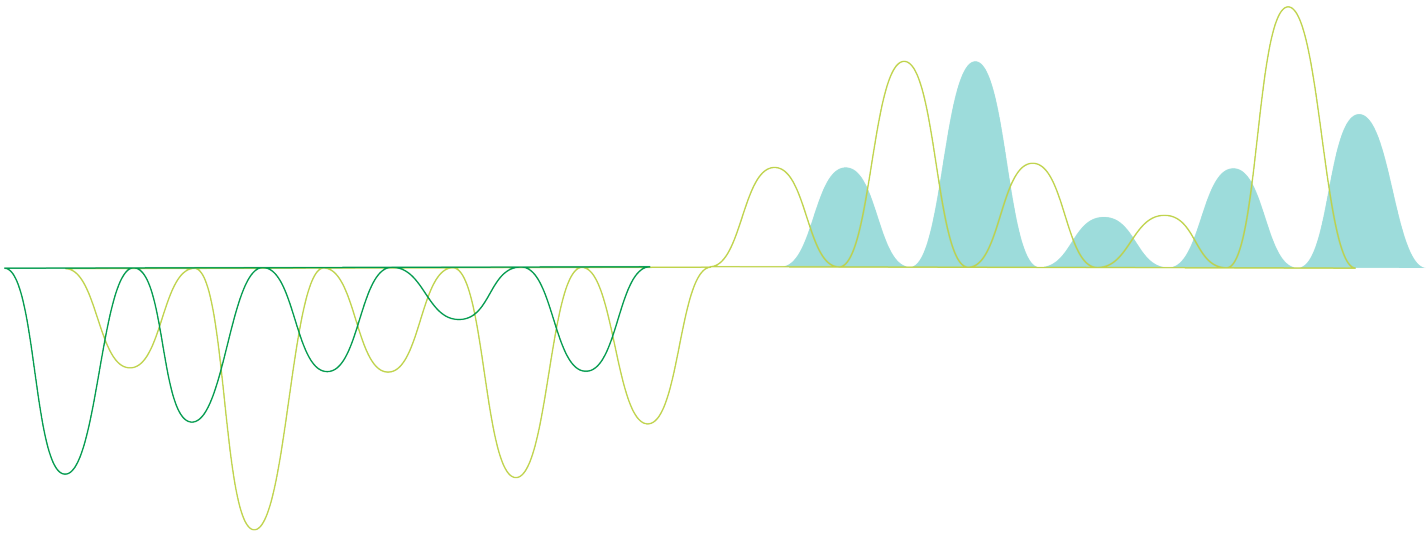
Kurs — kolejne etapy tworzenia skryptów

Qlik Sense®
November 2023



Contents

1 Witamy na kursie!

Witamy w kursie, podczas którego zostaną przedstawione zaawansowane opcje dotyczące skryptów w programie Qlik Sense.

Po zapoznaniu się z podstawowymi informacjami na temat skryptów w można przejść do bardziej zaawansowanych operacji na swoich danych ładowanych do programu Qlik Sense. Mogą to być na przykład przekształcenia danych za pomocą tabel krzyżowych, czyszczenie danych oraz tworzenie i ładowanie danych z plików danych Qlik nazywanych plikami QVD.

1.1 Czego się nauczysz?

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

1.2 Kto powinien wziąć udział w tym kursie?

Użytkownik powinien znać podstawy tworzenia skryptów w Qlik Sense. Oznacza to dysponowanie wiedzą w zakresie ładowania danych i ich obsługi.

Zalecamy ukończenie kursu Skrypty dla początkujących.

Wymagany jest dostęp do Edytora ładowania danych i dysponowanie możliwością ładowania danych w Qlik Sense Enterprise on Windows.

Niniejsze instrukcje są ogólnie dostosowane do rozwiązania Qlik Sense Cloud Business.

1.3 Zawartość pakietu

Pobrany pakiet zip zawiera następujące pliki danych, które są potrzebne do ukończenia kursu:

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

Pakiet zawiera również kopię aplikacji *Advanced Scripting Tutorial*. Dodatkowe sekcje skryptu w aplikacji zawierają skrypty dla innych aplikacji tworzonych w ramach niniejszego kursu. Aplikację można przesłać do swojego hubu.

Zalecamy samodzielne napisanie aplikacji zgodnie z instrukcjami w niniejszym kursie, co pozwoli zyskać największy zakres wiedzy i umiejętności. Dodatkowo wymagane będzie połączenie i przesłane plików danych w sposób opisany w kursie, aby zasoby danych działały.

W przypadku problemów można uzyskać pomoc w ich usunięciu. Wskazaliśmy, które segmenty skryptów są powiązane z daną lekcją.

1.4 Lekcje w kursie

W zależności od twojego doświadczenia w obsłudze Qlik Sense ukończenie tego kursu powinno trwać 3-4 godziny. Zagadnienia w kursie należy wykonywać kolejno. Jednak w każdej chwili można przerwać i kontynuować później. Na szczęście kurs jest pozbawiony testów.

Przekształcanie danych

Używanie prefiksu Crosstable

Łączenie tabel operatorami Join i Keep

Używanie funkcji międzywierszowych Funkcje Peek, Previous i Exists

Dopasowywanie interwałów i ładowanie iteracyjne

Czyszczenie danych

Obsługa danych hierarchicznych

Pliki QVD

1.5 Dodatkowe materiały i zasoby

- Qlik oferuje szeroką gamę zasobów, z których mogą korzystać osoby zainteresowane.
- Jest dostępna pomoc online Qlik.
- Szkolenia, w tym bezpłatne kursy online, są dostępne w Qlik Continuous Classroom.
- Fora dyskusyjne, blogi i więcej można znaleźć w Qlik Community.

2 Instrukcje LOAD i SELECT

Instrukcje LOAD i SELECT umożliwiają ładowanie danych do programu Qlik Sense. Każda z tych instrukcji powoduje wygenerowanie tabeli wewnętrznej. LOAD służy do ładowania danych z plików, a SELECT — do ładowania danych z baz danych.

W tym kursie używane będą dane z plików, więc używane będą instrukcje LOAD.

Załadowanymi danymi można również sterować za pomocą poprzedzającej instrukcji LOAD. Na przykład zmianę nazw pól należy wykonać w ramach instrukcji LOAD, natomiast instrukcja SELECT nie umożliwia żadnych zmian nazw pól.

Ładowanie danych do programu Qlik Sense podlega następującym regułom:

- Qlik Sense nie odróżnia tabel wygenerowanych przez instrukcję LOAD od tabel wygenerowanych instrukcją SELECT. W przypadku ładowania wielu tabel nie ma więc znaczenia, czy dana tabela została załadowana z użyciem instrukcji LOAD, instrukcji SELECT, czy też obu tych instrukcji.
- Z punktu widzenia logiki programu Qlik Sense kolejność pól w instrukcji i w pierwotnej tabeli w bazie danych jest nieistotna.
- W nazwach pól rozróżniana jest wielkość liter. Nazwy pól służą do określania powiązań pomiędzy tabelami danych. Z tego powodu czasami konieczna jest zmiana nazwy pól w skrypcie ładowania w celu uzyskania pożądanego modelu danych.

3 Przekształcanie danych

Przed użyciem danych w aplikacji można je przekształcać i obsługiwać w edytorze ładowania danych.

Jedną z zalet manipulowania danymi jest możliwość określenia, że ładowany jest tylko podzbiór danych z pliku, np. kilka kolumn z tabeli, co ułatwia obsługę danych. Dane można również załadować więcej niż raz, aby podzielić dane pierwotne na kilka nowych tabel logicznych. Możliwe jest również załadowanie danych z więcej niż jednego źródła oraz scalenie ich w jedną tabelę w programie Qlik Sense.

W poniższych ćwiczenia pokazano sposoby ładowania danych za pomocą prefiksu Crosstable. Dowiesz się również, jak łączyć tabele, używać funkcji międzywierszowych, takich jak Peek i Previous, a także jak ładować ten sam wiersz kilka razy przy użyciu opcji While Load.

3.1 Używanie prefiksu Crosstable

Tabela krzyżowa to często spotykany typ tabeli. Stanowi ona macierz wartości między dwiema ortogonalnymi listami danych nagłówka. W przypadku danych w tabeli krzyżowej można użyć prefiksu Crosstable, aby przetransformować je i utworzyć żądane pola.

Prefiks Crosstable

W poniższej tabeli *Product* każda kolumna odpowiada jednemu miesiącowi, wiersz jednemu produktowi.

Tabela produktów

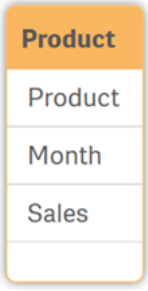
Produkt	Jan 2014	Feb 2014	Mar 2014	Apr 2014	May 2014	Jun 2014
A	100	98	100	83	103	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

Po załadowaniu tabeli wynikowo powstaje tabela z jednym polem dla *Product* i jednym polem dla każdego miesiąca.

Tabela Product z polem Product i jednym polem dla każdego miesiąca

Jeśli chcesz przeanalizować te dane, znacznie wygodniej jest dysponować wszystkimi liczbami w jednym polu i wszystkimi miesiącami w innym. W tym przypadku jest to tabela trzykolumnowa z jedną kolumną dla każdej kategorii (*Product, Month, Sales*).

Tabela Product z polami Product, Month i Sales



Użycie prefiksu Crosstable spowoduje przekształcenie danych w tabelę zawierającą jedną kolumnę dla kategorii *Month* i jedną dla kategorii *Sales*. Działanie prefiksu można też opisać jako przekształcenie nazw pól w wartości pól.

Wykonaj następujące czynności:

1. Utwórz nową aplikację i nazwij ją *Advanced Scripting Tutorial*.
2. Dodaj nową sekcję skryptu w **edytorze ładowania danych**.
3. Sekcji nadaj nazwę *Product*.
4. W sekcji **AttachedFiles** dostępnej po prawej stronie kliknij przycisk **Wybierz dane**.
5. Prześlij, a następnie wybierz plik *Product.xlsx*.
6. Wybierz tabelę *Product* w oknie **Wybierz dane z**.

   *Upewnij się, że w obszarze **Nazwy pól** zaznaczona jest opcja **Osadz. naz. pól**, aby podczas ładowania danych uwzględniać również nazwy pól tabeli.*

7. Kliknij polecenie **Wstaw skrypt**.

Skrypt powinien wyglądać następująco:

```
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);
```

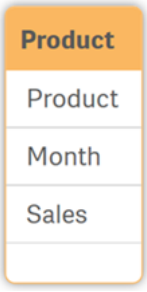
8. Kliknij polecenie **Ładuj dane**.
9. Otwórz **przeglądarkę modelu danych**. Model danych wygląda następująco:

   Tabela Product z polem Product i jednym polem dla każdego miesiąca

10. Kliknij kartę *Product* w **Edytorze ładowania danych**.
11. Wprowadź następujące dane powyżej instrukcji LOAD:
    `CrossTable(Month, Sales)`
12. Kliknij polecenie **Ładuj dane**.
13. Otwórz **przeglądarkę modelu danych**. Model danych wygląda następująco:

    Tabela Product z polami Product, Month i Sales

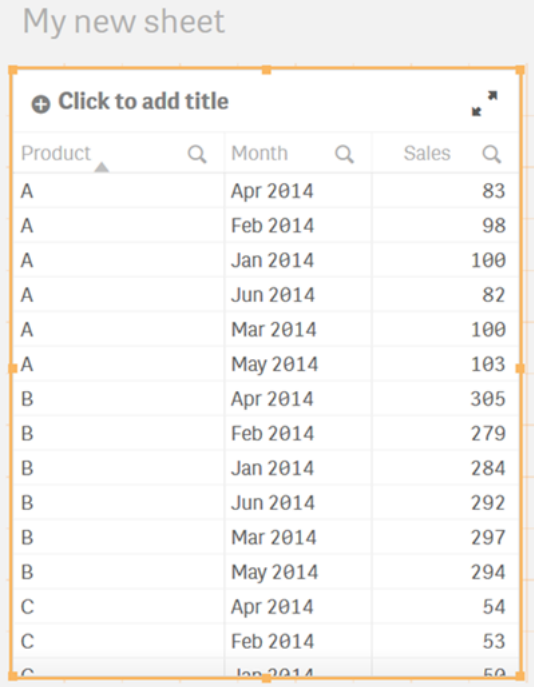
    

Należy pamiętać, że dane wejściowe zazwyczaj zawierają tylko jedną kolumnę jako pole kwalifikatora; jako klucz wewnętrzny (*Product* w powyższym przykładzie). Takich pól może jednak być kilka. W takim przypadku wszystkie pola kwalifikujące muszą być wymienione przed polami atrybutów w instrukcji

LOAD, a liczbę pól kwalifikujących należy określić trzecim parametrem prefiksu Crosstable. Poprzedni LOAD ani prefiks nie może występować przed słowem kluczowym Crosstable. Jednak można użyć opcji automatycznego konkatenowania.

W tabeli w Qlik Sense dane wyglądają następująco:

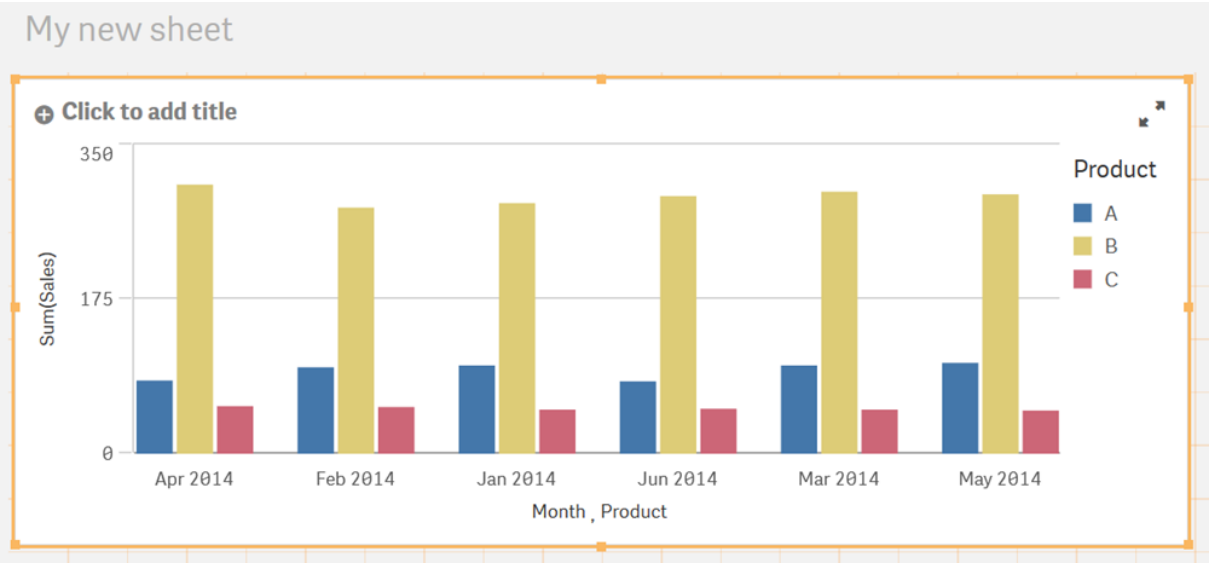
Tabela przedstawia dane załadowane przy użyciu prefiksu Crosstable



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

Teraz na podstawie danych można na przykład utworzyć wykres słupkowy:

Wykres słupkowy przedstawiający dane załadowane przy użyciu prefiksu Crosstable



Aby dowiedzieć się więcej na temat funkcji Crosstable, zapoznaj się z tym artykułem na blogu Qlik Community: The Crosstable Load (Ładowanie tabeli krzyżowej). *Działania są omawiane w kontekście rozwiązania QlikView. Logika jednak odnosi się również do Qlik Sense.*

Interpretacja liczbowa nie zadziała w przypadku pól atrybutów. Oznacza to, że miesiące podane jako nagłówki kolumn nie będą automatycznie interpretowane. Można to obejść, stosując prefiks Crosstable w celu utworzenia tabeli tymczasowej i uruchomienia drugiego przejścia w celu przeprowadzenia interpretacji zgodnie z poniższym przykładem.

Należy pamiętać, że jest to tylko przykład. Nie istnieją żadne ćwiczenia dodatkowe, które należy wykonać w 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;
```

Opróżnianie pamięci podręcznej

Można usunąć utworzone tabele, aby wyczyścić pamięć podręczną. Po załadowaniu danych do pamięci tymczasowej należy ją opróżnić, gdy przechowywane w niej dane nie są już potrzebne. Na przykład:

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

Możesz również opróżnić pola. Na przykład:

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

Jak widać, słowa kluczowe TABLE i FIELD mogą występować w liczbie pojedynczej i mnogiej.

3.2 Łączenie tabel operatorami Join i Keep

Sprzężenie to operacja polegająca na połączeniu dwóch tabel w jedną. Rekordami tabeli wynikowej są kombinacje rekordów oryginalnych tabel, zazwyczaj połączone w taki sposób, że rekordy składające się na poszczególne kombinacje w tabeli wynikowej mają wspólną wartość w jednym lub wielu polach wspólnych — jest to tzw. sprzężenie naturalne. W Qlik Sense połączenia mogą być wykonywane w skrypcie, co powoduje tworzenie tabel logicznych.

Tabele można sprzęgać bezpośrednio w skrypcie. Logika aplikacji Qlik Sense nie będzie wtedy widzieć poszczególnych tabel, tylko wynik sprzężenia w postaci pojedynczej tabeli wewnętrznej. W niektórych sytuacjach takie zachowanie jest pożądane, ale ma ono swoje wady:

- załadowane tabele często stają się większe, a aplikacja Qlik Sense działa wolniej;
- niektóre informacje mogą zostać utracone, na przykład częstotliwość (liczba rekordów) z pierwotnej tabeli.

Aby ograniczyć liczbę sytuacji wymagających jawnego sprzęgania tabel, stworzono funkcję Keep, której działanie polega na zredukowaniu jednej lub obu tabel do części wspólnej danych przed zapisaniem tych tabel w aplikacji Qlik Sense.

W tej dokumentacji termin „sprzężenie" jest zazwyczaj używany w odniesieniu do sprzężeń wykonywanych przed utworzeniem tabel wewnętrznych. Skojarzenie dokonywane po utworzeniu tabel wewnętrznych w gruncie rzeczy też jest jednak sprzężeniem.

Join

Najprostszym sposobem na uzyskanie sprzężenia jest zastosowanie w skrypcie prefiksu Join, który powoduje wykonanie sprzężenia tabeli wewnętrznej z inną wskazaną tabelą lub z ostatnio utworzoną tabelą. Wykonywane jest sprzężenie zewnętrzne, które daje wszystkie możliwe kombinacje wartości z obu tabel.

Przykład:

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

Wynikowa tabela wewnętrzna zawiera pola a, b, c oraz d. Liczba rekordów zależy od wartości pól w obu tabelach.

Nazwy pól, według których wykonywane jest sprzężenie, muszą być identyczne. Liczba pól, według których wykonywane jest sprzężenie, jest dowolna. Zazwyczaj tabele mają wspólne jedno pole lub kilka pól. Brak pola wspólnego spowoduje zwrócenie iloczynu kartezjańskiego tabel. Teoretycznie mogą też być wspólne wszystkie pola, ale zazwyczaj nie ma to sensu. Jeśli w instrukcji Join nie zostanie podana nazwa wcześniej załadowanej tabeli, prefiks Join użyje ostatnio utworzonej tabeli. Kolejność tych dwóch instrukcji nie jest zatem dowolna.

Używanie funkcji Join

Jawne użycie prefiksu Join w języku skryptowym Qlik Sense powoduje wykonanie pełnego sprzężenia dwóch tabel. Wynikiem jest pojedyncza tabela. Takie połączenia mogą często prowadzić do powstania bardzo dużych tabel.

Wykonaj następujące czynności:

1. Otwórz aplikację *Advanced Scripting Tutorial*.
2. Dodaj nową sekcję skryptu w **edytorze ładowania danych**.
3. Wywołaj sekcję *Transactions*.
4. W sekcji **AttachedFiles** dostępnej po prawej stronie kliknij przycisk **Wybierz dane**.
5. Prześlij, a następnie wybierz *Transactions.csv*.

   *Upewnij się, że w obszarze **Nazwy pól** zaznaczona jest opcja **Osadz. naz. pól**, aby podczas ładowania danych uwzględniać również nazwy pól tabeli.*

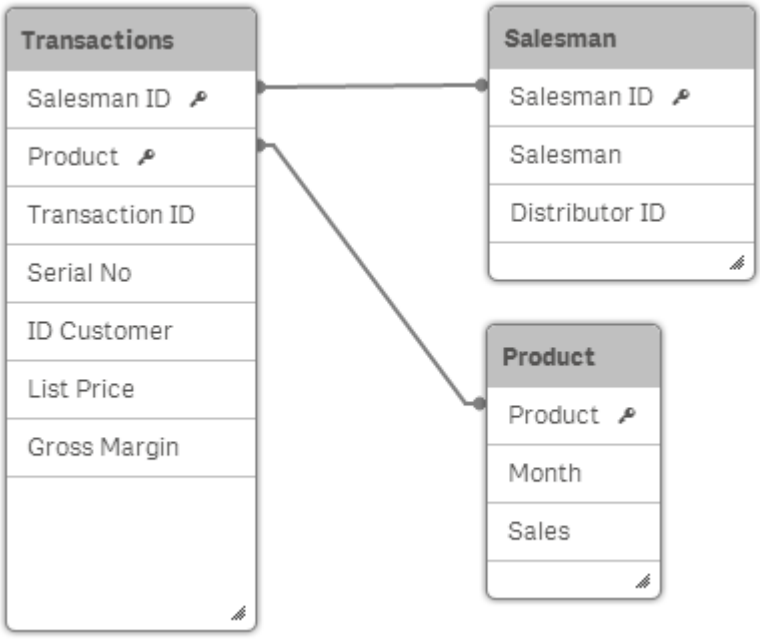
6. W oknie **Wybierz dane z** kliknij przycisk **Wstaw skrypt**.
7. Prześlij, a następnie wybierz *Salesman.xlsx*.
8. W oknie **Wybierz dane z** kliknij przycisk **Wstaw skrypt**.

   Skrypt powinien wyglądać następująco:

```
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. Kliknij polecenie **Ładuj dane**.
10. Otwórz **przeglądarkę modelu danych**. Model danych wygląda następująco:

Model danych: Tabele Transactions, Salesman i Product



Jednak osobne tabele *Transactions* i *Salesman* mogą nie być wymaganym rezultatem. Lepsze może być połączenie tych dwóch tabel.

Wykonaj następujące czynności:

1. Aby ustawić nazwę tabeli połączonej, dodaj następujący wiersz nad pierwszą instrukcją LOAD:
   `Transactions:`
2. Aby połączyć tabele *Transactions* i *Salesman*, dodaj następujący wiersz nad drugą instrukcją LOAD:
   `Join(Transactions)`

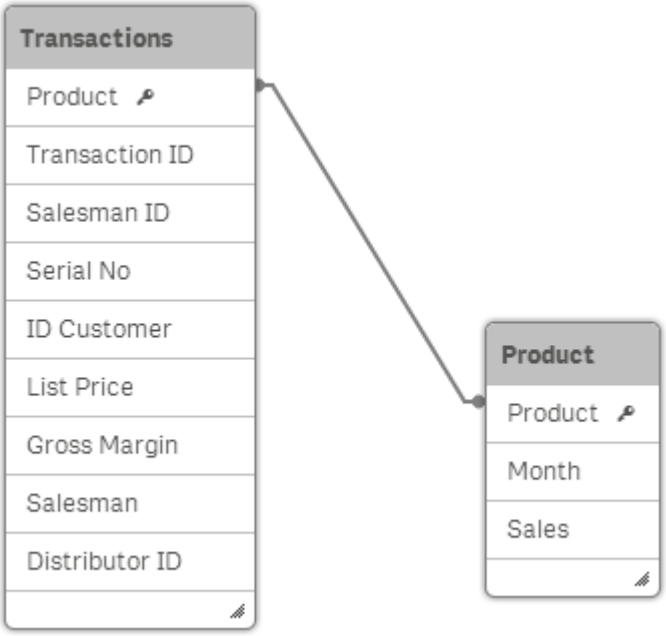
   Skrypt powinien wyglądać następująco:

```
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. Kliknij polecenie **Ładuj dane**.
4. Otwórz **przeglądarkę modelu danych**. Model danych wygląda następująco:

   Model danych:Tabele Transactions i Product



Wszystkie pola tabel *Transactions* i *Salesman* są teraz połączone w jednej tabeli *Transactions*.

Aby dowiedzieć się więcej na temat korzystania z Join, zapoznaj się z tymi artykułami na blogu Qlik Community: To Join or not to Join*,* Mapping as an Alternative to Joining (Łaczyć czy nie łączyć, Mapowanie jako alternatywa do łączenia)*. Działania są omawiane w kontekście rozwiązania QlikView. Logika jednak odnosi się również do Qlik Sense.*

Keep

Jedną z kluczowych zalet aplikacji Qlik Sense jest możliwość tworzenia skojarzeń między tabelami zamiast wykonywania sprzężeń. Pozwala to ograniczyć wymagania pamięciowe, przyspiesza działanie programu i daje ogromną elastyczność działania. Funkcję Keep zaprojektowano z myślą o ograniczeniu liczby sytuacji wymagających używania jawnych połączeń.

Przedrostek Keep pomiędzy dwoma instrukcjami LOAD lub SELECT redukuje jedną lub obie tabele do przecięcia danych tabeli przed zapisaniem ich w Qlik Sense. Prefiks Keep musi być zawsze poprzedzony jednym z następujących słów kluczowych: Inner, Left lub Right. Rekordy są wybierane z tabel w taki sam sposób, jak przy analogicznym sprzężeniu. Tabele nie są jednak sprzęgane i pozostaną zapisane w aplikacji Qlik Sense jako dwie odrębne tabele nazwane.

Inner

Prefiksy Join i Keep skryptu ładowania danych mogą być poprzedzone prefiksem Inner

Podanie go przed prefiksem Join oznacza, że do tabel ma być zastosowane sprzężenie wewnętrzne. Wynikowa tabela zawiera tylko te kombinacje wierszy z obu tabel, które mają po obu stronach pełen zestaw danych.

Podanie go przed prefiksem Keep oznacza, że przed zapisaniem wyniku w aplikacji Qlik Sense tabele mają zostać zredukowane do części wspólnej danych.

Przykład:

W tych przykładach użyjemy tabel źródłowych *Table1* i *Table2*.

Należy pamiętać, że są to tylko przykłady. Nie istnieją żadne ćwiczenia dodatkowe, które należy wykonać w Qlik Sense.

Table 1

A	**B**
1	aa
2	cc
3	ee

Table2

A	**C**
1	xx
4	yy

Inner Join

Najpierw wykonamy na tabelach operację Inner Join, której wynikiem będzie tabela *VTable* zawierająca tylko jeden wiersz (jedyny rekord występujący w obu tabelach) z połączonymi danymi z obu tabel.

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

VTable

A	**B**	**C**
1	aa	xx

Inner Keep

Natomiast jeśli na tabelach zostanie wykonana operacja Inner Keep, nadal będziemy mieli dwie tabele. Dwie tabele są skojarzone poprzez wspólne pole *A*.

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

VTab1

A	B
1	aa

VTab2

A	C
1	xx

Left

Prefiksy Join i Keep skryptu ładowania danych mogą być poprzedzone prefiksem left.

Podanie go przed prefiksem Join oznacza, że do tabel ma być zastosowane sprzężenie lewe (Left Join). Wynikowa tabela zawiera tylko te kombinacje wierszy z obu tabel, które mają pełen zestaw danych z pierwszej tabeli.

Podanie go przed prefiksem Keep oznacza, że przed zapisaniem wyniku w aplikacji Qlik Sense druga tabela ma zostać zredukowana do części wspólnej z pierwszą tabelą.

Przykład:

W tych przykładach użyjemy tabel źródłowych *Table1* i *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Najpierw wykonamy na tabelach operację Left Join, której wynikiem będzie tabela *VTable* zawierająca wszystkie wiersze z tabeli *Table1* połączone z polami z pasujących wierszy w tabeli *Table2*.

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

VTable

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

Natomiast jeśli na tabelach zostanie wykonana operacja Left Keep, nadal będziemy mieli dwie tabele. Dwie tabele są skojarzone poprzez wspólne pole *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

Prefiksy Join i Keep języka skryptowego Qlik Sense mogą być poprzedzone prefiksem right.

Podanie go przed prefiksem Join oznacza, że do tabel ma być zastosowane sprzężenie prawe (Right Join). Wynikowa tabela zawiera tylko te kombinacje wierszy z obu tabel, które mają pełen zestaw danych z drugiej tabeli.

Podanie go przed prefiksem Keep oznacza, że przed zapisaniem wyniku w aplikacji Qlik Sense pierwsza tabela ma zostać zredukowana do części wspólnej z drugą tabelą.

Przykład:

W tych przykładach użyjemy tabel źródłowych *Table1* i *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Najpierw wykonamy na tabelach operację Right Join, której wynikiem będzie tabela *VTable* zawierająca wszystkie wiersze z tabeli *Table2* połączone z polami z pasujących wierszy w tabeli *Table1*.

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

VTable

A	B	C
1	aa	xx
4	-	yy

Natomiast jeśli na tabelach zostanie wykonana operacja Right Keep, nadal będziemy mieli dwie tabele. Dwie tabele są skojarzone poprzez wspólne pole *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 Używanie funkcji międzywierszowych Peek, Previous i Exists

Tych funkcji używa się, gdy obliczenie wartości bieżącego rekordu wymaga wartości z wcześniej załadowanych rekordów danych.

W tej części kursu zajmiemy się funkcjami Peek(), Previous() i Exists().

Peek()

Peek() wyszukuje wartość pola w tabeli z wiersza, który został już załadowany. Numer wiersza może być określony, podobnie jak tabela. Jeśli nie określono numeru wiersza, zostanie użyty ostatnio załadowany rekord.

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

Wiersz musi być liczbą całkowitą. 0 oznacza pierwszy rekord, 1 drugi rekord itd. Liczby ujemne określają kolejność od końca tabeli. -1 oznacza ostatni wczytany rekord.

Jeśli nie zostanie podany wiersz, przyjmowana jest wartość -1.

Tablename jest etykietą tabeli bez końcowego dwukropka. Jeśli argument *tablename* nie zostanie podany, przyjmowana jest bieżąca tabela. W przypadku używania poza instrukcją **LOAD** lub podczas odwołania do innej tabeli należy uwzględnić *tablename*.

Previous()

Funkcja **Previous()** wyszukuje wartość wyrażenia **expr** przy użyciu danych z poprzedniego rekordu wejściowego, który nie został odrzucony z powodu klauzuli **where**. W przypadku pierwszego wiersza tabeli wewnętrznej funkcja zwróci wartość NULL.

Składnia:
```
Previous(expression)
```

Funkcja Previous() może być zagnieżdżona w celu uzyskiwania dostępu do bardziej odległych wierszy. Dane są pobierane bezpośrednio ze źródła danych wejściowych, co umożliwia odwoływanie się również do pól, które nie zostały załadowane do programu Qlik Sense, czyli nawet gdy nie zostały zapisane w powiązanej bazie danych.

Exists()

Funkcja **Exists()** określa, czy podana wartość pola została już załadowana w polu w skrypcie ładowania danych. Funkcja zwraca wartość TRUE lub FALSE, dzięki czemu może zostać użyta w klauzuli **where** instrukcji **LOAD** lub instrukcji **IF**.

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

Pole musi istnieć w danych załadowanych do tej pory przez skrypt. *Expression* jest wyrażeniem odwołującym się do poszukiwanej wartości pola w określonym polu. Pominięcie tego argumentu odpowiada przyjęciu we wskazanym polu wartości bieżącego rekord.

Używanie funkcji Peek() i Previous()

W najprostszej postaci funkcje Peek() i Previous() służą do identyfikowania konkretnych wartości w tabeli. Poniżej znajduje się próbka danych w tabeli *Employees*, która zostanie załadowana w tym ćwiczeniu.

Przykładowe dane z tabeli pracowników

Data	Zatrudnienie	Zwolnienie
1/1/2011	6	0
2/1/2011	4	2
3/1/2011	6	1
4/1/2011	5	2

Obecnie gromadzone są tylko dane dotyczące miesięcy, osób zatrudnianych i kończących zatrudnienie, dlatego dodamy pola dla wartości *Employee Count* i *Employee Var*, używając funkcji Peek() i Previous(), aby sprawdzić różnicę łącznej liczby pracowników w ujęciu miesięcznym.

Wykonaj następujące czynności:

1. Otwórz aplikację *Advanced Scripting Tutorial*.
2. Dodaj nową sekcję skryptu w **edytorze ładowania danych**.
3. Wywołaj sekcję *Employees*.
4. W sekcji **AttachedFiles** dostępnej po prawej stronie kliknij przycisk **Wybierz dane**.
5. Prześlij, a następnie wybierz *Employees.xlsx*.

   Upewnij się, że w obszarze Field names wybrana jest opcja Embedded field names, co pozwala na uwzględnienie nazw pól tabel podczas ładowania danych.

6. W oknie **Wybierz dane z** kliknij przycisk **Wstaw skrypt**.

   Skrypt powinien wyglądać następująco:

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

7. Zmodyfikuj skrypt w taki sposób, aby wyglądał następująco:

   ```
   [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);
```

Daty w polu *Date* w arkuszu Excel są zapisane w formacie MM/DD/RRRR. Aby upewnić się, że daty są poprawnie interpretowane z użyciem formatu ze zmiennych systemowych, względem pola *Date* stosowana jest funkcja Date.

Funkcja Peek() umożliwia zidentyfikowanie dowolnej wartości załadowanej dla zdefiniowanego pola. W tym wyrażeniu najpierw sprawdzamy, czy wartość rowno() jest równa 1. Jeżeli tak, to *Employee Count* nie będzie istnieć, więc wypełnimy pole z różnicą między *Hired* a *Terminated*.

Jeśli funkcja rowno() jest większa od 1, sprawdzamy wartość *Employee Count* z ostatniego miesiąca i dodajemy tę liczbę do różnicy liczby pracowników *Hired* minus *Terminated* z tego miesiąca.

Zauważ też, że w funkcji Peek() używamy (-1). Dla aplikacji Qlik Sense oznacza to, że ma sprawdzić rekord nad bieżącym rekordem. Jeśli wartość (-1) nie zostanie podana, aplikacja Qlik Sense przyjmie, że wymagane jest sprawdzenie poprzedniego rekordu.

8. Dodaj następujący kod na końcu skryptu:

```
[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];
```

Funkcja Previous() umożliwia zidentyfikowanie ostatniej wartości załadowanej dla zdefiniowanego pola. W wyrażeniu tym najpierw sprawdzamy, czy wartość rowno() jest równa 1. Jeżeli tak, to wiemy, że nie będzie istnieć Employee Var, ponieważ nie ma żadnego zapisu dla poprzedniego miesiąca w przypadku sekcji *Employee Count*. Więc dla tej wartości wpisujemy 0.

Jeśli funkcja rowno() jest większa niż 1, wiemy, że będzie istniała wartość *Employee Var*, więc sprawdzamy wartość *Employee Count* z ostatniego miesiąca i odejmujemy tę liczbę od wartości *Employee Count* z bieżącego miesiąca, aby utworzyć wartość w polu *Employee Var*.

Skrypt powinien wyglądać następująco:

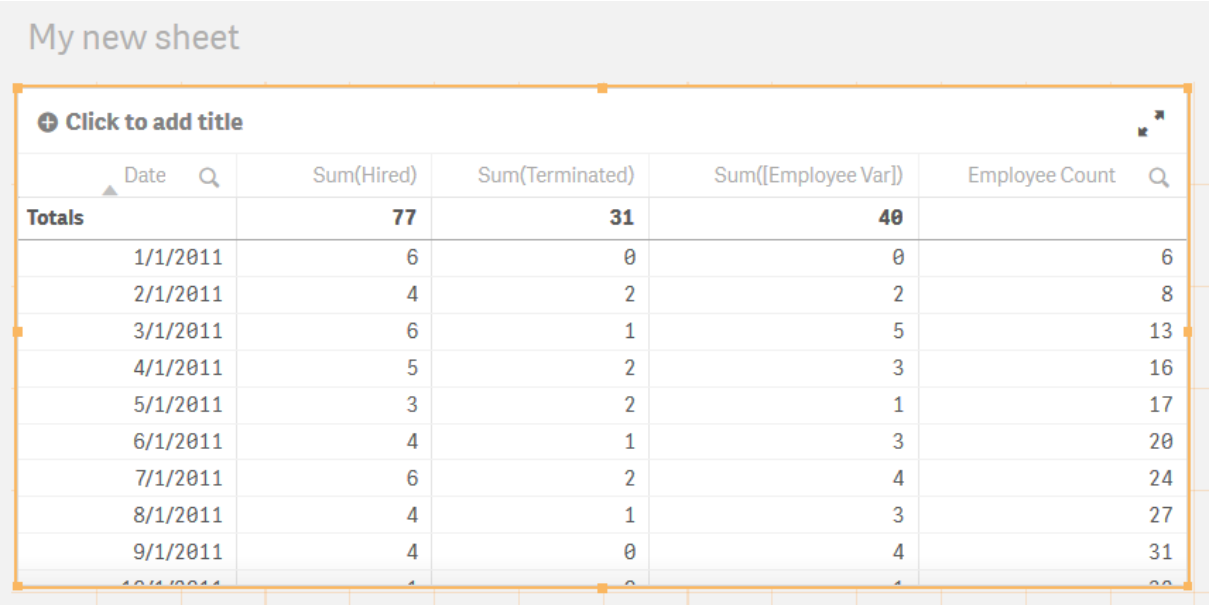
```
[Employees Init]:
LOAD
    rowno() as Row,
    Date(Date) as Date,
    Hired,
    Terminated,
    If(rowno()=1, Hired-Terminated,  peek([Employee Count], -1)+(Hired-Terminated)) as
[Employee Count]
FROM [lib://AttachedFiles/Employees.xlsx]
```

```
(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. Kliknij polecenie **Ładuj dane**.

   W nowym arkuszu w przeglądzie aplikacji utwórz tabelę z kolumnami *Date, Hired, Terminated, Employee Count* i *Employee Var*. Wynikowa tabela powinna wyglądać następująco:

   W poniższej tabeli przedstawiono wykorzystanie funkcji Peek i Previous w skrypcie



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

Funkcje Peek() i Previous() pozwalają wskazać zdefiniowane wiersze w tabeli. Największa różnica między tymi dwiema funkcjami polega na tym, że funkcja Peek() umożliwia użytkownikowi sprawdzanie pola, które nie było poprzednio ładowane do skryptu, natomiast funkcja Previous() może tylko sprawdzać pola, które zostały poprzednio załadowane. Funkcja Previous() działa na danych wejściowych dla instrukcji LOAD, natomiast funkcja Peek() działa na danych wyjściowych instrukcji LOAD. (Podobna różnica występuje między funkcjami RecNo() i RowNo()). Oznacza to, że obie te funkcje będą zachowywać się inaczej, jeśli istnieje klauzula Where.

Więc funkcja Previous() jest bardziej odpowiednia, kiedy wymagane jest wyświetlenie aktualnej wartości w porównaniu z poprzednią wartością. W przykładzie obliczyliśmy wariancję liczby pracowników z miesiąca na miesiąc.

Funkcja Peek() jest lepsza, gdy pole docelowe nie zostało wcześniej załadowane do tabeli lub gdy wybierany jest konkretny wiersz. Zostało to przedstawione w przykładzie, w którym obliczyliśmy wartość *Employee Count*, sprawdzając wartość *Employee Count* z poprzedniego miesiąca i dodając różnicę między liczbą zatrudnionych a zwolnionych pracowników dla miesiąca bieżącego. Należy pamiętać, że wartość *Employee Count* nie była polem w pierwotnym pliku.

Aby dowiedzieć się więcej na temat używania funkcji Peek() i Previous(), zapoznaj się z tym artykułem na blogu Qlik Community: Peek() vs Previous() – When to Use Each*. Działania są omawiane w kontekście rozwiązania QlikView. Logika jednak odnosi się również do Qlik Sense.*

Używanie funkcji Exists()

Funkcja Exists() jest często używana z klauzulą Where w skrypcie w celu ładowania danych w sytuacji, gdy dane powiązane zostały już załadowane w modelu danych.

W poniższym przykładzie używamy również funkcji Dual() w celu przypisania wartości liczbowych do łańcuchów.

Wykonaj następujące czynności:

1. Utwórz nową aplikację i nadaj jej nazwę.
2. Dodaj nową sekcję skryptu w **edytorze ładowania danych**.
3. Wywołaj sekcję *People*.
4. Wprowadź następujący skrypt:

```
//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;
```

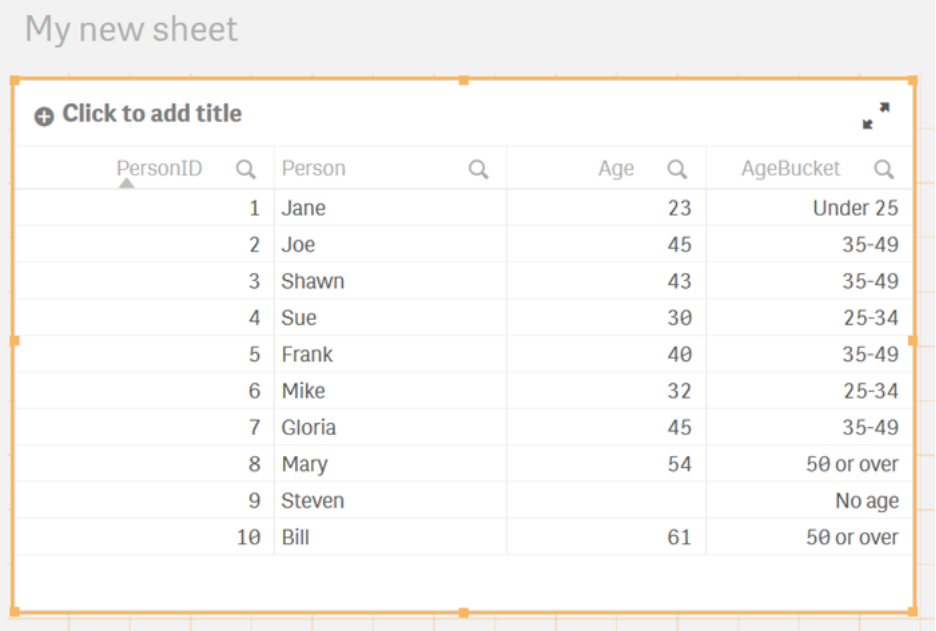
5. Kliknij polecenie **Ładuj dane**.

   W skrypcie pola *Age* i *AgeBucket* są ładowane tylko wtedy, gdy identyfikator *PersonID* został już załadowany w modelu danych.

   Zwróć uwagę na to, że tabela *AgeTemp* zawiera wartości wieku dla identyfikatorów *PersonID* 11 i 12, ale ponieważ te identyfikatory nie zostały załadowane w modelu danych (w tabeli *People*), dlatego zostały wykluczone przez klauzulę Where Exists(PersonID). Tę klauzulę można również zapisać następująco: Where Exists(PersonID, PersonID).

   Dane wyjściowe skryptu wyglądają następująco:

Tabela po użyciu Exists w skrypcie

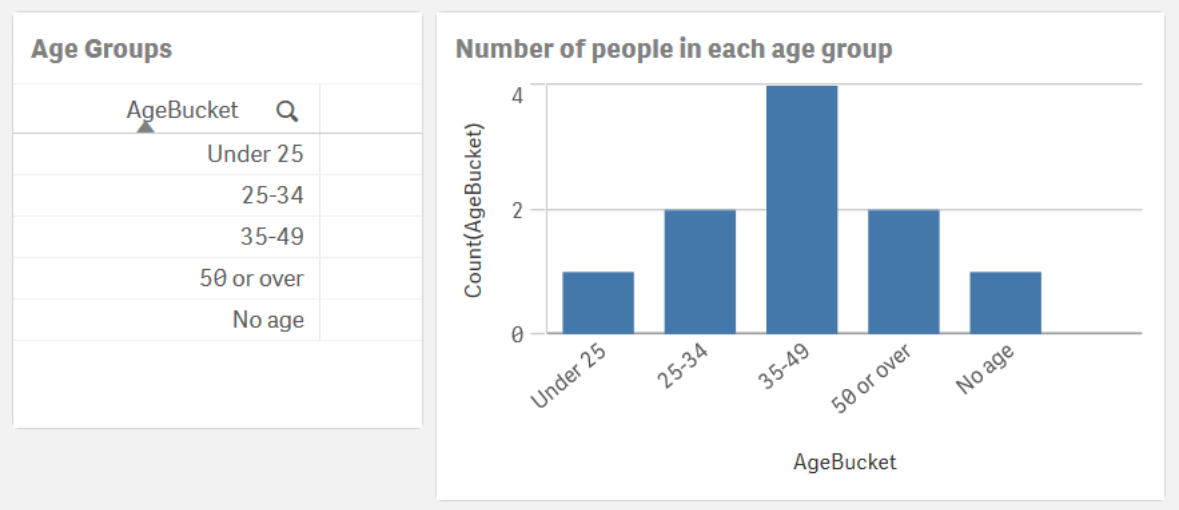


PersonID	Person	Age	AgeBucket
1	Jane	23	Under 25
2	Joe	45	35-49
3	Shawn	43	35-49
4	Sue	30	25-34
5	Frank	40	35-49
6	Mike	32	25-34
7	Gloria	45	35-49
8	Mary	54	50 or over
9	Steven		No age
10	Bill	61	50 or over

Gdyby żaden z identyfikatorów *PersonID* w tabeli *AgeTemp* nie został załadowany do modelu danych, wówczas pola *Age* i *AgeBucket* nie zostałyby sprzężone do tabeli *People*. Funkcja Exists() może zapobiegać powstawaniu osieroconych wierszy/danych w modelu danych, czyli pól *Age* i *AgeBucket*, które nie zawierają żadnych danych powiązanych osób.

6. Utwórz nowy arkusz i nadaj mu nazwę.
7. Otwórz nowy arkusz i kliknij przycisk **Edytuj arkusz**.
8. Do arkusza dodaj standardową tabelę z wymiarem *AgeBucket* i nadaj wizualizacji nazwę *Grupy wiekowe*.
9. Dodaj do arkusza wykres słupkowy z wymiarem *AgeBucket*oraz miarą *Count([AgeBucket])*. Nadaj nazwę wizualizacji *Number of people in each age group*.
10. Dostosuj właściwości tabeli i wykresu słupkowego zgodnie z preferencjami i kliknij polecenie **Gotowe**. Arkusz powinien wyglądać podobnie do niniejszego:

Arkusz z podziałem na grupy według wieku



Funkcja Dual() jest bardzo użyteczna w skrypcie albo w wyrażeniu wykresu, gdy nie ma potrzeby przypisywania wartości liczbowej do ciągu znaków.

W powyższym skrypcie dostępna jest aplikacja, która ładuje wartości wieku. Te wartości zostały umieszczone w przedziałach, co umożliwia utworzenie wizualizacji na podstawie przedziałów, zamiast na podstawie rzeczywistych wartości wieku. Istnieje przedział dla osób w wieku poniżej 25 lat, w zakresie od 25 do 35 lat itd. Za pomocą funkcji Dual() można przypisać do przedziałów wieku wartość liczbową, która później może być używana w celu sortowania przedziałów na liście wartości lub w wykresie. Sortowanie, podobnie jak w arkuszu aplikacji, umieszcza przedział „No age” na końcu listy.

Aby dowiedzieć się więcej na temat funkcji Exists() i Dual(), zapoznaj się z tym artykułem na blogu Qlik Community: Dual & Exists – Useful Functions (Przydatne funkcje Dual i Exists)

3.4 Dopasowywanie interwałów i ładowanie iteracyjne

Prefiks Intervalmatch przed instrukcją LOAD lub SELECT służy do łączenia dyskretnych wartości liczbowych z dowolną liczbą interwałów liczbowych. Jest to funkcja o ogromnych możliwościach, która może być stosowana na przykład w środowisku produkcyjnym.

Używanie prefiksu IntervalMatch()

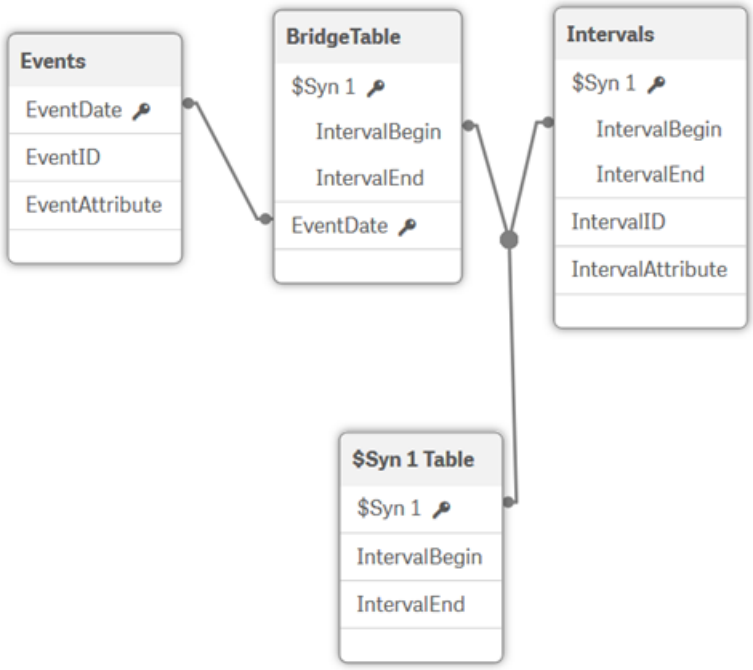
Podstawowy przykład dopasowywania interwałów to sytuacja, w której jedna tabela zawiera listę liczb lub dat (zdarzeń), a druga tabela listę przedziałów. Chodzi o powiązanie tych dwóch tabel. W przypadku ogólnym jest to relacja o krotności wiele do wielu — do interwału może należeć wiele dat, a każda data może należeć do wielu interwałów. Rozwiązanie tego problemu wymaga utworzenia tabeli pomostowej między dwiema tabelami pierwotnymi. Można to zrobić na kilka sposobów.

Najprostszym sposobem na rozwiązanie tego problemu w Qlik Sense jest wykorzystanie prefiksu IntervalMatch() przed każdą instrukcją LOAD lub SELECT. Instrukcja LOAD lub SELECT musi zawierać tylko dwa pola definiujące interwały: pole From i pole To. Prefiks IntervalMatch() posłuży następnie do generowania wszystkich kombinacji załadowanych interwałów i wcześniej załadowanego pola liczbowego wskazanego jako parametr prefiksu.

Wykonaj następujące czynności:

1. Utwórz nową aplikację i nadaj jej nazwę.
2. Dodaj nową sekcję skryptu w **edytorze ładowania danych**.
3. Wywołaj sekcje *Events*.
4. W sekcji **AttachedFiles** dostępnej po prawej stronie kliknij przycisk **Wybierz dane**.
5. Prześlij, a następnie wybierz *Events.txt*.
6. W oknie **Wybierz dane z** kliknij przycisk **Wstaw skrypt**.
7. Prześlij, a następnie wybierz *Intervals.txt*.
8. W oknie **Wybierz dane z** kliknij przycisk **Wstaw skrypt**.
9. W skrypcie pierwszej tabeli nadaj nazwę *Zdarzenia*, a drugiej *Intervals*.
10. Na końcu skryptu dodaj IntervalMatch, aby utworzyć trzecią tabelę łączącą dwie pierwsze tabele:
    ```
    BridgeTable:
    IntervalMatch (EventDate)
    LOAD distinct IntervalBegin, IntervalEnd
    Resident Intervals;
    ```
11. Skrypt powinien wyglądać następująco:
    ```
    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. Kliknij polecenie **Ładuj dane**.
13. Otwórz **przeglądarkę modelu danych**. Model danych wygląda następująco:

Model danych:Tabele Events, BridgeTable, Intervals i $Syn1



Model danych zawiera klucz złożony (pola *IntervalBegin* i *IntervalEnd*), który będzie wyświetlany w Qlik Sense jako klucz syntetyczny.

Podstawowe tabele to:

- tabela *Events* zawierająca dokładnie jeden rekord na zdarzenie;
- tabela *Intervals* zawierająca dokładnie jeden wiersz na interwał;
- tabela pomostowa zawierająca dokładnie jeden wiersz dla każdej kombinacji zdarzenia i interwału, łącząca dwie poprzednie tabele.

Należy pamiętać, że w przypadku zachodzących na siebie interwałów jedno zdarzenie może należeć do kilku interwałów, a do jednego interwału może oczywiście należeć wiele zdarzeń.

Jest to optymalny model danych: znormalizowany i zwarty. Tabele *Events* i *Intervals* pozostają bez zmian i każda zawiera oryginalną liczbę wierszy. Wszystkie obliczenia programu Qlik Sense wykonywane na tych tabelach, na przykład Count(EventID), będą wykonywane prawidłowo i z prawidłowymi wynikami.

Aby dowiedzieć się więcej na temat funkcji IntervalMatch(), zapoznaj się z tym artykułem na blogu Qlik Community: Używanie IntervalMatch()

Używanie pętli While i funkcji IterNo() do ładowania iteracyjnego

Bardzo podobną tabelę pomostową można utworzyć z użyciem pętli While i funkcji IterNo(), która generuje wartości nadające się do iterowania między dolnym a górnym limitem interwału.

Pętlę w instrukcji LOAD można utworzyć za pomocą klauzuli While. Na przykład:

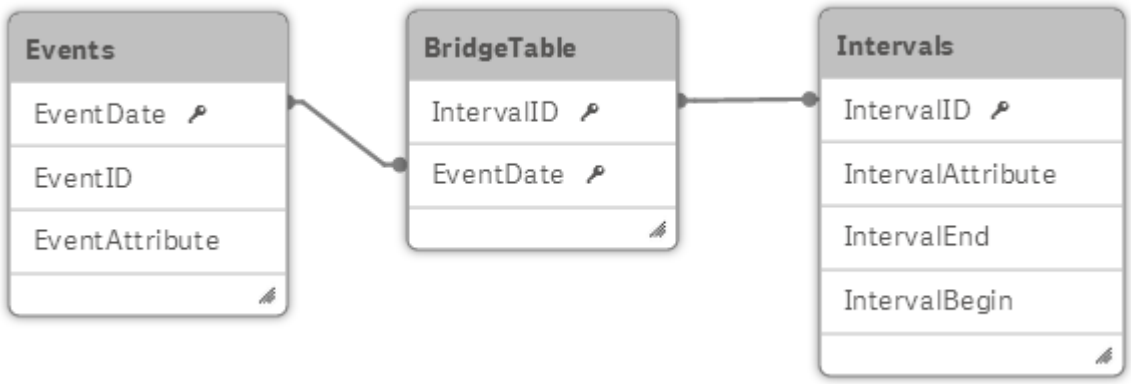
```
LOAD Date, IterNo() as Iteration From … While IterNo() <= 4;
```

Tak sformułowana instrukcja LOAD będzie w pętli odczytywać i ładować każdy kolejny wiersz wejściowy, dopóki wyrażenie podane w klauzuli While będzie mieć wartość true. Funkcja IterNo() zwraca wartość „1” przy pierwszej iteracji, „2” w drugiej i tak dalej.

W tabeli interwałów jest klucz główny (IntervalID), więc jedyną różnicą w skrypcie będzie sposób tworzenia tabeli pomostowej:

Wykonaj następujące czynności:

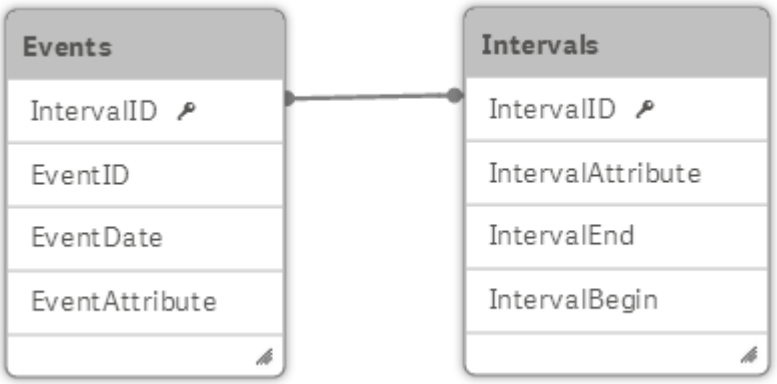
1. Zastąp istniejące instrukcje Bridgetable następującym skryptem:
   ```
   BridgeTable:
   LOAD distinct * Where Exists(EventDate);
   LOAD IntervalBegin + IterNo() - 1 as EventDate, IntervalID
       Resident Intervals
       While IntervalBegin + IterNo() - 1 <= IntervalEnd;
   ```
2. Kliknij polecenie **Ładuj dane**.
3. Otwórz **przeglądarkę modelu danych**. Model danych wygląda następująco:
   Model danych:Tabele Events, BridgeTable i Intervals



   Ogólnie rozwiązanie z trzema tabelami jest najlepsze, ponieważ pozwala na stosowanie relacji „wiele do wielu” między interwałami i zdarzeniami. Bardzo często wiadomo jednak z góry, że każde zdarzenie może przynależeć tylko do jednego interwału. W tym przypadku tabela pomostowa nie jest konieczna. Wartości kolumny *IntervalID* można zapisać bezpośrednio w tabeli zdarzeń. Można to zrobić na kilka sposobów, ale najwygodniejsze będzie sprzężenie tabeli Bridgetable z tabelą *Events*.

4. Dodaj następujący skrypt na końcu swojego skryptu:
   ```
   Join (Events)
   LOAD EventDate, IntervalID
   Resident BridgeTable;

   Drop Table BridgeTable;
   ```

5. Kliknij polecenie **Ładuj dane**.
6. Otwórz **przeglądarkę modelu danych**. Model danych wygląda następująco:
   Model danych:Tabele Events i Intervals



Interwały otwarte i zamknięte

Interwał jest otwarty lub zamknięty w zależności od tego, czy obejmuje punkty końcowe.

- Jeśli interwał obejmuje punkty końcowe, jest to interwał zamknięty:
  $[a,b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$
- Jeśli interwał nie obejmuje punktów końcowych, jest to interwał otwarty:
  $]a,b[= \{x \in \mathbb{R} \mid a < x < b\}$
- Jeśli interwał obejmuje tylko jeden punkt końcowy, jest to interwał półotwarty:
  $[a,b[= \{x \in \mathbb{R} \mid a \leq x < b\}$

Jeśli interwały zachodzą na siebie i niektóre liczby mogą należeć do więcej niż jednego interwału, zwykle wskazane jest użycie interwałów zamkniętych.

Czasami jednak interwały nie zachodzą na siebie i każda liczba ma należeć do dokładnie jednego interwału. Pojawia się zatem problem, jeśli jeden punkt jest końcem jednego interwału i początkiem następnego. Liczba o wartości odpowiadającej temu punktowi zostanie przypisana do obu interwałów. Odpowiednim rozwiązaniem będą więc interwały półotwarte.

Praktycznym rozwiązaniem tego problemu jest odjęcie bardzo małej liczby od wartości końcowej każdego interwału, aby uzyskać w ten sposób interwały zamknięte, ale niezachodzące na siebie. Jeśli używane liczby są datami, najprościej w tym celu skorzystać z funkcji DayEnd(), która zwraca ostatnią milisekundę doby:

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

Niewielką liczbę można również odjąć ręcznie. W takim przypadku trzeba uważać, by odejmowana liczba nie była za mała, gdyż wynik operacji zostanie zaokrąglony do 52 znaczących cyfr binarnych (14 cyfr dziesiętnych). Odjęcie zbyt małej liczby nie zmieni wartości w sposób znaczący, więc wynik będzie znów liczbą pierwotną.

4 Czyszczenie danych

Czasami konieczne jest przetworzenie danych źródłowych ładowanych do Qlik Sense, aby w aplikacji Qlik Sense miały one pożądaną postać. W rozwiązaniu Qlik Sense dostępnych jest wiele funkcji i instrukcji umożliwiających przekształcanie danych do odpowiedniego formatu.

W skrypcie Qlik Sense można stosować mapowanie w celu zastępowania lub modyfikowania wartości lub nazw pól podczas wykonywania skryptu. Mapowanie umożliwia więc czyszczenie danych i poprawianie ich spójności, jak również zastępowanie wartości pól w części lub w całości.

Podczas ładowania danych z różnych tabel często zdarza się, że pola o identycznym znaczeniu mają różne nazwy. Takie niespójności utrudniają tworzenie asocjacji, więc konieczne jest ich korygowanie. Elegancką metodą rozwiązania tego problemu jest utworzenie tabeli mapowania używanej do porównywania wartości pól.

4.1 Tabele mapowania

Tabele ładowane za pomocą Mapping load lub Mapping select są traktowane inaczej niż inne tabele. Są one przechowywane w odrębnym obszarze pamięci i używane wyłącznie jako tabele mapowania podczas wykonywania skryptu. Po wykonaniu skryptu tabele te są automatycznie usuwane.

Reguły:

- Tabela mapowania musi zawierać dokładnie dwie kolumny, z których pierwsza zawiera wartości sprawdzane, a druga pożądane wartości docelowe mapowania.
- Obie kolumny muszą mieć nazwy, ale same nazwy nie mają żadnego znaczenia. Podane tu nazwy kolumn nie są w żaden sposób związane z nazwami pól zwykłych tabel wewnętrznych.

4.2 Funkcje i instrukcje Mapping

W tym kursie zostaną omówione następujące funkcje i instrukcje mapowania:

• Prefiks Mapping

• ApplyMap()

• MapSubstring()

• Instrukcja Map … Using

• Instrukcja Unmap

4.3 Prefiks Mapping

Prefiks Mapping służy do utworzenia tabeli mapowania w skrypcie. Takiej tabeli mapowania można następnie używać z funkcją ApplyMap(), funkcją MapSubstring() lub instrukcją Map … Using.

Wykonaj następujące czynności:

1. Utwórz nową aplikację i nadaj jej nazwę.
2. Dodaj nową sekcję skryptu w **edytorze ładowania danych**.
3. Wywołaj sekcję *Countries*.
4. Wprowadź następujący skrypt:

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

   Tabela *CountryMap* zawiera dwie kolumny: *Country* i *NewCountry*. Kolumna *Country* zawiera różne wersje nazw krajów wpisane w polu *Country*. Kolumna *NewCountry* zawiera mapowania tych wartości. Tabela mapowania posłuży do zapisywania w polu *Country* spójnych wartości odpowiadających Stanom Zjednoczonym (w tym przypadku *US*). Jeśli na przykład pole *Country* zawiera wartość *U.S.A.*, zostanie ona zmapowana na *US*.

4.4 ApplyMap() funkcja

Użyj ApplyMap(), aby zastąpić dane w polu na podstawie wcześniej utworzonej tabeli mapowania. Przed użyciem funkcji ApplyMap() trzeba załadować odpowiednią tabelę mapowania. Dane w tabeli *Data.xlsx*, które zostaną załadowane, wyglądają następująco:

Tabela danych

Identyfikator	Name	Country	Kod
1	John Black	U.S.A.	SDFGBS1DI
2	Steve Johnson	U.S.	2ABC
3	Mary White	United States	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	US	KSD111DKFJ1

Należy zauważyć, że kraj jest wprowadzany na różne sposoby. Aby zawartość pola kraju była spójna, ładujemy tabelę mapowania i używamy funkcji **ApplyMap()**.

Wykonaj następujące czynności:

1. Poniżej skryptu wprowadzonego powyżej wybierz i załaduj plik *Data.xlsx*, a następnie wstaw skrypt.
2. Wprowadź następujące dane powyżej nowej utworzonej instrukcji LOAD:

```
Data:
```

Skrypt powinien wyglądać następująco:

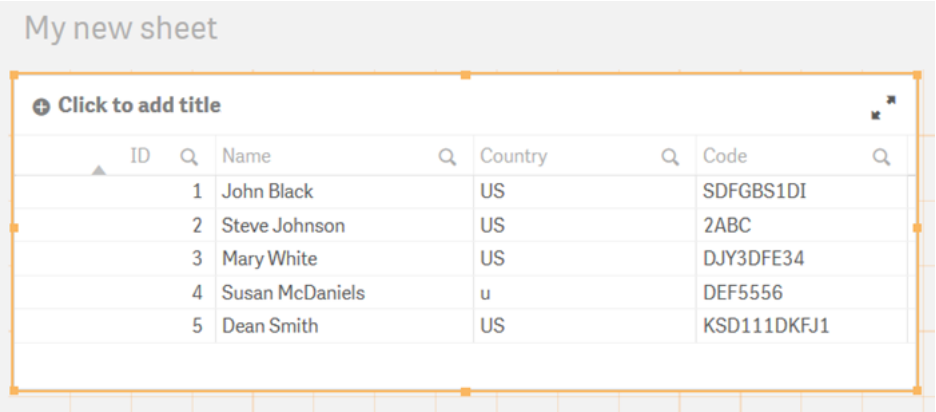
```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

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

3. Zmodyfikuj wiersz zawierający pole `Country`, w następujący sposób:
   `ApplyMap('CountryMap', Country) as Country,`

   Pierwszy parametr funkcji ApplyMap() to nazwa mapy w pojedynczych cudzysłowach. Drugi parametr określa pole zawierające dane, które będą zastępowane.

4. Kliknij polecenie **Ładuj dane**.
   Wynikowa tabela wygląda następująco:

   Tabela przedstawia dane załadowane przy użyciu funkcji ApplyMap()



   Wszystkie warianty nazwy *United States* zostały zamienione na *US*. Jeden z rekordów zawierał błąd pisowni, więc funkcja ApplyMap() nie zmieniła tej wartości pola. Do funkcji ApplyMap() można dodać trzeci parametr w celu określenia wyrażenia domyślnego, które będzie używane w razie braku pasującej wartości w tabeli mapowania.

5. Dodaj wartość `'US'` jako trzeci parametr funkcji ApplyMap(), co pozwoli obsłużyć przypadki nieprawidłowego wpisania kraju:
   `ApplyMap('CountryMap', Country, 'US') as Country,`

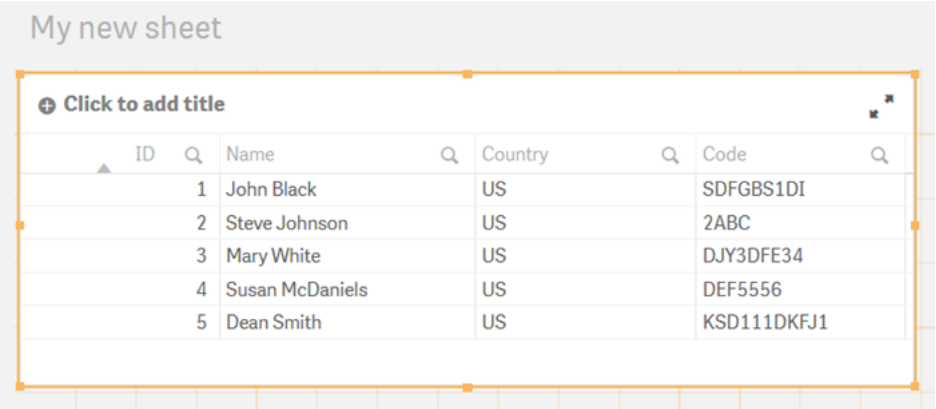
Skrypt powinien wyglądać następująco:

```
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. Kliknij polecenie **Ładuj dane**.

   Wynikowa tabela wygląda następująco:

   Tabela przedstawia dane załadowane przy użyciu funkcji ApplyMap



Aby dowiedzieć się więcej na temat funkcji ApplyMap(), zapoznaj się z tym artykułem na blogu Qlik Community: Don't join - use Applymap instead (Nie łącz, używaj opcji Applymap)

4.5 MapSubstring() funkcja

Funkcja MapSubstring() umożliwia mapowanie części pola.

Przetworzymy tabelę utworzoną przez funkcję ApplyMap(), aby liczby były zapisane słownie. Funkcja MapSubstring() posłuży do zastąpienia danych liczbowych tekstem.

W tym celu trzeba najpierw utworzyć tabelę mapowania.

Wykonaj następujące czynności:

1. Dodaj następujące wiersze skryptu po sekcji *CountryMap*, ale przed sekcją *Data*.

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

   Tabela *CodeMap* mapuje liczby od 1 do 5 oraz liczbę 11.

2. W sekcji *Data* skryptu zmodyfikuj instrukcję `Code` następująco:

   ```
   MapSubString('CodeMap', Code) as Code
   ```

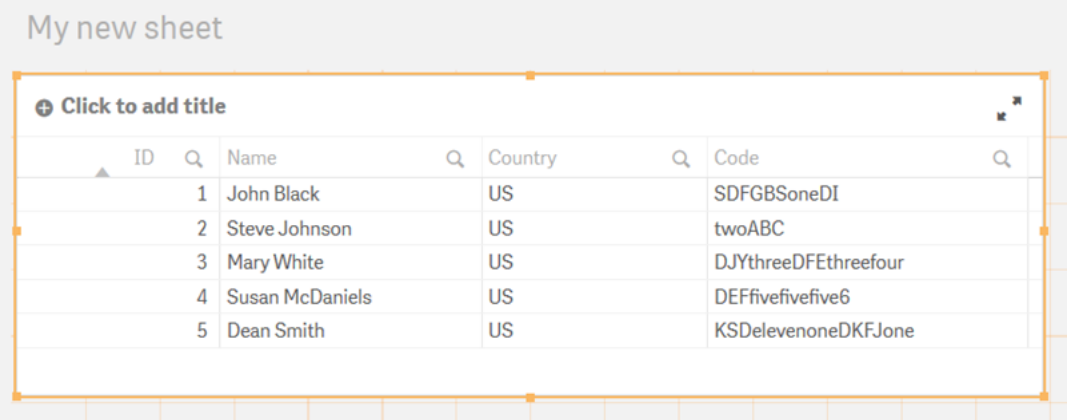
   Skrypt powinien wyglądać następująco:

   ```
   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. Kliknij polecenie **Ładuj dane**.

   Wynikowa tabela wygląda następująco:

Tabela przedstawia dane załadowane przy użyciu funkcji 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

Zapisy liczbowe w polu *Code* zostały zastąpione tekstem. Jeśli liczba występuje więcej niż raz, jak w przypadku ID=3 i ID=4, tekst również jest powtarzany. ID=4. *Susan McDaniels* zawiera wartość 6 w kodzie. Ponieważ dla liczby 6 nie określono mapowania w tabeli *CodeMap*, pozostała ona bez zmian. W kodzie osoby o ID=5 (*Dean Smith*) występowała liczba 111. Została ona zmapowana jako „elevenone”.

Aby dowiedzieć się więcej na temat funkcji MapSubstring(), zapoznaj się z tym artykułem na blogu Qlik Community: Mapping … and not the geographical kind (Mapowanie, ale nie takie geograficzne)

4.6 Map … Using

Za pomocą funkcji Map … Using można zastosować mapę do pola. Jednak, działa to trochę inaczej niż w przypadku funkcji ApplyMap(). W przypadku funkcji ApplyMap() mapowanie jest stosowane przy każdym napotkaniu nazwy pola przy przetwarzaniu, natomiast instrukcja Map … Using dokonuje mapowania na etapie zapisywania wartości w tabeli wewnętrznej w polu o określonej nazwie.

Przyjrzyjmy się przykładowi. Załóżmy, że wielokrotnie ładujemy pole *Country* w skrypcie i chcemy stosować mapowanie przy każdym załadowaniu. Można do tego celu użyć funkcji ApplyMap(), jak we wcześniejszej części tego kursu, lub skorzystać z instrukcji Map … Using.

Zastosowanie instrukcji Map … Using spowoduje zastosowanie mapy do pola na etapie zapisywania pola w tabeli wewnętrznej. W poniższym przykładzie mapowanie zostanie zastosowane do pola *Country* w tabeli Data1, ale nie do pola *Country2* w tabeli *Data2*. Wynika to stąd, że instrukcja Map … Using jest stosowana jedynie do pól o nazwie *Country*. Po zapisaniu w tabeli wewnętrznej pole *Country2* nie ma już nazwy *Country*. Jeśli mapowanie miałoby być stosowane również do tabeli *Country2*, należałoby użyć funkcji ApplyMap().

Instrukcja Unmap kończy instrukcję Map … Using, więc gdyby pole *Country* zostało załadowane po instrukcji Unmap, tabela mapowania *CountryMap* nie zostałaby zastosowana.

Wykonaj następujące czynności:

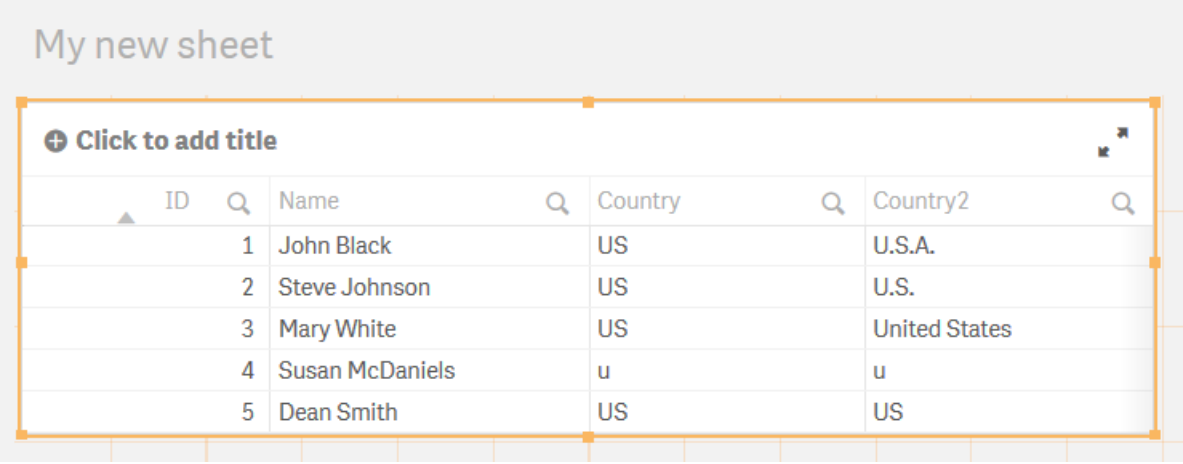
1. Skrypt dla tabeli *Data* zastąp następującym skryptem:

```
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. Kliknij polecenie **Ładuj dane**.

   Wynikowa tabela wygląda następująco:

   Tabela przedstawia dane załadowane przy użyciu funkcji Map … Using



5 Obsługa danych hierarchicznych

Hierarchie stanowią ważny element każdego rozwiązania Business Intelligence. Umożliwiają one do opisywanie wymiarów, które ze swej natury obejmują różne poziomy szczegółowości. Niektóre hierarchie są proste i intuicyjne, podczas gdy inne są bardziej złożone i prawidłowe ich zamodelowanie wymaga dokładnego przemyślenia.

Przechodząc od szczytu hierarchii do jej dołu, kolejne poziomy są coraz bardziej szczegółowe. Na przykład w wymiarze obejmującym poziomy Rynek, Kraj, Stan i Miasto element Ameryki będzie należeć do pierwszego poziomu hierarchii, element Stany Zjednoczone do drugiego poziomu, element Kalifornia do trzeciego, a element San Francisco do poziomu najniższego. Kalifornia to wartość bardziej szczegółowa niż Stany Zjednoczone, a San Francisco to z kolei wartość bardziej szczegółowa niż Kalifornia.

Przechowywanie hierarchii w relacyjnym modelu danych jest znanym problemem o licznych rozwiązaniach. Istnieje tu kilka podejść:

- hierarchia pozioma,
- model list sąsiadowania,
- metoda wyliczania ścieżki,
- model zbiorów zagnieżdżonych,
- lista węzłów nadrzędnych.

Dla potrzeb tego kursu utworzymy listę węzłów nadrzędnych, ponieważ pozwala ona zapisać hierarchię w formacie bezpośrednio nadającym się do wykonywania zapytań. Więcej informacji na temat innych strategii można znaleźć w temacie Qlik Community.

5.1 Prefiks Hierarchy

Prefiks Hierarchy to polecenie skryptu umieszczane przed instrukcją LOAD lub SELECT w celu załadowania tabeli węzłów sąsiadujących. Instrukcja LOAD musi zawierać co najmniej trzy pola: identyfikator stanowiący niepowtarzalny klucz węzła, odwołanie do węzła nadrzędnego i nazwę.

Zastosowanie prefiksu spowoduje przekształcenie załadowanej tabeli do postaci tabeli z węzłami rozwiniętymi, zawierającej kilka dodatkowych kolumn – po jednej dla każdego poziomu hierarchii.

Wykonaj następujące czynności:

1. Utwórz nową aplikację i nadaj jej nazwę.
2. Dodaj nową sekcję skryptu w **edytorze ładowania danych**.
3. Wywołaj sekcję *Wine*.
4. W sekcji **AttachedFiles** dostępnej po prawej stronie kliknij przycisk **Wybierz dane**.
5. Prześlij, a następnie wybierz *Winedistricts.txt*.
6. W oknie **Wybierz dane z** usuń zaznaczenie pól *Lbound* i *RBound*, aby nie zostały one załadowane.
7. Kliknij polecenie **Wstaw skrypt**.
8. Wprowadź następujące dane powyżej instrukcji LOAD:

```
Hierarchy (NodeID, ParentID, NodeName)
```

Skrypt powinien wyglądać następująco:

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

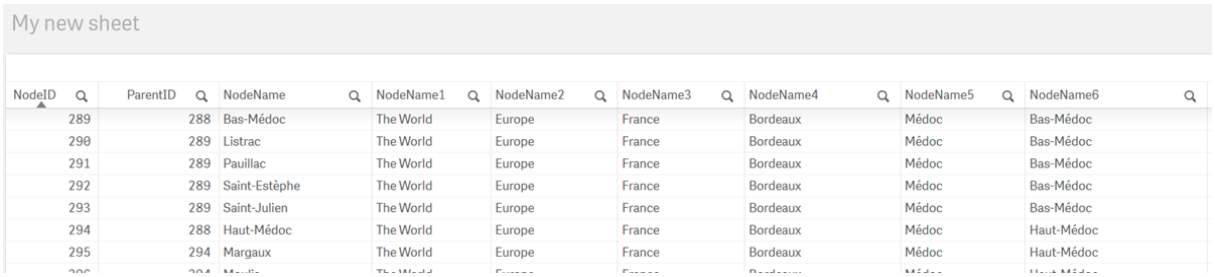
9. Kliknij polecenie **Ładuj dane**.
10. W sekcji **Podgląd** w **Przeglądarce modelu danych** wyświetl wynikową tabelę.

    Wynikowa tabela węzłów rozwiniętych zawiera dokładnie tyle samo rekordów, co tabela źródłowa, czyli po jednym dla każdego węzła. Tabela węzłów rozwiniętych to bardzo praktyczna postać danych, ponieważ spełnia szereg wymogów dotyczących analizowania hierarchii w modelu relacyjnym:

    - Jedna kolumna zawiera wszystkie nazwy węzłów, więc można jej używać w wyszukiwaniach.
    - Poszczególne poziomy węzłów zostały rozwinięte w osobne pola, których można używać w grupach hierarchicznych lub jako wymiary w tabelach przestawnych.
    - Poszczególne poziomy węzłów zostały rozwinięte w osobne pola, których można używać w grupach hierarchicznych.
    - Można sprawić, by tabela zawierała unikatową ścieżkę prowadzącą do określonego węzła, zawierającą wszystkie elementy nadrzędne w odpowiedniej kolejności.
    - Można sprawić, by tabela zawierała głębokość węzła, czyli jego odległość od węzła głównego.

    Wynikowa tabela wygląda następująco:

    Tabela przedstawia część danych załadowanych przy użyciu prefiksu Hierarchy



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

5.2 Prefiks HierarchyBelongsTo

Podobnie jak Hierarchy prefiks HierarchyBelongsTo to polecenie skryptu umieszczane przed instrukcją LOAD lub SELECT w celu załadowania tabeli węzłów sąsiadujących.

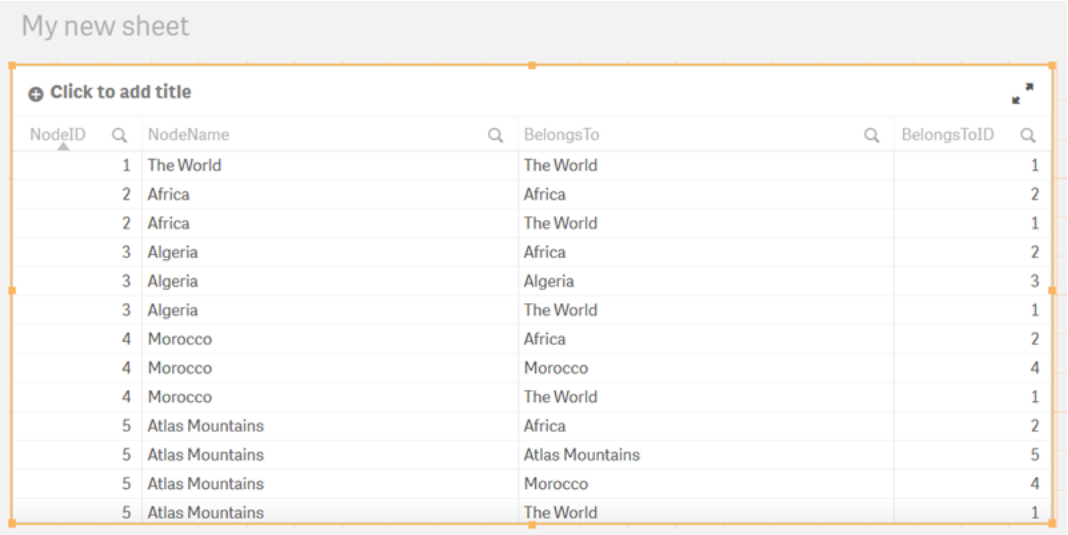
Również w tym przypadku instrukcja LOAD musi zawierać co najmniej trzy pola: identyfikator stanowiący niepowtarzalny klucz węzła, odwołanie do węzła nadrzędnego i nazwę. Prefiks powoduje przekształcenie załadowanej tabeli w tabelę węzłów nadrzędnych, czyli taką, która zawiera osobny wiersz dla każdej kombinacji węzła nadrzędnego i podrzędnego. Dzięki temu bardzo łatwo jest znaleźć wszystkie węzły nadrzędne lub podrzędne określonego węzła.

Wykonaj następujące czynności:

1. Zmodyfikuj instrukcję Hierarchy w **Edytorze ładowania danych** do następującej postaci:
   `HierarchyBelongsTo (NodeID, ParentID, NodeName, BelongsToID, BelongsTo)`
2. Kliknij polecenie **Ładuj dane**.
3. W sekcji **Podgląd** w **Przeglądarce modelu danych** wyświetl wynikową tabelę.
   Tabela nadrzędna spełnia szereg wymogów dotyczących analizowania hierarchii w modelu relacyjnym:
   - Podczas gdy identyfikator węzła reprezentuje tylko pojedynczy węzeł, identyfikator węzła nadrzędnego może reprezentować całe drzewo hierarchii lub poddrzewo w jej obrębie.
   - Wszystkie nazwy węzłów istnieją zarówno w roli węzłów, jak i w roli drzew, i mogą być używane w wyszukiwaniach w obu rolach.
   - Można sprawić, by tabela zawierała różnicę głębokości węzła i określonego węzła nadrzędnego, czyli odległość od węzła głównego poddrzewa.

   Wynikowa tabela wygląda następująco:

   Tabela przedstawia dane załadowane przy użyciu prefiksu 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

Autoryzacja

Częstym zastosowaniem hierarchii jest dokonywanie autoryzacji. Typowym przykładem jest hierarchia organizacyjna. Każdy menedżer powinien oczywiście mieć wgląd w działalność wszystkich jednostek w obrębie swojego działu, w tym wszelkich poddziałów. Niekoniecznie musi jednak mieć uprawnienia wglądu do innych działów.

Przykład hierarchii organizacyjnej

Oznacza to, że różne osoby mogą widzieć różne poddrzewa w ramach organizacji. Tabela autoryzacji może wyglądać następująco:

Tabela autoryzacji

DOSTĘP	NTNAME	OSOBA	STANOWISKO	UPRAWNIENIA
USER	ACME\JRL	John	CPO	HR
USER	ACME\CAH	Carol	CEO	CEO
USER	ACME\JER	James	Dyrektor ds. inżynierii	Inżynieria
USER	ACME\DBK	Diana	CFO	Finanse
USER	ACME\RNL	Bob	COO	Sales
USER	ACME\LFD	Larry	CTO	Produkt

W tym przypadku *Carol* ma uprawnienia do wyświetlania wszystkiego od węzła *CEO* w dół, *Larry* ma uprawnienia do wyświetlania całej organizacji *Product*, a *James* widzi tylko organizację *Engineering*.

Przykład:

Często hierarchia jest przechowywana w tabeli węzłów sąsiadujących. W tym przykładzie, aby rozwiązać ten problem, można załadować tabelę węzłów sąsiadujących za pomocą HierarchyBelongsTo, a polu nadrzędnemu nadać nazwę *Tree*.

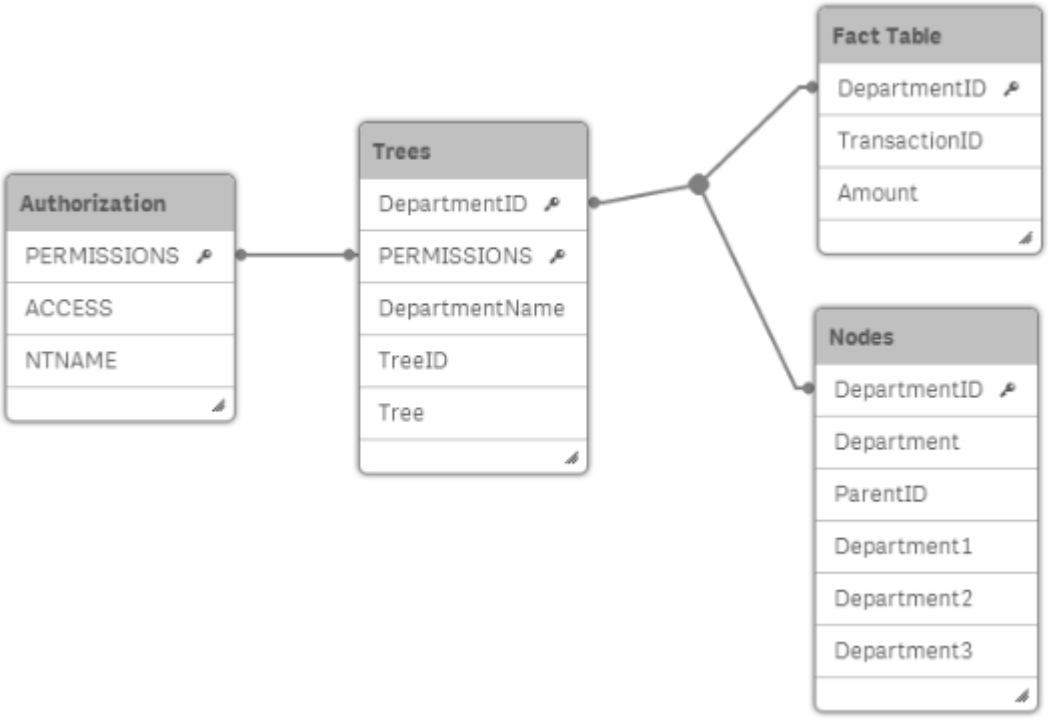
Jeśli chcesz użyć Section Access, załaduj kopię *Tree* zapisaną wielkimi literami i wywołaj to nowe pole *PERMISSIONS*. Na koniec należy załadować tabelę autoryzacji. Dwa ostatnie kroki można wykonać, używając poniższego kodu. Zauważ, że tabela TempTrees została utworzona przez instrukcję HierarchyBelongsTo.

Należy pamiętać, że jest to tylko przykład. Nie ma żadnego ćwiczenia towarzyszącego, które należy wykonać w 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;
```

Na podstawie tego przykładu powstałby następujący model danych:

Model danych: Tabele Authorization, Trees, Fact i Nodes



6 Pliki QVD

Plik QVD (QlikView Data) zawiera tabelę danych eksportowaną z programu Qlik Sense lub QlikView. QVD to własnościowy format Qlik, który może być zapisywany i odczytywany wyłącznie przez programy Qlik Sense i QlikView. Format pliku jest zoptymalizowany w celu uzyskania maksymalnej szybkości odczytu danych przez skrypty Qlik Sense przy zachowaniu niewielkich rozmiarów. Format pliku jest zoptymalizowany w celu uzyskania maksymalnej szybkości odczytu danych przez skrypty przy zachowaniu niewielkich rozmiarów. Odczyt danych z pliku QVD jest zazwyczaj 10–100 razy szybszy niż odczyt z innych źródeł danych.

Dostępne są dwa tryby odczytu plików QVD: standardowy (szybki) i zoptymalizowany (szybszy). Wybrany tryb jest automatycznie określany przez silnik obsługi skryptów Qlik Sense. Trybu zoptymalizowanego można używać jedynie w przypadku odczytywania wszystkich ładowanych pól bez transformacji (formuł wykonujących operacje na polach), choć dozwolona jest zmiana nazw pól. Ładowanie zoptymalizowane jest też wyłączane w przypadku klauzuli Where wymagającej od programu Qlik Sense rozpakowania wierszy.

Plik QVD mieści dokładnie jedną tabelę danych i obejmuje trzy części:

- Nagłówek XML w kodowaniu znaków UTF-8 opisujący pola tabeli, układ następujących dalej informacji i metadane;
- Tabele symboli w formacie ze wstawianiem bajtów.
- Rzeczywiste dane tabeli w formacie ze wstawianiem bitów.

Pliki QVD mogą być używane do różnych celów. Można wskazać cztery typowe zastosowania, ale używanie tych plików często przynosi kilka korzyści naraz:

- Przyspieszenie ładowania danych

  Buforowanie niezmiennych lub rzadko modyfikowanych bloków danych wejściowych w plikach QVD pozwala znacznie przyspieszyć wykonywanie skryptów na dużych zestawach danych.

- Przyspieszenie ładowania danych

  Buforowanie niezmiennych lub rzadko modyfikowanych bloków danych wejściowych w plikach QVD pozwala znacznie przyspieszyć wykonywanie skryptów na dużych zestawach danych.

- Zmniejszenie obciążenia serwerów baz danych

  Można znacznie zmniejszyć ilość danych pobieranych z zewnętrznych źródeł danych. Pozwala to ograniczyć obciążenie zewnętrznych baz danych i natężenie ruchu w sieci. Poza tym gdy kilka skryptów Qlik Sense korzysta z tych samych danych, wystarczy tylko raz załadować takie dane ze źródłowej bazy danych do pliku QVD. Inne aplikacje będą mogły następnie używać tych samych danych za pośrednictwem tego pliku QVD.

- Konsolidowanie danych z wielu aplikacji Qlik Sense

  Instrukcja skryptu Binary pozwala tylko załadować dane z jednej aplikacji Qlik Sense do innej, natomiast użycie plików QVD umożliwia łączenie w skrypcie Qlik Sense danych z dowolnie wielu aplikacji Qlik Sense. Stwarza to liczne możliwości konsolidowania podobnych danych z różnych jednostek organizacyjnych itp.

- Ładowanie przyrostowe

Typowym zastosowaniem plików QVD jest wspomaganie ładowania przyrostowego, czyli ładowania jedynie nowych rekordów z rosnącej bazy danych.

Aby zobaczyć, jak społeczność Qlik wykorzystuje Qlik Application Automation do poprawy czasów ładowania QVD, zobacz temat Jak podzielić QVD za pomocą automatyzacji, aby poprawić przeładowania.

6.1 Tworzenie plików QVD

Istnieją dwa sposoby utworzenia pliku QVD:

- Jawne utworzenie i nazwanie pliku z użyciem polecenia Store w skrypcie Qlik Sense.
  W skrypcie należy określić, że wcześniej wczytana tabela lub część takiej tabeli ma zostać eksportowana do pliku o podanej nazwie i lokalizacji.
- Automatyczne tworzenie i utrzymywanie przez skrypt.
  Jeśli instrukcja LOAD lub SELECT zostanie poprzedzona prefiksem Buffer, program Qlik Sense automatycznie utworzy plik QVD, który w określonych warunkach może zastępować pierwotne źródło danych podczas przeładowywania danych.

Niezależnie od metody utworzenia wszystkie pliki QVD dają taką samą szybkość odczytu.

Store

Ta instrukcja skryptów tworzy jawnie nazwane pliki QVD, CSV lub txt.

Składnia:

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

Instrukcja może eksportować pola tylko z jednej tabeli danych. W przypadku eksportowania pól z kilku tabel należy wcześniej jawnie wykonać w skrypcie sprzężenie, aby utworzyć tabelę danych do wyeksportowania.

Wartości tekstowe są eksportowane do pliku CSV w formacie UTF-8. Możliwe jest określenie ogranicznika — zobacz opis instrukcji **LOAD**. Instrukcja store zapisująca do pliku CSV nie obsługuje eksportu BIFF .

Przykłady:

```
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';
```

Wykonaj następujące czynności:

1. Otwórz aplikację *Advanced Scripting Tutorial*.
2. Kliknij sekcję skryptu *Product*.
3. Dodaj następujący kod na końcu skryptu:
   ```
   Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
   ```

   Skrypt powinien wyglądać następująco:

   ```
   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. Kliknij polecenie **Ładuj dane**.
   Na liście plików powinien teraz widnieć plik *Product.qvd*.

   Ten plik danych jest wynikiem wykonania skryptu **Crosstable** i zawiera tabelę o trzech kolumnach, po jednej dla każdej kategorii (Product, Month, Sales). Używając tego pliku danych, można by teraz zastąpić całą sekcję skryptu *Product* .

6.2 Odczyt danych z plików QVD

Istnieje kilka metod wczytania lub otwarcia pliku QVD w programie Qlik Sense:

- Załadowanie pliku QVD jako jawnego źródła danych. Instrukcje load w skrypcie Qlik Sense mogą się odwoływać do plików QVD tak samo jak do innych plików tekstowych (csv, fix, dif, biff itp.).

  Przykłady:

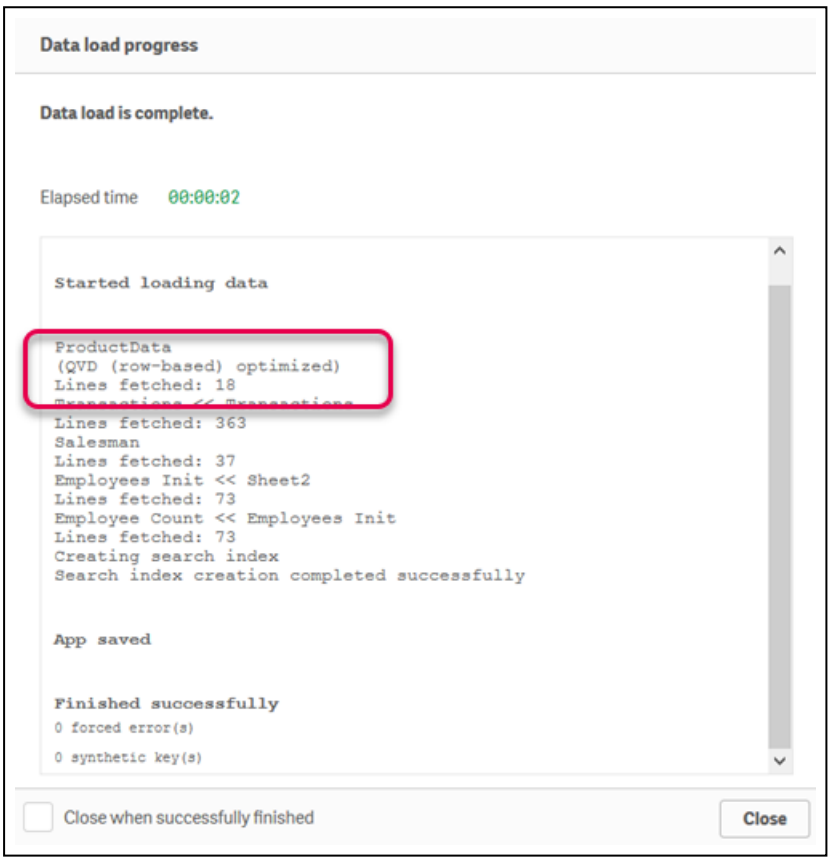
  ```
  LOAD * from 'lib://FolderConnection/xyz.qvd' (qvd);
  LOAD fieldname1, fieldname2 from [lib://FolderConnection/xyz.qvd] (qvd);
  LOAD fieldname1 as newfieldname1, fieldname2 as newfieldname2 from
  [lib://AttachedFiles/xyz.qvd](qvd);
  ```

- Automatyczne ładowanie buforowanych plików QVD. W przypadku użycia prefiksu buffer przed instrukcją load lub select odczyt nie wymaga żadnych dodatkowych instrukcji. Program Qlik Sense będzie automatycznie ustalać, w jakim zakresie używać danych z pliku QVD zamiast pobierania danych za pośrednictwem pierwotnej instrukcji LOAD lub SELECT.
- Dostęp do plików QVD z poziomu skryptu. Dane zawarte w nagłówku XML pliku QVD można pobierać za pomocą odpowiednich funkcji skryptowych o nazwach zaczynających się na QVD.

Wykonaj następujące czynności:

1. W sekcji skryptu *Product* ujmij w komentar cały skrypt.
2. Wprowadź następujący skrypt:
   ```
   Load * from [lib://AttachedFiles/ProductData.qvd](qvd);
   ```
3. Kliknij polecenie **Ładuj dane**.
   Dane są wczytywane z pliku QVD.

   Okno postępu ładowania danych



Aby zapoznać się z używaniem plików QVD w celu wykonywania ładowania przyrostowego, przeczytaj ten wpis na blogu – Qlik Community: Overview of Qlik Incremental Loading (Przegląd funkcji ładowania przyrostowego Qlik)

Buffer

Pliki QVD można tworzyć i utrzymywać automatycznie z użyciem prefiksu Buffer. Prefiksu tego można używać w większości instrukcji LOAD i SELECT w skryptach. Sygnalizuje on, że do buforowania wyniku instrukcji używane są pliki QVD.

Składnia:

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

Jeśli nie zostanie podana żadna opcja, bufor QVD utworzony przez pierwsze wykonanie skryptu będzie używany przez czas nieokreślony.

Przykład:

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

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

Amount to liczba określająca okres. Mogą być używane liczby dziesiętne (Decimals). Jeśli parametr nie zostanie określony, przyjmowana jest wartość dni.

Opcja stale after jest zazwyczaj używana w przypadku źródeł bazodanowych, gdy nie istnieje prosty znacznik czasu oryginalnych danych. Klauzula stale after jedynie określa czas od momentu utworzenia bufora QVD, po którego upływie bufor będzie uważany za nieważny. Przed upływem tego czasu źródłem danych będzie bufor QVD, natomiast po upływie tego czasu używane będzie oryginalne źródło danych. Plik bufora QVD zostanie automatycznie zaktualizowany i rozpocznie się nowy okres.

Przykład:

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

Incremental

Opcja incremental umożliwia odczytywanie tylko części pliku bazowego. Poprzedni rozmiar pliku jest zapisany w nagłówku XML pliku QVD. Jest to szczególnie przydatne w przypadku plików dziennika. Wszystkie rekordy załadowane już wcześniej są wczytywane z pliku QVD, natomiast dalsze nowe rekordy są wczytywane z oryginalnego pliku źródłowego. Na koniec tworzony jest zaktualizowany plik QVD.

Należy pamiętać, że opcji incremental można używać wyłącznie z instrukcjami LOAD i plikami tekstowymi oraz że ładowania przyrostowego nie można używać w przypadku zmiany lub usuwania starych danych.

Przykład:

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

Bufory QVD są zazwyczaj usuwane, gdy nie ma już do nich żadnych odniesień po wykonaniu całego skryptu w aplikacji tworzącej bufor lub gdy aplikacja tworząca bufor przestaje istnieć. Aby zachować zawartości bufora w postaci pliku QVD lub CSV, należy użyć instrukcji Store.

Wykonaj następujące czynności:

1. Utwórz nową aplikację i nadaj jej nazwę.
2. Dodaj nową sekcję skryptu w **edytorze ładowania danych**.
3. W sekcji **AttachedFiles** dostępnej po prawej stronie kliknij przycisk **Wybierz dane**.
4. Prześlij, a następnie wybierz *Cutlery.xlsx*.

5. W oknie **Wybierz dane z** kliknij przycisk **Wstaw skrypt**.
6. Obejmij komentarzem pola w instrukcji ładowania i zmień instrukcję ładowania do następującej postaci:
   ```
   Buffer LOAD *
   ```

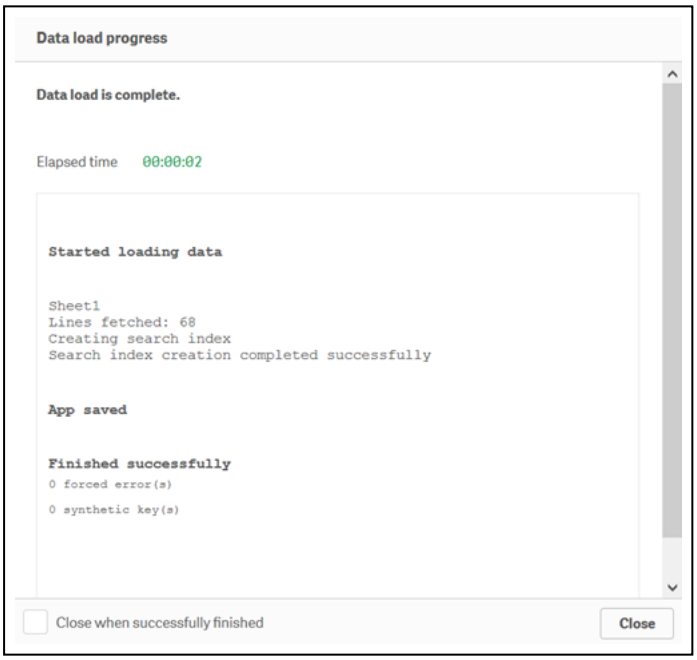
   Skrypt powinien wyglądać następująco:

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

7. Kliknij polecenie **Ładuj dane**.

   Podczas pierwszego ładowania danych są one ładowane z *Cutlery.xlsx*.

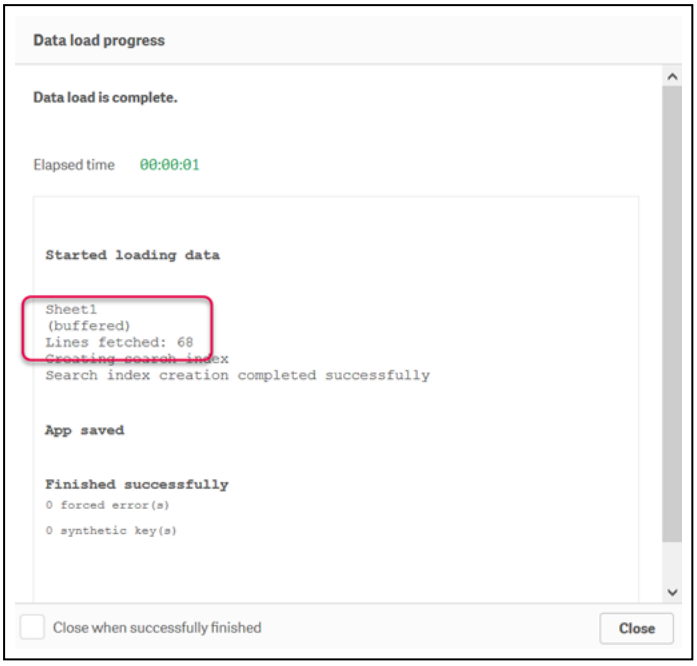
   Okno postępu ładowania danych

   Instrukcja Buffer tworzy również plik QVD i zapisuje go w Qlik Sense. W przypadku wdrożenia Qlik Sense Enterprise on Windows jest on przechowywany w katalogu na serwerze Qlik Sense.

8. Kliknij ponownie przycisk **Ładuj dane**.
9. Tym razem dane są ładowane z pliku QVD utworzonego przez instrukcję Buffer podczas pierwszego ładowania danych.

Okno postępu ładowania danych



6.3 Dziękujemy!

Kurs został ukończony i masz już dodatkową wiedzę na temat tworzenia skryptów w programie Qlik Sense. Informacje o innych dostępnych szkoleniach znajdziesz w naszej witrynie internetowej.