

Wydanie II

Microsoft Power BI dla zaawansowanych

Eksperckie techniki tworzenia
interaktywnych analiz
w świecie biznesu

Greg Deckler
Brett Powell



Helion 

<packt>

Tytuł oryginału: Mastering Microsoft Power BI: Expert techniques to create interactive insights for effective data analytics and business intelligence, 2nd Edition

Tłumaczenie: Zbigniew Waśko

ISBN: 978-83-8322-445-9

Copyright © Packt Publishing 2022. First published in the English language under the title 'Mastering Microsoft Power BI - Second Edition - (9781801811484)'

Polish edition copyright © 2023 by Helion S.A.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/mipoz2>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- **Lubię to!** » Nasza społeczność

Spis treści |

O autorach	15
O redaktorze	16
Słowo wstępne	17
Wstęp	18
ROZDZIAŁ 1	
Planowanie projektów Power BI	23
Tryby wdrożenia usługi Power BI	24
Korporacyjna analityka biznesowa	26
Wizualizacja samoobsługowa	27
Samoobsługowa analityka biznesowa	27
Wybór trybu wdrożenia	28
Wyszukiwanie i pozyskiwanie danych	31
Przykładowy szablon projektu Power BI	32
Role w projektach Power BI	35
Projektant zbioru danych	37
Twórca raportów	39
Administrator Power BI	39
Współpraca między rolami projektowymi	41
Licencje usługi Power BI	41
Pojemność współdzielona	42
Pojemność dedykowana	44
Scenariusze licencjonowania usługi Power BI	48
Projektowanie zbioru danych	49
Macierz magistrali hurtowni danych	50
Proces projektowania zbioru danych	51
Profilowanie danych	56
Profilowanie danych w Power BI Desktop	60

Planowanie zbioru danych	61
Transformacje danych	62
Tryby pobierania i przechowywania danych	64
Analiza przykładowego projektu	68
Podsumowanie	70

ROZDZIAŁ 2

Przygotowanie źródeł danych	71
Składanie zapytań	72
Częściowe składanie zapytań	73
Projektowanie zapytań w zależności od trybu zbioru danych	74
Zapytania do zbiorów danych w trybie importowym	76
Zapytania do zbiorów danych w trybie DirectQuery	77
Złożone zbiory danych	79
Źródła danych	82
Uwierzytelnianie	83
Ustawienia źródła danych	85
Poziomy prywatności	87
Usługa Power BI jako źródło danych	89
Opcje programu Power BI Desktop	90
Widoki SQL	95
Widoki SQL a zapytania w języku M	96
Przykłady widoków SQL	99
Wolno zmieniające się wymiary	105
Podsumowanie	106

ROZDZIAŁ 3

Łączenie się ze źródłami i przekształcanie danych za pomocą języka M	107
Rodzaje zapytań Power Query M	108
Organizacja zapytań	109
Parametry źródła danych	110
Kwerendy przejściowe	112
Zapytania o fakty i wymiary	114
Kwerendy tabel parametrów	118
Kwerendy tabel zabezpieczeń	121
Niestandardowe zapytania funkcyjne	122
Tworzenie zapytań w języku M	122
Numeryczne typy danych	123
Dostęp do poszczególnych danych w języku M	124

Przykłady zapytań w języku Power Query M	125
Filtrowanie poprzednich trzech lat	125
Przepływy danych	136
Funkcje przepływu danych w Power BI Premium	138
Narzędzia edycyjne Power Query M	138
Edytor zaawansowany	139
Visual Studio Code	140
Visual Studio	141
Podsumowanie	142

ROZDZIAŁ 4

Projektowanie modeli danych w trybach importu,

DirectQuery i złożonym	143
Warstwy zbioru danych	144
Power BI jako supersieć usługi Azure Analysis	145
Cele tworzenia zbiorów danych	146
Model danych	149
Widok modelu	149
Widok danych	152
Widok raportu	153
Tabele faktów	155
Tabele wymiarów	162
Tabele parametrów	167
Relacje	173
Unikatowość	173
Jednoznaczność	174
Relacje jednokierunkowe	175
Relacje dwukierunkowe	177
Funkcja CROSSFILTER()	181
Metadane modelu	182
Widoczność	183
Metadane kolumn i miar	184
Podsumowanie	184
Opis	187
Optymalizacja wydajności modelu danych	189
Import	190
DirectQuery	194
Modele złożone	196
Podsumowanie	199

ROZDZIAŁ 5

Tworzenie miar DAX i ról zabezpieczeń	200
Podstawy miar DAX	201
Kontekst filtra	202
Proces obliczania miar	205
Kontekst wiersza	207
Funkcje skalarne i tabelaryczne	209
Funkcja CALCULATE()	211
Zmienne w języku DAX	214
Miary bazowe	219
Miary pomocnicze	220
Metryki analizy czasowej	224
Bieżący w stosunku do poprzedniego i wskaźniki wzrostu	227
Okresy kroczące	228
Grupy obliczeniowe	229
Metryki wymiarów	233
Brakujące wymiary	234
Metryki rankingowe	236
Dynamiczne miary rankingowe	238
Role zabezpieczeń	240
Dynamiczne zabezpieczenia na poziomie wierszy	244
Testowanie wydajności	247
Analizator wydajności	248
DAX Studio	249
Podsumowanie	250

ROZDZIAŁ 6

Planowanie raportów Power BI	251
Proces planowania raportu	251
Określenie odbiorców	252
Formułowanie pytań biznesowych wymagających odpowiedzi	253
Sprawdzenie, czy zbiór danych dostarczy odpowiedzi na pytania biznesowe	254
Określenie interaktywności	255
Ustalenie metod dostępu i dystrybucji	256
Szkielet układu raportu	256
Diagram architektury raportu	257
Najlepsze praktyki wizualizacyjne	260
Wybór właściwych elementów wizualnych	264
Tabele i macierze a wykresy	265

Wybór wykresu	267
Antywzorce wizualizacji	269
Interakcje elementów wizualnych	271
Edycja interakcji	272
Strony uściślające raport	274
Etykiety niestandardowe i przycisk Wstecz	276
Przeglądanie szczegółowe z użyciem wielu kolumn	277
Zakresy filtrów raportu	280
Warunki filtrowania na poziomie raportu	282
Filtry na poziomach raportu i strony	285
Filtrowanie według daty względnej	287
Filtrowanie na poziomie elementu wizualnego	288
Zakładki	291
Panel Wybór oraz właściwość Wysuń na pierwszy plan	293
Niestandardowa nawigacja po raporcie	295
Tryb Widok	297
Połączenia na żywo z zestawami danych Power BI	297
Modyfikowanie raportów w trybie połączenia na żywo	300
Przełączanie źródłowych zbiorów danych w trybie połączenia na żywo	301
Przełączanie między zbiorami danych w trybach importu i połączenia na żywo	302
Podsumowanie fazy projektowania raportu	302
Podsumowanie	304

ROZDZIAŁ 7

Tworzenie i formatowanie wizualizacji	305
Panel Wizualizacje	306
Fragmentatory	307
Synchronizacja fragmentatorów	309
Niestandardowe parametry fragmentatora	311
Parametry co-jeśli	313
Panel filtrów czy fragmentator?	316
Wizualizacje jednowartościowe	318
Wizualizacja Karta	318
Wizualizacja KPI	318
Wizualizacja Miernik	319
Wizualizacje mapowe	320
Mapa bąbelkowa	322
Kartogram	323
Wykres kaskadowy	325

Wizualizacje Power Platform	326
Power Apps dla Power BI	327
Power Automate dla Power BI	328
Wizualizacje premium	331
Wizualizacje Metrics	331
Raporty podzielone na strony	333
Elementy raportu	334
Formatowanie wizualizacji	336
Etykiety	336
Wykresy kolumnowe i liniowe	339
Wizualizacje tabel i macierzy	342
Wykresy punktowe	351
Podsumowanie	352

ROZDZIAŁ 8

Stosowanie analizy zaawansowanej	353
Elementy wizualne AI	353
Kluczowe elementy mające wpływ	354
Drzewo dekompozycji	358
Pytania i odpowiedzi	359
Inteligentna narracja	361
Wizualizacje skryptów w językach R i Python	363
Element wizualny skryptu R	365
Wizualizacja języka Python	367
ArcGIS Maps for Power BI	368
Niestandardowe elementy wizualne	371
Dodawanie niestandardowego elementu wizualnego	372
Animacje i data storytelling	375
Odtwarzanie osi dla wykresów punktowych	376
Wizualizacja Pulse chart	377
Karta Analiza	379
Linia trendu	380
Prognozowanie	382
Szybki wgląd w szczegóły	384
Wyjaśnianie wzrostu lub spadku	385
Strony raportów zoptymalizowane pod kątem urządzeń mobilnych	388
Podsumowanie	390

ROZDZIAŁ 9

Projektowanie pulpitu nawigacyjnych	391
Pulpity nawigacyjne a raporty	391
Projekt pulpitu nawigacyjnego	395
Dobór wizualizacji	398
Układ graficzny pulpitu	400
Kafelki pomocnicze	403
Architektury pulpitu nawigacyjnych	403
Architektura jednopulpitowa	405
Architektura wielopulpitowa	406
Architektura korporacyjna	407
Wiele zbiorów danych	410
Kafelki pulpitu nawigacyjnego	411
Szczegóły kafelków i niestandardowe linki	413
Kafelki z danymi czasu rzeczywistego	414
Motywy pulpitu nawigacyjnych	416
Raporty podzielone na strony	417
Skoroszyty Excela	421
Strony raportów w trybie live	423
Pulpity nawigacyjne zoptymalizowane pod kątem urządzeń mobilnych	425
Podsumowanie	427

ROZDZIAŁ 10

Zarządzanie obszarami roboczymi i zawartością	428
Obszary robocze	429
Role i uprawnienia w obszarze roboczym	431
Zestawy danych w obszarach roboczych	437
Mój obszar roboczy	438
Wdrożenia etapowe	438
Zestawy danych w obszarze roboczym	441
Interfejs Power BI REST API i moduł PowerShell	442
Potoki wdrożeniowe Power BI	447
Treści wrażliwe i ich ochrona	450
Ochrona informacji	450
Zapobieganie utracie danych	453
Kontrola wersji	455
OneDrive dla Biznesu	456
Kontrola wersji dla kodu M i DAX	458
Zarządzanie metadanymi	460
Opisy pól	462
Metadata Mechanic	465

Raportowanie metadanych	466
Standardowe raporty z metadanymi	467
Podsumowanie	471

ROZDZIAŁ 11

Zarządzanie lokalną bramą danych472

Planowanie lokalnej bramy danych	473
Najważniejsze zadania związane z planowaniem bramy	477
Tryb standardowy a tryb osobisty	483
Różne aspekty bram	484
Klastry bram	484
Architektury bram danych	486
Zabezpieczenia bram	489
Instalacja i konfigurowanie bramy danych	491
Konto usługi lokalnej bramy danych	493
TCP kontra HTTPS	494
Łączniki danych	495
Klucze odzyskiwania	496
Zarządzanie klastrami bram	497
Administratorzy bram danych	499
Źródła danych i użytkownicy bramy	499
Obsługa klastrów danych za pomocą modułu PowerShell	501
Rozwiązywanie problemów i monitorowanie bram	502
Przywracanie, przenoszenie i przejmowanie bramy	502
Diagnostyka bramy	503
Raporty z monitoringu bramy	505
Odświeżanie danych	506
Zaplanowane odświeżanie danych	506
Zestawy danych w trybie DirectQuery	507
Połączenia na żywo z modelami Analysis Services	509
Odświeżenie bufora pulpitu nawigacyjnego	509
Podsumowanie	511

ROZDZIAŁ 12

Opracowywanie raportów podzielonych na strony512

Raporty podzielone na strony w usłudze Power BI	513
Planowanie raportów podzielonych na strony	514
Przygotowywanie i publikowanie raportów podzielonych na strony	515
Korzystanie z raportów podzielonych na strony	521

Przenoszenie raportów do usługi Power BI	524
Inwentaryzacja	524
Ocena	525
Planowanie	527
Migracja	528
Testy akceptacyjne i ostateczne wdrożenie	528
Planowanie serwera raportów Power BI (PBRs)	529
Różnice w porównaniu z usługą Power BI	531
Kompatybilność z SQL Server Reporting Services	532
Źródła danych i opcje połączeń	534
Licencjonowanie sprzętu i użytkowników	535
Alternatywne i hybrydowe modele wdrażania	536
Instalowanie i aktualizowanie serwera PBRs	540
Pozyskiwanie klucza produktu PBRs	541
Cykle aktualizacji PBRs	543
Aplikacje klienckie PBRs	544
Równoległe uruchamianie wersji desktopowych	544
Aplikacje mobilne Power BI	545
Podsumowanie	546

ROZDZIAŁ 13

Tworzenie aplikacji Power BI i dystrybucja treści	547
Metody dystrybucji treści	548
Aplikacje Power BI	550
Licencjonowanie aplikacji	550
Wdrażanie aplikacji	552
Uprawnienia odbiorców i zabezpieczenia aplikacji	554
Tworzenie i publikowanie aplikacji	557
Instalowanie aplikacji	562
Aktualizowanie aplikacji	564
Aplikacje Power BI w systemach mobilnych	565
Udostępnianie treści	566
Zakresy udostępniania	570
Udostępnianie bezpośrednie kontra aplikacje Power BI	571
Osadzanie treści	572
Licencjonowanie osadzania treści	572
Publikowanie w sieci	573
Bezpieczne osadzanie w internecie	575
Aplikacje Microsoft 365	576
Aplikacje niestandardowe	579

Alerty danych	583
Integracja z usługą Power Automate	585
Subskrypcje e-mailowe	586
Analizowanie w Excelu	588
Samoobsługowe obszary robocze BI	590
Samoobsługowa dystrybucja treści	591
Zagrożenia związane z samoobsługową analityką biznesową	592
Podsumowanie	593

ROZDZIAŁ 14

Zarządzanie Power BI na potrzeby organizacji	594
Rola administratora systemu Power BI	595
Zarządzanie danymi w Power BI	597
Implementacja zarządzania danymi	599
Usługa Azure Active Directory	601
Współpraca B2B w usłudze AAD	601
Zasady dostępu warunkowego	604
Portal administracyjny Power BI	607
Ustawienia dzierżawy	609
Metryki użycia	615
Użytkownicy i dzienniki inspekcji	615
Premium na użytkownika	616
Ustawienia pojemności	617
Kody osadzania	617
Wizualizacje organizacyjne	618
Połączenia platformy Azure	621
Obszary robocze	622
Znakowanie niestandardowe	623
Metryki ochrony	623
Polecana zawartość	623
Raporty metryk użycia	626
Dzienniki inspekcji	629
Rozwiązanie monitorujące dziennik inspekcji	634
Interfejsy Power BI REST dla administratorów	635
Podsumowanie	636

ROZDZIAŁ 15**Tworzenie korporacyjnej analizy biznesowej**

przy użyciu Power BI Premium	638
Pojemność Power BI Premium	639
Możliwości Power BI Premium	640
Węzły pojemności Premium	642
Zasoby frontendowe i backendowe	645
Szacowanie potrzebnej pojemności	647
Administrowanie pojemnością Premium i jej alokowanie	649
Alokacja pojemności	650
Tworzenie, ustalanie rozmiaru i monitorowanie pojemności	654
Przypisanie obszaru roboczego	659
Optymalizacja zasobów pojemności Premium	662
Optymalizacja modelu danych	662
Optymalizacja raportów i wizualizacji	664
Obciążenia	665
Zarządzanie cyklem życia treści w ramach pojemności Premium	667
Narzędzie ALM Toolkit	667
Zarządzanie zbiorami danych za pomocą SSMS	670
Tworzenie kopii zapasowych pojemności Premium	673
Podsumowanie	674
Skorowidz	675

Łączenie się ze źródłami i przekształcanie danych za pomocą języka M

Rozdział

3

W tym rozdziale, po przygotowaniu środowiska i źródła danych opisanym w rozdziale 2. „Przygotowanie źródeł danych”, przechodzimy do implementacji zapytań **Power Query (M)** w nowym pliku Power BI Desktop w celu pobrania wymaganych tabel faktów i wymiarów. Zapytania Power Query są zwykle pisane ręcznie w języku transformacji danych zwanym potocznie M, ale mogą też być generowane przy użyciu interfejsu o nazwie **Power Query Editor (Edytor Power Query)**. Zapytania te uzyskują dostęp do źródeł danych i opcjonalnie stosują logikę transformacyjną, aby przygotować tabele dla modelu danych w Power BI.

Wyrażenia Power Query (M) stają się wszechobecne w całej microsoftowej platformie przetwarzania danych. Są używane z przepływami danych, które mogą być stosowane wielokrotnie przez liczne zestawy danych w Power BI. Wyrażenia są również obsługiwane przez **Azure Data Factory (ADF)**, co oznacza, że procesy łączenia danych rozpoczęte w Power BI, mogą być w razie potrzeby skalowane. Wreszcie zapytania M stanowią podstawę przepływów danych w **Dataverse**, operacyjnym magazynie danych utrzymywanym przez firmę Microsoft.

Język M zawiera setki funkcji, a na temat niego samego i jego zastosowań napisano wiele książek. Głównym celem tego rozdziału jest ułatwienie zrozumienia zapytań M w kontekście korporacyjnego rozwiązania Power BI, które w głównej mierze wykonywane jest przez zarządzaną przez IT hurtownię danych.

W omawianych przykładach podstawowym źródłem danych dla zapytań Power Query (M) są system SQL Server i pliki Excela. Przez dobór odpowiednich parametrów i zmiennych uzyskujemy dostęp do zestawu widoków SQL odzwierciedlających tabele hurtowni

danych w systemie SQL Server oraz dane rocznego planu sprzedaży zawarte w skoroszycie Excela.

Za pomocą dodatkowych zapytań tworzymy relacje między planem sprzedaży i tabelami wymiarów, a także zwiększamy użyteczność zbioru danych i ułatwiamy zarządzanie nim. Przedstawiamy też przykłady przekształcania danych i logiki zapytań, np. przez utworzenie dynamicznej kolumny historii klienta. Na koniec omawiamy kwerendy wielokrotnego użytku zwane przepływami danych, a także narzędzia do edycji i zarządzania kwerendami, takie jak rozszerzenia dla **Visual Studio** i **Visual Studio Code**.

W tym rozdziale omawiamy następujące zagadnienia:

- rodzaje zapytań Power Query (M),
- tworzenie zapytań Power Query (M),
- przykłady zapytań Power Query (M),
- przepływy danych,
- narzędzia edycji zapytań Power Query (M).

Zacznijmy od przyjrzenia się różnym typom zapytań M.

Rodzaje zapytań Power Query M

W rozdziale 2. „Przygotowanie źródeł danych” utworzyliśmy widoki SQL, skonfigurowaliśmy źródła danych i ustawiliśmy opcje środowiska Power BI Desktop. Po wykonaniu tych zadań projektant zbioru danych może wreszcie przystąpić do opracowania zapytań pobierających dane i ustalenia parametrów zbioru danych. Kwerendy Power Query (M) są środkiem, za pomocą którego zbiory danych w Power BI łączą się ze źródłami i importują dane. Kwerendy M są niezbędne do łączenia się ze źródłami, takimi jak widoki SQL, a także mogą przekształcać dane zgodnie z wymaganiami.

Język M stosowany w Power Query jest językiem programowania nazywanym bardziej formalnie językiem formuł Power Query. M zawiera ponad 700 funkcji służących do łączenia się z danymi i wykonywania transformacji tych danych. Kod, który nawiązuje połączenie z danymi i przekształca je, nazywany jest zapytaniem lub kwerendą.

Istnieje wiele różnych typów zapytań, które służą różnym celom:

- zapytania o parametry źródła danych,
- kwerendy przejściowe,
- zapytania o fakty i wymiary,
- zapytania do tabeli parametrów,

- zapytania do tabeli zabezpieczeń,
- niestandardowe kwerendy funkcyjne.

W tym rozdziale szczegółowo omówimy wszystkie typy, jednak przed zagłębieniem się w te rozważania należałoby powiedzieć kilka słów na temat organizowania kwerend w celu uzyskania rozwiązań łatwiejszych do zrozumienia i możliwych do utrzymania przez dłuższy czas.

Organizacja zapytań

W **Edytorze Power Query** działającym w ramach aplikacji Power BI Desktop foldery grupowe służą do organizowania zapytań w określone kategorie, takie jak Data Source Parameters (parametry źródła danych), Staging Queries (kwerendy przejściowe), Parameter Tables (tabele parametrów), Fact Table Queries (zapytania tabeli faktów), Dimension Table Queries (zapytania tabeli wymiarów) i Bridge Table Queries (zapytania tabeli pomostowej), jak w przykładzie z rysunku 3.1.

ID	Name	Data	Schema	Item	Kind
1	BI.vDim_Account	Table	BI	vDim_Account	View
2	BI.vDim_Currency	Table	BI	vDim_Currency	View
3	BI.vDim_Customer	Table	BI	vDim_Customer	View
4	BI.vDim_Date	Table	BI	vDim_Date	View
5	BI.vDim_Department	Table	BI	vDim_Department	View
6	BI.vDim_Employee	Table	BI	vDim_Employee	View
7	BI.vDim_Geography	Table	BI	vDim_Geography	View
8	BI.vDim_Organization	Table	BI	vDim_Organization	View
9	BI.vDim_Product	Table	BI	vDim_Product	View
10	BI.vDim_ProductCategory	Table	BI	vDim_ProductCategory	View
11	BI.vDim_ProductSubcategory	Table	BI	vDim_ProductSubcategory	View
12	BI.vDim_Promotion	Table	BI	vDim_Promotion	View
13	DI.vDim_Reseller	Table	DI	vDim_Reseller	View
14	BI.vDim_SalesReason	Table	BI	vDim_SalesReason	View
15	BI.vDim_SalesTerritory	Table	BI	vDim_SalesTerritory	View
16	BI.vDim_Scenario	Table	BI	vDim_Scenario	View

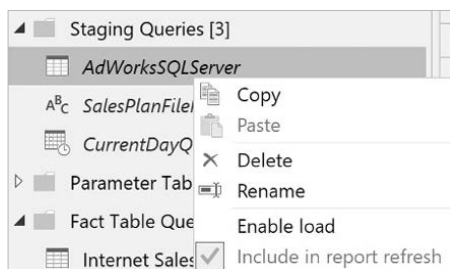
Rysunek 3.1. Edytor Power Query w Power BI Desktop z folderami grupowymi

Aby utworzyć nową grupę, należy kliknąć prawym przyciskiem myszy zapytanie w panelu *Queries (Zapytania)*, a następnie wybrać kolejno polecenia *Move to Group (Przenieś do grupy)* i *New Group (Nowa grupa)*. Po utworzeniu grup zapytania można przenosić pomiędzy nimi metodą przeciągania i upuszczania lub przez kliknięcie prawym przyciskiem myszy na zapytaniu, wybranie polecenia *Move to Group (Przenieś do grupy)*, a następnie wskazanie grupy docelowej.

Parametry i zapytania zapisane kursywą są uwzględniane w procesie odświeżania zbioru danych, ale nie są ładowane do zbioru danych w ramach Power BI Desktop. Na przykład zapytanie *AdWorksSQLServer* zaznaczone na rysunku 3.1 jedynie eksponuje obiekty

bazy danych SQL Server jako wynik działania funkcji `Sql . Database() M`, aby inne zapytania mogły się do nich odwoływać. Zapytanie to, wraz z parametrami źródła danych, jest zapisane kursywą i służy do usprawnienia procesu pobierania danych w taki sposób, że pojedyncza zmiana może być uwzględniana w aktualizacji wielu zapytań zależnych.

Kliknij prawym przyciskiem myszy zapytanie lub parametr w panelu *Queries (Zapytania)*, aby otworzyć menu z opcjami *Enable load (Włącz ładowanie)* i *Include in report refresh (Uwzględnij w odświeżeniu raportu)*, jak na rysunku 3.2.



Rysunek 3.2. Włączenie ładowania i uwzględniania w odświeżeniu raportu

Dla wielu zbiorów danych jedynymi zapytaniami, które powinny zostać załadowane do modelu, są te dotyczące tabel wymiarów i faktów oraz niektóre dotyczące tabeli parametrów. Ponadto może się zdarzyć, że dane w systemie źródłowym będą statyczne czyli niezmiennie. W takich przypadkach nie ma powodu, aby odświeżać dane w zbiorze po pierwszym załadowaniu, dlatego opcję *Include in report refresh (Uwzględnij w odświeżeniu raportu)* można wyłączyć, aby zaoszczędzić zasoby systemowe i skrócić czas odświeżania.

Przyjrzymy się teraz szczegółowo różnym typom zapytań M, zaczynając od parametrów źródła danych.

Parametry źródła danych

Na rysunku 3.1 w poprzednim punkcie pokazano grupę o nazwie *Data Source Parameters* zawierającą parametry źródła danych. Parametry to specjalne zapytania, które nie uzyskują dostępu do zewnętrznego źródła danych i zwracają jedynie konkretną wartość skalarną, taką jak data, liczba lub ciąg znaków tekstowych.

Podstawowym zastosowaniem parametrów jest zdefiniowanie jednego wspólnego elementu, takiego jak nazwa serwera lub bazy danych, a następnie odwołanie się do niego w wielu innych zapytaniach. Podobnie jak zmienne globalne, tak i parametry usprawniają zarządzanie dużymi zbiorami danych, ponieważ projektant takiego zbioru w razie potrzeby zmiany tego elementu może to po prostu zrobić w jednym miejscu (przez modyfikację stosownego parametru), a nie w każdym zapytaniu oddzielnie.

Poza tym twórcy zbiorów danych Power BI mogą używać parametrów, aby załadować do lokalnego pliku w Power BI Desktop tylko próbkę danych z tabeli źródłowej, którą w całości załadują później do opublikowanego już zbioru danych Power BI. Można na przykład utworzyć parametry daty początkowej i końcowej, a następnie wstawić je do wyrażenia ustalającego warunek filtrowania ładowanej tabeli.

Lokalny plik aplikacji Power BI Desktop może używać parametrów reprezentujących tylko pojedynczy miesiąc lub określony zakres lat, ale opublikowany zbiór danych może załadować wiele lat w oparciu o zmienione wartości parametrów daty początkowej i końcowej. Autor zbioru danych może manipulować wartościami parametrów ręcznie w usłudze Power BI lub za pomocą skryptu modyfikującego parametry za pośrednictwem Power BI REST API. Interfejs Power BI REST jest opisany w rozdziale 10. „Zarządzanie obszarami roboczymi i zawartością”, w punkcie „Power BI REST API i moduł PowerShell”.

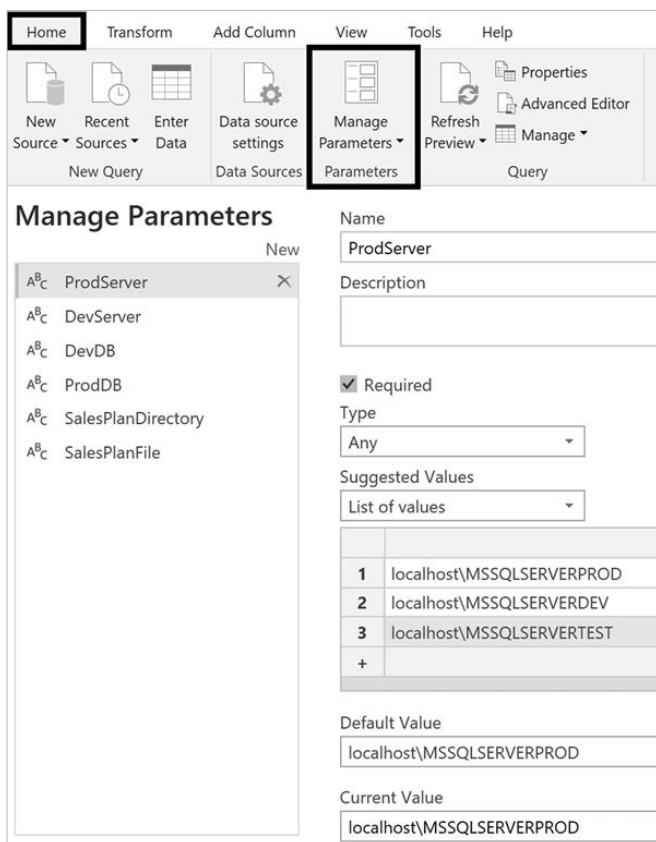
Parametry zapytań można tworzyć i modyfikować za pomocą okna dialogowego *Manage Parameters (Zarządzaj parametrami)* dostępnego na karcie *Home (Narzędzia główne) Edytora Power Query*. Na rysunku 3.3 przedstawiono to okno z sześcioma parametrami zdefiniowanymi dla bazy danych SQL Server i skoroszytu Microsoft Excel.

Dla tego zbioru danych parametry bazy danych środowiska programistycznego i produkcyjnego (na przykład ProdServer i ProdDB) mają skonfigurowaną listę możliwych wartości, co ułatwia pracę i pozwala uniknąć błędów przy przełączaniu źródeł danych. W tym samym celu zarówno nazwa skoroszytu Excela zawierającego roczny plan sprzedaży i marży, jak i katalog jego plików są również zapisane jako parametry.

Lista rozwijana *Suggested Values (Sugerowane wartości)* pozwala na ręczne wprowadzenie dowolnej wartości, wybór wartości z zakodowanej listy lub użycie zapytania zwracającego stosowną listę (typ wartości w języku M, taki jak tabela czy rekord). Ze względu na niewielką liczbę nazw serwerów w tym przykładzie oraz rzadkość zmiany nazw serwerów produkcyjnych i deweloperskich wprowadzono ręcznie trzy sugerowane wartości.

Parametry są często używane z plikami **szablonów Power BI**, aby umożliwić użytkownikom biznesowym modyfikowanie własnych raportów za pomocą predefiniowanych oraz wstępnie przefiltrowanych zapytań i miar. Na przykład użytkownik otwiera szablon i wybiera konkretny dział, a ten wybór jest wykorzystywany do filtrowania zapytań w zbiorze danych.

Poza tym parametry mogą być przydatne podczas definiowania wartości używanych w filtrach zapytań (np. daty początkowe i końcowe) oraz w logice obliczeniowej używanej do tworzenia niestandardowych kolumn w zapytaniach. Parametry są zwykle wykorzystywane tylko przez kwerendy i dlatego nie są ładowane (kursywa na rysunku 3.3),



Rysunek 3.3. Zarządzanie parametrami w Edytorze Power Query

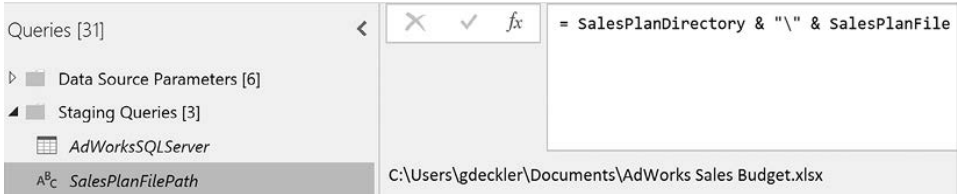
choć mogą być ładowane do modelu danych jako pojedyncze tabele z pojedynczą kolumną i pojedynczym wierszem. Po załadowaniu parametry mogą być dostępne dla wyrażeń DAX, tak jak inne tabele w modelu.

Teraz zajmiemy się kwerendami przejściowymi.

Kwerendy przejściowe

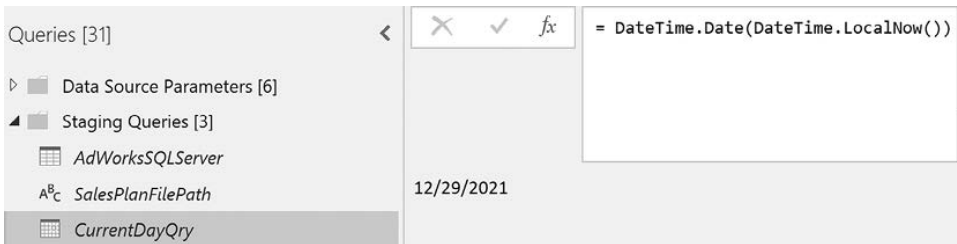
Po skonfigurowaniu parametrów źródła danych tworzy się zapytania przejściowe (ang. *staging queries*) w celu wyeksponowania źródła danych dla zapytań dotyczących tabel wymiarów i faktów w zbiorze danych. Na przykład zapytanie przejściowe `AdWorksSQLServer` przekazuje jedynie wartości parametrów określające serwer produkcyjny i produkcyjną bazę danych do funkcji `Sql.Database()`, tak jak na rysunku 3.1. Wynikiem tego zapytania jest tabela zawierająca schematy i obiekty przechowywane w bazie danych, w tym widoki SQL z tabelami faktów i wymiarów.

Kwerenda przejściowa `SalesPlanFilePath` używana w odniesieniu do źródła `Annual Sales Plan` (roczny plan sprzedaży) w Excelu jest bardzo podobna, gdyż tak samo odwołuje się jedynie do sparametryzowanych nazw pliku i katalogu w celu utworzenia pełnej ścieżki, tak jak na rysunku 3.4.



Rysunek 3.4. Kwerenda przejściowa rocznego planu sprzedaży — skoroszyt Excela

Trzecie i ostatnie zapytanie przejściowe, `CurrentDateQry`, po prostu wyznacza bieżącą datę, co pokazano na rysunku 3.5.



Rysunek 3.5. Kwerenda przejściowa bieżącej daty

Podobnie jak parametry, wyniki kwerend przejściowych, takich jak `CurrentDateQry`, mogą być przywoływane przez inne kwerendy, np. w warunku filtrowania tabeli faktów. W poniższym przykładzie funkcja `Table.SelectRows()` (wybierz wiersze) ma za zadanie pobrać tylko te wiersze z tabeli faktów, w których kolumna `Order Date` (data zamówienia) zawiera wartości mniejsze lub równe temu, co zwraca kwerenda `CurrentDateQry` (np. `12/29/2021`).

```
let
    Source = AdWorksSQLServer,
    ISales = Source[Schema = "BI", Item = "vFact_InternetSales"][Data],
    CurrentDateFilter = Table.SelectRows(ISales, each [Order Date]
    <= CurrentDateQry)
in
    CurrentDateFilter
```

W tym prostym przypadku ten sam warunek filtrujący może być łatwo wbudowany w widok SQL (`vFact_InternetSales`) zawierający tabelę faktów i takie podejście jest na ogół preferowane. Należy jednak zauważyć, że silnik M jest w stanie przekształcić

zmienną zapytania (`CurrentDateFilter`) wraz z odwołaniem do zapytania przejściowego (`CurrentDateQry`) w pojedynczą instrukcję SQL przez **składanie zapytań**.

W niektórych scenariuszach transformacji danych, szczególnie przy szybkich iteracjach i cyklach życia projektów w podejściu zwinnym, wskazane może być przynajmniej tymczasowe wykorzystanie wydajnych zapytań M w ramach zbioru danych Power BI (lub modelu Analysis Services) zamiast wprowadzania zmian w źródle danych (na przykład w tabelach lub widokach hurtowni danych).

Jak zostało to opisane w podrozdziale „Składanie zapytań” rozdziału 2. „Przygotowanie źródeł danych”, jeśli jest konieczne użycie języka M do implementacji transformacji zapytań lub logiki, projektant zbioru danych powinien upewnić się, że ta logika jest składana w instrukcję SQL i w ten sposób wykonywana przez system źródłowy. Jest to szczególnie ważne w przypadku dużych zapytań pobierających miliony wierszy przy ograniczonych zasobach serwera bramy danych (jeśli jest stosowany) lub przydzielonych mocy przerobowych (sprzętowych) w Power BI Premium.

Kwerendy przejściowe mogą być również używane z zapytaniami `DirectQuery`.

Przejściowość w `DirectQuery`

Możliwe jest użycie zapytań przejściowych nawet podczas pracy w trybie `DirectQuery`. Przejściowe zapytanie bazodanowe dla zbioru danych w trybie `DirectQuery` jest nieco inne niż dla zbioru danych w trybie importu. W przypadku takiego zapytania do wyrażenia `let` dodawana jest nowa zmienna, jak pokazano w poniższym przykładzie:

```
let
    Source = Sql.Database(ProdServer, ProdDB),
    DummyVariable = null
in
    Source
```

Dodatkowa zmienna, `DummyVariable`, jest pomijana przez zapytanie, a sama funkcja, `Sql.Database()`, która odwołuje się do parametrów serwera i bazy danych ze zbioru danych w trybie importu, może być również użyta w odniesieniu do zbioru danych `DirectQuery`.

Po zdefiniowaniu parametrów i zapytań przejściowych możemy skupić się na głównych zapytaniach o dane dla naszych faktów i wymiarów.

Zapytania o fakty i wymiary

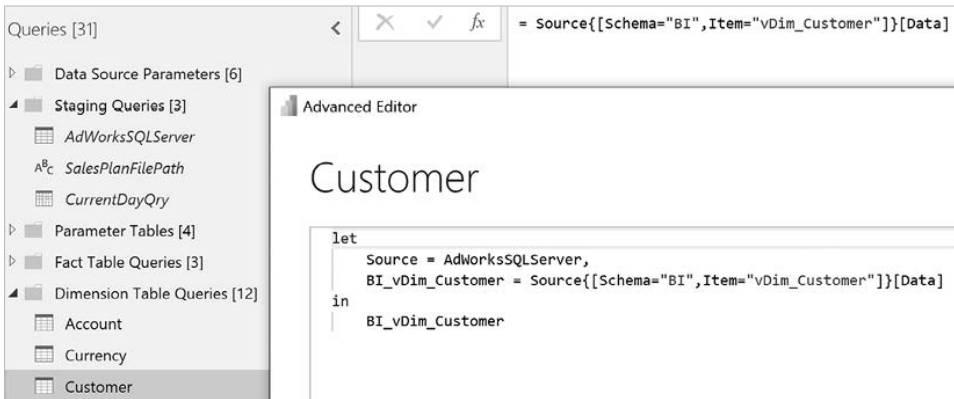
Cała dotychczasowa praca była tak naprawdę tylko przygotowaniem do opracowania zapytań łączących się ze źródłem i importujących interesujące nas dane dla naszych faktów i wymiarów.

Dla zbiorów danych w trybie importu zapytanie jest wykonywane podczas zaplanowanego odświeżenia, a wyniki są ładowane do skompresowanego formatu kolumnowego. Zbiory danych w trybie DirectQuery i zapytania importowe z wyłączoną właściwością *Enable load (Włącz ładowanie)* (zob. punkt „Organizacja zapytań”) definiują jedynie instrukcję SQL reprezentującą określone tabele wymiarów lub faktów. Źródło danych DirectQuery wykorzystuje tego typu instrukcje do tworzenia zapytań SQL niezbędnych do wykonania zapytań raportowych, takich jak połączenie zapytania Internet Sales (sprzedaż internetowa) z zapytaniem Product (produkt).

Po odpowiednim przygotowaniu większość zapytań w języku M powinna po prostu uzyskać dostęp do pojedynczego widoku SQL, zastosować niewielkie lub żadne transformacje, a następnie przekazać wyniki do zbioru danych jako tabelę wymiarów lub faktów. Takie zapytania są określane jako zapytania źródłowe wyłącznie referencyjne (ang. *source reference-only queries*).

Tylko odniesienie do źródła

Zapytanie w języku M przedstawione na rysunku 3.6 odwołuje się do widoku SQL (BI.vDim_Customer) poprzez zapytanie przejściowe (AdWorksSQLServer) i nie wprowadza żadnych dalszych transformacji:



Rysunek 3.6. Zapytanie o wymiar klienta

Jak pokazano na rysunku 3.6, zapytanie Customer (klient) uzyskuje dostęp do unikatowego rekordu związanego ze schematem (BI) i widokiem SQL (vDim_Customer) tabeli wytworzonej przez zapytanie przejściowe (AdWorksSQLServer). Rekord ten zawiera wszystkie nazwy pól tabeli zapytania przejściowego, w tym pole Data (dane) przechowujące widok SQL. Odwołanie się do pola Data we wspomnianym rekordzie powoduje pobranie wyników widoku SQL.

Ponieważ nie są stosowane żadne transformacje, zapytanie odzwierciedla źródłowy widok SQL, a zmiany w tym widoku, takie jak usunięcie kolumny, są automatycznie przenoszone do zbioru danych Power BI przy następnym odświeżeniu. Relacja jeden do jednego pomiędzy widokiem SQL a zapytaniem jest jednym z głównych powodów, dla których warto zaimplementować lub przenieść logikę transformacji danych do źródła hurtowni danych, a nie do zbioru danych Power BI.

Połączenie z bazami hurtowni danych, takimi jak Azure SQL Database lub Azure SQL Managed Instance, powinno zazwyczaj skutkować prostymi wyrażeniami zapytań w języku M z niewielką liczbą wymaganych transformacji. Źródła częściowo ustrukturyzowane i nieustrukturyzowane, takie jak pliki JSON i Excela, wymagają oczywiście więcej transformacji w celu przygotowania danych do analizy. Następnie przyjrzymy się kwerendzie o większej złożoności spowodowanej nawiązaniem połączenia z nieustrukturyzowanym źródłem danych, jakim jest Microsoft Excel.

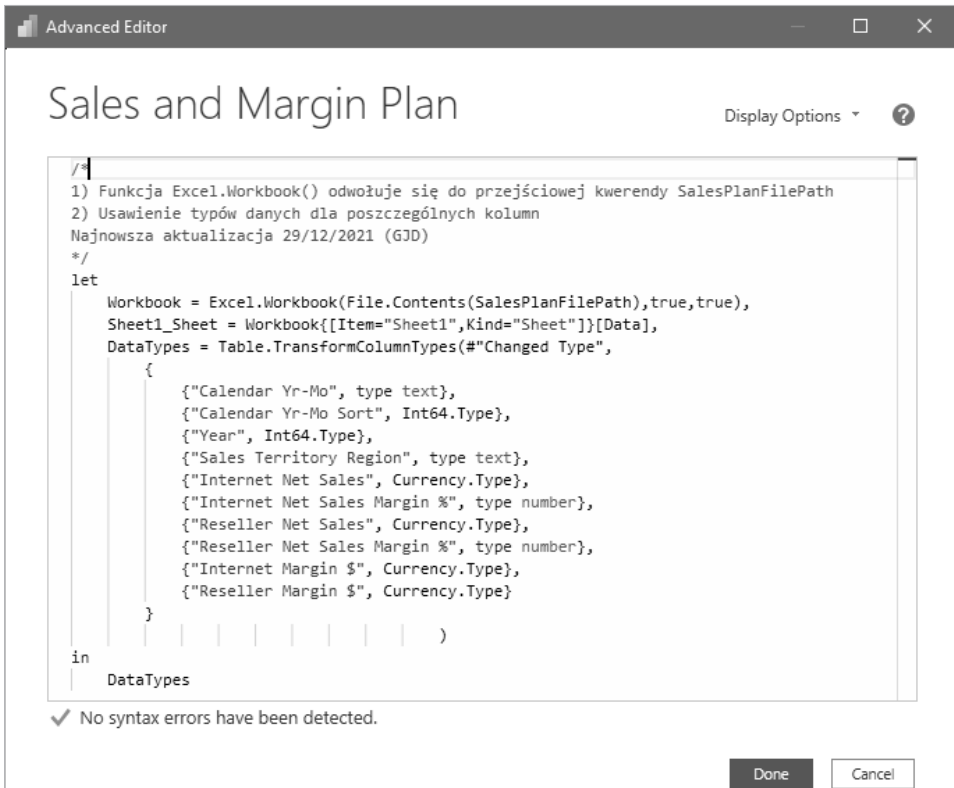
Skoroszyt Excela — roczny plan sprzedaży

Jak wykazaliśmy w poprzednim punkcie, zapytania o fakty i wymiary dla ustrukturyzowanych źródeł danych, takich jak bazy SQL Server, powinny zasadniczo mieć charakter wyłącznie referencyjny. Jednak w przypadku zbioru danych w trybie importu dane dotyczące rocznego planu sprzedaży i marży są pobierane z obiektu tabelarycznego należącego do skoroszytu Excela, czyli do nieustrukturyzowanego źródła danych.

W kwerendzie tabeli faktów pokazanej na rysunku 3.7, Sales and Margin Plan (plan sprzedaży i marży), funkcja udostępniająca dane, Excel.Workbook(), wywołuje zapytanie przejściowe SalesPlanFilePath (ścieżka pliku z planem sprzedaży).

Zgodnie z tym, co napisaliśmy w rozdziale 2. „Przygotowanie źródeł danych” (punkt „Opcje programu Power BI Desktop”), opcja automatycznego wykrywania typów danych dla źródeł nieustrukturyzowanych powinna być wyłączona. Źródła danych ustrukturyzowane, takie jak SQL Server, jawnie określają typy danych dla kolumn. W przypadku źródeł nieustrukturyzowanych funkcja automatycznego wykrywania typu danych próbuje określić odpowiedni typ dla każdej kolumny.

Przy wyłączonym automatycznym wykrywaniu typów danych konieczne jest jawne określenie odpowiedniego typu danych dla każdej kolumny tabeli Excela za pomocą funkcji Table.TransformColumnTypes(). Argumenty Int64.Type, Currency.Type i type number użyte w tej funkcji odpowiadają typom danych *Whole Number* (Liczba całkowita), *Fixed Decimal Number* (Stałoprzecinkowa liczba dziesiętna) i *Decimal Number* (Liczba dziesiętna).



Rysunek 3.7. Zapytanie Sales and Margin Plan dla skoroszytu Excela

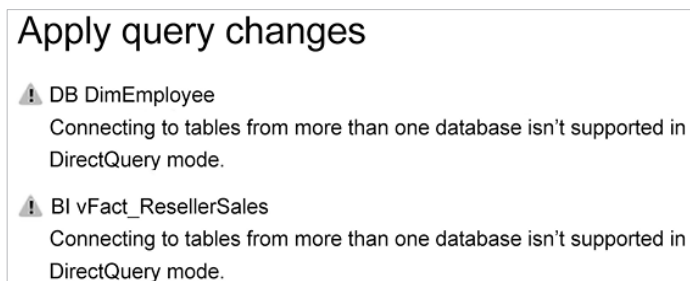
W przypadku zbioru danych w trybie DirectQuery dane dotyczące planu sprzedaży i marży byłyby pobierane z widoku SQL w ramach tej samej bazy danych, z której pobierane są inne tabele faktów i wymiarów, jak pokazano na rysunku 3.8.



Rysunek 3.8. Zapytanie Sales and Margin Plan dla zbioru danych w trybie DirectQuery

Koszt i czas integracji danych z kwerendy Sales and Margin Plan z bazą hurtowni danych są jednym z powodów, dla których w tym projekcie wybrano dla zbioru danych domyślny tryb importu. Ograniczenie pojedynczej bazy danych w ramach jednego źródła danych

jest obecnie jednym z podstawowych czynników ograniczających zbiory danych w trybie DirectQuery. Na rysunku 3.9 widoczny jest błąd zgłoszony przy próbie połączenia się w trybie DirectQuery z dwoma bazami danych funkcjonującymi na tym samym serwerze.



Rysunek 3.9. Ograniczenie trybu DirectQuery — jedna baza danych

To ograniczenie DirectQuery można ominąć, używając złożonych modeli danych, ale wtedy wzrasta złożoność, która jest na ogół niepożądana.

A teraz przyjrzyjmy się kwerendom przeznaczonym do tworzenia tabel wspomagających budowanie relacji w obrębie modelu danych.

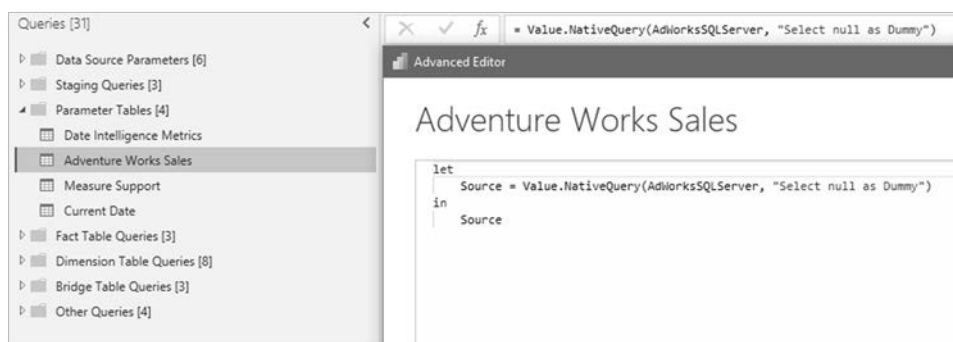
Kwerendy tabel parametrów

Zapytania tabel parametrów tworzy się dla celów użyteczności i zarządzania.

Z punktu widzenia użyteczności kwerendy Data Intelligence Metrics (metryki analizy danych) i Adventure Works Sales (sprzedaż w Adventure Works) służą do konsolidacji powiązanych miar na liście pól i udostępniania ich autorom raportów oraz analitykom, aby ci nie musieli przewijać całych tabel w poszukiwaniu potrzebnych miar. Kwerenda Current Date (bieżąca data) dostarcza raportom wiadomość tekstową informującą o dacie ostatniej operacji odświeżenia danych. Z punktu widzenia zarządzania zapytanie Measure Support (obsługa miar) służy do centralizacji pośrednich lub rozgałęzionych wyrażen DAX, z których korzysta wiele miar DAX.

Jak widać na rysunku 3.10, w przypadku trzech z czterech zapytań zastosowano wyrażenie trywialne, ponieważ chodzi wyłącznie o przekazanie nazwy tabeli do modelu danych.

Kwerendy Date Intelligence Metrics, Adventure Works Sales i Measure Support pobierają pustą wartość, a właściwość *Include in report refresh* (*Uwzględnij w odświeżeniu raportu*) mają wyłączoną. W rozdziale 4. „Projektowanie modeli danych w trybach importu, DirectQuery i złożonym” pokazujemy, jak te puste tabele można wykorzystać w systemie metadanych modelu danych, a w rozdziale 5. „Tworzenie miar DAX i ról zabezpieczeń” omawiamy dodawanie miar DAX do zbioru danych.



Rysunek 3.10. Kwerenda Adventure Works Sales z grupy Parameter Tables

Zapytanie `Current Date` jest jedynym zapytaniem tabeli parametrów wykonywanym przy każdym odświeżeniu raportu. Poniższy skrypt M dla zapytania `Current Date` tworzy tabelę z jedną kolumną i jednym rekordem reprezentującym datę w momencie wykonania.

```
let
    RefreshDateTime = DateTime.LocalNow(),
    TimeZoneOffset = -5,
    RefreshDateTimeAdjusted = RefreshDateTime +
#duration(0,TimeZoneOffset,0,0),
    RefreshDateAdjusted = DateTime.Date(RefreshDateTimeAdjusted),
    TableCreate = Table.FromRecords({[CurrentDate = RefreshDateAdjusted]}),
    DateType = Table.TransformColumnTypes(TableCreate,{"CurrentDate", type
    ↪date})
in
    DateType
```

Wszystkie raportowane czasy w Microsoft Azure są wyrażone w **uniwersalnym czasie koordynowanym (UTC)**. Aby więc mieć pewność, że ostatni odświeżony komunikat o dacie odzwierciedla lokalną strefę czasową, należy w zapytaniu zawrzeć stosowną logikę.

W powyższym przykładzie wartość zwracana przez funkcję `DateTime.LocalNow()` jest pomniejszana o pięć godzin, co odpowiada różnicy pomiędzy czasami wschodnioamerykańskim i UTC. Ze zmodyfikowanej w ten sposób wartości wyodrębniono datę i zapisano ją w utworzonej właśnie w tym celu tabeli.

Jak wynika z rysunku 3.11, zapytanie `Current Date` jest wykorzystywane przez miarę DAX do informowania o ostatnio zaktualizowanej dacie.

Obliczenie bieżącej daty i godziny w zapytaniu Power BI przechwytuje datę i godzinę w momencie odświeżenia i może być potencjalnie wykorzystane jako zapytanie pomocnicze dla innych zapytań w obrębie zbioru danych.



Rysunek 3.11. Tabele parametrów w liście Data (Dane) i komunikat o odświeżeniu danych

Wyrażenie DAX `Current Date` po prostu umieszcza te informacje w raporcie i jednocześnie wprowadza dodatkowy kontekst dla przeglądających raport.

Wyrażenie DAX obsługujące ostatni zaktualizowany komunikat jest następujące:

```
Last Refresh Msg =
VAR __CurrentDateValue = MAX("Bieżąca data"[CurrentDate])
RETURN
    "Last Refreshed: " & __CurrentDateValue
```

Dodatkowy przykład wykorzystania języka DAX do zwracania wartości łańcuchowej na potrzeby tytułu lub etykiety znajduje się w rozdziale 6. „Planowanie raportów Power BI” (podrozdział „Strony uściślające raport”).

W miarę jak zbiory danych stają się coraz większe i bardziej złożone, można starać się lepiej zorganizować zapytania M przez wprowadzanie nowych grup lub zmienianie nazw grup istniejących. Na przykład grupa czterech zapytań prezentowana w tym rozdziale spełnia trzy różne funkcje (lista pól danych, data ostatniego odświeżenia i centralizacja logiki DAX).

Dla doświadczonych programistów Power BI i Analysis Services Tabular tabela parametrów jest po prostu niestandardową tabelą wartości parametrycznych załadowaną do modelu i dostępną dla interfejsu raportowania. Miary DAX powinny być tak zaprojektowane, aby wykrywały, która wartość (parametr) została wybrana przez użytkownika (na przykład 10% wzrostu, 20% wzrostu), i dynamicznie obliczały wynik. W takim zbiorze danych koncepcja tabeli parametrów ulega rozszerzeniu na każde zapytanie załadowane do modelu danych, ale niezwiązane z żadną inną tabelą w tym modelu.

Większość dużych zbiorów danych Power BI zawiera parametry źródła danych, kwerendy przejściowe, kwerendy faktów i wymiarów, kwerendy tabel relacji oraz kwerendy tabel parametrów. Teraz omówimy dwa opcjonalne typy kwerend: kwerendy tabel zabezpieczeń i kwerendy funkcji niestandardowych.

Kwerendy tabel zabezpieczeń

Kwerendy tabel bezpieczeństwa ułatwiają wdrażanie **zabezpieczeń na poziomie wierszy (RLS)** dla zbioru danych. Takie zapytania mogą być wymagane w sytuacji, gdy każdy użytkownik powinien widzieć wyłącznie swoje dane. Wówczas tabela zabezpieczeń może importować **nazwy główne użytkowników (UPN)** z dostępem do raportów tworzonych na podstawie zbioru danych. Zazwyczaj UPN-y pokrywają się z adresami e-mailowymi użytkowników i za pomocą funkcji `DAX USERPRINCIPALNAME()` można je pobierać w celu ustanowienia zabezpieczeń lub wprowadzenia innych filtrów w usłudze Power BI.

W przypadku projektu opisanego w rozdziale 1. „Planowanie projektów Power BI” zabezpieczenie danych przez ustawienie roli RLS nie wymaga pobierania żadnych tabel.

Zgodnie z tym, co powiedzieliśmy w rozdziale 1. „Planowanie projektów Power BI” (w punkcie „Przykładowy szablon projektu Power BI”), kierownicy sprzedaży i ich współpracownicy powinni mieć dostęp tylko do swoich grup terytorialnych (Sales Territory), natomiast wiceprezesi powinni mieć dostęp globalny.

Przy takich prostych wymaganiach można utworzyć grupy bezpieczeństwa użytkowników (na przykład Ameryka Północna, Europa, region Pacyfiku) i przypisać je do odpowiednich ról RLS ustanowionych w modelu danych. Szczegółowe informacje na temat implementacji tych ról znajdują się w rozdziale 5. „Tworzenie miar DAX i ról zabezpieczeń”.

W projektach o bardziej złożonych lub drobiazgowych wymaganiach bezpieczeństwa często konieczne jest załadowanie do modelu danych dodatkowych tabel, takich jak Users (użytkownicy) i Permissions (uprawnienia). Na przykład, jeśli na użytkowników miałyby być nałożone ograniczenia dotyczące kodów pocztowych, a nie terytorialnych grup sprzedaży, lepsze byłoby podejście dynamiczne oparte na tabelach i filtrach uwzględniających użytkownika żądającego raportu niż tworzenie (i utrzymywanie) dużej ilości różnych ról RLS i grup bezpieczeństwa.

Ze względu na znaczenie zabezpieczeń dynamicznych (opartych na użytkownikach), szczególnie w przypadku dużych zbiorów danych, w rozdziale 5. „Tworzenie miar DAX i ról zabezpieczeń” omówimy szczegółowe przykłady implementacji takich zabezpieczeń zarówno dla zbiorów danych importowanych, jak i w trybie DirectQuery.

Teraz przyjrzymy się kolejnemu opcjonalnemu typowi zapytań, czyli zapytaniom funkcyjnym.

Niestandardowe zapytania funkcyjne

Ponieważ język zapytań w Power Query (M) jest funkcyjnym językiem programowania, możliwe jest tworzenie niestandardowych funkcji w formie zapytań. Kwerendy te umożliwiają tworzenie kodu wielokrotnego użytku, który można stosować przy wykonywaniu powtarzających się transformacji danych, takich jak niestandardowa analiza składniowa lub obliczenia. Oto prosty przykład niestandardowej funkcji do obliczania wieku klientów w tabeli Customer (klient):

```
let CalculateAge = (BirthDate as date) =>
    Date.Year(CurrentDayQuery) - Date.Year(BirthDate)
in CalculateAge
```

Ta niestandardowa funkcja przyjmuje jeden parametr, BirthDate (data urodzenia), który musi być typu data. Funkcja Date.Year (data.rok) jest używana zarówno w odniesieniu do zapytania CurrentDayQuery (data bieżąca), jak i parametru BirthDate, a różnica tych dwóch wyników zwraca liczbę lat. Całość można zapisać jako zapytanie o nazwie GetAge (oblicz wiek) i używać go w ramach niestandardowej formuły dla kolumny w tabeli Customer w następujący sposób:

```
= GetAge([BirthDate])
```

Dodatkowe przykłady i zastosowania funkcji niestandardowych można znaleźć w oficjalnej dokumentacji firmy Microsoft pod adresem <https://bit.ly/33VJfRz>.

Na tym kończymy omawianie rodzajów zapytań Power Query (M), które można tworzyć podczas opracowywania zbioru danych. Teraz zajmiemy się bardziej szczegółowo samymi zapytaniami pisanymi w języku M.

Tworzenie zapytań w języku M

Jak wspomniano, język M jest funkcyjnym językiem programowania, który zawiera ponad 700 funkcji. Podobnie jak inne języki programowania, M ma swoją specyficzną składnię, strukturę, operatory i typy danych, których należy używać podczas kodowania. Doświadczeni programiści Power Query (M) bardzo dobrze znają listy, rekordy i tabele oraz funkcje dostosowane do pracy z tymi specyficznymi typami.

Pełny opis języka M wykracza poza zakres tej książki, ale w kolejnych punktach poruszymy wiele ważnych tematów dotyczących tego języka, jak również podamy przykłady bardziej złożonych zapytań. Czytelników zainteresowanych lepszym poznaniem języka M odsyłamy do jego oficjalnej dokumentacji, którą można znaleźć pod adresem <https://bit.ly/3vmFSyr>.

Na początek przyjrzymy się numerycznym typom danych w M.

Numeryczne typy danych

W przypadku strukturalnych źródeł danych, takich jak SQL Server, typy danych kolumn źródłowych określają typy danych stosowane w Power BI. Na przykład typ danych waluty lub pieniędzy w SQL Server skutkuje typem danych *Fixed Decimal Number* (*Stałoprzecinkowa liczba dziesiętna*) w Power BI. Podobnie liczby całkowite w SQL Server wymuszają typ danych *Whole Number* (*Liczba całkowita*), a liczby dziesiętne w SQL Server przekładają się na typ *Decimal Number* (*Liczba dziesiętna*) w Power BI.

Gdy zapytanie jest ładowane do modelu danych w zbiorze Power BI, typ numeryczny w SQL Server (np. 19,4) staje się typem *Fixed Decimal Number* (*Stałoprzecinkowa liczba dziesiętna*). Dzięki czterem cyfrom po przecinku typ ten pozwala uniknąć błędów zaokrąglania. Typ *Decimal Number* (*Liczba dziesiętna*) jest odpowiednikiem zmiennoprzecinkowego lub przybliżonego typu danych z ograniczeniem do 15 cyfr znaczących.

Ze względu na możliwość wystąpienia błędów zaokrąglenia w przypadku danych typu *Decimal Number* (*Liczba dziesiętna*) oraz przewagę wydajnościową typu *Fixed Decimal Number* (*Stałoprzecinkowa liczba dziesiętna*), zaleca się przechowywanie liczb ułamkowych jako *Fixed Decimal Number* (*Stałoprzecinkowa liczba dziesiętna*), jeśli tylko wystarczająca jest precyzja czterocyfrowa. Wszystkie kolumny liczbowe typu całkowitego powinny być przechowywane w Power BI jako typy *Whole Number* (*Liczba całkowita*).

Kolumnom numerycznym w zapytaniach M można przypisywać typ danych *Whole Number* (*Liczba całkowita*), *Fixed Decimal Number* (*Stałoprzecinkowa liczba dziesiętna*) lub *Decimal Number* (*Liczba dziesiętna*) za pomocą następujących wyrażeń, odpowiednio: `Int64.Type`, `Currency.Type` i `type number`. Do konwersji typów danych służy funkcja `Table.TransformColumnTypes()` — w poniższym przykładzie dotyczy to kolumn `Discount Amount` (wielkość rabatu), `Sales Amount` (wielkość sprzedaży) i `Extended Amount` (wartość):

```
let
    Source = AdWorksSQLServer,
    Sales = Source{[Schema = "BI", Item = "vFact_InternetSales"]}[Data],
    TypeChanges = Table.TransformColumnTypes(Sales,
        {
            {"Discount Amount", Int64.Type}, // liczba całkowita
            {"Sales Amount", Currency.Type}, // stałoprzecinkowa liczba
                // dziesiętna
            {"Extended Amount", type number} // liczba dziesiętna
        })
in
    TypeChanges
```

Ponieważ język M rozróżnia wielkość liter, wyrażenia określające typ danych muszą być wprowadzane z zachowaniem dokładnej pisowni, np. `type number`, a nie `Type Number`. Zauważ, że w zapytaniach M mogą być zawarte komentarze jedno- i wielowierszowe.

Więcej szczegółów na ten temat podajemy w podrozdziale „Przykłady zapytań w języku Power Query M”.

Ze względu na wydajność i możliwość wystąpienia błędów zaokrąglenia ważne jest sprawdzenie, czy poszczególnym kolumnom numerycznym zostały przypisane właściwe typy danych. Dodatkowe szczegóły dotyczące typów danych znajdują się w rozdziale 4. „Projektowanie modeli danych w trybach importu, DirectQuery i złożonym”.

Teraz przyjrzymy się metodom dostępu do elementów (rekordów) przy użyciu języka M.

Dostęp do poszczególnych danych w języku M

Dostęp do rekordów w tabelach, elementów w listach i wartości w rekordach ma fundamentalne znaczenie w formułowaniu zapytań kierowanych do źródeł danych. W poniższym przykładzie wyniki widoku SQL BI.vDim_Account są przekazywane do Power BI przy użyciu nieco innej składni kwerendy M niż w przypadku zapytania o wymiar Customer z poprzedniego podrozdziału.

```
let
    Source = AdWorksSQLServer,
    AccountRecord = Source{[Name = "BI.vDim_Account"]},
    Account = AccountRecord[Data]
in
    Account
```

W tym zapytaniu rekord jest pobierany z zapytania przejściowego AdWorksSQLServer tylko na podstawie kolumny Name (nazwa). Pole Data (dane) tego rekordu jest następnie przenoszone do osobnej zmiennej (Account) i za jej pomocą przekazywane do Power BI jako rezultat wykonania widoku BI.vDim_Account. Projektanci zbiorów danych mogą wybrać metodę dostępu do elementów zawartych w zapytaniu przejściowym.

Poniższy przykładowy kod pobiera łańcuch "Cherry" z listy zdefiniowanej za pomocą instrukcji języka M:

```
let
    Source = {"Apple", "Banana", "Cherry", "Dates"},
    ItemFromList = Source{2}
in
    ItemFromList
```

W języku M numeracje rozpoczynają się od zera, więc wyrażenie Source{0} zwróciłoby wartość "Apple", a Source{4} zwróciłoby błąd, ponieważ na liście są tylko cztery pozycje. To samo dotyczy również wyodrębniania znaków z łańcucha tekstowego. Na przykład wyrażenie Text.Range("Brett",2,2) zwraca znaki et.

Listą w języku M jest uporządkowana sekwencja wartości. Istnieje wiele funkcji służących do analizowania i przekształcania list, np. List.Count() (liczebność listy) czy List.Distinct() (bez duplikatów). Funkcje agregujące wartości zawarte w listach, takie

jak `List.Average()` (wartość średnia), są często używane w zapytaniach grupujących, które wywołują funkcję `Table.Group()` (grupowanie wierszy tabeli). Pełną listę funkcji języka M czytelnik znajdzie w artykule „Dokumentacja funkcji języka Power Query M” opublikowanym pod adresem <https://learn.microsoft.com/pl-pl/powerquery-m/powerquery-m-function-reference>.

Następnie przyjrzymy się być może jednemu z najważniejszych aspektów języka M, jakim jest składnia zapytań.

Przykłady zapytań w języku Power Query M

Jak wynika z dotychczasowych przykładów, połączenie dojrzałej hurtowni danych i warstwy widoków SQL w ramach jednego źródła może wyeliminować potrzebę dalszego przekształcania danych. Projektanci zbiorów danych Power BI powinni mimo wszystko znać podstawy formułowania zapytań w języku M i najczęstsze przypadki ich stosowania, ponieważ często konieczne jest dalsze rozszerzanie i wzbogacanie danych źródłowych.

W następnych punktach zademonstrujemy trzy typowe scenariusze przekształcania danych, które można zrealizować za pomocą języka M. Zapytania M nie tylko pobierają poprawne wyniki, ale też generują instrukcje SQL przeznaczone do wykonania przez system źródłowy (mechanizm składania zapytań), a komentarze są dołączane dla celów długoterminowej konserwacji.

Jeśli jeszcze nie tworzyłeś zapytań w języku M, możesz zacząć od utworzenia zapytania pustego w kategorii *Other (Inne)* łączników ze źródłami danych dostępnych w oknie dialogowym *Get Data (Pobierz dane)*.

Można też powielić istniejące zapytanie za pomocą polecenia wybranego w menu kontekstowym po kliknięciu prawym przyciskiem jednego z istniejących zapytań w **Edytorze Power Query**. Potem można zmienić nazwę duplikatu i zmodyfikować jego treść.

Filtrowanie poprzednich trzech lat

Celem tego przykładu jest pobranie dat z trzech lat poprzedzających rok bieżący i wszystkich dat z roku bieżącego aż do dnia, w którym pytanie jest formułowane. Na przykład zapytanie sformułowane 30 grudnia 2021 roku powinno pobrać daty od 1 stycznia 2018 roku do 30 grudnia 2021 roku. Wymóg ten zapewni, że trzy pełne lata danych historycznych plus rok bieżący będą zawsze dostępne dla opracowywanych raportów.

Daty początkowa i końcowa dla warunku filtra są obliczane za pomocą funkcji języka M `Date` i `DateTime`, a następnie są przypisywane do zmiennych, odpowiednio: `StartDate` (data początkowa) i `CurrentDate` (data bieżąca). Ponieważ data początkowa wypada zawsze 1 stycznia, potrzebne jest tylko wyznaczenie roku początkowego i przekazanie tej wartości do konstruktora `#date`.

Na koniec dwie wspomniane zmienne są przekazywane do funkcji `Table.SelectRows()` (wybierz wiersze) w celu nałożenia filtra na widok tabeli faktów `Reseller Sales` (sprzedaj przez pośredników):

```
let
    // Daty z trzech poprzednich lat
    CurrentDate = DateTime.Date(DateTime.LocalNow()),
    StartYear = Date.Year(CurrentDate)-3,
    StartDate = #date(StartYear,1,1),
    // Widok tabeli faktów Reseller Sales
    Source = AdWorksSQLServer,
    ResellerSales = Source[{Schema = "BI", Item = "vFact_ResellerSales"}]
    ↪ [Data],
    // Filtrowanie danych z trzech poprzednich lat
    FilterResellerSales = Table.SelectRows(ResellerSales,
    each Date.From([OrderDate]) >= StartDate and Date.From([OrderDate])
    ↪ <= CurrentDate)
in
    FilterResellerSales
```

Jak widać w oknie dialogowym polecenia *View Native Query* (*Wyświetl zapytanie natywne*) dostępnym w menu kontekstowym panelu *Applied Steps* (*Zastosowane kroki*) **Edytora Power Query**, niestandardowy warunek filtra jest tłumaczony na instrukcję w języku T-SQL do wykonania w źródłowej bazie danych SQL Server:

```
from [BI].[vFact_ResellerSales] as [_]
where [_.OrderDate] >= convert(datetime2, '2018-01-01 00:00:00') and
[_.OrderDate] < convert(datetime2, '2021-12-31 00:00:00')
```

Warto zauważyć, że kolejność zmiennych w wyrażeniu nie ma wpływu na końcowe zapytanie. Na przykład dwie zmienne widoku `Reseller Sales` mogą być określone przed trzema zmiennymi daty, a końcowa zmienna `FilterResellerSales` nadal będzie generować to samo zapytanie SQL. Trzeba też pamiętać, że język M rozróżnia wielkość liter. Na przykład odwołanie się do zmiennej zdefiniowanej jako `StartDate` za pomocą nazwy `Startdate` powoduje błędne działanie kwerendy.

Komentarze jednowierszowe mogą być wprowadzane w zapytaniach M po dwóch ukośnikach (`//`), jak w przykładzie z trzema poprzednimi latami. Komentarze wielowierszowe zaczynają się od znaków (`/*`) i kończą znakami (`*/`), tak jak zapytania T-SQL dla bazy SQL Server.

Gdyby wymaganie dotyczyło jedynie pobierania danych z trzech lat wstecz względem bieżącej daty (na przykład od 30 grudnia 2018 do 30 grudnia 2021), zmienną `StartDate` można by obliczyć za pomocą funkcji `Date.AddYears()` w następujący sposób:

```
// Trzy poprzednie lata (np. od 18 października 2018 r. do 18 października 2021 r.)
CurrentDate = DateTime.Date(DateTime.LocalNow()),
StartDate = Date.AddYears(CurrentDate,-3)
```

Na koniec należy zauważyć, że standardowa baza danych `AdventureWorksDW` ma dane na temat wielkości sprzedaży w sieci pośredników tylko do roku 2013, więc użycie funkcji `Date.AddYears()` w celu odjęcia lat od obliczonej zmiennej `CurrentDate` jest w tym przypadku konieczne.

W następnym przykładzie użyjemy ponownie tego zapytania o trzy poprzednie lata, ale rozszerzamy je tak, aby w środowiskach testowych i produkcyjnych ładowane były wszystkie lata.

Łączenie zapytań

Wiele zapytań można łączyć (scalać) za pomocą funkcji `Table.Combine`. Funkcja ta może być niezwykle przydatna w przypadku zapytań typu `Folder`, gdy w ramach modelu danych trzeba wiele plików o tym samym formacie złączyć w jedną tabelę.

W kolejnym przykładzie utworzono parametr o nazwie `Mode` (tryb) z listą dostępnych wartości: `Dev`, `Test` i `Prod`. Przytoczone niżej zapytanie sprawdza wartość tego parametru. Jeśli parametr ma wartość `Dev`, to za pomocą zapytania z poprzedniego przykładu zwracane są tylko trzy poprzednie lata. W przeciwnym razie zapytanie z poprzedniego przykładu jest łączone z wyrażeniem tabelarycznym, które pobiera wszystkie dodatkowe lata. Dwa wyrażenia tabelaryczne są łączone ze sobą za pomocą funkcji `Table.Combine`:

```
let
// Daty wyznaczające trzy poprzednie lata
CurrentDate = DateTime.Date(DateTime.LocalNow()),
StartYear = Date.Year(CurrentDate)-3,
StartDate = #date(StartYear,1,1),
Results =
    if Mode = "Dev"
    then Trailing3Years
    else
        Table.Combine(
            {
                Trailing3Years,
                Table.SelectRows(
                    AdWorksSQLServer{[Schema = "BI", Item =
                    ↪"vFact_ResellerSales"]}[Data],
                    each Date.From([OrderDate]) < StartDate)
            }
        )
```

```

    )
in
    Results

```

W tym przykładzie zapytanie `Trailing3Years` (poprzednie 3 lata) jest zwykle ustawiane tak, aby nie łąadowało się do modelu danych, a całe powyższe zapytanie służy jako główna tabela faktów dla sprzedaży przez pośredników. Dzięki takiemu podejściu programiści mogą pracować z dużo mniejszym lokalnym zbiorem danych i łatwo dołączyć wszystkie wymagane dane podczas przenoszenia zbioru ze środowiska deweloperskiego do testowego i potem do produkcyjnego. Takie etapowe wdrażanie projektu omawiamy dokładniej w rozdziale 10. „Zarządzanie obszarami roboczymi i wartością” (w podrozdziale „Wdrożenia etapowe”).

Mamy tu również pokazane zastosowanie instrukcji `if`, której ogólny format przedstawia się następująco:

```
if <wyrażenie prawdziwe lub fałszywe> then <wyrażenie> else <wyrażenie>
```

W tym przykładzie użycie instrukcji `if` zapobiega włączeniu do zapytania wszystkich wierszy z tabeli `vFact_ResellerSales` (sprzedaże przez pośredników), gdy tryb jest ustawiony na `Dev` (tryb deweloperski), zapewniając mniejszy ogólny rozmiar zbioru danych i szybsze jego ładowanie w środowisku programistycznym.

Następny przykład również dotyczy tabel faktów i wiąże się z odświeżaniem przyrostowym.

Odświeżanie przyrostowe dla tabel faktów

Odświeżanie przyrostowe jest funkcją Power BI, która może znacznie zmniejszyć czas odświeżania bardzo dużych tabel faktów.

Odświeżanie przyrostowe pozwala na odświeżenie tylko części danych (nowych i zmienionych) w obrębie tabeli, w przeciwieństwie do przeładowywania wszystkich wierszy podczas każdego cyklu odświeżania, co jest domyślnym zachowaniem Power BI.

Korzystanie z odświeżania przyrostowego wymaga użycia dwóch parametrów o zarezerwowanych nazwach: `RangeStart` (początek zakresu) i `RangeEnd` (koniec zakresu). Parametry te muszą mieć przypisany typ daty lub czasu. W poniższym przykładzie zapytanie o sprzedaż internetową zostało zmodyfikowane przez dodanie filtrowania specyficznego dla implementacji odświeżania przyrostowego.

```

let
    Source = AdWorksSQLServer,
    InternetSales = Source[{Schema="BI",Item="vFact_InternetSales"}] [Data],
    FilterRows = Table.SelectRows(InternetSales, each [OrderDateKey] >
        ↪ ConvertDateKey(RangeStart) and [OrderDateKey] <= ConvertDateKey(RangeEnd))
in
    FilterRows

```

Kod w kroku `FilterRows` (wiersze filtrowane) wykorzystuje funkcję `Table.SelectRows()` w połączeniu z parametrami `RangeStart` i `RangeEnd` oraz niestandardową funkcją `ConvertDateKey` (konwersja klucza daty). Funkcja `ConvertDateKey` jest konieczna, ponieważ w zapytaniu odwołano się do kolumny klucza zastępczego `OrderDateKey` (klucz daty zamówienia). Kolumna ta jest kluczem zastępczym, ponieważ przechowuje datę jako wartość całkowitą w formacie `YYYYMMDD`, a nie jako wartość typu `date` lub `datetime`.

Kod niestandardowej funkcji `ConvertDateKey` wygląda następująco:

```
let
    ConvertDateKey = (DateTime as datetime) => Date.Year(DateTime) * 10000
    ↳+ Date.Month(DateTime) * 100 + Date.Day(DateTime)
in
    ConvertDateKey
```

Podczas filtrowania tabeli z użyciem parametrów `RangeStart` i `RangeEnd` ważne jest, aby tylko jeden z warunków porównywania z tymi parametrami zawierał opcję równości (=). W przeciwnym razie może dojść do powielania danych, ponieważ niektóre wiersze danych mogą spełniać warunek końca jednego cyklu odświeżania i warunek początku następnego cyklu odświeżania.

Początkowy cykl odświeżania dla tego zbioru danych ładuje wszystkie wiersze w ramach zbioru, a parametr `RangeStart` jest ustawiany automatycznie przez usługę. Kolejny cykl odświeżania ustawia parametr `RangeEnd` na bieżącą datę i godzinę, dzięki czemu do tabeli danych dodawane są tylko nowe i zaktualizowane dane.

Teraz skierujemy uwagę na przykłady zapytań M dla wymiarów i zaczniemy od zapytania na temat klienta.

Kolumna historii klienta

W tym przykładzie celem jest dodanie kolumny do tabeli wymiaru `Customer` w celu pogrupowania klientów w czterech kategoriach na podstawie daty ich pierwszego zakupu. W szczególności nowa kolumna musi wykorzystać istniejącą kolumnę dat pierwszego zakupu i przypisać wiersze klientów do jednej z następujących czterech kategorii: `First Year Customer` (klient roczny), `Second Year Customer` (klient dwuletni), `Third Year Customer` (klient trzyletni) i `Legacy Customer` (klient starszy).

Kolumna jest obliczana codziennie przy każdym zaplanowanym odświeżeniu danych i korzystają z niej zespoły sprzedaży i marketingu, aby koncentrować swoje wysiłki na segmentach nowych i starszych klientów.

W celu utworzenia nowej kolumny zastosowano funkcję `Table.AddColumn()` z generatorem kolumn łączącym funkcje operujące datami z logiką warunkową (`if..then..else`):

```

let
// Przedziały dat w historii klientów
    CurrentDate = DateTime.Date(DateTime.LocalNow()),
    OneYearAgo = Date.AddYears(CurrentDate,-1),
    TwoYearsAgo = Date.AddYears(CurrentDate,-2),
    ThreeYearsAgo = Date.AddYears(CurrentDate,-3),
// Wymiar Customer (klient)
    Source = AdWorksSQLServer,
    Customer = Source{[Schema = "BI", Item = "vDim_Customer"]} [Data],
    CustomerHistoryColumn = Table.AddColumn(Customer, "Customer History
Segment",
    each
    if [DateFirstPurchase] >= OneYearAgo then "First Year Customer"
    else if [DateFirstPurchase] >= TwoYearsAgo and [Customer First Purchase
↪Date] < OneYearAgo then "Second Year Customer"
    else if [DateFirstPurchase] >= ThreeYearsAgo and [Customer First
↪Purchase Date] < TwoYearsAgo then "Third Year Customer"
    else "Legacy Customer", type text)
in
    CustomerHistoryColumn

```

Jak widać na rysunku 3.12, kolumna Customer History Segment (segment historii klientów) generuje jedną z czterech wartości tekstowych na podstawie kolumny Date ↪FirstPurchase (data pierwszego zakupu).

	123 CustomerKey	DateFirstPurchase	A ^B C Customer History Segment
382	11381	6/12/2011	Legacy Customer
383	11382	11/3/2012	Third Year Customer
384	11383	1/1/2014	First Year Customer
385	11384	8/2/2013	Second Year Customer

Rysunek 3.12. Kolumna segmentu historii klientów w Edytorze Power Query

Podobnie jak we wcześniejszym przykładzie zapytania M z filtrowaniem trzech poprzednich lat, logika warunkowa dla nowej kolumny klienta jest tłumaczona na język T-SQL przez mechanizm składania zapytań:

```

[].[DateFirstPurchase] as [DateFirstPurchase],
[].[CommuteDistance] as [CommuteDistance],
case
    when [].[DateFirstPurchase] >= convert(date, '2013-12-30')
    then 'First Year Customer'
    when [].[DateFirstPurchase] >= convert(date, '2012-12-30')
    ↪and [].[DateFirstPurchase] < convert(date, '2013-12-30')
    then 'Second Year Customer'
    when [].[DateFirstPurchase] >= convert(date, '2011-12-30')
    ↪and [].[DateFirstPurchase] < convert(date, '2012-12-30')
    then 'Third Year Customer'

```

```

        else 'Legacy Customer'
    end as [Customer History Segment]
from [BI].[vDim_Customer] as [_]

```

Dwie dynamiczne kolumny, `Calendar Year Status` i `Calendar Month Status`, zawarte w widoku SQL wymiaru daty, o których była mowa wcześniej w tym rozdziale, mogły być również obliczone za pomocą funkcji `M`.

Przedstawimy teraz dodatkowe szczegóły dotyczące ostatniego parametru funkcji `Table.AddColumn()`, czyli `type text` (typ tekstowy).

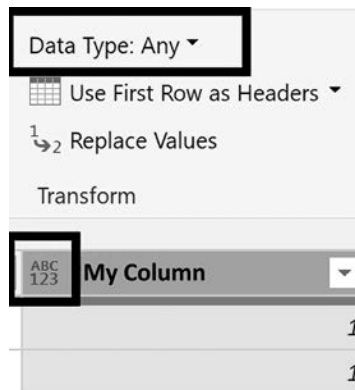
Typy danych kolumny pochodnej

Ostatni parametr funkcji `Table.AddColumn()` jest opcjonalny, ale powinien zostać podany w celu określenia typu danych nowej kolumny.

W przykładzie z segmentem historii klientów nowa kolumna ma przypisany tekstowy typ danych. Gdyby tworzono kolumnę z liczbami całkowitymi, przypisano by jej typ `Int64.Type`, jak w poniższym przykładzie:

```
MyNewColumn = Table.AddColumn(Product, "My Column", each 5, Int64.Type)
```

Jeśli nie określimy typu danych tworzonej kolumny w funkcji `Table.AddColumn()` lub później w zapytaniu przy użyciu funkcji `Table.TransformColumnTypes()`, to nowa kolumna przyjmie typ danych `Any` (dowolny), co pokazano na rysunku 3.13.



Rysunek 3.13. Typ danych Any

Kolumny o typie danych `Any` są ładowane do modelu jako dane tekstowe. Projektanci zbiorów danych powinni zadbać, aby każda kolumna w każdym zapytaniu miała określony typ danych. Innymi słowy, typ danych `Any` (czyli tak naprawdę nieznanym) nie powinien być dopuszczany w zapytaniach `M`.

Przejdziemy teraz do przedstawienia przykładowego zapytania `M` dla wymiaru `Product`.

Integracja wymiaru produktu

Widok SQL dla wymiaru produktu przywołany w rozdziale 2. „Przygotowanie źródeł danych” zawierał następujące cztery operacje:

1. Połączenie tabel wymiarów Product, ProductSubcategory i ProductCategory w jedno zapytanie.
2. Utworzenie niestandardowej kolumny grup kategorii produktów (na przykład rowery i nierowery).
3. Zastosowanie wygodnych w raportowaniu nazw kolumn ze spacjami i właściwą wielkością liter.
4. Zastąpienie wszelkich pustych wartości w kolumnach Product Subcategory i Product Category wartością Undefined (nieokreślona).

Prawie wszystkie operacje dostępne dla zapytań SQL typu SELECT mają swoje odpowiedniki w funkcjach języka M. Jeśli w ramach źródła danych nie można utworzyć widoku SQL dla wymiaru produktu, to taki sam rezultat można uzyskać za pomocą następującej kwerendy M:

```
let
    Source = AdWorksSQLServer,
    // Widoki tabeli wymiaru Product
    Product = Source{[Schema = "BI", Item = "vDim_Product"]}[Data],
    ProductSubCat = Source{[Schema = "BI", Item =
        ↪"vDim_ProductSubcategory"]}[Data],
    ProductCat = Source{[Schema = "BI", Item = "vDim_ProductCategory"]}[Data],
    // Złączenia zewnętrzne tabeli Product
    ProductJoinSubCat = Table.NestedJoin(Product,"ProductSubcategoryKey",
        ↪ProductSubCat,"ProductSubcategoryKey","ProductSubCatTableCol",
        ↪JoinKind.LeftOuter),
    ProductJoinSubCatCol = Table.ExpandTableColumn(ProductJoinSubCat,
        ↪"ProductSubCatTableCol",{ "EnglishProductSubcategoryName",
        ↪"ProductCategoryKey"},{"Product Subcategory", "ProductCategoryKey"}),

    ProductJoinCat = Table.NestedJoin(ProductJoinSubCatCol,
        ↪"ProductCategoryKey",ProductCat,"ProductCategoryKey",
        ↪"ProductCatTableCol",JoinKind.LeftOuter),
    ProductJoinCatCol = Table.ExpandTableColumn(ProductJoinCat,
        ↪"ProductCatTableCol", {"EnglishProductCategoryName"},{"Product Category"}),
    // Zmianianie nazw wybranych kolumn
    ProductDimCols = Table.SelectColumns(ProductJoinCatCol,{"ProductKey",
        ↪"ProductAlternateKey","EnglishProductName","Product Subcategory",
        ↪"Product Category"}),
    ProductDimRenameCols = Table.RenameColumns(ProductDimCols,
        ↪{"ProductKey", "Product Key"},{"ProductAlternateKey","Product
        ↪Alternate Key"},{"EnglishProductName","Product Name"
    }),
```



```
// Kolumna grupująca kategorie produktu
ProductCatGroupCol = Table.AddColumn(ProductDimRenameCols, "Product
↳Category Group",
    each
        if [Product Category] = "Bikes" then "Bikes"
        else if [Product Category] = null then "Undefined"
        else "Non-Bikes",
        type text),
// Usuwanie wszystkich wartości Null
UndefinedCatAndSubcat = Table.ReplaceValue(ProductCatGroupCol, null,
↳"Undefined", Replacer.ReplaceValue, {"Product Subcategory", "Product
↳Category"})
in
    UndefinedCatAndSubcat
```

Trzy tabele wymiaru produktu w schemacie db0 hurtowni danych są przywoływane z zapytania przejściowego AdWorksSQLServer opisanego wcześniej w tym rozdziale.

Funkcja `Table.NestedJoin()` (złączenie zagnieżdżone) działa analogicznie jak klauzula `LEFT JOIN` (złączenie lewostronne) w SQL, a funkcja `Table.ExpandTableColumn()` (rozwiniecie kolumny tabel) ekstrahuje potrzebne kolumny `Product Subcategory` i `Product Category` oraz zmienia ich nazwy.

Po wybraniu i zmianie nazwy kolumn wyrażenie warunkowe w funkcji `Table.AddColumn()` tworzy kolumnę grupową o nazwie `Product Category`. Na koniec funkcja `Table.ReplaceValue()` zastępuje w kolumnach `Product Category` i `Product Subcategory` wszelkie wartości null ciągiem tekstowym "Undefined". Edytor Power Query umożliwia podgląd wyników, co pokazano na rysunku 3.14.

Product Key	Product Alternate Key	Product Name	Product Subcategory	Product Category	Product Category Group
308	FR-M945-38	HL Mountain Frame - Silver, 38	Mountain Frames	Components	Non-Bikes
309	FR-M945-38	HL Mountain Frame - Silver, 38	Mountain Frames	Components	Non-Bikes
310	BK-R93R-62	Road-150 Red, 62	Road Bikes	Bikes	Bikes
311	BK-R93R-44	Road-150 Red, 44	Road Bikes	Bikes	Bikes

Rysunek 3.14. Podgląd wyniku zapytania Product wyświetlany w Edytorze Power Query

Pomimo dodatkowych kroków i złożoności tego zapytania w stosunku do poprzednich przykładów zapytań w języku M (filtr trzech poprzednich lat, kolumna `Customer History Segment`) całe zapytanie jest tłumaczone na pojedynczą instrukcję SQL i wykonane przez źródłową bazę danych w systemie SQL Server. Polecenie *View Native Query* (Wyświetl zapytanie natywne) dostępne w menu kontekstowym panelu *Applied Steps* (Zastosowane kroki) **Edytora Power Query** ujawnia specyficzną składnię zapytania SQL wygenerowanego przez złożenie zapytania:

```

select [_].[ProductKey] as [Product Key],
       [_].[ProductAlternateKey] as [Product Alternate Key],
       [_].[EnglishProductName] as [Product Name],
       case
         when [_].[EnglishProductSubcategoryName] is null
           then 'Undefined'
         else [_].[EnglishProductSubcategoryName]
       end as [Product Subcategory],
       case
         when [_].[EnglishProductCategoryName] is null
           then 'Undefined'
         else [_].[EnglishProductCategoryName]
       end as [Product Category],
       case
         when [_].[EnglishProductCategoryName] = 'Bikes' and [_].
[EnglishProductCategoryName] is not null
           then 'Bikes'
         when [_].[EnglishProductCategoryName] is null
           then 'Undefined'
         else 'Non-Bikes'
       end as [Product Category Group]
from

```

Zauważ, że dedykowany obiekt widoku SQL w schemacie BI (na przykład `BI.vDim_ ↪ProductSubcategory`) jest dostępny dla każdej z trzech tabel wymiaru produktu. Zgodnie z tym, co napisaliśmy w rozdziale 2. „Przygotowanie źródeł danych” (w podrzdziale „Widoki SQL”), powinno się zawsze uzyskiwać dostęp do widoków SQL, gdyż jest to równoznaczne z dostępem do tabel źródłowych.

Zauważ, że ze względu na wymóg zewnętrznego złączenia lewostronnego i obecność wspólnych nazw kolumn nie można tu użyć funkcji `Table.Join()`. Próba takiego złączenia tabel o wspólnych nazwach kolumn, takich jak `ProductSubcategoryKey` czy `Product ↪CategoryKey`, spowodowałaby błąd.

Mimo że złączenie zewnętrzne lewostronne jest domyślnym ustawieniem dla funkcji `Table.NestedJoin()`, zaleca się jawne określenie rodzaju złączenia — na przykład `JoinKind.Inner` (wewnętrzne), `JoinKind.LeftOuter` (zewnętrzne lewostronne) lub `JoinKind.LeftAnti` (antyłączenie lewostronne) — zgodnie z wartościami zmiennych `ProductJoinSubCat` i `ProductJoinCat`. W celu odświeżenia sobie wiadomości na temat typów złączeń dostępnych w Power Query proszę zapoznać się z artykułem zamieszczonym pod adresem <https://learn.microsoft.com/pl-pl/power-query/merge-queries-overview>.

Jeśli tylko użycie funkcji `Table.Join` jest możliwe, to na ogół należy dać jej pierwszeństwo przed `Table.NestedJoin`, a to dlatego, że ta druga wykonuje operację łączenia tabel w oparciu o zasoby lokalne, podczas gdy pierwszą można przerzucić do systemu źródłowego, co zazwyczaj zwiększa wydajność kwerendy. Warto jednak zauważyć, że

operacja **scalania kwerend** dostępna w **graficznym interfejsie (GUI) Edytora Power Query** domyślnie stosuje funkcję `Table.NestedJoin`.

Gdy źródłem danych dla zbioru Power BI ma być jakiekolwiek źródło nieustrukturyzowane lub należące do użytkownika biznesowego, zazwyczaj zaleca się zaimplementowanie w ramach zapytania M dodatkowej logiki jakości danych i obsługi błędów.

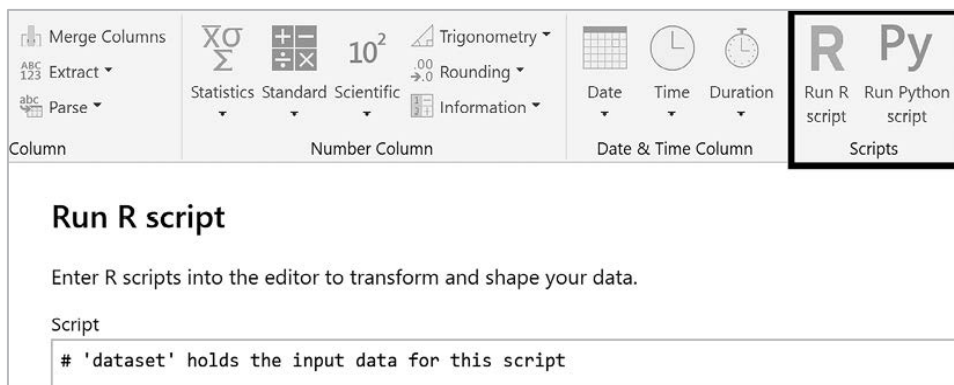
Na przykład do zapytania `Sales and Margin Plan`, które pobiera dane ze skoroszytu Excela, można dodać krok wywołujący funkcję `Table.Distinct()`, aby usunąć wszelkie zduplikowane wiersze. Podobnie trzeci parametr funkcji `Table.SelectColumns()` (wybierz kolumny), np. `MissingField.UseNull` (brakujące pole zastąp wartością `Null`), można wykorzystać do obsługi przypadków braku kolumny źródłowej lub zmiany jej nazwy.

Wprowadzie język M sam zapewnia szerokie możliwości transformacji danych, ale niektórzy analitycy mogą lepiej znać innymi języki, takie jak R czy Python. Właśnie taką sytuację omówimy w kolejnym przykładzie.

Przekształcanie danych za pomocą skryptów R i Pythona

W ramach zapytania M jako część procesu transformacji danych można wykorzystywać skrypty R lub Pythona.

Jak pokazano na rysunku 3.15, na karcie *Transform (Przekształć)* Edytora Power Query w Power BI Desktop są dostępne polecenia *Run R script (Uruchom skrypt języka R)* i *Run Python script (Uruchom skrypt języka Python)*.



Rysunek 3.15. Polecenia Run R script (Uruchom skrypt języka R) i Run Python script (Uruchom skrypt języka Python) w Edytorze Power Query

Żeby aplikacja Power BI Desktop mogła wykonywać skrypty napisane w R i Pythonie, języki te muszą być zainstalowane na lokalnym komputerze wraz ze wszystkimi potrzebnymi pakietami. Jeśli używana jest **brama danych**, to R, Python i odpowiednie

pakiety muszą być zainstalowane również na serwerze uruchamiającym tę bramę. I rzecz najważniejsza, aby skrypty działały poprawnie, gdy zbiór danych jest publikowany w usłudze Power BI, poziomy prywatności dla źródeł danych muszą być ustawione na *Public (Publiczne)*.

Dla większości organizacji to ograniczenie wyklucza stosowanie skryptów R i Pythona w Power BI. Poza tym obecność takich skryptów tworzy kolejną warstwę złożoności dla rozwiązania zawierającego już kwerendy w językach SQL, M i DAX.

Przyjrzyjmy się jeszcze innemu sposobowi tworzenia zapytań w języku M, a mianowicie z użyciem **przepływów danych**.

Przepływy danych

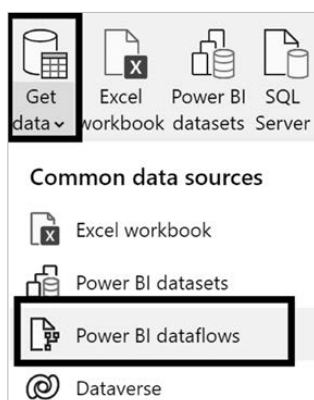
Mówiąc najprościej, **przepływy danych** to zapytania Power Query M utworzone w usłudze Power BI. W przypadku obszarów roboczych innych niż *My Workspace (Mój obszar roboczy)* usługa Power BI udostępnia interfejs niemal identyczny jak **Edytor Power Query** w Power BI Desktop do tworzenia i edycji zapytań Power Query.

Przepływy danych uzyskują dostęp do systemów źródłowych w taki sam sposób jak zapytania Power Query utworzone w **Edytorze Power Query** dostępnym w Power BI Desktop. Jednakże dane pobierane z tych zapytań dla przepływów danych w trybie importu są przechowywane w folderach zgodnych z Dataverse w ramach usługi Azure Data Lake Storage Gen2. Przepływy danych mogą być również tworzone z dostępem bezpośrednim (w trybie DirectQuery) do systemów źródłowych.

Istnieje kilka kluczowych zalet stosowania przepływów danych w analityce firmowej. Główną z nich jest możliwość wielokrotnego wykorzystania. Raz utworzony przepływ danych może być użyty jako źródło danych w wielu różnych plikach aplikacji Power BI Desktop podczas projektowania zbioru danych. Przepływy danych w Power BI są opcją przy korzystaniu z polecenia *Get data (Pobierz dane)* zarówno w Power BI Desktop, jak i w **Edytorze Power Query**, co pokazano na rysunku 3.16.

Oznacza to, że można raz opracować logikę transformacji danych w ramach zapytania i potem wykorzystywać ją w wielu różnych zestawach danych, dzięki czemu tworzenie tych ostatnich staje się wydajniejsze, bardziej ustandaryzowane i mniej podatne na błędy.

Drugą kluczową zaletą przepływów danych jako źródeł dla zestawów danych w trybie importu jest to, że przepływ danych izoluje systemy źródłowe od zbiorów danych Power BI. Oznacza to, że podczas odświeżania zbiorów danych Power BI pobierane są dane z instancji Azure Data Lake Storage Gen2 zamiast z samych systemów źródłowych, dzięki czemu proces odświeżania w żaden sposób nie wpływa na te systemy.



Rysunek 3.16. Przepływy danych w Power BI jako źródło danych

Przepływ danych może być odświeżony raz, a potem odświeżenia we wszystkich zbiorach danych Power BI wykorzystujących ten przepływ danych wpływają tylko na wysoce skalowalną instancję Azure Data Lake Storage Gen2, a nie na systemy źródłowe. Nabiera to jeszcze większego znaczenia, gdy źródłowy system danych, do którego ma dostęp przepływ danych, jest lokalny i wymaga zastosowania bramy danych.

Ponieważ dane pobierane przez zbiory danych Power BI Desktop znajdują się w instancji Azure Data Lake Storage Gen2, do ich odświeżania nie jest wymagana brama danych. Brama jest konieczna tylko podczas odświeżania samego przepływu danych.

Izolacja systemów źródłowych od zbiorów danych Power BI jest korzystna również z punktu widzenia bezpieczeństwa. Zamiast przekazywać twórcom zbiorów danych poświadczenia dostępu do systemów źródłowych, firmowy zespół BI może po prostu przekazać im poświadczenia dostępu do samego przepływu danych.

Idealnym stanem dla BI w przedsiębiorstwie byłaby jedna hurtownia danych i odpowiadający jej jeden zbiór danych Power BI do celów raportowania, jednak często coś takiego jest niemożliwe. Na przykład lista produktów dostosowanych do potrzeb klienta może być przechowywana przez zespół marketingowy w witrynie SharePoint, a jej włączenie do korporacyjnej hurtowni danych nie jest na razie planowane. Jednak ta lista może być przydatna w wielu zbiorach danych Power BI używanych przez dział sprzedaży, marketingu i dostaw.

W takiej sytuacji można by stworzyć jeden przepływ danych, który łączyłby się z tym plikiem źródłowym i stosował niezbędne transformacje. Zbiory danych dotyczące sprzedaży, marketingu i łańcucha dostaw mogłyby być połączone z tym jednym przepływem, tak że wszelkie aktualizacje tego centralnego zasobu w naturalny sposób przepływałyby do wszystkich zależnych zbiorów danych.

Jak wspominaliśmy, przepływy danych zapewniają kilka istotnych korzyści, a w przypadku zastosowania ich w Power BI Premium użytkownik uzyskuje dostęp do dodatkowych funkcji związanych z ich obsługą.

Funkcje przepływu danych w Power BI Premium

Power BI Premium obsługuje dodatkowe funkcje przepływu danych, w tym *Enhanced compute engine* (ulepszony aparat obliczeniowy), *DirectQuery* (zapytanie bezpośrednie), *Computed entities* (jednostki obliczone), *Linked entities* (jednostki połączone), *Incremental refresh* (odświeżanie przyrostowe) oraz *AutoML* (zautomatyzowane uczenie maszynowe).

Usprawniony silnik obliczeniowy może wyraźnie zwiększyć prędkość odświeżania dla złożonych transformacji, takich jak złączenia, grupowanie, filtrowanie i operacje rozróżniania. Jak wcześniej wspomniano, tworzenie przepływów danych dla źródeł *DirectQuery* jest obsługiwane, ale tylko w wersji Premium. Należy zauważyć, że modele złożone, które mają źródła w trybach zarówno importu, jak i *DirectQuery*, obecnie dają możliwości włączenia przepływów danych jako źródeł.

Encje obliczane pozwalają na wykonywanie w magazynie obliczeń łączących dane z wielu przepływów w nową, zespoloną encję lub wzbogacających istniejącą. Na przykład przepływ danych o nazwie Produkt może zostać wzbogacony o informacje z przepływów Kategoria produktu i Podkategoria produktu.

Podobnie jak w przypadku encji obliczanych, encje łączone pozwalają na odwołanie się do innych przepływów danych w celu wykonania obliczeń (encje obliczane) lub ustanowienia tabeli służącej jako jedyne źródło prawdy dla innych przepływów danych.

I na koniec funkcja *AutoML* automatyzuje tworzenie modeli **uczenia maszynowego (ML)**, umożliwiając automatyczne rozpoznawanie wzorców, analizę nastrojów itp.

Na zakończenie rozdziału omówimy różne narzędzia edycyjne używane do tworzenia zapytań w języku M.

Narzędzia edycyjne Power Query M

Podobnie jak w przypadku innych języków i typów projektów, dostępne są różne narzędzia do edycji kodu, które wspierają tworzenie i dokumentowanie wersji zapytań w języku M oraz zarządzanie nimi.

Projektanci zbiorów danych mogą tworzyć zapytania M dla Power BI i innych projektów Microsoft za pomocą zaawansowanego edytora w Power BI Desktop i interfejsu edycji

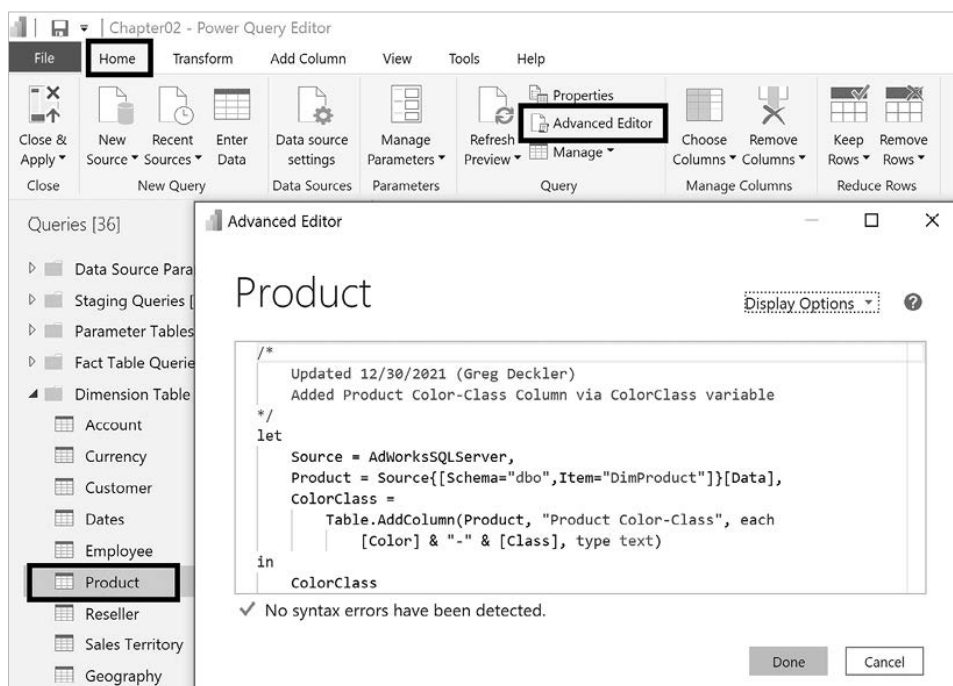
przepływow danych w usłudze Power BI, ale także przy użyciu aplikacji Visual Studio lub Visual Studio Code. Wszystkie te narzędzia zawierają podobne funkcje programistyczne, takie jak IntelliSense, podświetlanie składni czy zintegrowana kontrola źródeł.

W kolejnych punktach omówimy te narzędzia nieco dokładniej.

Edytor zaawansowany

W Power BI Desktop i dla przepływów danych w usłudze Power BI kod M każdego zapytania jest dostępny w oknie *Advanced Editor* (*Edytor zaawansowany*) będącym częścią aplikacji *Edytor Power Query*.

Aby otworzyć to okno, w Edytorze Power Query wybierz z listy zapytań po lewej stronie interesującą Cię pozycję i na karcie *Home* (*Narzędzia główne*) kliknij ikonę *Advanced Editor* (*Edytor zaawansowany*). Wygląd okna edytora zaawansowanego jest pokazany na rysunku 3.17.



Rysunek 3.17. Edytor zaawansowany dostępny w Power BI Desktop

Doświadczeni autorzy zapytań M często używają ikon transformacji danych dostępnych w Edytorze Power Query, aby szybko stworzyć wstępną wersję jednego kodu zapytania. Potem przechodzą do okna *Advanced Editor* (*Edytor zaawansowany*) lub do

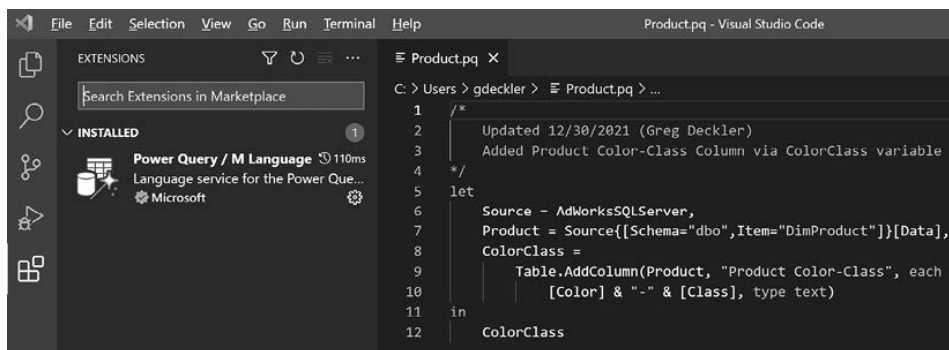
jakiegoś zewnętrznego narzędzia i tam analizują, a w razie potrzeby również modyfikują ten kod, np. przez zmianę nazw zmiennych lub wykorzystanie opcjonalnych parametrów niektórych funkcji języka M.

W przypadku popularnych i prostych zadań transformacji danych, takich jak filtrowanie wierszy na podstawie jednej wartości kolumny (np. `State = "Kansas"`), kod M wygenerowany przez **Edytor Power Query** zwykle wymaga tylko niewielkiej korekty. Dla bardziej złożonych zapytań z niestandardowymi lub mniej powszechnymi warunkami interfejs graficzny **Edytora Power Query** okazuje się niewystarczający i trzeba sięgać po narzędzia umożliwiające kodowanie w sposób bezpośredni.

Edytor zaawansowany jest wygodny, ale brakuje mu odpowiedniej integracji z kontrolą źródła. Mankament ten można ominąć przez zastosowanie alternatywnych narzędzi, takich jak Visual Studio Code.

Visual Studio Code

Visual Studio Code to darmowe, lekkie narzędzie edytorskie firmy Microsoft, które jest dostępne na wszystkich platformach (Windows, Mac i Linux). Po zainstalowaniu rozszerzenia o nazwie *Power Query / M Language* zapewnia obsługę edycji kodu dla języka Power Query M, co pokazano na rysunku 3.18.



Rysunek 3.18. Zapytanie M w Visual Studio Code

W prezentowanym przykładzie to samo zapytanie o nazwie `Product` przeglądane w Edytorze zaawansowanym będącym częścią programu Power BI Desktop zostało skopiowane do pliku z rozszerzeniem `.pq` obsługiwany przez Visual Studio Code. Zmiana rozszerzenia pozwoliła na uaktywnienie funkcji edycyjnych, takich jak kolorowanie słów kluczowych, automatyczne uzupełnianie kodu i wyświetlanie informacji o wpisywanym kodzie. Pliki zapytań w języku M mogą być otwierane i zapisywane z następującymi czterema rozszerzeniami: `.m`, `.M`, `.pq` i `.PQ`.

Należy zauważyć, że kod M opracowany w Visual Studio Code nie może być wykorzystany bezpośrednio przez Power BI. Należy go skopiować z Visual Studio Code lub repozytorium kontroli źródeł i wkleić do Edytora zaawansowanego. Doświadczone zespoły BI w firmach przywiązują jednak dużą wagę do właściwej kontroli wersji, zwłaszcza jeśli chodzi o zmiany w bazowych tabelach danych i transformacjach danych.

Ponieważ rozszerzenie *.pq* jest używane przez Power Query SDK dla Visual Studio, zaleca się stosowanie go do przechowywania zapytań M w Visual Studio Code, a także w omawianym poniżej Visual Studio.

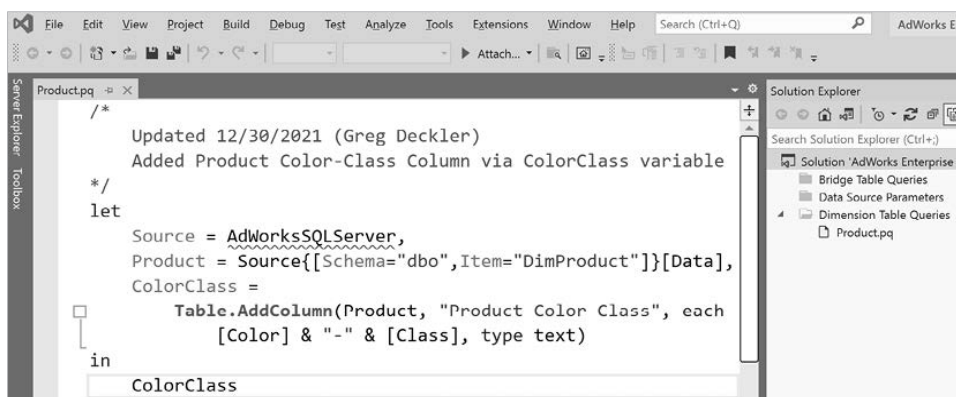
Visual Studio

Visual Studio jest rozbudowaną wersją lżejszego Visual Studio Code i jest **zintegrowanym środowiskiem programistycznym (IDE)** szeroko stosowanym w całej branży informatycznej. W Visual Studio 2015 i nowszych wersjach Power Query SDK może służyć do tworzenia łączników danych i zapytań M, co pokazano na rysunku 3.19.



Rysunek 3.19. Typy projektów Power Query w Visual Studio

Dzięki nowo opracowanemu formatowi plików PQ można do projektu realizowanego w Visual Studio dodać zapytania M jako oddzielne pliki (*.pq*), tak jak na rysunku 3.20.



Rysunek 3.20. Projekt Power Query w Visual Studio 2019

W przeciwieństwie do Visual Studio Code rozszerzenie pliku dla projektów Power Query jest tylko jedno (.pq). Co najważniejsze, w pełni obsługiwane są funkcje IntelliSense dla języka M, co znacznie ułatwia znalezienie funkcji M istotnych dla konkretnych operacji przekształcania danych. IntelliSense jest ogólnym terminem dla funkcji wspomagających pisanie kodu, takich jak automatyczne uzupełnianie, podpowiadanie treści (informacje o parametrach) i sygnalizowanie błędów składniowych.

Ponadto w przeciwieństwie do Visual Studio Code zapytania M mogą być wykonywane z poziomu Visual Studio za pośrednictwem Power Query SDK. Aby wykonać zapytanie M w Visual Studio, takie jak w poprzednim przykładzie, należy kliknąć przycisk *Start* na pasku narzędzi (zielona ikona *play*) lub nacisnąć klawisz *F5*.

Można również kliknąć prawym przyciskiem myszy projekt Power Query (np. AdWorks Enterprise Import), aby skonfigurować właściwości opracowywanego projektu, takie jak maksymalna liczba wierszy wyjściowych i możliwość wykonywania zapytań natywnych.

Aby zainstalować Power Query SDK dla Visual Studio, należy otworzyć witrynę *Visual Studio Marketplace (Extensions/Manage Extensions)* i wyszukać rozszerzenie o nazwie *Power Query SDK*.

Power Query SDK dla Visual Studio umożliwia typową integrację z narzędziami do kontroli źródeł i zarządzania projektami, takimi jak usługi Azure DevOps (dawniej Visual Studio Team Services).

To kończy naszą eksplorację metod łączenia się ze źródłami i przekształcania danych za pomocą języka M.

Podsumowanie

W tym rozdziale omówiliśmy wiele składników procesu pobierania danych ze źródła i dostarczania ich do zbioru danych w Power BI zgodnie z tym, co napisaliśmy w rozdziale 1. „Planowanie projektów Power BI”. Opisaliśmy konstruowanie warstwy dostępu do danych i procesu pobierania danych z użyciem zapytań M służących do definiowania i ładowania tabel wymiarów oraz faktów.

W następnym rozdziale wykorzystamy opisane tu kwerendy i techniki ich projektowania do tworzenia modeli danych w trybach importu i DirectQuery.

Skorowidz |

A

- administrator Power BI, 39, 595
- administratorzy globalni, 40
- agregacje automatyczne, 195
- AI, Artificial intelligence, 353
- aktualizowanie aplikacji, 564
- alerty danych, 583
- ALM, Application Life-cycle Management, 455
 - Toolkit, 667
- analitika biznesowa, BI, 597
 - samoobsługowa, SSBI, 597
- analiza, 379
 - czasowa, 224
 - pamięci, 193
 - projektu, 68
- analizator wydajności, 248
- Analysis Services, 509
- animacje, 375
 - odtworzenie osi, 376
- antywzorce wizualizacji, 269
- aplikacje
 - klienckie PBRs, 544
 - Microsoft 365, 576
 - niestandardowe, 579
 - Power BI, 550, 571
 - aktualizowanie, 564
 - licencjonowanie, 550
 - publikowanie, 557
 - tworzenie, 557
 - uprawnienia, 554
 - w systemach mobilnych, 565
 - wdrażanie, 552
 - zabezpieczenia, 554
- ArcGIS Maps for Power BI, 368

architektury

- bram danych, 486
- pulpitów nawigacyjnych, 403
- zbiorów danych, 50

automatyczne uczenie maszynowe, 354

Azure

- Active Directory, AAD, 40, 475, 595, 601
 - grupa zabezpieczeń, 242
- Analysis, 145
- Data Factory, ADF, 96, 107
- Machine Learning, 415

B

- baza danych SQL Server, 57
- bezpieczeństwo, 240, 244, 489
- bezpieczne osadzanie raportu, 575
- brama danych
 - administratorzy, 499
 - architektury, 486
 - diagnostyka, 503
 - dodawanie źródła danych, 499
 - instalacja, 491
 - klastry bram, 484
 - klucze odzyskiwania, 482, 496
 - konfigurowanie, 491
 - konto, 493
 - łączniki danych, 495
 - monitorowanie, 502
 - odświeżanie danych, 506
 - planowanie, 473, 477
 - przejmowanie, 502
 - przenoszenie, 502
 - przywracanie, 502
 - raporty, 505
 - role i uprawnienia, 481

brama danych
 tryb osobisty, 483
 tryb standardowy, 483
 zabezpieczenia, 489
 zadania, 477
 zarządzanie klastrami, 497
 buforowanie zapytań, 190

C

cele zbioru danych, 147
 Columnstore, 194
 CRM, Customer Relationship Management, 58
 cykl życia
 aplikacji, ALM, 455
 treści, 667

D

data storytelling, 375
 DAX, Data Analysis eXpressions, 37, 144, 200
 dynamiczne zabezpieczenia, 244
 funkcja CALCULATE(), 211
 funkcje, 209
 miary, 201
 role zabezpieczeń, 240
 zmienne, 214
 DAX Studio, 249
 DirectQuery, 114
 DLP, Data Loss Prevention, 453
 DMV, Dynamic Management Views, 193, 461
 dostęp
 do danych, 124
 warunkowy w usłudze AAD, 605
 drzewo dekompozycji, 358
 dynamiczne
 miary rankingowe, 238
 relacje RLS, 245
 widoki zarządzania, DMV, 461
 zabezpieczenia, 244
 dystrybucja treści, 547
 alerty danych, 583
 analizowanie w Excelu, 588
 aplikacje
 Microsoft 365, 576
 niestandardowe, 579
 Power BI, 550, 571

metody, 548
 osadzanie treści, 572
 Power Automate, 585
 samoobsługowa, 591
 subskrypcje e-mail, 586
 udostępnianie
 bezpośrednie, 571
 raportów i pulpitów nawigacyjnych, 566
 zakresy udostępniania, 570
 dziennik inspekcji, 629
 monitorowanie, 634

E

edycja poświadczeń, 86
 edytor
 Power Query, 61, 72, 85, 109, 139
 zaawansowany, 139
 element wizualny
 AI, 353
 ArcGIS Maps for Power BI, 370
 Decomposition tree, 358
 Key influencers, 355
 niestandardowy, 371
 dodawanie, 372
 pobieranie, 374
 Pulse chart, 377
 Q&A, 360
 skryptu R, 365
 Smart narrative, 362
 ELT, Extract-Load-Transform, 64, 173
 e-mail, 586
 ERP, Enterprise Resource Planning, 58
 ETL, Extract-Transform-Load, 57, 98
 Excel, 116, 421, 588

F

fakty, 55, 114
 filtrowanie, 202, 280
 na poziomie
 elementu wizualnego, 288
 raportu i strony, 285
 według daty względnej, 287

formatowanie
warunkowe, 346
warunkowe wykresów, 340
wizualizacji, 336

fragmentatory, 307
niestandardowe parametry, 311
synchronizacji, 309, 310
wizualne, 308

funkcja
CALCULATE(), 211
CROSSFILTER(), 181
FILTER(), 212
Quick Insights, 384
RELATEDTABLE(), 210
Table.NestedJoin, 135

funkcje
DAX, 209
przepływu danych, 138
skalarne, 209
tabelaryczne, 209

G

GIS, Geographic Information Systems, 368

główna
nazwa usługi, SPN, 509
nazwa użytkownika, UPN, 508

grupa zabezpieczeń, 242

grupy obliczeniowe, 229

H

historia klienta, 129

HRM, Human Resource Management, 58

HTAP, 194

HTML, HyperText Markup Language, 514

HTTPS, 494

I

IDE, Integrated Development Environment, 363

identyfikacja osób, 453

identyfikator globalnie unikatowy, GUID, 446

implementacja zarządzania danymi, 599

infrastruktura jako usługa, IaaS, 474

instalacja
bramy danych, 491
serwera PBRs, 540

inteligentna narracja, 361

interfejs Power BI REST API, 442, 635

internet rzeczy, IoT, 414

ISV, Independent Software Vendor, 46, 573

J

jakość danych, 61

jednokrotne logowanie, SSO, 613

jednostki magazynowe, SKU, 190, 572

język
DAX, 37, 200
HTML, 514
M, 73, 96, 107, *Patrz także* zapytania
Power Query M
dostęp do danych, 124
tworzenie zapytań, 122
Python, 363
R, 363
wizualizacje skryptów, 363
RDL, 515
XML, 512

K

kafelki
danych strumieniowych, 415
dashboardu, 411, 413
pomocnicze dashboardu, 403
z danymi czasu rzeczywistego, 414

karta
Analiza, 379
Audit logs, Dzienniki inspekcji, 615
Capacity settings, Ustawienia pojemności, 654
Custom branding, Znakowanie niestandardowe, 623
Embed Codes, Kody osadzania, 617, 618
Featured content, Polecana zawartość, 623
Organizational visuals, Wizualizacje organizacyjne, 618
Premium Per User, Premium na użytkownika, 616

karta
 Protection metrics, Metryki ochrony, 623
 Tenant settings, Ustawienia dzierżawcy, 609
 Usage metrics, Metryki użycia, 615
 Users, Użytkownicy, 615
 Workspaces, Obszary robocze, 622

kartogram, 323

klucze odzyskiwania, 482, 496

kluczowe elementy mające wpływ, 354

kluczowy wskaźnik wydajności, KPI, 221, 318, 391

kompresja kolumnowa, 191

konfigurowanie bramy danych, 491

kontrola wersji, 455
 dla kodu DAX, 458
 dla kodu M, 458

korporacyjna analityka biznesowa, 26

KPI, Key Performance Indicator, 221, 318, 391

L

licencja, 41
 darmowa, 42
 Power BI Pro, 604
 profesjonalna, 43

linia
 prognozy, 383
 trendu, trend line, 380

linki niestandardowe, 413

logowanie jednokrotne, SSO, 508

Ł

łańcuchowanie, chaining, 80

łączenie zapytań, 127

łącznik danych, 495
 internetowych, WDC, 477

M

macierz, 265
 magistrali hurtowni danych, 50

mapa bąbelkowa, 322

MDM, Master Data Management, 55

MDX, Multi-Dimensional eXpressions, 204

metadane, 460
 kolumn i miar, 184
 opis, 187
 podsumowanie, 184
 modelu, 182
 raportowanie, 466, 467

metadata Mechanic, 465

metody dystrybuowania treści, 548, 549

metryki
 analizy czasowej, 224
 ochrony, 623
 rankingowe, 236
 użycia, 626
 wymiarów, 233

MFA, Multi-Factor Authentication, 601

MHTML, 514

miary
 bazowe, 219
 dynamiczne, 238
 jako wiersze, 350
 pomocnicze, 220

miary DAX, 201
 grupy obliczeniowe, 229
 kontekst
 filtra, 202
 wiersza, 207

metryki
 analizy czasowej, 224
 rankingowe, 236, 238
 wymiarów, 233
 proces obliczania, 205
 wydajność, 247

Microsoft Information Protection, 450

miniwykresy, 349

ML, Machine Learning, 195

model danych, 144, 149
 dynamiczne relacje RLS, 245
 metadane, 182
 optymalizacja, 189, 662

tabele
 faktów, 155
 parametrów, 167
 wymiarów, 162

widok
 danych, 152
 modelu, 149
 raportu, 153

- złożony, 196
- moduł
 - Power BI PowerShell, 444
 - PowerShell, 501
- monitorowanie
 - bramy danych, 502
 - pojemności Premium, 654, 658
- motywy pulpitów nawigacyjnych, 416

N

- narzędzie
 - ALM Toolkit, 667
 - MSHGQM, 459
 - RDL Migration Tool, 526
- nazwy główne użytkowników, UPN, 121
- niestandardowe ciągi formatujące, 344
- niezależny
 - dostawca oprogramowania, 46
 - producent oprogramowania, ISV, 573

O

- obszar roboczy BI, 429, 590, 659
 - identyfikatory, 446
 - okno dialogowe Access, 432
 - rola
 - administratora, 436
 - członka, 436
 - osoby przeglądającej, 433
 - współautora, 435
 - uprawnienia, 431
 - zestawy danych, 437, 441
- ochrona informacji, 450
- ODBC, Open DatabaseConnectivity, 474
- odświeżanie przyrostowe, 128
- okresy kroczące, 228
- OLE DB, 474
- OLTP, Online Transaction Processing, 65
- OneDrive dla Biznesu, 456
- OOS, Office Online Server, 529
- opcje poziomu prywatności, 88
- opisy
 - miar, 467
 - pól, 462

- oprogramowanie jako usługa, SaaS, 40, 77, 474, 531, 609
- optymalizacja
 - modelu danych, 662
 - raportów, 664
 - wizualizacji, 664
- osadzanie treści, 572
 - licencjonowanie, 572
 - publikowanie w sieci, 573
 - w internecie, 575
 - w Microsoft Teams, 576
 - w SharePoint Online, 576

P

- PaaS, Platform as a Service, 77, 474, 531
- panel
 - filtrów, 316
 - synchronizacji fragmentatorów, 310
 - Wizualizacje, 306
- parametry
 - co-jeśli, 313
 - źródła danych, 110
- PBRs, Power BI Report Server, 529
- planowanie
 - raportu, 251
 - zasobów przedsiębiorstwa, ERP, 58
 - zbiorów danych, 61, 98
 - transformacje danych, 62
 - tryb pobierania danych, 64
 - tryb przechowywania danych, 64
- platforma jako usługa, PaaS, 77, 474, 531
- pliki .abf, 674
- POC, Proof of Concept, 30
- pojedyncze logowanie, SSO, 476
- pojemność
 - BI
 - korporacyjna, 652
 - samoobsługowa, 652
 - dedykowana, 44
 - embedded, 46
 - premium, 45
 - premium na użytkownika, 86
 - współdzielona, 42
 - licencja darmowa, 42
 - licencja profesjonalna, 43

- polecenie Explain the increase, 386
- połączenia platformy Azure, 621
- portal administracyjny, 607
 - dzienniki inspekcji, 615
 - kody osadzania, 617
 - metryki ochrony, 623
 - metryki użycia, 615
 - obszary robocze, 622
 - polecana zawartość, 623
 - połączenia platformy Azure, 621
 - premium na użytkownika, 616
 - ustawienia dzierżawy, 609
 - ustawienia pojemności, 617
 - użytkownicy, 615
 - wizualizacje organizacyjne, 618
 - znakowanie niestandardowe, 623
- potoki wdrożeniowe Power BI, 447
 - opcje, 448
 - reguły, 449
- Power
 - Apps dla Power BI, 327
 - Automate, 585
 - Automate dla Power BI, 328
 - BI Desktop
 - edycja interakcji, 273
 - opcje, 90
 - opcje bieżącego pliku, 92
 - opcje globalne, 92
 - profilowanie danych, 60
 - BI Premium, 138, 638
 - administrowanie pojemnością, 649
 - alokacja pojemności, 650
 - cykl życia treści, 667
 - funkcje, 641
 - monitorowanie pojemności, 654, 658
 - obciążenia, 665
 - optymalizacja zasobów pojemności, 662
 - pojemność, 639
 - przypisanie obszaru roboczego, 659
 - szacowanie pojemności, 647
 - tworzenie kopii zapasowych, 673
 - ustalanie rozmiaru, 654
 - węzły pojemności, 642
 - zasoby backendowe, 645
 - zasoby frontendowe, 645
 - BI Report Builder, 515
 - źródła danych, 516
 - BI Report Server, PBRs, 392, 418, 640
 - BI REST API, 442, 635
 - Platform, 326
 - Query, 71
- PowerShell
 - obsługa klastrów danych, 501
 - poziomy prywatności, 87
 - pozyskiwanie danych, 31
 - PPU, Premium Per User, 46
 - profilowanie danych, 66
 - w Power BI Desktop, 60
 - prognozowanie, 382
 - projekt pulpitu nawigacyjnego, 395
 - projektant zbioru danych, 37
 - projektowanie
 - zapytań, 74
 - zbioru danych, 49, 51
 - identyfikacja wymiarów, 54
 - określanie faktów, 55
 - określenie ziarna, 53
 - wybór procesu biznesowego, 51
- protokół
 - HTTPS, 494
 - TCP, 494
- przepływy danych, 136
 - funkcje, 138
- przetwarzanie powolnych zmian wymiarów, 105
- przycisk Nawigator, 335
- publikowanie w sieci, 573
- pulpit nawigacyjny, dashboard, 392–394
 - architektura
 - jednopulpitowa, 405
 - korporacyjna, 407
 - wielopulpitowa, 406
 - dla urządzeń mobilnych, 425
 - dobór wizualizacji, 398
 - kafelki, 411
 - kafelki pomocnicze, 403
 - motywy, 416
 - odświeżenie bufora, 509
 - projekt, 395

- przypinanie
 - strony raportu, 424
 - wizualizacji raportowej, 419
 - układ graficzny, 400
 - wiele zbiorów danych, 410
 - pytania i odpowiedzi, 359
 - Python, 135, 363
 - wizualizacje skryptów, 363
- R**
- raportowanie
 - jakości danych, 60
 - metadanych, 466
 - raporty
 - bezpieczne osadzanie, 575
 - metryk użycia, 626
 - optymalizacja, 664
 - podzielone na strony, 417, 513
 - drukowanie, 522
 - eksportowanie, 522
 - inwentaryzacja, 524
 - korzystanie, 521
 - migracja, 528
 - ocena, 525
 - planowanie, 514, 527
 - przenoszenie, 524
 - przygotowywanie, 515
 - publikowanie, 515
 - subskrybowanie, 522
 - testy akceptacyjne, 528
 - udostępnianie, 522
 - Power BI, 393, 394
 - antywzorce wizualizacji, 269
 - diagram architektury, 257
 - dla urządzeń mobilnych, 388
 - dystrybucja, 256
 - elementy, 334
 - etykiety niestandardowe, 276
 - faza projektowania, 302
 - filtrowanie, 287, 288
 - interakcje elementów wizualnych, 271
 - macierze, 265
 - metody dostępu, 256
 - modyfikowanie, 300
 - najlepsze praktyki wizualizacyjne, 260
 - niestandardowa nawigacja, 295
 - oddzielanie od zbioru danych, 297
 - odpowiedzi na pytania biznesowe, 253, 254
 - określenie interaktywności, 255
 - określenie odbiorców, 252
 - panel Analiza, 379
 - panel Wybór, 293
 - proces planowania, 251
 - przeglądanie szczegółowe, 277
 - przełączanie zbiorów danych, 301, 302
 - przycisk Wstecz, 276
 - strony uściślające, 274
 - tabele, 265
 - tryb Widok, 297
 - tworzenie szkicu, 256
 - w trybie live, 423
 - warunki filtrowania, 282
 - właściwość Wysuń na pierwszy plan, 293
 - wybór elementów wizualnych, 264
 - wybór wykresu, 267
 - wykresy, 265
 - z metadanymi, 467
 - zakładki, 291
 - zakresy filtrów, 280
 - udostępnianie, 568, 569
 - w aplikacji SharePoint, 578
 - w trybie DirectQuery, 78
 - RDL, Report Definition Language, 515
 - Migration Tool, 526
 - reguły
 - alertu, 584
 - DLP, 454
 - relacje
 - dwukierunkowe, 177
 - funkcja CROSSFILTER(), 181
 - jednokierunkowe, 175
 - jednoznaczność, 174
 - RLS, 245
 - unikatowość, 173
 - REST, 635
 - RLS, Row-Level Security, 37
 - ROI, Return of Investment, 32

rola, 35
 Admin, Administrator, 436
 Contributor, Współautor, 435
 Member, Członek, 436
 Viewer, Osoba przeglądająca, 433

role
 w obszarze roboczym, 431
 zabezpieczeń, 240

Row-Level Security, RLS, 88

RPA, Robotic Process Automation, 326

S

SaaS, Software as a Service, 40, 77, 474, 531, 609

samoobsługowa analityka biznesowa, 27, 29, 592

scalanie kwerend, 135

scenariusze licencjonowania, 48

schemat bazy danych, 57

serwer
 OOS, 529
 raportów Power BI, PBRS, 529
 aktualizowanie, 540
 aplikacje klienckie, 544
 aplikacje mobilne, 545
 cykle aktualizacji, 543
 instalowanie, 540
 kompatybilność z SSRS, 532
 modele wdrażania, 536
 opcje połączeń, 534
 planowanie, 529
 pozyskiwanie klucza, 541
 skalowanie, 539
 topologia referencyjna, 538
 zakup licencji, 535
 źródła danych, 534

SharePoint Online, 576

składanie zapytań, query folding, 72

skoroszyty Excela, 421

skrypty
 PowerShell, 447
 R, 135

SME, Subject Matter Expert, 31

SPN, Service Principal Name, 509

SQL Server
 Analysis Services, SSAS, 475, 509, 534
 Integration Services, SSIS, 57
 Management Studio, SSMS, 509, 670
 zarządzanie zbiorami danych, 670
 Reporting Services, SSRS, 255, 333, 417, 532

SSO, Single Sign-On, 476, 508, 613

strony
 raportów pobierających dane, 423
 uściślające raport, 274

subskrypcje e-mailowe, 586

systemy informacji geograficznej, GIS, 368

szablon projektu, 32

sztuczna inteligencja, AI, 353

szybki wgląd w szczegóły, 384

Ś

śledzenie historii produktu, 58

T

tabele, 265
 agregacji, 196
 konfiguracja, 198
 relacje, 197

faktów, 155

obliczeniowe, 209

parametrów, 118, 167

tryby przechowywania, 81

wymiarów, 162
 filtry, 202
 zabezpieczeń, 121

TCP, Transmission Control Protocol, 494

Teams, 576

testowanie wydajności, 247

testy akceptacyjne użytkownika, UAT, 439, 528

transformacje danych, 62

tryb
 DirectQuery, 77, 194, 507
 projektowanie modeli danych, 143
 DirectQuery (Live), 67
 dualny, 81

- importu, 66, 76, 190, 302
 - projektowanie modeli danych, 143
- koncentracji uwagi, 397
- live, 423
- pobierania danych, 64
- połączenia na żywo, 300–302
- przechowywania danych, 64
- przechowywania tabel, 81
- wdrożenia, 24
 - korporacyjna analityka biznesowa, 26
 - samoobsługowa analityka biznesowa, 27
 - wizualizacja samoobsługowa, 27
 - wybór, 28
- Widok, 297
- złożony, 68
 - projektowanie modeli danych, 143
- tworzenie
 - aplikacji Power BI, 557
 - kopii zapasowych, 673
 - miar DAX, 200
 - opisów, 462
 - ról zabezpieczeń, 240
 - wizualizacji, 305
 - zapytań w języku M, 122
- twórca raportów, 39
- typy danych
 - Any, 131
 - kolumny pochodnej, 131
 - numeryczne, 123

U

- UAT, User Acceptance Testing, 439
- uczenie maszynowe, ML, 195
- udostępnianie
 - bezpośrednie treści, 571
 - raportu, 568
 - opcje, 569
- UPN, User Principal Name, 508
- uprawnienia w obszarze roboczym, 431
- uprawnienie twórcze, 438
- urządzenia mobilne
 - aplikacje Power BI, 565
 - pulpity nawigacyjne, 425
 - strony raportów, 388

- usługa
 - AAD, 601
 - współpraca B2B, 601
 - zasady dostępu warunkowego, 604
 - Azure Analysis, 145
 - OneDrive dla Biznesu, 456
 - Power Automate, 585
- uwierzytelnianie, 83
 - wieloskładnikowe, MFA, 601, 605

V

- VertiPaq Analyzer, 193
- Visual Studio, 141
- Visual Studio 2019, 141
- Visual Studio Code, 140
 - zapytanie M, 140

W

- warstwy zbioru danych, 144
- WDC, Web Data Connector, 477
- wdrożenia etapowe, 438
- weryfikacja koncepcji, 30
- widok
 - danych, 152
 - modelu, 149
 - raportu, 153
 - SQL, 95, 96, 99
 - wymiaru daty, 100
 - wymiaru produktu, 104
- wizualizacje, 269, 305, *Patrz także* element wizualny
 - formatowanie, 336
 - etykiety, 336
 - miniwykresy, 349
 - tabele i macierze, 342
 - wykresy, 339, 351
 - fragmentatory, 307, 316
 - jednwartościowe, 318
 - języka Python, 367
 - Karta, 318
 - karta Analiza, 379
 - kartogram, 323
 - kluczowe elementy mające wpływ, 354
 - KPI, 318

wizualizacje

- mapa bąbelkowa, 322
- mapowe, 320
- Metrics, 331
- Miernik, 319
- optymalizacja, 664
- organizacyjne, 618
- panel, 306
- panel filtrów, 316
- parametry co-jeśli, 313
- Power Apps dla Power BI, 327
- Power Automate dla Power BI, 328
- Power Platform, 326
- premium, 331
- raporty podzielone na strony, 333
- samoobsługowe, 27
- skryptów, 363
- wykres kaskadowy, 325
- wskaźniki
 - rentowności, 32
 - wzrostu, 227
- współpraca między firmami, B2B, 601
- wydajność
 - miar DAX, 247
 - modelu danych, 189
- wyjaśnianie wzrostu lub spadku, 385
- wykres, 265, 267
 - kaskadowy, 325
 - kolumnowy, 339
 - liniowy, 339, 342
 - pulsacyjny, 377
 - punktowy, 351
 - z odtwarzaniem osi, 376
 - słupkowy, 339
 - wstążkowy, 340
- wymiar produktu, 132
- wymiary, 54, 105, 114
 - daty, 180
 - współdzielone, 178
- wyszukiwanie danych, 31

X

- XML, eXtensible Markup Language, 512

Z

- zabezpieczenia
 - dynamiczne, 244
 - na poziomie wiersza, RLS, 37, 121, 432
- zadanie profilowania danych, 59
- zakładki, 291
- zakres projektu, 33–35
- zapobieganie utracie danych, DLP, 453
- zapytania, 72, 73
 - do zbiorów danych
 - w trybie DirectQuery, 77
 - w trybie importowym, 76
 - Power Query, 71
 - Power Query M, 108
 - do tabeli parametrów, 118
 - do tabeli zabezpieczeń, 121
 - filtrowanie, 125
 - funkcyjne, 122
 - kwerendy przejściowe, 112
 - łączenie zapytań, 127
 - narzędzia edycyjne, 138
 - o fakty i wymiary, 114
 - o parametry źródła danych, 110
 - przepływy danych, 136
 - projektowanie, 74
 - składanie, 72
 - w języku M, 96, 108
- zarządzanie
 - alertami, 583
 - cyklem życia treści, 667
 - danymi, 597
 - implementacja, 599
 - podstawowymi, MDM, 55
 - kłastrami bram, 497
 - lokalną bramą danych, 472
 - metadanymi, 460
 - obszarami roboczymi, 428
 - relacjami z klientami, CRM, 58
 - uprawnieniami, 570
 - zasobami ludzkimi, HRM, 58
 - zbiorami danych, 670
- zbiór danych, 61, *Patrz także* model danych
 - cele tworzenia, 146
 - relacje, 173

tryb

DirectQuery, 77, 507

DirectQuery (Live), 67

importu, 66, 76

złożony, 68

w obszarze roboczym, 437, 441

warstwy, 144

zarządzanie, 670

złożony, 79

ziarnistość tabel faktów, 53

zintegrowane środowiska programistyczne,

IDE, 141, 363

zmiennie w języku DAX, 214

zrobotyzowana automatyzacja procesów,

RPA, 326

Ż

źródło danych, 82

dodanie do klastra bram, 500

Organizational, organizacyjne, 88

parametry, 110

Private, prywatne, 88

przepływy danych, 137

usługa Power BI, 89

ustawienia, 85

PROGRAM PARTNERSKI

— GRUPY HELION —

1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion

Power BI: Twój sposób na ekspercką analizę dla biznesu!

Profesjonalna analiza danych przekłada się na sukces rynkowy i zyski. Liderem w dziedzinie analityki biznesowej jest Microsoft Power BI, pakiet obejmujący usługi w chmurze, aplikacje mobilne, aplikacje do modelowania danych i tworzenia raportów, a także inne narzędzia, takie jak lokalne bramy danych. Zdobycie zaawansowanych umiejętności korzystania z Power BI umożliwi tworzenie interaktywnych i pełnych treści analiz, które ułatwią podejmowanie trafnych decyzji.

To drugie, w pełni zaktualizowane wydanie podręcznika dla profesjonalistów. Zarówno osoby zawodowo zajmujące się tworzeniem rozwiązań w Power BI, jak i zarządzający czy administrujący wdrożeniami takich rozwiązań znajdą tu coś dla siebie. Książka zawiera rozbudowaną analizę narzędzi i funkcji Power BI, dzięki którym nauczysz się kształtować i rozbudowywać dane źródłowe, jak również tworzyć modele analityczne. Dowiesz się, jak stosować niestandardowe wizualizacje, implementować nowe polecenia DAX, zarządzać obszarami roboczymi i metadanymi. Poznasz najlepsze praktyki projektowania raportów i interaktywnych pulpitów nawigacyjnych. Dowiesz się, jakie są nowe możliwości aplikacji mobilnych Power BI i techniki samoobsługowej analizy danych. Zapoznasz się z metodami zarządzania systemem, w tym z zarządzaniem cyklem życia aplikacji z użyciem potoków Power BI.

Najciekawsze zagadnienia:

- język Power Query M i przepływy danych
- skalowalne modele danych w trybach DirectQuery
- proste i zaawansowane miary DAX
- korzystanie z map ArcGIS
- skalowanie rozwiązań za pomocą Power BI Premium

Greg Deckler posiada kilka tytułów Microsoft MVP w obszarze platformy danych. Jest ekspertem w dziedzinie Power BI, a także autorem licznych rozwiązań i narzędzi dla Power BI.

Brett Powell jest konsultantem Microsoftu i autorem wielu książek technicznych. Ma bogate doświadczenie w analityce biznesowej, jest uznawany za świetnego programistę, architekta i administratora.

	KOD KORZYŚCI Sięgnij po więcej! ▶	
 helion.pl	ISBN 978-83-8322-445-9	
 HELION SA ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	 9 788383 224459	
Cena: 149,00 zł		

<packt>