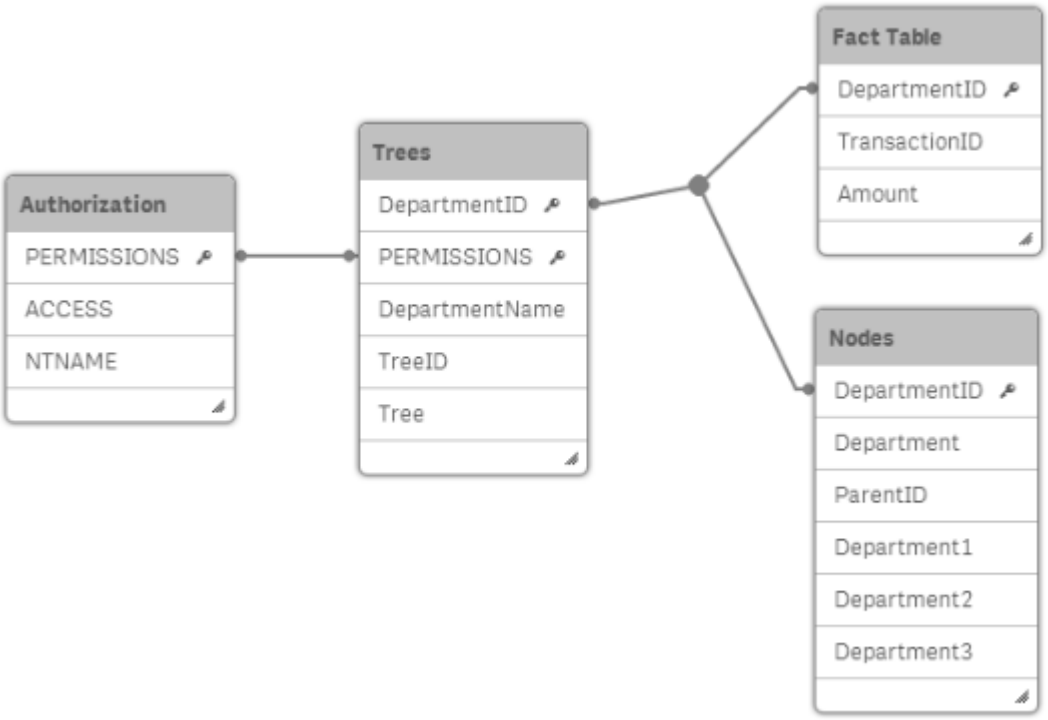
Se você quiser usar o Section Access, carregue uma cópia em maiúsculas de *Tree* e chame esse novo campo de *PERMISSIONS*. Por último, você precisa carregar a tabela de autorização. Essas duas últimas etapas podem ser realizadas usando as seguintes linhas de script. Observe que a tabela TempTrees é a tabela criada pelo comando HierarchyBelongsTo.

Observe que este é apenas um exemplo. Não há exercícios complementares a serem concluídos no Qlik Sense.

```
Trees: LOAD *,     Upper(Tree) as PERMISSIONS     Resident TempTrees; Drop Table TempTrees;
Section Access; Authorization: LOAD     ACCESS,         NTNAME,         UPPER(Permissions) as PERMISSIONS Fro
Organization; Section Application;
```

Este exemplo produziria o seguinte modelo de dados:

Modelo de dados: Tabelas Authorization, Trees, Fact e Nodes



6 Arquivos QVD

Um arquivo QVD (QlikView Data) é um arquivo que contém uma tabela de dados exportada do Qlik Sense ou do QlikView. O QVD é um formato nativo do Qlik e pode ser gravado e lido apenas pelo Qlik Sense ou QlikView. O formato de arquivo é otimizado para velocidade na leitura de dados de um script do Qlik Sense e ao mesmo tempo é compacto. A leitura de dados de um arquivo QVD é geralmente de 10 a 100 vezes mais rápida do que a leitura de outras fontes de dados.

Os arquivos QVD podem ser lidos em dois modos, padrão (rápido) e otimizado (mais rápido). O modo selecionado é determinado automaticamente pelo mecanismo de script do Qlik Sense. O modo otimizado pode ser usado apenas quando todos os campos são lidos sem transformações (fórmulas que atuam nos campos), embora a renomeação de campos seja permitida. Uma cláusula Where que faz o Qlik Sense descompactar os registros também desativará a carga otimizada.

Um arquivo QVD contém exatamente uma tabela de dados e é composto por três partes:

- Um cabeçalho XML (com o conjunto de caracteres UTF-8), que descreve os campos da tabela, o layout das informações posteriores e alguns outros metadados.
- Tabelas de símbolos em um formato com bytes.
- Dados da tabela em um formato com bits.

os arquivos QVD podem ser usados para vários fins. Pelo menos quatro usos principais podem ser facilmente identificados. Mais de um deles pode se aplicar em determinadas situações:

- Aumento da velocidade da carga de dados

  Se forem armazenados em buffer blocos de dados de entrada de arquivos QVD, que não mudam ou mudam aos poucos, a execução do script ficará consideravelmente mais rápida para conjuntos grandes de dados.

- Diminuindo a carga nos servidores de base de dados

  O volume de dados lidos de fontes de dados externas pode também ser bastante reduzido. Isso reduz a carga de trabalho dos bancos de dados externos e o tráfego de rede. Além disso, quando vários scripts Qlik Sense compartilham os mesmos dados, basta carregá-los uma vez do banco de dados de origem em um arquivo QVD. Os outros aplicativos podem usar os mesmos dados por meio desse arquivo QVD.

- Consolidação de dados de vários aplicativos Qlik Sense

  Com o comando de script Binary, é possível carregar dados de um único aplicativo Qlik Sense em outro aplicativo, mas com os arquivos QVD, um script do Qlik Sense é capaz de combinar dados de vários aplicativos do Qlik Sense. Isso aumenta as possibilidades de aplicativos que consolidam dados semelhantes de unidades de negócios diferentes etc.

- Carga incremental

  Em muitos casos comuns, a funcionalidade QVD pode ser usada para facilitar a carga incremental, isto é, para carregar exclusivamente novos registros de uma base de dados crescente.

6.1 Criando arquivos QVD

Os arquivos QVD podem ser criados de duas maneiras:

- Criação e nomeação explícitas, usando o comando Store no script Qlik Sense.
  Indique no script que uma tabela lida anteriormente ou parte dela deve ser exportada para um arquivo nomeado de forma explícita em um local de sua escolha.
- Criação e manutenção automáticas a partir do script.
  Se um comando load ou select for precedido do novo prefixo Buffer, o Qlik Sense criará automaticamente um arquivo QVD que, se determinadas condições forem atendidas, pode ser usado no lugar da fonte de dados original ao recarregar os dados.

Não há diferença entre os arquivos QVD resultantes, por exemplo, em relação à velocidade de leitura.

Store

Essa função de script cria um arquivo explicitamente nomeado QVD, CSV ou txt.

Sintaxe:

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

O comando só pode exportar campos de uma tabela lógica. Se os campos de várias tabelas forem exportados, uma junção explícita deve ser feita anteriormente no script para criar a tabela de dados que deve ser exportada.

Os valores de texto são exportados para o arquivo CSV no formato UTF-8. É possível especificar um separador. Para isso, consulte **LOAD**. O comando store de um arquivo CSV não suporta a exportação de BIFF .

Exemplos:

```
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';
```

Faça o seguinte:

1. Abra o aplicativo *Tutorial de uso de scripts avançado*.
2. Clique em *Product* na seção de script.
3. Adicione o seguinte ao final do script:
   ```
   Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
   ```
   Seu script deve ter a seguinte aparência:

```
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. Clique em **Carregar dados**.
   Agora, o arquivo *Product.qvd* deve constar na lista de arquivos.

   Este arquivo de dados é o resultado do script **Crosstable** e é uma tabela de três colunas, com uma coluna para cada categoria (Product, Month, Sales). Este arquivo de dados agora pode ser usado para substituir toda a seção do script de *Product* .

6.2 Lendo os dados de arquivos QVD

Um arquivo QVD pode ser lido ou acessado pelo Qlik Sense pelos seguintes métodos:

- Carregando um arquivo QVD como uma fonte de dados explícita. Os arquivos QVD podem ser referenciados por um comando load no script do Qlik Sense, como qualquer outro tipo de arquivo de texto (csv, fix, dif, biff, etc).

  Exemplos:

  ```
  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);
  ```

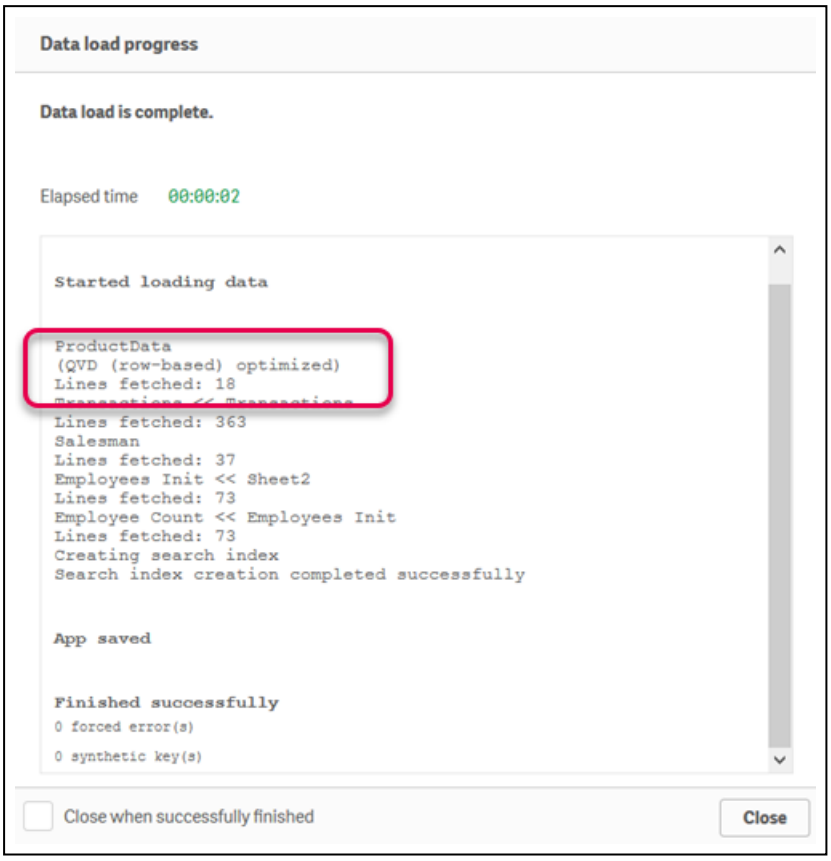
- Carregando automaticamente os arquivos QVD armazenados em buffer. Ao usar o prefixo buffer em comandos load ou select, nenhuma declaração explícita para leitura é necessária. O Qlik Sense determinará como usará os dados do arquivo QVD, ao contrário da obtenção de dados com o uso do comando original LOAD ou SELECT.
- Acessando arquivos QVD a partir do script. Várias funções de script (todas começando com QVD) podem ser usadas para recuperar diversas informações dos dados contidos no cabeçalho XML de um arquivo QVD.

Faça o seguinte:

1. Comente todo o script na seção de script do *Product*.
2. Digite o seguinte script:
   ```
   Load * from [lib://AttachedFiles/ProductData.qvd](qvd);
   ```
3. Clique em **Carregar dados**.

Os dados são carregados do arquivo QVD.

Janela do progresso do carregamento de dados



Para saber como usar arquivos QVD para cargas incrementais, consulte esta postagem de blog no Qlik Community: Overview of Qlik Incremental Loading *(Visão geral do carregamento incremental da Qlik)*

Buffer

Os arquivos QVD podem ser criados e mantidos automaticamente usando o prefixo Buffer. Esse prefixo pode ser usado com a maioria dos comandos LOAD e SELECT no script. Ele indica se os arquivos QVD serão usados para armazenar em cache/buffer o resultado do comando.

Sintaxe:

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

Se não for usada nenhuma opção, o buffer de QVD criado pela primeira execução do script será usado indefinidamente.

Exemplo:

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

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

Amount é um número que especifica o período de tempo. Decimals pode ser usado A unidade adotada será dias, se for omitida.

A opção stale after é geralmente usada com fontes do banco de dados nas quais não há nenhum carimbo de data/hora simples nos dados originais. Uma cláusula stale after simplesmente determina um período de tempo a partir da hora de criação do buffer do QVD, após a qual o buffer deixará de ser considerado válido. Antes desse período, o buffer de QVD será usado como fonte dos dados e, após o período, será usada a fonte de dados original. O arquivo do buffer de QVD será automaticamente atualizado e um novo período será iniciado.

Exemplo:

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

Incremental

A opção incremental permite a leitura apenas de parte de um arquivo subjacente. O tamanho anterior do arquivo é armazenado no cabeçalho XML no arquivo QVD. Isso é especificamente útil com arquivos de log. Todos os registros carregados anteriormente são lidos do arquivo QVD, enquanto que os novos registros posteriores são lidos na fonte original e, por último, um arquivo QVD atualizado é criado.

Observe que a opção incremental só pode ser usada com comandos LOAD e arquivos de texto e essa carga incremental não pode ser usada quando dados antigos são alterados ou excluídos.

Exemplo:

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

Os buffers de QVD normalmente serão excluídos quando deixarem de ser referenciados durante a execução completa do script no aplicativo que os criou ou quando o aplicativo que os criou não existir mais. O comando Store deve ser usado se você desejar manter o conteúdo do buffer como um arquivo QVD ou CSV.

Faça o seguinte:

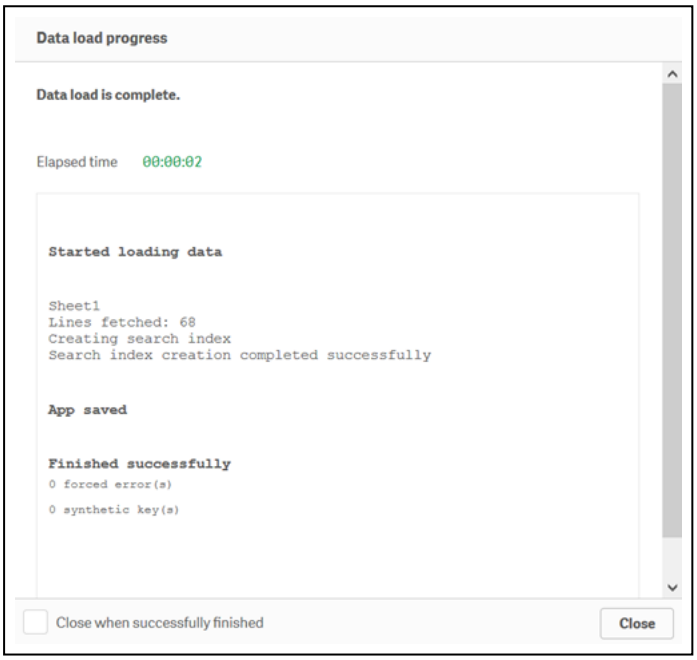
1. Crie um novo aplicativo e nomeie-o.
2. Adicione uma nova seção de script no **Editor de carregamento de dados**.
3. Em **AttachedFiles** no menu direito, clique em **Selecionar dados**.
4. Carregue e, em seguida, selecione *Cutlery.xlsx*.
5. Na janela **Selecionar dados de**, clique em **Inserir script**.
6. Comente os campos no comando de LOAD e altere o comando de LOAD para o seguinte:
   ```
   Buffer LOAD *
   ```

   Seu script deve ter a seguinte aparência:

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

7. Clique em **Carregar dados**.
   Na primeira vez em que você carrega dados, eles são carregados de *Cutlery.xlsx*.

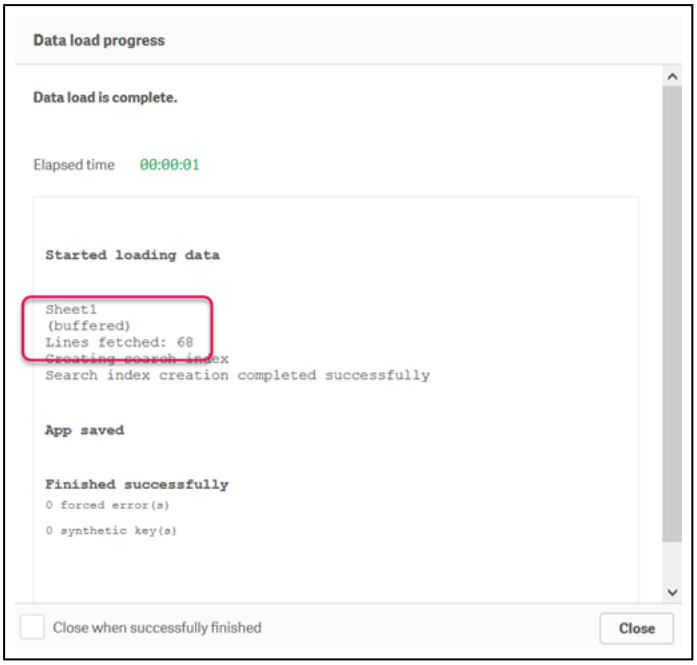
   Janela do progresso do carregamento de dados



   O comando Buffer também cria um QVD arquivo e o armazena no Qlik Sense. Em uma implementação Qlik Sense Enterprise on Windows, ele é armazenado em um diretório no servidor do Qlik Sense.

8. Clique em **Carregar dados** novamente.
9. Desta vez, os dados são carregados do arquivo QVD criado pelo comando Buffer quando você carregou os dados pela primeira vez.

Janela do progresso do carregamento de dados



6.3 Obrigado!

Você concluiu este tutorial e, com certeza, obteve um conhecimento básico sobre o uso de scripts no Qlik Sense. Acesse nosso site para obter mais informações sobre mais treinamentos disponíveis.