Pro Power BI Architecture

Development, Deployment, Sharing, and Security for Microsoft Power BI Solutions

—

Second Edition

—

Reza Rad

Apress®

Pro Power BI Architecture

Development, Deployment, Sharing, and Security for Microsoft Power BI Solutions

Second Edition

Reza Rad

Apress®

Pro Power BI Architecture: Development, Deployment, Sharing, and Security for Microsoft Power BI Solutions

Reza Rad
Auckland, New Zealand

ISBN-13 (pbk): 978-1-4842-9537-3 ISBN-13 (electronic): 978-1-4842-9538-0
https://doi.org/10.1007/978-1-4842-9538-0

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Editorial Project Manager: Shaul Elson
Copy Editor: Kezia Endsley

Cover image by wal_172619 from Pixabay

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, USA. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at at https://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To Leila

Who showed me how strong a human being can be

Table of Contents

About the Author xxix

About the Technical Reviewers xxxi

Acknowledgments xxxiii

Introduction xxxv

Part I: Getting Started 1

Chapter 1: Power BI Components 3

What Is Power BI? 4

The Power BI Desktop 4

Power Query for Data Transformation 5

The Power BI Report 6

Analytical Engine (Tabular Engine) 7

Datasets 8

The Power BI Dashboard 8

Dataflows 9

Datamarts 10

The DAX Language 11

The Power BI Service for Hosting Reports 12

The Power BI Mobile App 13

The Power BI Report Server 14

Paginated Reports 15

The Power BI Report Builder 16

Metrics and Scorecards 17
The On-Premises Gateway 18
Workspaces 18
Power BI App 19
Power BI Premium 20
Power BI Embedded 20
Developer API or REST API 20
Summary 21

Chapter 2: Tools and Preparation 23

Using the Power BI Desktop 23
The Power BI Desktop: Report Authoring Tool 24
The Power BI Desktop from the Microsoft Store 24
The Power BI Desktop as a Separate Download Link 26
Auto-Update from the Microsoft Store App 26
Flexibility on Installation: The Download Link 27
Development Work 27
Setting Up Power BI Accounts 27
Creating Power BI Accounts from Office 365 28
Is Your Account Pro, Free, or PPU? 29
Installing Power BI Mobile App 30
Power BI Report Server 30
On-Premises Gateway 30
Microsoft SQL Server Database and Management Tools 30
The Power BI Helper 30
Downloading Dataset Files 30
Summary 30

Part II: Development 31

Chapter 3: Import Data or Scheduled Refresh 33

The Connection Type Is Not the Data Source Type 33

Import Data or Scheduled Refresh 33

A Closer Look at Import Data 33

Loading Data to the Model 35

How Power BI Stores Data in Memory 36

How xVelocity Compresses Data 36

Where Data Is Stored in Memory 38

How About Power BI Service? 38

Is There a Limitation on Size? 39

Combining Data Sources; Power Query Unleashed 40

DAX: A Powerful Analytical Expression Language 43

Publishing a Report 44

Gateway Configuration 44

Scheduled Refresh 46

Advantages and Disadvantages of Import Data 49

Advantages of Import Data 49

Disadvantages of Import Data 49

When Should You Use Import Data? 50

Summary 50

Chapter 4: DirectQuery 51

What Is DirectQuery? 51

Which Data Sources Support DirectQuery? 51

How to Use DirectQuery 52

No Data Tab in DirectQuery Mode 54

How DirectQuery Works 55

Performance 58

Performance Tuning on the Data Source 58

Query Reduction 60
Maximum Connections Per Data Source 62
Adding Extra Data Sources: Composite Mode 64
Limited Power Query 65
Limited Modeling and DAX 66
No Refresh Needed 66
Large Scale Dataset 66
Summary 67
Chapter 5: Live Connection 69
What Is Live Connection? 69
Create a Report with Live Connection 69
Live Connection Behind the Scenes 72
Performance 73
Single Data Source 74
No Power Query Transformations 74
Modeling and Report-Level Measure 76
Publishing the Report and Gateway Configuration 78
Live Connection to Power BI Service 83
How Live Connection Differs from DirectQuery 84
Summary 84
Chapter 6: Composite Model 87
What Is Composite Model? 87
Why Use Composite Model? 88
How Does Composite Model Work? 90
Dual Storage Mode 93
Modeling with the Power BI Composite Model 95
Relationships in the Power BI Composite Model 95
Calculations and DAX 96
DirectQuery to Power BI Datasets 96
Summary 97

Chapter 7: DirectQuery for Power BI Datasets and Analysis Services 99

Power BI with Import Data 99

One Important Problem with Import Data 99

Power BI Using Live Connection 100

Big Limitation of Live Connection 100

DirectQuery for Power BI Datasets or Analysis Services Datasets 101

Why Is This Feature a Big Deal? 101

How It Works Under the Hood 102

Not from My Workspace 102

Creating a DirectQuery to Power BI Dataset 103

DirectQuery to Power BI Dataset vs Live Connection to Power BI Dataset 106

Multiple DirectQuery Connections Are Supported 107

Creating a Relationship Between Different Data Sources 107

Publish to Service 108

Lineage View 113

Sharing the Report 114

Building Access to the Source Power BI Datasets 114

Summary 116

Chapter 8: Choosing the Right Connection Type: DirectQuery, Live, Import, or Composite Model 117

Why Is This a Tough Decision? 117

What Is Import Data (Scheduled Refresh)? 118

Compression Engine of xVelocity 118

Important Pros and Cons of This Method 118

What Is DirectQuery? 119

Important Pros and Cons of This Method 119

What Is Live Connection? 121

Important Pros and Cons of This Method 121

Composite Model: The Best of Both Worlds 122

Differences Between Live Connection and DirectQuery 123

Relationship Configuration 123

Report-Level Measures 124
No Power Query in Live Connection 124
Pros and Cons of Each Method 124
Import Data or Scheduled Refresh 124
DirectQuery 124
Live Connection 124
Which Method Is the Best and Fastest? 125
Which Method Is More Flexible? 125
Which Method Is More Scalable? 125
Which Architecture Scenarios Are Best for Each Method? 125
Import Data for Agility and Performance 125
Live Connection for an Enterprise Solution 126
Live Connection for Team Development 126
Direct Query for Non-Microsoft Sources 126
Summary 128
Chapter 9: Dataflows 129
What Is a Dataflow? 129
A Dataflow Is Service-Only (Cloud-Only) Object 130
A Dataflow Is Not Just for Power BI 130
Where Do Dataflows Store Data? 131
Standard vs. Analytical Dataflows 132
Dataflows Are Powered by Power Query Online 133
Dataflows Can Be Used in Power BI, Excel, and Other Services 135
Example Dataflow Scenarios 136
Using One Power Query Table in Multiple Power BI Reports 136
Different Data Sources with Different Refresh Schedules 138
Centralized Data Warehouse 139
Getting Started with Dataflows in Power BI 140
Administrator's Control 141
Creating Your First Dataflow 142
Sample Datasets 145

Power Query Editor Online 146
Setting Up a Gateway 147
Blank Query as the Data Source 148
Scheduled Refreshes 150
Get Data from Dataflow in the Power BI Desktop 151
Get Data from Dataflows Using Excel 154
Summary 155
Chapter 10: Shared Datasets 157
What Is the Dataset in Power BI? 157
What Is Included in the Dataset? 159
What Is a Shared Dataset? 159
How Do You Create a Shared Dataset? 162
Sharing Datasets Across Multiple Workspaces 162
External Dataset: How Does Shared Dataset Work Behind the Scenes? 163
Why Use Shared Datasets? 163
Shared Datasets in the Power BI Architecture 164
Endorsement: Certified and Promoted Datasets 165
Summary 168
Chapter 11: Datamarts 169
Power BI for the Citizen Data Analyst 169
Governance and Reusability 169
What are Datamarts in Power BI? 170
Who Are Datamarts for? 172
Power BI Datamart for the Citizen Data Analyst 173
Power BI Datamart for Enterprises 173
Power BI Datamart for Developers 173
What Are the Features of Datamarts that Empower Power BI Development? 173
Datamarts Complete the BI Ecosystem 173
One Place to Manage and Build, No Other Tools Needed 174
One License Is Enough 175

The Vast Horizon and the Future of Datamarts 175
Governance 175
Getting Started with Power BI Datamarts 175
Premium Workspace 175
Creating a Datamart 177
Get Data: Power Query Online 179
The Datamart Editor 182
Power BI Datamart Components 190
Power BI Datamarts Under the Hood 190
Summary 202
Chapter 12: The Multi-Layer Architecture 203
Challenges of Using a Single PBIX File for Everything 203
Using Dataflows to Decouple the Data Preparation Layer 204
Shared Datasets for the Data Modeling Layer 206
Thin Power BI Reports, Paginated Reports, and the Analyze in Excel Option 206
Power BI Architecture with Dataflows, Shared Datasets, and Thin Reports 208
Benefits of a Multi-Layered Architecture 209
Enhancing the Architecture Using Power BI Datamarts 210
Layered Architecture for Dataflow 211
Chained Datasets 214
What About Enterprise Architecture? 215
Summary 216
Chapter 13: Datamarts vs. Dataflows vs. Datasets 217
What Is a Dataflow? 217
What Is a Dataset? 218
What Is a Datamart? 218
Differences Between Dataflows and Datasets 219
Dataflows vs. Datasets 224

What About Datamarts? 225
Will Dataflows Be Replaced by Datamarts? 226
Are Datasets Being Replaced by Datamarts? 227
Summary 228
Chapter 14: Paginated Reports 229
What Is a Paginated Report in Power BI? 229
Why Paginated Reports? 231
Differences Between Power BI Reports and Paginated Reports 233
Do You Need a Paginated Report? 234
Architecture Best Practice: One Dataset to Rule Them All 235
The Power BI Report Builder 235
Summary 240
Chapter 15: Excel and Power BI Integration 241
The Analyze in Excel Feature 241
The Analyze in Excel Feature from the Power BI Service 241
Implicit Measures Don't Work in Excel 244
Excel Is Connected Live to the Power BI Model in the Service 247
Getting Data from Power BI Dataset 248
Getting Data from a Power BI Dataflow 251
Why is Analyze in Excel Better than Using Export Data? 252
Publish to Power BI from Excel 254
Upload Your Workbook to Power BI 255
Export Workbook Data to Power BI 256
Import Excel into the Power BI Desktop 256
Summary 257
Chapter 16: Real-Time Streaming Datasets in Power BI 259
Real-time vs. Live or DirectQuery 259
Sample Scenario 260
Capturing Inputs with a Microsoft Form 260

The Power BI Streaming Dataset 261
Power Automate Pushes Data to the Power BI Dataset 265
The Power BI Real-Time Dashboard 268
Streaming Dataset Types 269
Push Datasets 269
Streaming Datasets 269
Hybrid Datasets 269
Streaming Services 270
Power BI Streaming Dataset and REST API 270
Azure Stream Analytics 270
PubNub 270
Creating Calculations on Real-Time Datasets 270
Writing Calculations Using Q&A 270
Live Connection to a Streaming Dataset 274
Summary 278
Chapter 17: Performance Tuning with Aggregations 279
How Does Aggregation Work? 280
Aggregated Table 281
Layers of Aggregation 282
Step 1: Create the Aggregated Table 283
Dual Storage Mode; The Most Important Configuration for Aggregations! 293
Step 2: Power BI Aggregations 293
Step 3: Configure Aggregation Functions and Test Aggregations in Action 310
Multiple Layers of Aggregations 318
Aggregation for Imported Data 328
DAX Measures Instead of Manage Aggregation 331
Automatic Aggregation 336
Summary 337

Chapter 18: Big Data with Incremental Refresh and Hybrid Tables 339

What Is Incremental Refresh? 339

Hybrid Tables 339

Configuring Incremental Refresh 340

Setting Up Incremental Refresh 340

Parameters in Power Query 340

Filter Data Based on the Parameters 342

Power BI Desktop Incremental Refresh Setup 344

Hybrid Table Setup 347

Incremental Refresh for Multiple Tables 348

Publish to Service 349

What Do the Partitions Look Like? 350

Detecting Data Changes 354

Only Refresh When Period Is Complete 355

Incremental Refresh for Dataflows or Datamarts 356

Summary 357

Chapter 19: Development Best Practices 359

Reuse Tables Generated in Power Query 359

Reuse DAX Calculations 359

Use Power Query Functions for Reusable Data Transformation 360

Parametrize the Data Transformation Process Using Power Query Parameters 360

Categorize Power Query Using Power Query Groups (or Folders) 360

Disable the Load of Temp Tables in Power Query 361

Only Load What You Need for Reporting: Filter the Data 361

Use Reference and Duplicate in Their Rightful Places 361

Multi-Layer Architecture Everywhere 361

Sync Slicers 362

Use a Theme File 362

Use a Background Image for Report Pages 362

Incorporate Conditional Formatting Using DAX, Parameters, and Tables 362
Create a Measure Table 363
Create and Use Hierarchies 363
Set Auto Summarization at the Field Level 363
Consider Using a Custom Date Table 363
Design Mobile Reports 364
Use Aggregations for Big Tables 364
Use Incremental Refresh and Hybrid Tables When Possible 364
Design Star-Schema Models 365
Avoid Both-Directional Relationships 365
Clean Up Your Models 365

Part III: Deployment and Collaboration 367

Chapter 20: The Power BI Service 369
The Power BI Desktop: A Report Authoring Tool 369
Hosting Options for Power BI Reports 370
What Is the Power BI Service? 371
Publish Reports to the Service 372
Capacity-Based or User-Based 373
The Power BI Service 374
Template Apps 375
Apps 375
Workspaces 376
Dataflows 377
Shared Datasets 378
Datamarts 378
Deployment Pipelines 379
Dashboards 380
Metrics 380

Content Certification 381
Data Lineage 382
Summary 382
Chapter 21: The Power BI Report Server 383
What Is the Power BI Report Server? 383
Requirements for Setting It Up 384
Installing the Power BI Report Server 385
Configuring the Power BI Report Server 390
Database Set Up 392
Web URL Set Up 396
Installing the Power BI Desktop Report Server 402
Developing Reports with the Power BI Report Server 403
Publish Report to the Report Server 405
Managing Datasets on the Report Server 407
Schedule Refresh Requirement 409
Pros and Cons of the Report Server 412
No Gateway Is Needed 412
All Types of Connections Are Supported 413
The Power BI Report Server Is a Fully On-Premises Solution 413
The Power BI Service Features Are Not Available 413
Licensing the Report Server 413
Summary 414
Chapter 22: Power BI Gateways 415
What Is a Gateway? 415
Do You Always Need a Gateway? 415
Types of Gateways 415
Gateway Execution Flow 417
Important Points to Consider Before Installing a Gateway 418
How Many Gateways Are Required? 418
Installing Gateways 419

Adding Data Sources 424
Set Up a Gateway Connection to the Dataset 428
Users and Access Controls for Gateways 429
Gateway Installers 429
Gateway Users 430
Data Source Users 431
Additional Settings for Gateway 432
Troubleshooting 435
Summary 436
Chapter 23: Power BI Licensing 437
Two Types of Licensing 437
The Power BI Desktop 438
Why Do I Need a License if the Power BI Desktop Is Free? 438
User-Based Licensing 439
Power BI Free 439
Power BI Pro 439
Power BI PPU: Premium Per User 441
Capacity-Based Licensing 442
Power BI Embedded 442
Power BI Premium 444
The Difference Between A SKUs and P/EM SKUs 446
SQL Server Enterprise Edition Plus Software Assurance 447
Summary 449
Chapter 24: Power BI Admin Portal and Tenant Settings 451
Power BI Administrator 451
Tenant Settings 452
Help and Support Settings 453
Workspace Settings 455
Information Protection 455
Export and Sharing Settings 456

Content Pack and App Settings 459
Integration Settings 460
Custom Visual Settings 462
Audit and Usage Settings 463
Developer Settings 464
Dataflow, Template Apps, and Datamarts 464
Usage Metrics 465
Embed Codes 465
Organization Visuals 466
Workspaces 468
Custom Branding 468
Summary 469
Chapter 25: PowerShell Cmdlets for Power BI 471
PowerShell 471
PowerShell Cmdlets for Power BI 471
Getting Started 472
Example: Export a List of All Workspaces in the Organization as a CSV File 478
Example: Deploy a Power BI Report from One Workspace to Another 479
Are PowerShell Cmdlets the Same as REST APIs in Power BI? 483
Summary 483
Chapter 26: Power BI REST API 485
The Power BI REST API 485
Getting Started 485
Calling the API from .NET 490
Embedding into a Custom Application 492
Power Automate 493
PowerShell to Call REST API 493
Summary 493

Chapter 27: Power BI Audit Log for the Tenant 495
The Problem 495
The Audit Log 498
Using PowerShell to Extract the Audit Log 502
The Power BI Report for the Audit Log 505
Summary 512
Chapter 28: XMLA Endpoint 513
Behind the Scenes of a Power BI Dataset 513
SSAS Is More Than What You See 514
Can I Access the SSAS Model of My Power BI Dataset? 516
XMLA Endpoint 517
How Does an XMLA Connection Work? 518
The XMLA Endpoint URL 519
Admin Control under Capacity Configuration 519
Client Tools 520
Sample Connection 521
The Power BI Helper 522
Summary 522
Chapter 29: Dashboard and Report Sharing 523
Power BI Content Owner 523
How Does Dashboard Sharing Work? 523
Three Levels of Access 528
Manage Permissions 528
Advantages of Dashboard Sharing 532
Disadvantages of Dashboard Sharing 532
No Edit Access 533
Share Objects One at a Time 533
Licensing for a Large User Base 533
Summary 533

Chapter 30: Power BI Workspaces 535
What Are Power BI Workspaces? 535
Creating Workspaces 536
Adding Content to a Workspace 541
Four Levels of Workspace Access 543
Tenant Admin Control for Workspaces 544
Advantages of Workspaces 547
Disadvantages of Workspaces 547
Summary 548

Chapter 31: Power BI Apps 549
Power BI Apps 549
App Workspaces 549
Creating an App 551
Setup 552
Content 554
Audience 556
Getting Apps 559
Making Changes to an App 561
Isolated Environments for Developers and Users 562
What About Data Refresh? 564
Advantages of Power BI Apps 564
Cons of Power BI Apps 565
Summary 565

Chapter 32: Publish to Web 567
What Is Publish to Web? 567
Security Issues with Publish to Web 570
Removing Access 572

Central Monitoring for all Embed Codes 574
Who Can Publish Reports on the Web? 574
Differences with Other Sharing Methods 575
Summary 578
Chapter 33: Power BI Embedded 579
Using Power BI Embedded 579
Myths about Power BI Embedded 580
What Is Power BI Embedded? 580
Embedded Playground 580
Licensing Power BI Embedded 582
Some SKUs Can Be Turned On and Off 583
Power BI Users or Custom Application Users 583
Implementing Power BI Embedded 584
Summary 585
Chapter 34: Secure Embed 587
Why Another Embedding Method? 587
Publish to Web 587
Power BI Embedded 588
Secure Embed: Good Features from Both Worlds! 589
Using Secure Embed 589
Who Can View the Report? 590
Licensing Needs 593
Advantages of Secure Embed 593
Limitations 594
Scenarios for Using this Method 594
Summary 595

Chapter 35: Embed in SharePoint Online and Teams 597

How to Use Embed in SharePoint Online 597

Sharing a SharePoint Page with an Embedded Power BI Report 604

Access to SharePoint 604

Power BI Permissions 605

Advantages and Disadvantages of Embedding in SharePoint Online 608

Advantages of Embed in SharePoint Online 608

Disadvantages of Embed in SharePoint Online 609

Power BI and Teams 609

Power BI and PowerPoint 610

Summary 611

Chapter 36: Comparison of Sharing Methods for Power BI 613

Types of Sharing Methods 613

Basic Sharing for Dashboards and Reports 614

Advantages of Dashboard Sharing 615

Disadvantages of Dashboard Sharing 615

Workspaces 615

Advantages of Workspaces 616

Disadvantages of Workspaces 617

Power BI Apps 617

Advantages of Power BI Apps 618

Cons of Power BI Apps 619

The Publish to Web Option 619

Security Issues with Publish to Web 620

Embed in SharePoint Online 620

Advantages of Embed in SharePoint Online 621

Disadvantages of Embed in SharePoint Online 622

Power BI Embedded 622

Pros and Cons of Power BI Embedded 622

Secure Embed 623
Advantages of Secure Embed 623
Disadvantages and Limitations of Secure Embed 624
Summary 624
Cheat Sheet for Choosing a Method 624
Chapter 37: Types of Power BI Users 625
The Different Types of Power BI Users 625
Access Control 626
Training 627
The Layers and User Categories Are Different in Each Organization 628
Users Can Move Between Layers 629
Summary 629
Chapter 38: Best Practices for Setting Up Workspace Roles 631
The Modern Power BI Workspace 631
Roles in the Workspace 632
Viewer 632
Contributor 633
Member 634
Admin 637
Best Practices for Each Role 638
Viewer Role: Use Power BI Apps Instead 638
Contributor Role: Developers Only 639
Member Role: Deployment Group Only 639
Admin Role: Admin Group 640
Use Groups, Not Individual Accounts 640
Summary 640

Chapter 39: Build Access Level 641
Build Access Example 641
Are Build and Edit the Same? A Row-Level Security Example 643
Providing Build But Not Edit Access 645
Manage Permissions: Add Build Access 645
Create an App with Build Access 646
Who Should Have Build Access? 648
Licensing 648
Summary 649
Chapter 40: How to Organize Workspaces in a Power BI Environment 651
Considering a Power BI Workspace as a Single Development-Sharing Unit 651
Separating Audiences Using Power BI Apps 651
Separating Developers with Multiple Workspaces 652
Split the Load, Use the Capacity 654
Sharing Workspaces Among Multiple Developers 654
Layers of Shared Workspaces 655
Separating the Environments 656
Summary 660
Chapter 41: Content Certification and Endorsement 661
The Importance of Endorsement 661
Levels of Endorsement 661
What Kinds of Content Can Be Endorsed? 663
Who Can Endorse It? 663
Dataset Discoverability 665
An Example of an Endorsement and Content Certification 666
Summary 668

Chapter 42: Deployment Pipelines 669
Why Multiple Environments? 669
How Many Environments Do You Need? 672
What Is a Deployment Pipeline? 673
How Does a Deployment Pipeline Work? 674
Creating Deployment Pipelines 674
Assigning a Workspace 675
Comparing Content 676
Deploy 678
Rules and Connections 680
Automate the Deployment 680
What If There Is No Premium License? 681
Summary 681
Chapter 43: Data-Level Security 683
Introduction 683
Row-Level Security 683
Static Row-Level Security 684
Dynamic Row-Level Security 693
Column-Level Security 703
Object-Level Security 704
Page-Level Security 704
Summary 704
Chapter 44: Power BI Helper 705
Documenting a Power BI File and Report 705
Download and Install Power BI Helper for Free 705
Open Power BI Helper as an External Tool 705
Connect to Model 706
Visualization Information 707
Documentation 708

What Is Included in the Documentation? 710
Configurations 710
Exporting the Data in a Power BI Table 712
Connect to Model 712
Export the Data 713
Any Size, Any Table Type 716
Consider Using Analyze in Excel 717
Reducing the Size of Power BI File 717
Steps to Reduce File Size 718
Documenting Power BI Tenant Objects 722
Service Tenant Settings in Power BI Helper 723
Scan Service Objects 730
Configuring the Documentation's Output 732
Exporting the Power BI Audit Log 735
What Is the Power BI Audit Log? 735
Difficulties Exporting the Audit Log 735
A Simple Way to Export the Audit Log Without Limitations 736
Power BI File Cleanup in a Few Steps 739
Visualization Information 740
Hiding Technical Fields 743
Summary 745
Index 747

About the Author

Reza Rad is a Microsoft regional director, a best-selling author, a Microsoft certified trainer, a consultant, and a noted international speaker featured at Microsoft Ignite, Microsoft Business Applications Summit, Data Insight Summit, and PASS Summit, among others. He has a BSc in computer engineering and more than 20 years of experience in data analysis, BI, databases, programming, and development, mostly related to Microsoft technologies. He has been a Microsoft Data Platform MVP for 13 years. A leader in his field, Reza leads the New Zealand Business Intelligence users group and is the co-founder and co-organizer of RADACAD, a training and consulting business focusing on Microsoft data analytics technologies, the Difinity conference in New Zealand, the Power BI Summit (the largest Power BI conference), and the Data Insight Summit (Chicago, USA). Reza has also been recognized as a Dynamic Communities Emerald award winner, a Power BI All-Star award winner, and a Microsoft Fast Track Recognized Solution Architect. Reza has also earned the MCP, MCSE, and MCITP for BI certifications.

An active blogger, Reza's articles on different aspects of technologies, especially on BI, can be found on his blog at `radacad.com/blog`. He was also an active member on online technical forums such as MSDN and Experts-Exchange, and was a moderator of MSDN SQL Server forums. Reza has written more than 12 books on BI, Power BI, and analytics, including the popular *Power BI: From Rookie to Rock Star* and *Power BI Pro Architecture,* both published by Apress.

When Reza is not doing data-related work, he enjoys playing fetch with his Akitas, Khersi and Lucy, rewatching the *Lord of the Rings* and *Star Wars,* strumming classical guitar, and working his way toward a helicopter flying license. You can connect with Reza on LinkedIn (`www.linkedin.com/in/rezarad/`), Twitter (`twitter.com/Rad_Reza`), and his blog (`radacad.com/contact-us`).

About the Technical Reviewers

Dr. Greg Low is a member of the Microsoft Regional Director program that Microsoft describes as "150 of the world's top technology visionaries chosen specifically for their proven cross-platform expertise, community leadership, and commitment to business results." He is the founder and principal consultant at SQL Down Under, a boutique data-related consultancy firm in Australia. Greg is a long-term data platform MVP, a well-known data community leader, and a public speaker at conferences worldwide. He is known for his pragmatic attitude to business transformation and to solving issues for businesses of all sizes.

Gilbert Quevauvilliers has been working in the data analytics space for the past 14 years, with experience in Azure and Power BI. Gilbert has also been recognized by Microsoft as a Microsoft MVP for the past six years. Gilbert works at FourMoo, consulting with clients to use their data as an asset. You can find Gilbert on Twitter `@gilbertQue` and at LinkedIn.

Eugene Meidinger works as an independent BI consultant and Pluralsight author, specializing in Power BI and the Azure Data Platform. He is a Microsoft MVP for the data platform. He has been working with data for over ten years and speaks regularly at user groups and conferences. He also helps run the GroupBy online conference.

Acknowledgments

Writing a book at this scale is a big project. There are many whom I want to thank for helping me complete this project. First, I want to thank my book's wonderful team of reviewers—Gilbert, Eugene, and Greg—for their wealth of expertise and knowledge of Power BI. They reviewed every chapter of this book, and they are the main reason for the high quality of the book's content. I must also thank Apress and my editor Shonmirin for helping with this project.

Big thanks go to the readers of my blog articles, my YouTube channel subscribers, and the students of my Power BI courses worldwide. They provided the most valuable feedback. They brought up discussion points so that I could challenge the solutions and develop new ways of doing things.

I'd also like to thank Microsoft's Power BI team for their outstanding work building a fantastic service and product, which is the core of this book. It is no wonder that Power BI has been at the forefront of all the analytical tools and services in the market these past few years: The team behind it is the greatest team I have seen building a Microsoft product.

Last but not least, I'd like to thank Leila for the lessons and the feedback, and for sharing her wealth of experience and support while I completed this project.

Introduction

Since 2018, when I wrote the first edition of this book, many features and significant updates have been added to the world of Power BI. Many of these features changed the architecture of Power BI solutions. To design the right architecture, you must understand all the features, services, tools, and components of Power BI and make informed decisions about using these components in the right situations.

The first edition, written in 2018, covered many aspects of architecture (which are still valid now), but this newer edition has many additions. When I was thinking about the second edition of this book, my initial thought was that it would require 50 percent new content, but in the end, I have to say about seventy percent of the content is new and updated from the first edition.

This book targets architects of Power BI solutions. This might be someone with the title of Power BI architect or data analytics architect, or a developer designing an architecture for a Power BI solution. Because architecture is the focus of this book, many features and components are explained throughout this book, but they are not discussed at a deep technical level. Architects must know what each component is, what it does, and how it works with other components, but they might not necessarily know how to implement them all. That is the job of development. Because of that, many aspects of Power BI development are not discussed in this book. This is not a book about creating a Power BI report or writing a DAX measure. However, this book includes chapters that explain best practices for some of these actions and many other important facts.

A good Power BI architecture leads to good Power BI adoption and an easy-to-maintain Power BI implementation that will last for years. Your entire organization can benefit from it.

This book is for Power BI developers, analysts, architects, and managers who want to look at the Power BI implementation holistically and consider all aspects. The book is split into parts for developing, deploying, and sharing Power BI solutions.

The development part of this book starts with looking at the different types of connections in the Power BI environment. It takes a detailed look at Import Data, DirectQuery, and Live Connection and explains the differences between these connection methods. It also expands the DirectQuery connection topic to include the Power BI datasets, composite mode, and the pros and cons of each.

The book then continues with a detailed look at dataflow, including its use cases, how it works with the shared dataset concept, and how the datamart plays a role in the architecture. This book explains the differences between dataflows, datasets, and datamarts, and how they work together to build a multi-layered Power BI architecture.

The book continues with aspects related to Power BI development, which are explained for specific cases such as paginated reports for printing or using Analyze in Excel and real-time streaming datasets for an IoT dashboard.

The development part wraps up with performance-tuning tips for big data projects, such as aggregations and incremental refresh, and includes general Power BI development best practices.

The deployment part of the book covers the Power BI service as the cloud-based hosting solution for Power BI and compares it with the Power BI Report Server, which is a fully-on-premises hosting option for Power BI. The role of the on-premises gateway is fully explained in this book. Some chapters explain the licensing of Power BI in detail.

The administration part of Power BI tenant, such as tenant settings and PowerShell cmdlets or REST API calls for admins, is explained throughout this book. Features such as the XMLA endpoint and its role in managing a Power BI solution are also discussed, with examples.

The book continues with a detailed look at all Power BI sharing methods. It starts from individual object sharing to the workspaces to sharing via Power BI apps. It also covers features such as Power BI Embedded, the embed code, and the differences between the Publish to Web option and options such as embedding in SharePoint Online. The book explains the differences between all these sharing methods and discusses their pros and cons.

The deployment part then continues with a look at how workspaces should be set up, how their roles and access levels need to be set, and the deployment pipeline needed to manage multiple environments. The book also covers some aspects of governance, such as content certification, and some aspects of security, such as row-level security.

I cover a lot of components in this book. Based on years of experience designing Power BI architecture, I find them all related to architecture design. The Power BI service and its features are a moving target when it comes to writing books. I encourage you to check out my blog at `radacad.com` for updates on new features added after this book's publish date. I hope you enjoy the book.

—Reza Rad

CHAPTER 1

Power BI Components

Power BI is more than just the Power BI Desktop. It is a combination of several components. Learning about these components and their roles is important before you go further. It is impossible to talk about architecture when you don't know about Power Query or Power BI. These components play an important role in the Power BI solution. This chapter covers the following components:

- Power BI
- The Power BI Desktop, a tool for report development
- Power Query, a tool for data transformation
- DAX and modeling, the analytical engine
- The Power BI Service, for hosting reports
- Power BI mobile apps
- The Power BI Report Server, an on-premises host for Power BI
- The Power BI Report Builder, for building paginated reports
- On-premises gateways
- Dataflows
- Datamarts
- Datasets
- Reports
- Paginated reports
- Dashboards
- Metrics and scorecards
- Deployment pipelines
- Workspaces
- Power BI apps
- Power BI Premium
- Power BI Embedded
- The Developer API

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_1

What Is Power BI?

Power BI is a cloud-based technology from Microsoft for reporting and analyzing data. This reporting technology allows developers to create reports. Power BI is a simple, easy-to-use, and user-friendly environment for creating reports. It is based on several powerful components that help you create reports from complex scenarios.

Each component in Power BI is responsible for a specific part of the technology. There are components for building reports, connecting to data sources, performing analytics calculations, sharing reports, and so on. The following sections explain the applicable components. Other components are explained in detail in the future chapters of this book.

The Power BI Desktop

The Power BI Desktop is an important component of Power BI. This tool is the report development or report authoring editor for Power BI reports. It's free to download, install, and use. This tool is lightweight, about 450MB, and can be easily downloaded and installed. There is no configuration necessary when installing it. The Power BI Desktop supports installation only on Windows machines at the time of writing this chapter. There are some ways to install it on non-Windows machines—such as on Macs—using utility tools, but that topic is outside this chapter's scope.

Figure 1-1 shows the Power BI Desktop at the time of writing this chapter.

Figure 1-1. *The Power BI Desktop, a tool for authoring reports*

The Power BI Desktop is not a complex tool. Although most report development these days uses the Power BI Desktop, the Power BI team is currently enabling the Power BI Service's user interface to provide the same functionality. This means that without installing a tool and on any operating system, users will be able to build Power BI reports in the future.

Power Query for Data Transformation

A component of every reporting tool is connecting to data sources and preparing data (or data transformation). Power Query, also called Get Data and Transform, is used for that purpose. Power Query connects to different types of data sources, gets data from those sources, gives you the ability to apply transformations, and finally loads the data into your Power BI dataset.

Power Query comes in different shapes and sizes. It is not just available within the Power BI Desktop. You can even find it in Excel. This component is part of the Power BI Desktop. When you install the Power BI Desktop, Power Query (shown in Figure 1-2) is automatically installed. You can start working with it by choosing Get Data from Power BI.

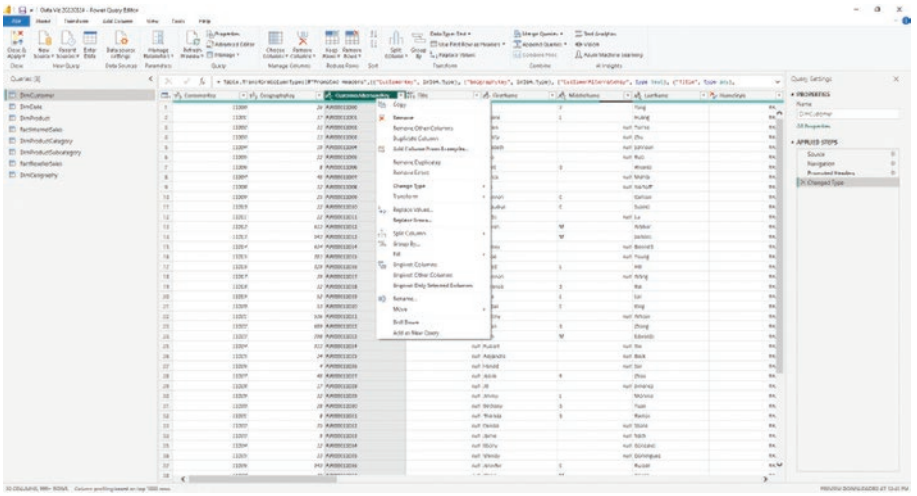


Figure 1-2. *Power Query, the data transformation engine*

There are many data sources you can access data from when using Power Query. You can even create connectors yourself.[1] Power Query has a window for development called the Power Query Editor. The Power Query Editor is where you apply all data transformations, and it prepares data to load into Power BI. This process usually happens before the data loads into Power BI.

Power Query uses a functional language called M,[2] as shown in Figure 1-3. M is the behind-the-scenes language for data transformation in Power BI.

[1] To learn more, visit `radacad.com/power-bi-custom-connector-connect-to-any-data-sources-hello-world`

[2] To learn more about M, visit `radacad.com/basics-of-m-power-query-formula-language`

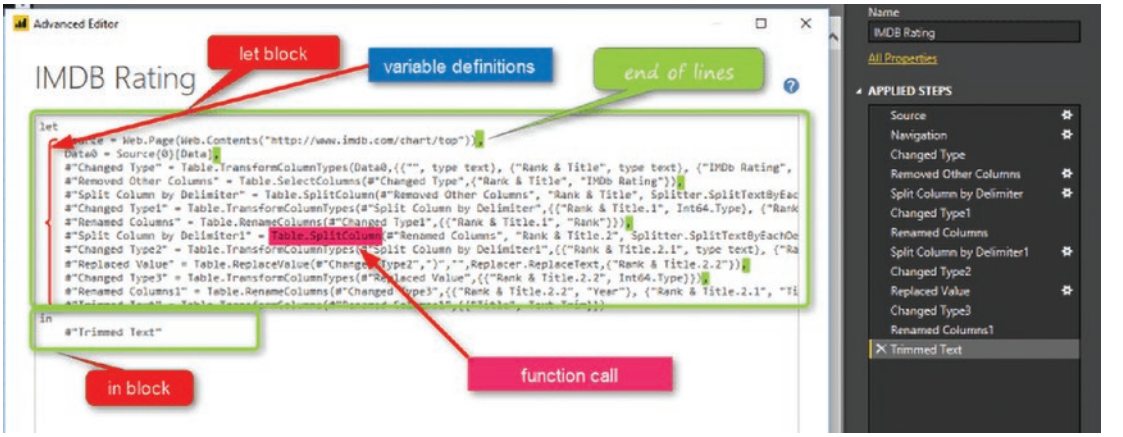


Figure 1-3. *The M scripting language for Power Query*

The Power BI Report

The Power BI Report (shown in Figure 1-4) is the visualization layout of Power BI. The Power BI Report is an interactive set of visualizations across multiple pages. This visualization allows the users to slice and dice the data. The Power BI Report is designed either in the Power BI Desktop or through the Power BI Service.

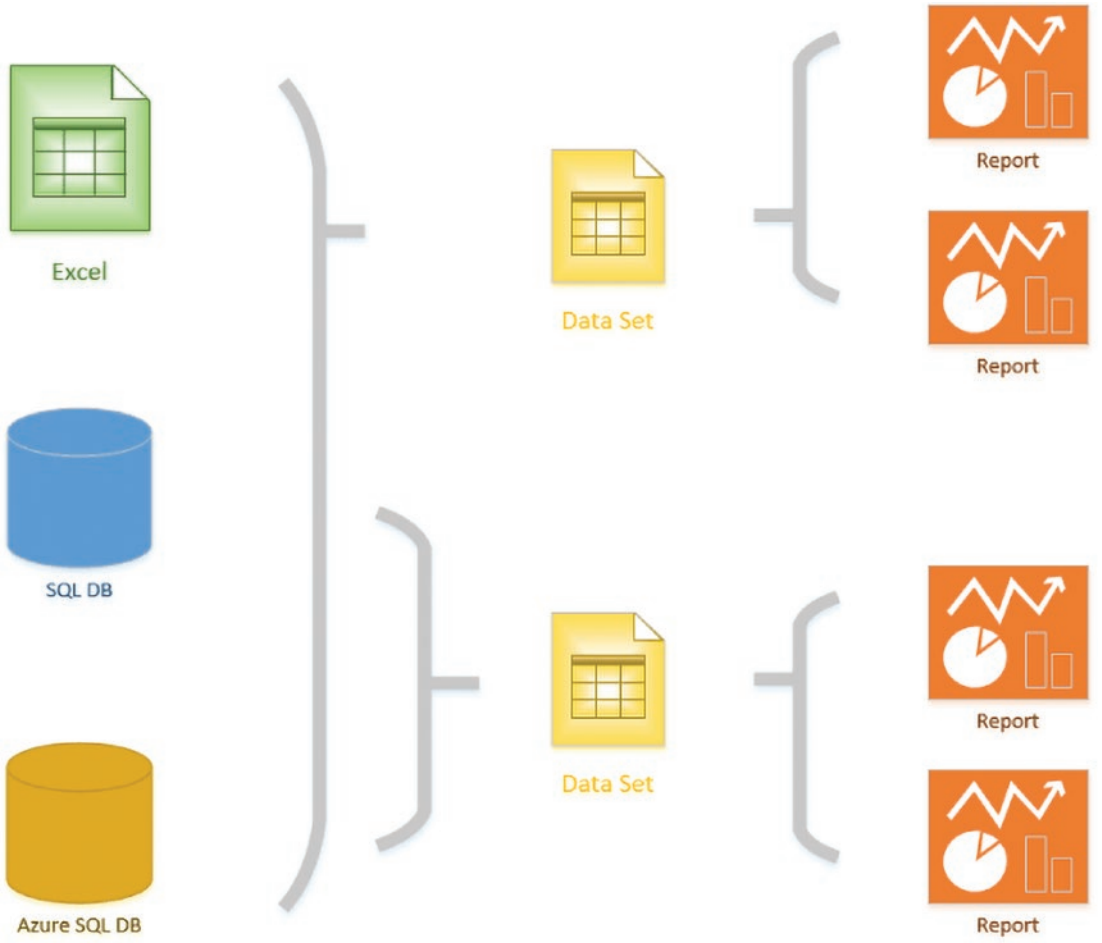


Figure 1-4. *The Power BI Report connected to Power BI datasets*

Analytical Engine (Tabular Engine)

This component is the analytical core and engine of the Power BI data model. The Tabular Engine installs as part of the Power BI Desktop, with no specific editor window. All modeling, calculations, and configurations are shown in the Power BI Desktop window. DAX and modeling are built into the Power BI Desktop in a way that most people consider a non-detachable part of the Power BI Desktop. However, the data model component of Power BI is a separate object, called a *dataset*.

The Power BI Analytical Engine is similar to the SQL Server Analysis Services Tabular or Azure Analysis Service, as shown in Figure 1-5. Depending on the Power BI license you use, there might be differences between those two services.

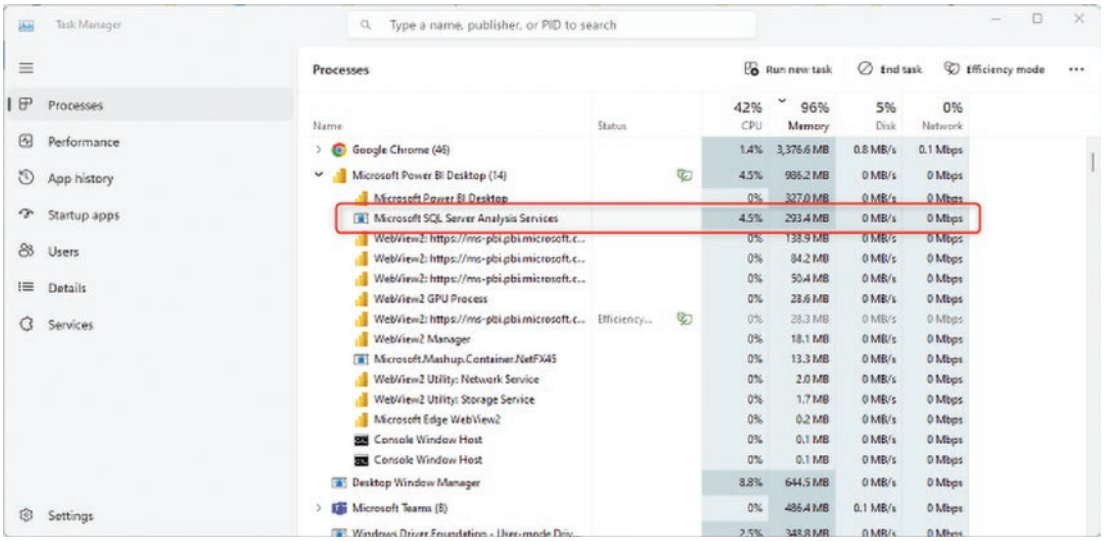


Figure 1-5. *The analytics engine of Power BI is called Analysis Services Tabular*

Datasets

A dataset in the Power BI environment is where all the data, relationships, connections to the data source, DAX calculations, and field- or table-level configurations live. The report is connected live to this dataset to produce visualizations. Multiple reports can be created from the same dataset, which is called a *shared dataset.*

There are different types of datasets depending on how the connection is created. Sometimes a dataset uses a DirectQuery connection. Sometimes, a dataset imports data from other data sources. Sometimes, a dataset is a DirectQuery connection to other datasets. A dataset can even be a real-time streaming dataset.

The Power BI Dashboard

Although the report is the core of visualization in the Power BI environment, the *dashboard* can sometimes be used to provide a holistic view across multiple reports. A dashboard in the Power BI environment is a visualization element that is not interactive (unlike the report). It provides a single view of multiple reports, as illustrated in Figure 1-6. It also comes with features such as data-driven alerts, real-time tiles, and auto-refresh.

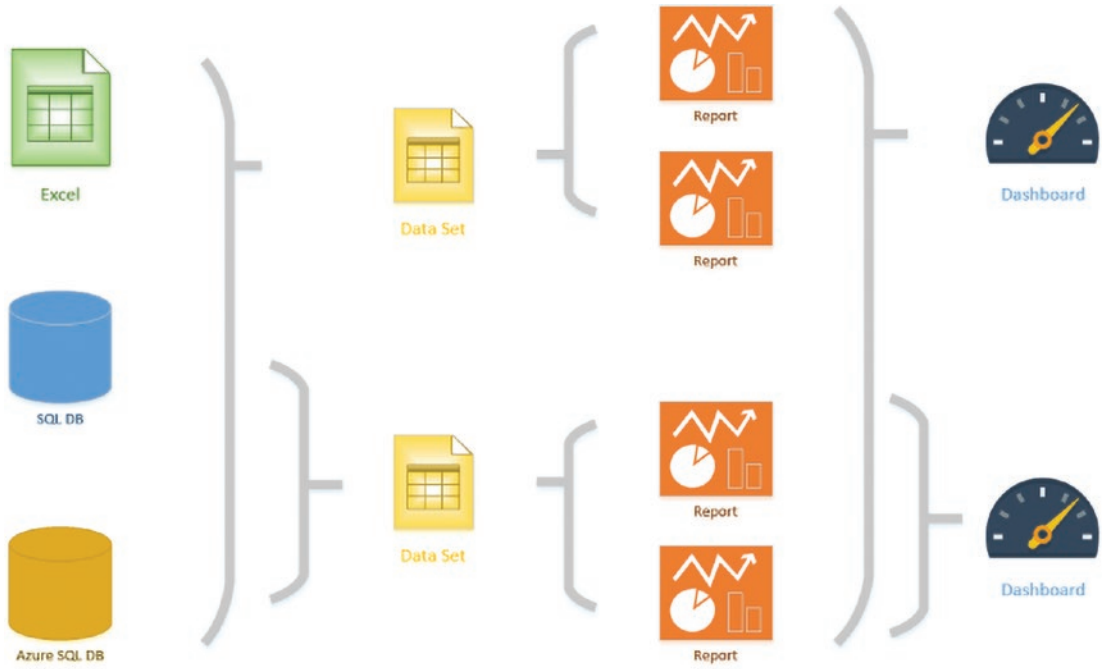


Figure 1-6. *The Power BI Dashboard can consolidate visualizations from multiple reports*

Dataflows

The data transformation layer of Power BI can be separated from the dataset so that it provides better storage capabilities. This enables multiple datasets to use the tables generated from Power Query. The component and engine that provide such a feature is called a *dataflow*. Dataflows exist in Power BI as service-only components. The data for the tables is stored in Azure Data Lake Storage, as shown in Figure 1-7 (other storage options will be available in the future, such as an Azure SQL database). Dataflows enable the Power BI developer to separate the data transformation layer of the Power BI implementation from the rest of the model and, as a result, have a reusable solution.

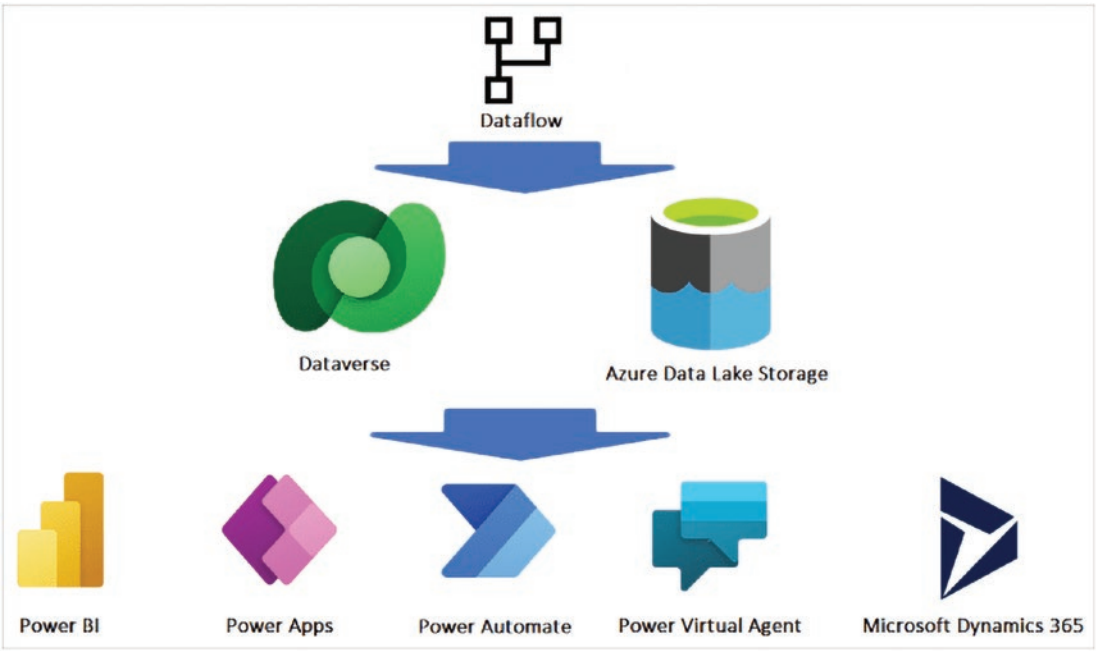


Figure 1-7. *Dataflow*

Datamarts

Datamarts simplify the creation of multiple Power BI components through one unified editor. Using a datamart, you can have the dataflow, a central repository of the data based on Azure SQL database, and the dataset through one object. Datamarts can be created through the Power BI Service. Datamarts are a recent addition to the Power BI components. Figure 1-8 shows what datamarts offer.

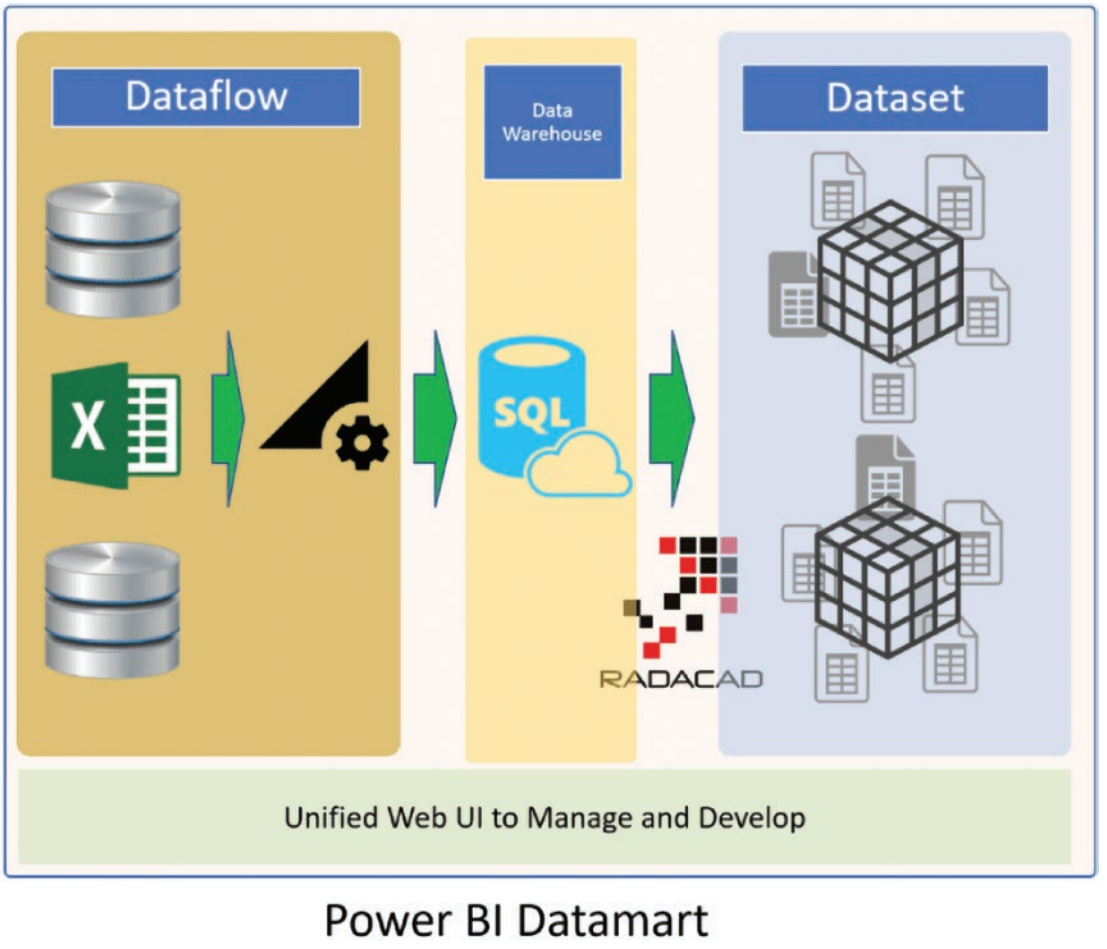


Figure 1-8. *The Power BI datamart*

The DAX Language

DAX stands for Data Analysis Expression. It is a language you use to write calculations in Power BI. This language is very powerful and gives you analytical power over your model. DAX is not just part of Power BI. It is also part of Excel Power Pivot, SQL Server Analysis Services Tabular Model, and Azure Analysis Services. DAX has a steep and long learning curve. Some consider DAX the hardest part of learning about Power BI. However, the most important part of learning about Power BI is its data modeling services.

Figure 1-9 shows the process of writing a calculation in DAX. Don't worry if you don't understand what this DAX expression does at this time.

```
1 InernetSales YTD =
2 TOTALYTD(
3         SUM('FactInternetSales'[InernetSales]),
4         'DimDate'[FullDateAlternateKey].[Date]
5     )
```

Figure 1-9. *DAX is the analytical language in Power BI*

The Power BI Service for Hosting Reports

The Power BI Desktop, Power Query, and DAX are all components used to create and develop reports. After developing reports, you need to publish them and share them with others. The Power BI Service, previewed in Figure 1-10, is the online service hosted on the Power BI website for hosting and sharing reports. You must have a Power BI account to work with the Power BI Service. There are a few options when choosing an account; you learn about them in full detail in the licensing chapter.

After publishing your reports to the website, they are accessible via the Power BI Service (or website) through `app.powerbi.com/`.

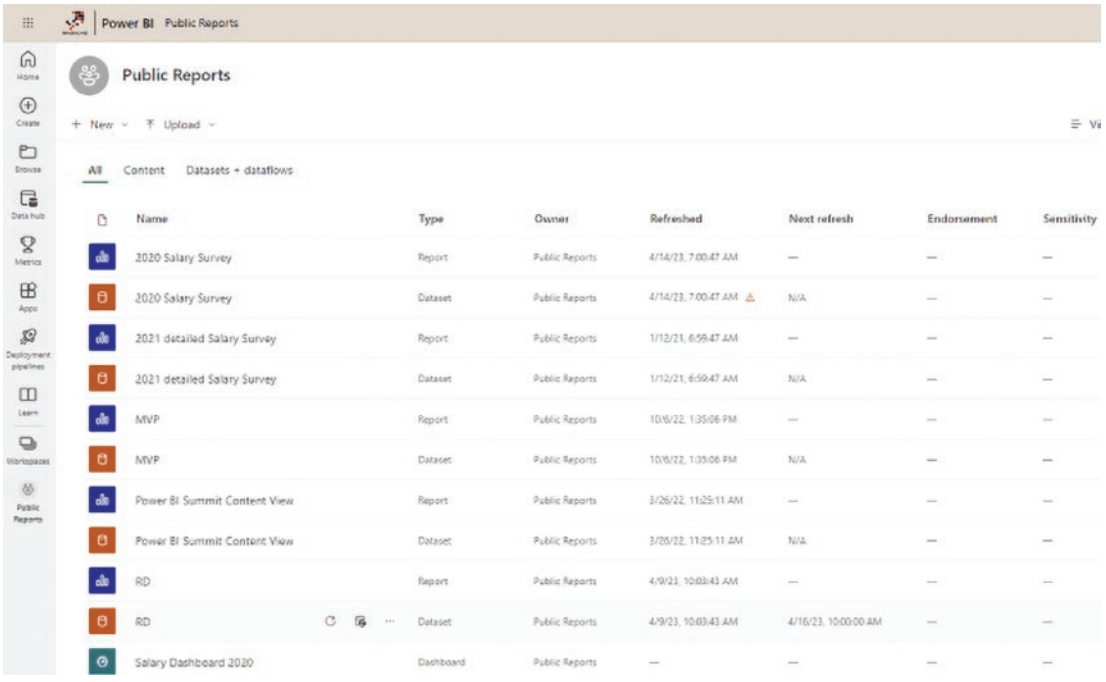


Figure 1-10. *The Power BI Service for hosting Power BI objects*

As shown in Figure 1-11, you can edit or author reports in the Power BI Service. However, not all the Power BI Desktop development functionality is available on the service.

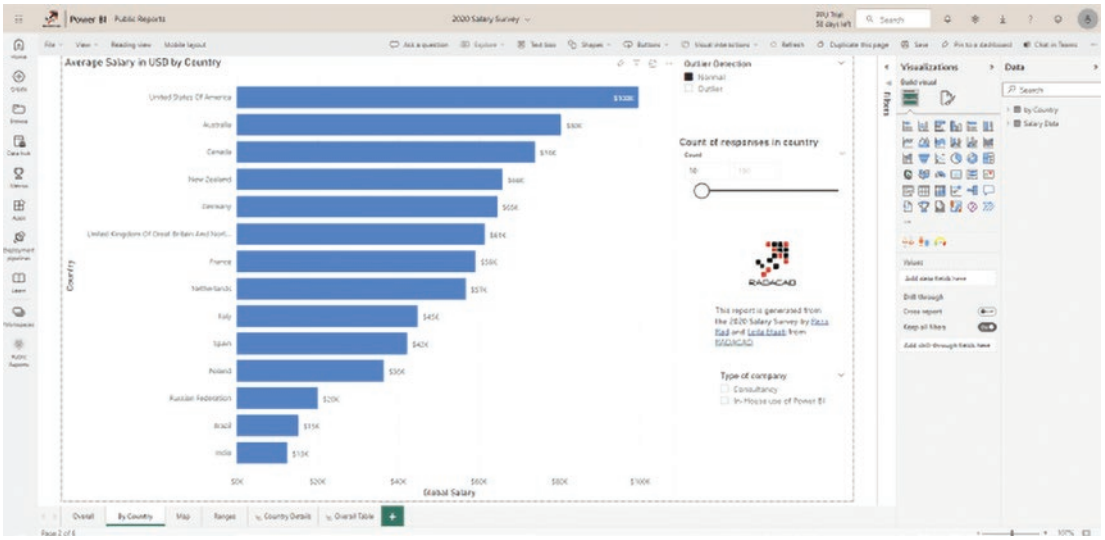


Figure 1-11. *Editing reports in the Power BI Service directly*

The Power BI Service can be accessed via different browsers if you use their recent versions. The Power BI Service works based on the HTML5 standard, which is supported in all new browsers—IE, Edge, Safari, Chrome, Firefox, and so on.

The Power BI Mobile App

Besides using browsers to connect to the Power BI Service and browse reports, there is another way to access reports interactively. The Power BI mobile app is available from the Microsoft Windows store, the Apple App Store, or the Google Play store. The app is free; you can easily download and install it. After logging in to the app, you can browse the Power BI reports from your mobile device. Mobile reports can be designed to be different from normal reports. Figure 1-12 shows a Power BI report opened on a mobile app.

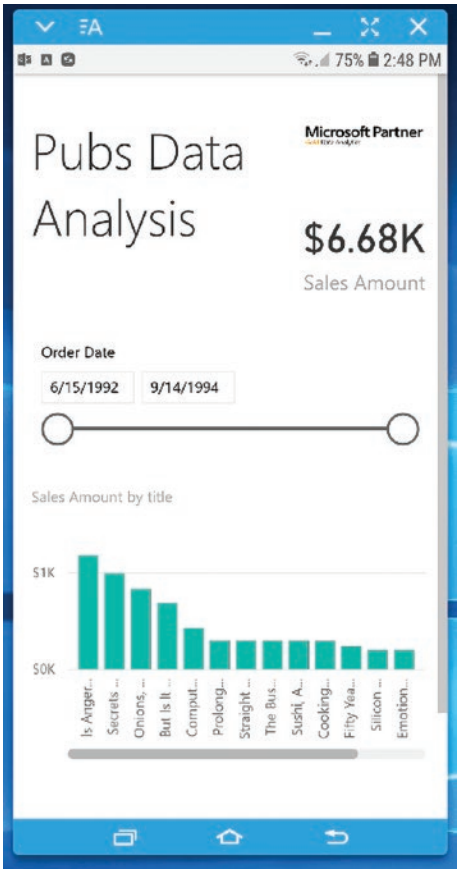


Figure 1-12. *The Power BI mobile app*

Having the report on the mobile app gives you some specific features, such as annotation and sharing the annotation with others.

The Power BI Report Server

The Power BI Report Server is an on-premises hosting spot for Power BI. Many businesses still prefer to keep their solutions and data on-premises. Power BI is a cloud-based technology. However, it has a version that works on-premises. This edition of Power BI keeps the report, data, and everything on-premises. Power BI on-premises leverages a component called the Power BI Report Server.

The Power BI Report Server, shown in Figure 1-13, is a specific version of SQL Server Reporting Services developed for hosting Power BI reports. Reports are shared with other users through this environment. There are a lot of features in the Power BI Service that have not yet been implemented on the report server. However, this solution gives users an on-premises experience of Power BI reports. There is a chapter in this book that explains the Power BI Report Server in detail, where you learn about installing, configuring, and using it.

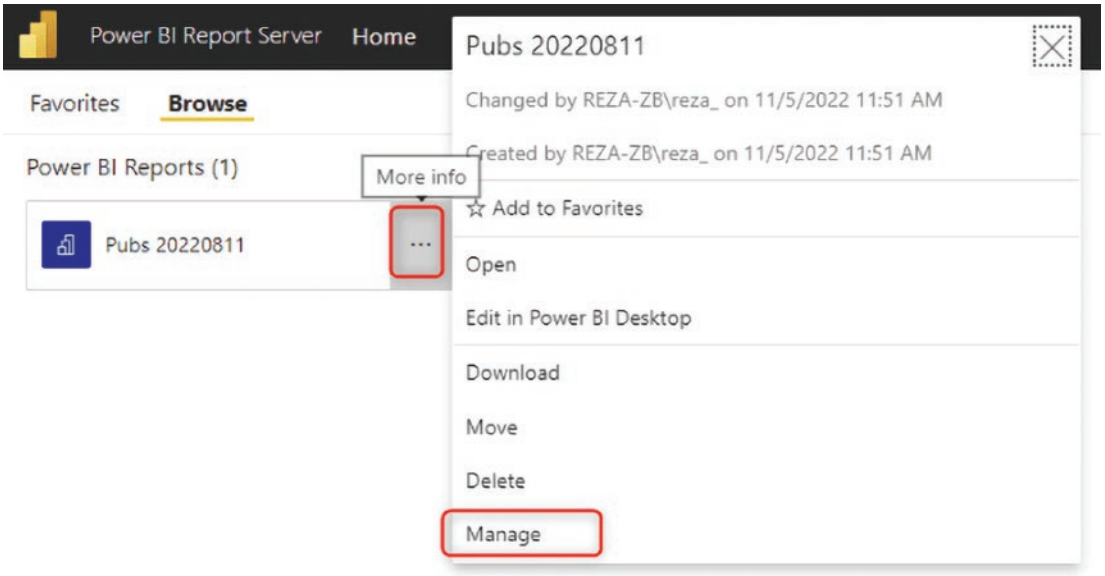


Figure 1-13. *The Power BI Report Server*

Paginated Reports

If you need to print your Power BI reports, it is better to design a different kind of report, called a Power BI Paginated Report (see Figure 1-14). Power BI Paginated Report has major differences from the normal Power BI Report; one of them is that the paginated report is not as interactive, as easy to use, or as easy to build as the Power BI Report. On the other hand, the paginated report can be designed for fine-printing details, whereas the normal Power BI report can't. Paginated reports cannot be developed by the Power BI Desktop. They have to be developed using a different tool, the Power BI Report Builder.

Contoso

Contoso Suites
123 Coffee Street
Buffalo,NY 98052
USA
Telephone 012345678
http://www.contososuites.com

Owl Wholesales
123 Violet Road , Phoenix, CO 85003 USA

Invoice CIV-000676 000007-1
30 November 2019
Payment terms: Net 45 days
Payment due 1/14/2020
$321,113.52

ITEM	DESCRIPTION	QUANTITY		SALES PRICE	DISCOUNT	AMOUNT
D011	Lens	2	Each	2	0	4.00
L0001	Mid-Range Speaker	35	Each	500	0	17,500.00
P0001	Acoustic Foam Panel	117	Each	37	0	4,329.00
D0003	Standard Speaker	23	Each	220	0	5,060.00
T0001	Speaker cable 10	65	Each	500	0	32,500.00
D0004	High End Speaker	12	Each	2000	0	24,000.00
T0004	Television M120 37" Silver	53	Each	350	0	18,550.00
T00002	Projector Television	23	Each	3750	0	86,250.00
T0005	Television HDTV X590 52" White	33	Each	2890	0	95,370.00
T0003	Surround Sound Receiver	56	Each	450	0	25,200.00
				SALES SUBTOTAL AMOUNT		4.00
				SALES TAX		12,350.52
				USD TOTAL		**321,113.52**

Sales invoice notes

METHODS OF PAYMENT

Electronic payment
Payment reference **US-009**
Sort code
Account No. **34567**

Check
Make check payable to **Contoso Suites**.
Write reference **US-009** on the check.

OTHER INFORMATION

Tax registration no. **1234123400**
Our reference **Karl Bystrom**

Figure 1-14. *A Power BI Paginated Report for printing*

The Power BI Report Builder

The Power BI Report Builder, previewed in Figure 1-15, is the tool that creates paginated reports. This tool can be installed on a Windows machine. This tool was inherited from the SQL Server Reporting Services Report Builder. Power BI Paginated Reports are descendants of Reporting Services Reports and have an RDL extension.

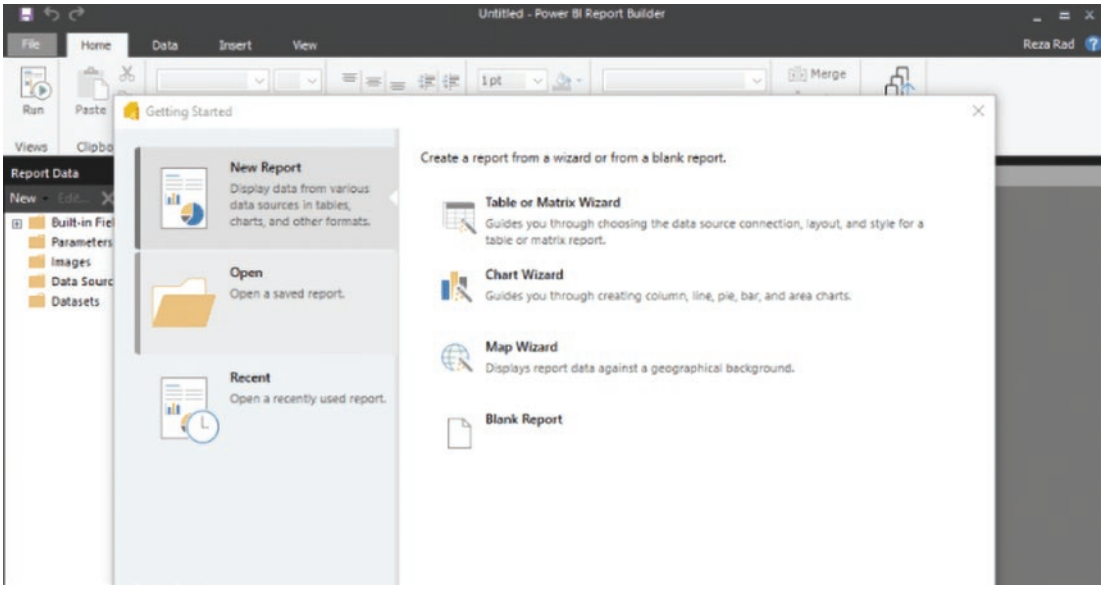


Figure 1-15. *The Power BI Report Builder*

Metrics and Scorecards

In every business, key performance indicators (KPIs) are measured often, and when compared to targets, you can tell if your business is on track or behind schedule. In the Power BI environment, the component used to create these KPIs is called *Metrics*. You can create Power BI Metrics with scorecards and get their values from a Power BI dataset. Rules and status identifiers can be defined on the scorecard to automatically track the KPI's current status, as demonstrated by Figure 1-16. Metrics and scorecards must be created in the Power BI Service, but they can be embedded into a Power BI Report using the Power BI Desktop. Power BI Metrics was previously called Power BI Goals.

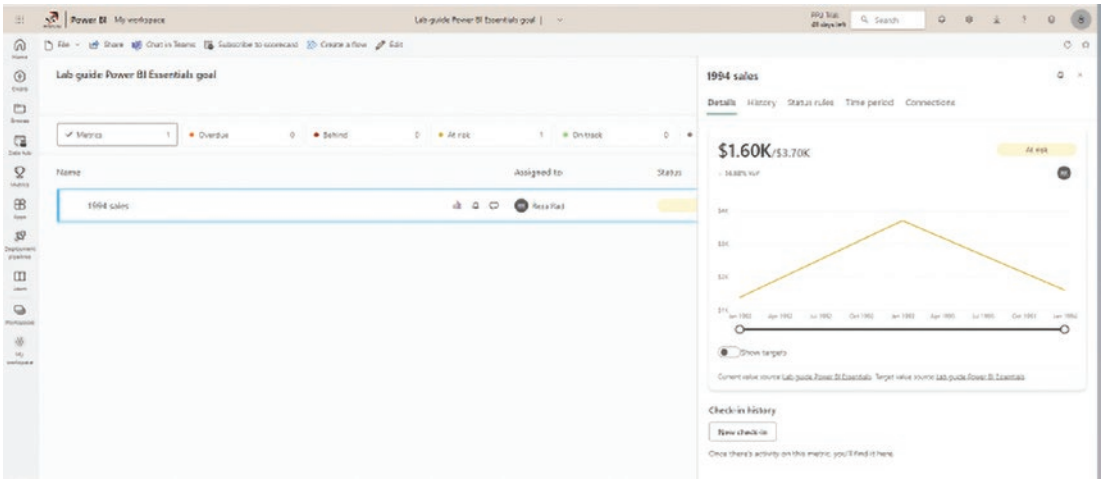


Figure 1-16. *Power BI Metrics to track KPIs*

The On-Premises Gateway

The on-premises gateway is another component of Power BI used to create a connection between the Power BI Service (a cloud-based technology) and the on-premises data source. This component is installed on an on-premises server and manages the connection for the Power BI Reports. This gateway is unnecessary if the data source is cloud-based (such as an Azure SQL Database).

Gateway installation has some options and configurations that need careful consideration. A chapter later in this book explains the details of installing a gateway, configuring it, adding data sources to it, and scheduling datasets to be refreshed through a gateway connection.

Workspaces

In the Power BI Service, objects are placed inside *workspaces*. Workspaces can contain multiple Power BI objects (reports, paginated reports, dashboards, datasets, dataflows, datamarts, and so on). Workspaces are the core of deployment and sharing in the Power BI Service environment. The access level can be defined on the workspace itself. Users can consume the content of a workspace through Power BI Apps. Figure 1-17 offers a glimpse of a workspace landing page.

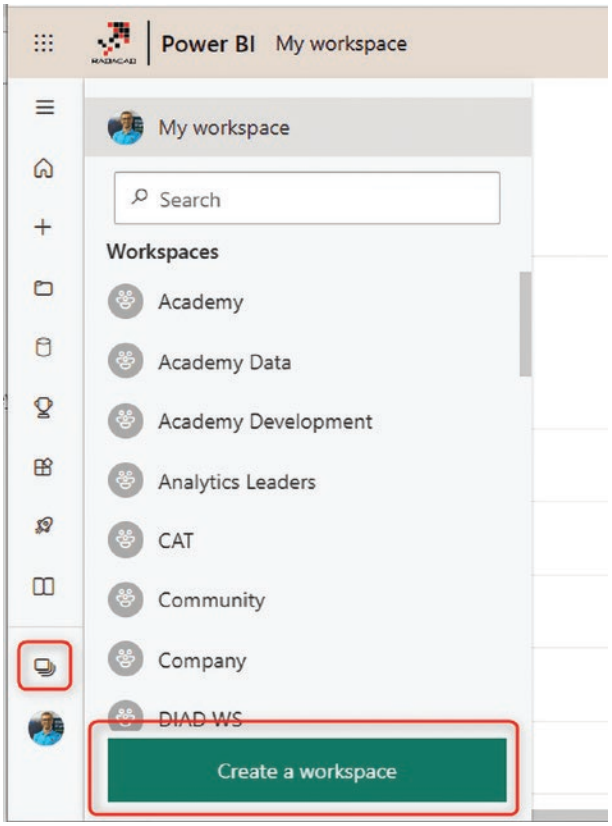


Figure 1-17. *Power BI workspaces*

Power BI App

Although the workspace's content can be shared directly with the users, creating a separate environment for the users is better. In the user environment, you can also have custom navigation to make it easier for the users to consume reports. Power BI has a component that offers these features, called *Power BI App*. Power BI App, depicted in Figure 1-18, differs from the Power BI mobile app. Power BI App shares content with the users.

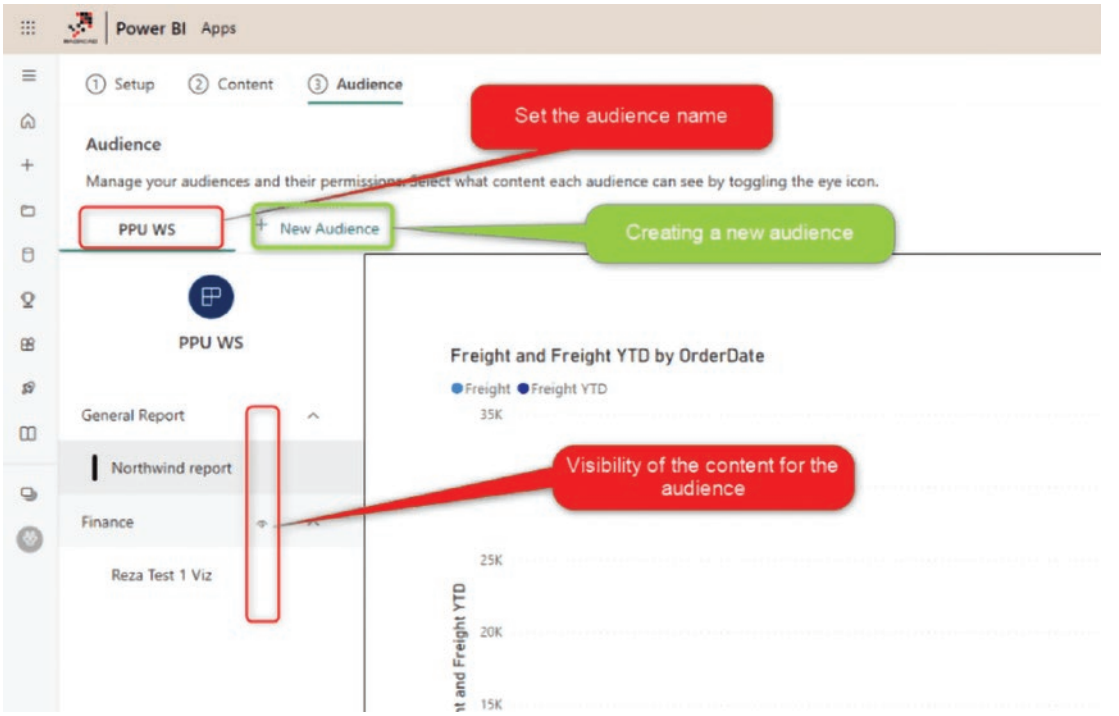


Figure 1-18. *Power BI App*

Power BI Premium

The Power BI licensing has a variety of options. The most comprehensive license for Power BI is called Power BI Premium. This license provides full functionality throughout all the components of Power BI. It also provides a dedicated capacity through the Power BI Service, which can be used to enhance the performance of reports and datasets. Power BI Premium comes in various nodes, each with different specifications for the dedicated capacity.

Power BI Embedded

If you need the Power BI Report to be embedded in a custom web application so the users of that custom application can use the Power BI Report, you can use Power BI Embedded. It allows you to embed Power BI reports inside a web application. Power BI Embedded works with a licensing plan. Web developers must embed the Power BI content inside the web application and manage the authentication and authorization processes.

Developer API or REST API

Power BI is not just a tool for report developers; it is also for application developers and programmers. There are few APIs in Power BI that developers can interact with. REST API is an API for .NET (C# or VB), and developers can use it through ASP.NET or mobile applications. This API lets you interact with the report in

the Power BI Service. You can embed a report into an application (see Figure 1-19), refresh the dataset in the Power BI Service, change the gateway configuration, and more. You learn about the REST API of Power BI later in this book.

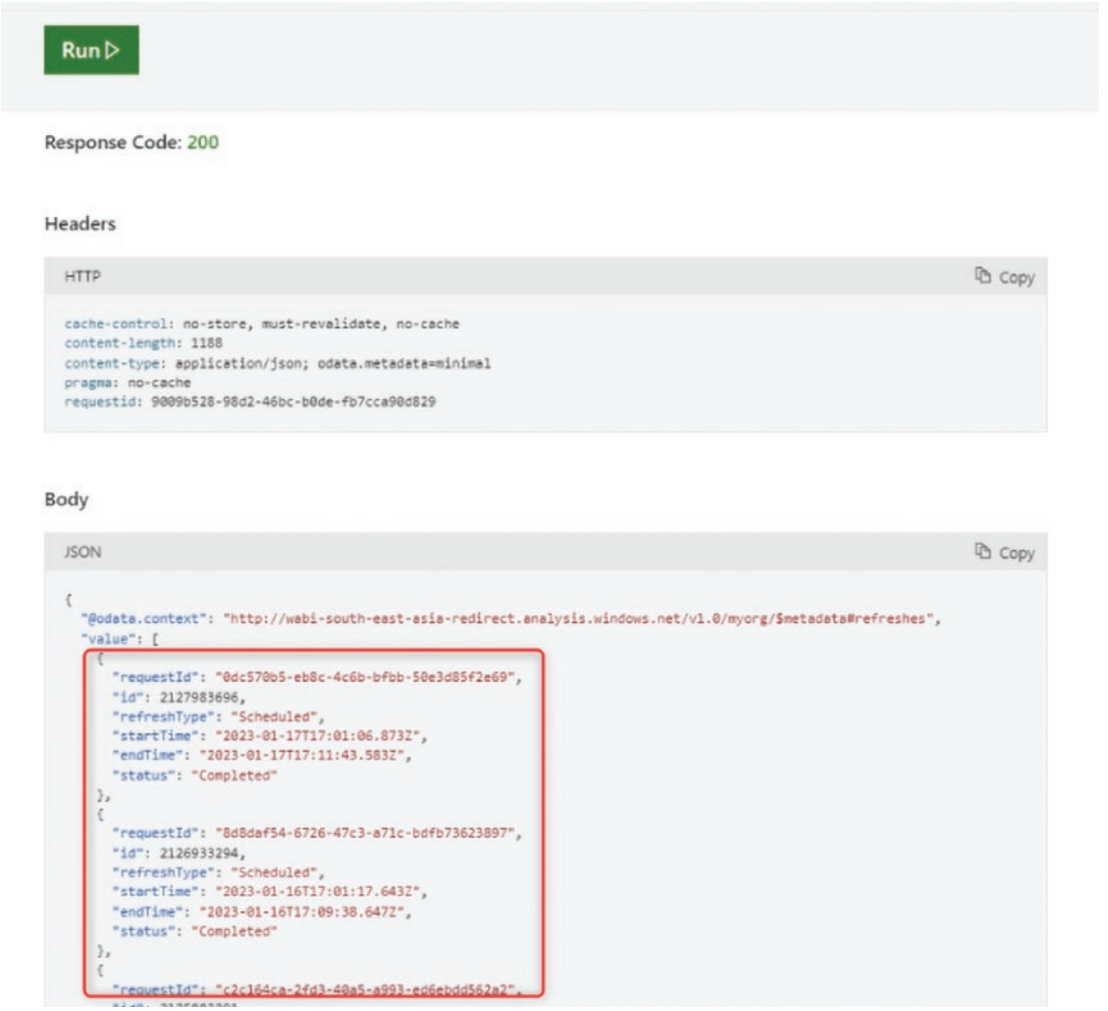


Figure 1-19. *Power BI Embedded*

There are also other SDKs that developers can work with, including an SDK for creating custom connectors in Visual Studio and another for creating custom visuals.

Summary

It is important to understand these Power BI components before developing solutions with Power BI. This chapter explained some of those concepts very briefly. There are still many terms that are not included here, and many of them are explained in future chapters. A Power BI architect, developer, and data analyst should be familiar with these components, so they can choose whether they need to use them.

CHAPTER 2

Tools and Preparation

This book discusses the features and components of Power BI. It is not a to-do book. However, there are examples throughout the book that you can follow if you have the tools you need. This chapter explains the tools you need to install and set up. The following topics are covered:

- Getting the Power BI Desktop
- Setting up multiple Power BI Service accounts
- Installing the Power BI mobile app
- Downloading dataset files

Using the Power BI Desktop

If you want to develop Power BI reports, you need the Power BI Desktop application. It comes in two ways—you can download it separately and install it on your Windows machine, or you can get it from Microsoft Store as an app. The question that I often get is, what is the difference and which one is better? This section addresses that question.

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_2

The Power BI Desktop: Report Authoring Tool

The Power BI Desktop, previewed in Figure 2-1, is a tool that every Power BI developer needs to build Power BI reports. This tool is free.

Figure 2-1. *Power BI in action*

When you open the Power BI Desktop the first time, a pop-up window asks you to log in, but that is not mandatory. You can use the Power BI Desktop without an account, a username, or login credentials.

The Power BI Desktop from the Microsoft Store

One of the ways to download the Power BI Desktop is to get it from the Microsoft Store. Again, that doesn't mean you need to pay for it. All you need is a Microsoft account (which can be live account) to access apps from the store. The Power BI Desktop app is free, as you can see in Figure 2-2.

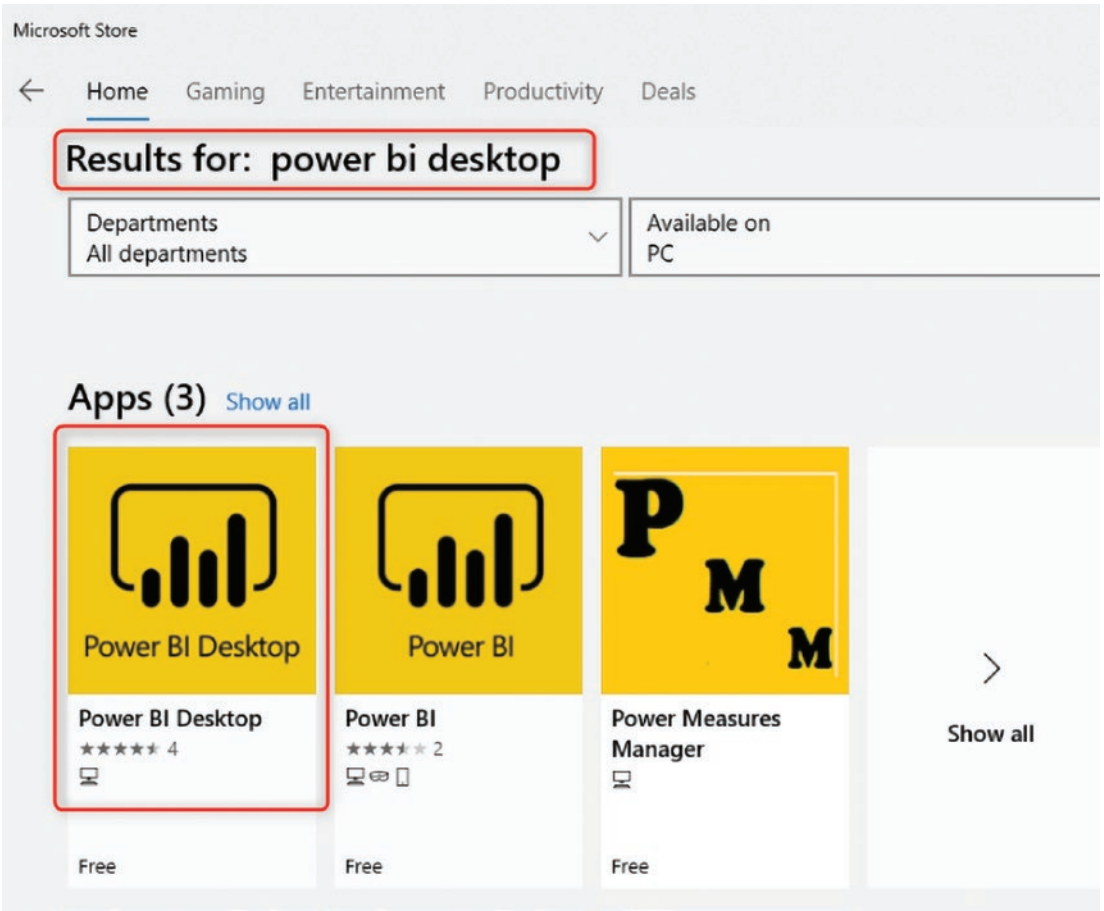


Figure 2-2. *Finding the Power BI Desktop in the app store*

The Power BI Desktop can be searched in the Microsoft Store and easily downloaded. You can use this option if you have the Windows operating system on your machine.

The Power BI Desktop as a Separate Download Link

Another way to get the Power BI Desktop is to download it from the following link: `aka.ms/pbiSingleInstaller`. Clicking this link will bring up the page shown in Figure 2-3.

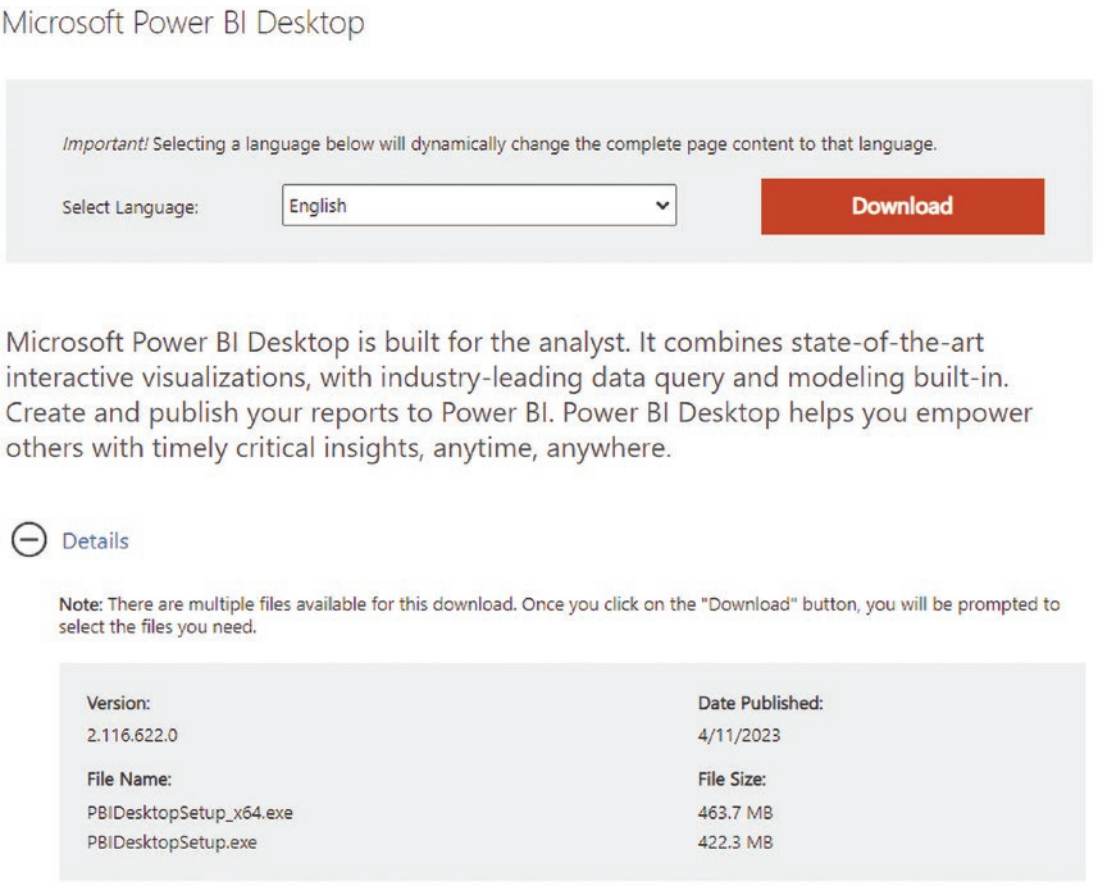


Figure 2-3. *Downloading Power BI*

You can simply download the file and install it on your machine.

Auto-Update from the Microsoft Store App

Now that you are aware of the two methods to access Power BI Desktop, let's look at the differences. The first difference is that the Microsoft Store version of Power BI Desktop updates automatically. As soon as a new update is available from the Power BI Team, the application automatically updates itself to the latest version. You don't need to do anything to keep it up-to-date.

The Power BI Team updates the Power BI Desktop tool every month, and keeping that manually up to date is a big hassle. If you want to always be on the cutting edge of updates and enjoy the fascinating new features announced every month, consider getting the Power BI Desktop from the Microsoft Store.

Flexibility on Installation: The Download Link

One of the problems of the Microsoft Store version of Power BI Desktop is that it is not available for all versions of Windows. For some versions, you can only use the download link. Another scenario is that for some reason, someone might want to use an older version of Power BI Desktop for their report development work. Using the Microsoft Store app won't allow this, because it automatically updates the version. You have more flexibility on what you want to have installed and where using the download link.

Development Work

In terms of building the report, there is no difference between the two versions (if both versions are up to date). You can build the same report using either method. There is no difference in the Power BI file (`*.PBIX` file) that is created. You can use either version to open a file created with either of these, as long as they are both up to date. In the examples throughout this book, there aren't any differences between the two types of installs.

Setting Up Power BI Accounts

To go through the examples in this book, you may also need Power BI accounts. You need more than one account to test some of the functions, such as sharing and security. Power BI accounts can be free or Pro. The Free account does not support sharing (it should be part of paid subscription, which is Pro or Premium only).

To run the examples in this book, you can create a free account and apply for a trial Pro account. Trial Pro accounts last for 60 days and don't cost you anything. The process of creating a Power BI account is explained next.

Go to the Power BI website, `powerbi.microsoft.com`, and click Try Free, as shown in Figure 2-4. (Use this option if you don't already have a Power BI account. If you have an existing account, click Sign In.)

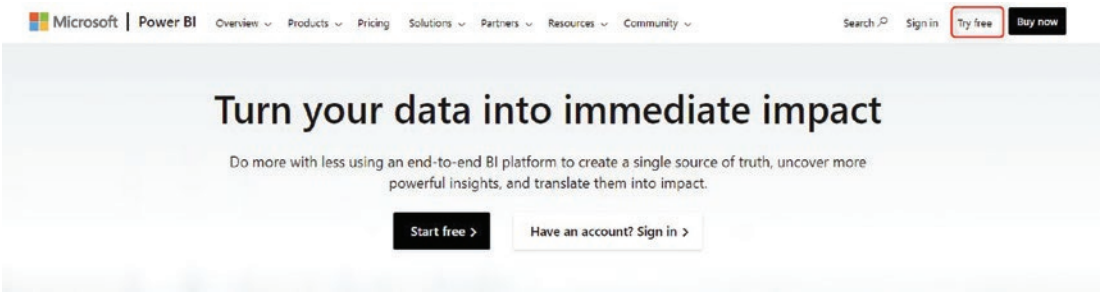


Figure 2-4. *Try Power BI for free*

In the next window, enter your email address, as shown in Figure 2-5.

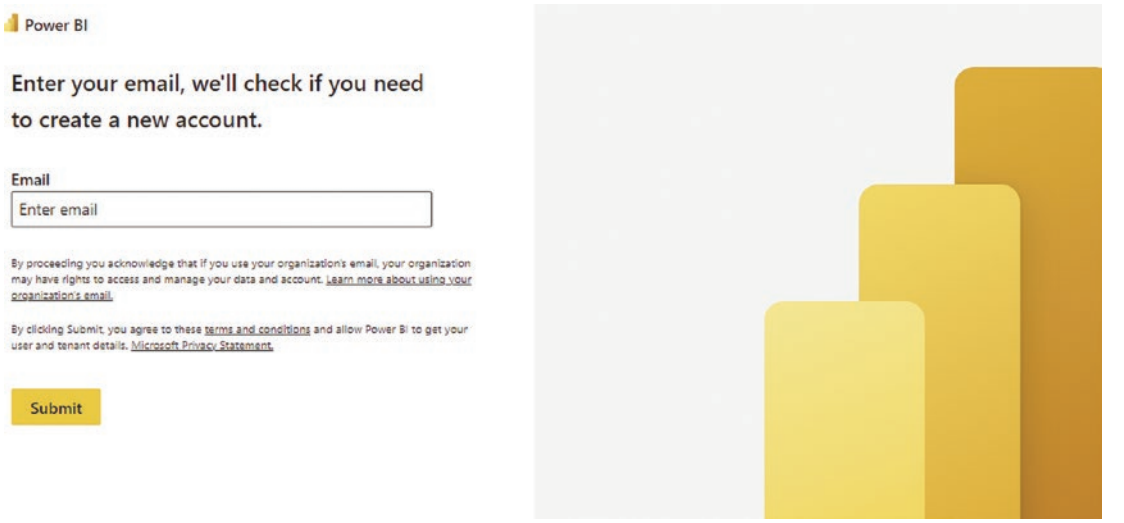


Figure 2-5. *Registering login information with Power BI*

The email used in this step should *not* be a public domain email address. You cannot use email domains such as Gmail, Yahoo, or Live. You can use your organization, company, or university account for this. After entering the email, click Sign Up.

After clicking Sign Up, you will receive an email from support@email.microsoftonline.com. Make sure that your company firewall doesn't block this email as a sender. This email will contain a verification code, which you need to enter in the next step.

After this step, you will get a message that your account has been set up correctly. You can then go to the Power BI service at `powerbi.microsoft.com` and log in.

Creating Power BI Accounts from Office 365

Another way to assign Power BI licensing is through the Office 365 Portal. If you have administrator access to Office 365, you can log in to `portal.office.com` and then find the user in the list of users. Then you assign Power BI Free for the user (or Power BI Pro if you want). This process is shown in Figure 2-6.

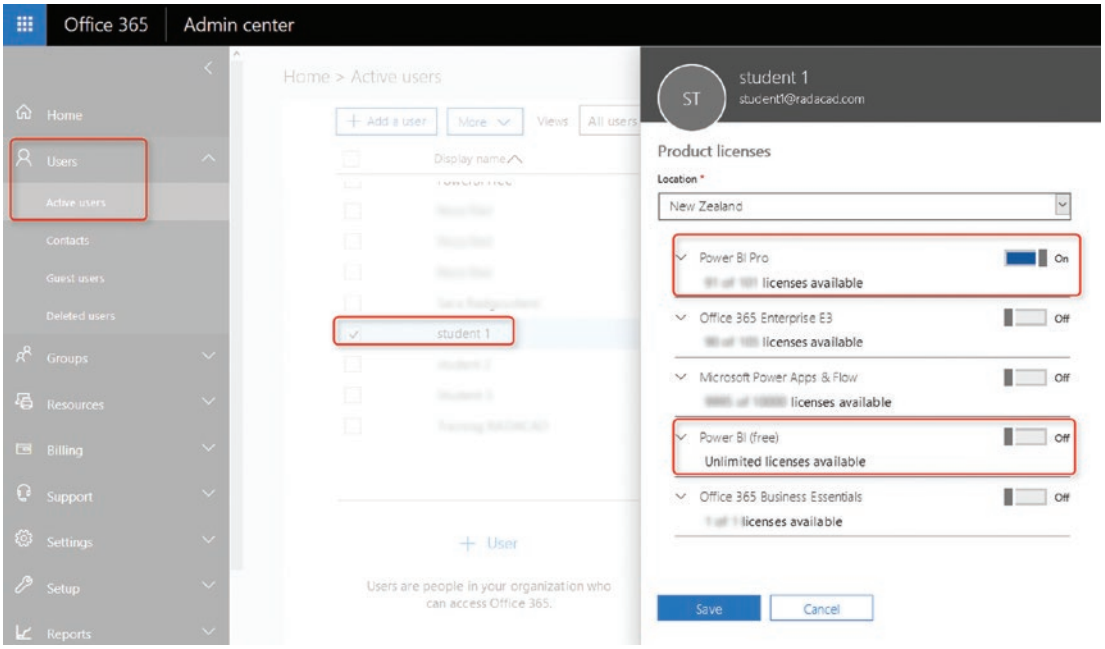


Figure 2-6. *Assigning Power BI licensing through the Office 365 Portal*

Is Your Account Pro, Free, or PPU?

One way to check your account designation—Pro, Free, or Premium Per User—is by using the Power BI Service and clicking the Profile icon, as shown in Figure 2–7.

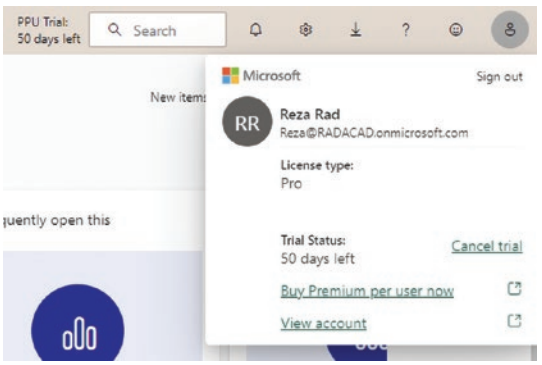


Figure 2-7. *Checking your account tier designation*

In the window, you can identify your account type—Free, Pro Trial, or Pro. If your account is free, you can choose to try Pro for free (the trial lasts 60 days) or upgrade it to Pro.

Installing Power BI Mobile App

Although you don't need Power BI Mobile App for this book, you may still want to install it. With Windows, Mac, and Android devices, you can simply search for the app and install it. After installing the app, you can log in using the same account you created in the previous step.

Power BI Report Server

You learn how to install the Power BI Report Server in a later chapter of this book.

On-Premises Gateway

You learn how to install an on-premises gateway in a later chapter of this book.

Microsoft SQL Server Database and Management Tools

To run through some of the examples in this book, you need to install Microsoft SQL Server and services such as SQL Server Analysis Services Tabular. However, they are only useful with a few chapters, such as the DirectQuery and Live Connection chapters. Explaining how to install SQL Server is outside this book's scope. I suggest that if you have it on your machine, use it; otherwise, skip that example.

The Power BI Helper

The last chapter of this book explains some of the useful features that you can use by installing Power BI Helper. Power BI Helper is a free tool that can be downloaded from `powerbihelper.org/` and installed through an installation wizard.

Downloading Dataset Files

All the sample files, including datasets and sample Power BI (`*.pbix`) files, are available for download at https://github.com/Apress/pro-power-BI-architecture-2nd.

Summary

This chapter explained the tools and setup you need to run some of the code examples in this book. In the next chapter, you learn about connection types in Power BI.

CHAPTER 3

Import Data or Scheduled Refresh

Power BI supports more than one type of connection. Each connection type has its pros and cons. This chapter covers everything about the Import Data or Scheduled Refresh type of connection. You learn how Power BI stores data in the xVelocity in-memory engine, and the pros and cons of using this method.

The Connection Type Is Not the Data Source Type

The connection type doesn't refer to the data source type. Power BI supports more than 160 data source types. The connection type is the way that the data source reaches the connection. One data source can support multiple types of connections. For example, you can connect to a SQL Server database with Import Data or DirectQuery. It is the same data source, regardless of how to access it.

Every connection type has advantages and disadvantages. Some connection types are suitable for smaller datasets, others for bigger datasets, yet others provide better flexibility, and so on. You cannot change the connection type in the middle of your Power BI development cycle. You should decide on the connection type at the beginning of the process. Otherwise, you could end up with a lot of re-work.

This book covers the following connection types:

- Import Data
- DirectQuery
- Live Connection
- Composite Mode

Import Data or Scheduled Refresh

The Import Data connection type imports the entire dataset into memory. This is the memory of the machine that hosts the Power BI dataset. If you have a Power BI dataset opened on the Power BI Desktop, it will be the memory of the machine where Power BI Desktop is running. When you publish your Power BI file to the Power BI website, it will be the memory of that machine in the cloud.

Loading data into memory also means something more; data needs to be refreshed. You need to schedule data updates if you are using this method. Otherwise, the data will become obsolete. That is why this technique is called Import Data or Scheduled Refresh.

A Closer Look at Import Data

To take a closer look at import data, create a report with this type of connection. Open the Power BI Desktop and click Get Data. Select Excel, as shown in Figure 3-1.

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_3

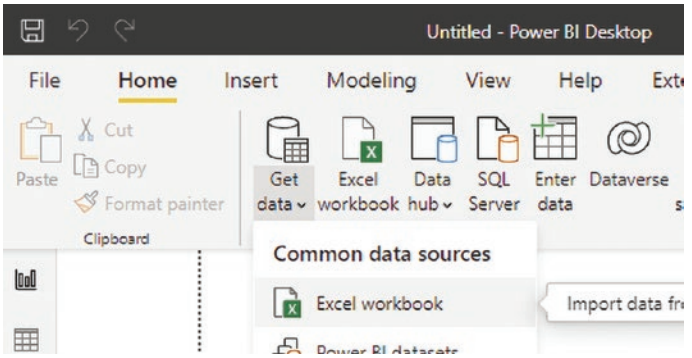


Figure 3-1. *Selecting Excel as the source for data import*

Get the data from the Excel workbook. Select the Excel file called `Pubs Descriptions.xlsx`, select all the tables from the list shown in the Navigator, and then click Load. See Figure 3-2.

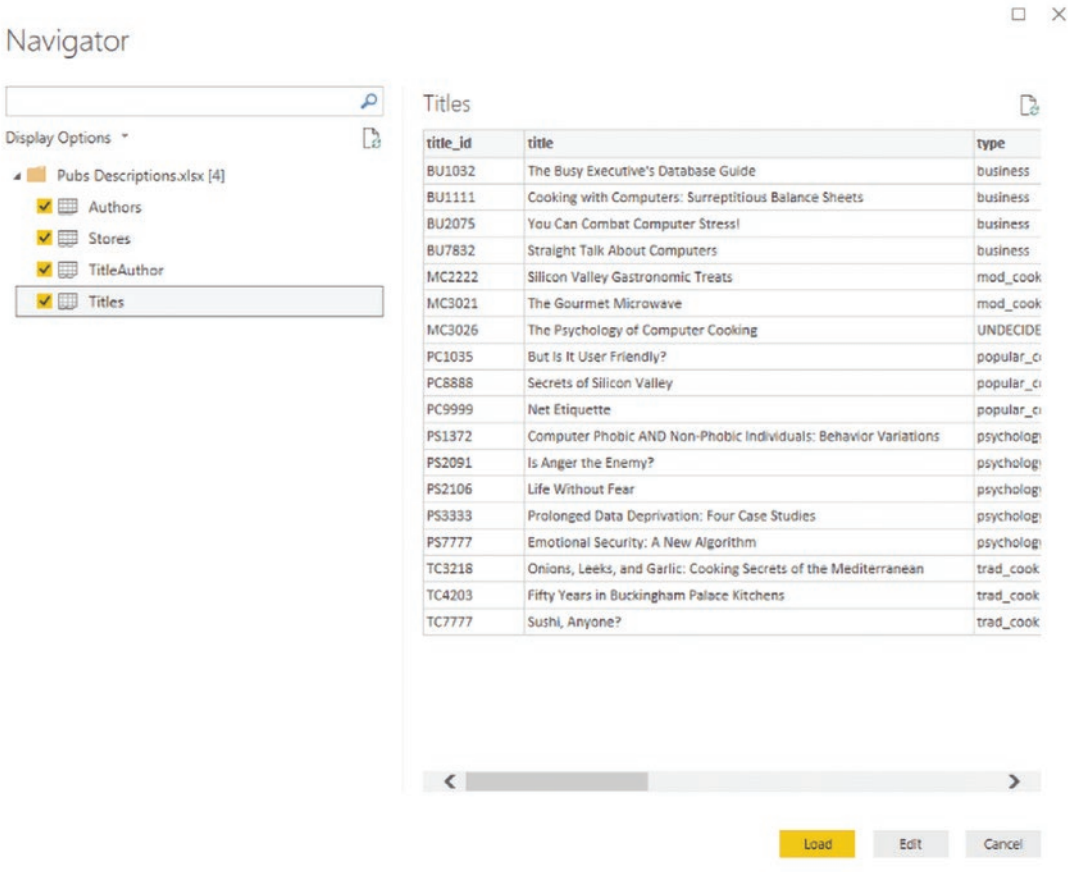


Figure 3-2. *The Navigator window*

Loading Data to the Model

You will see that Power BI loads the data into the model, as shown in Figure 3-3.

Figure 3-3. *Loading data to into the Power BI model*

As you probably guessed already, this is Import Data Connection Type. The Excel data source only supports the Import Data Connection type. If multiple connection types are supported, you can choose one when you connect to that source. For example, if you are connecting to SQL Server database, you will see, as in Figure 3-4, an option to choose Import Data.

Figure 3-4. *Getting data from SQL Server database*

How Power BI Stores Data in Memory

The first question that might come to your mind is how big the memory needs to be. What if you have hundreds of millions of records? Or many gigabytes of data? How does Power BI store data in memory? To answer these questions, you need to first learn a little bit about the in-memory engine called *xVelocity*.

Power BI, SQL Server Analysis Services Tabular, and Power Pivot are three technologies that use the same engine for storing data—the in-memory engine called xVelocity. xVelocity stores data in memory. However, it applies a few compression steps to the data before storing it. The data stored in memory will not be the same size as your data source in the majority of the cases. Data is compressed significantly. However, the compression is not always at the same rate. Depending on many factors, it might behave differently.

How xVelocity Compresses Data

To really understand the concept of data compression in xVelocity, you would need to read an entire book. However, in this section, I'm going to briefly explain what happens when the xVelocity engine works on compressing the data.

Traditional database technologies stored data on disk because the dataset was too large. Consider Table 3-1.

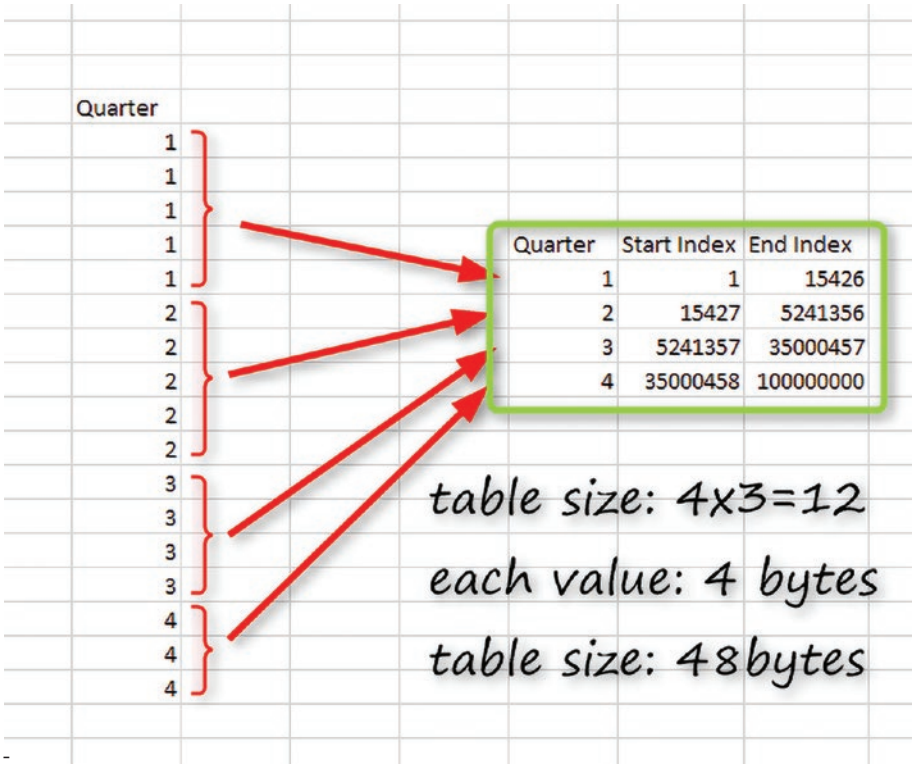
Table 3-1. *Traditional Way of Storing Data*

4 bytes	4 bytes	4 bytes	8 bytes	3 bytes	16 bytes	16 bytes		Total each row: 100 bytes
Order ID	Quarter	Quantity	Sales Amount	Oder Date	uniqueidentifier	text	other columns	
1	1	5	43254	7/01/2017				
2	1	2	123423	8/01/2017				
3	1	67	234	9/01/2017				Number of rows: 100 Million
4	1	1	523	10/01/2017				
5	1	7	132	11/01/2017				Total space needed: 100M * 100bytes= 10GB
6	1	34	675	12/01/2017				
7	1	3	79678	13/01/2017				
8	2	8	90780	10/05/2017				
9	2	45	89	11/05/2017				
10	2	9	868	12/05/2017				

As you can see in Table 3-1, every column consumes some space depending on the data type of the column. The whole row ends up with 100 bytes. Then for 100 million rows in the table, you need ten gigabytes of space on the disk. That is why the traditional technologies stored their data on the disk.

xVelocity uses column-store technology combined with other performance-tuned compression algorithms. In simple words, xVelocity stores data column by column. It first sorts every column by its values. Then it creates a mapping table (see Table 3-2) for it with indexes at the beginning and end of each section (this is also run-length encoding [RLE] compression).

Table 3-2. *xVelocity Compression and Mapping Table*



The whole point of this compression is that when you have a huge column, a lot of values are repetitive. For the Quarter column in Table 3-2, you can have only four unique values. So instead of storing that 100 million times, which takes about 400MB of space (4 bytes for every value, multiplied by 100 million rows), you can store unique values and their start and end indexes.

The mapping table shown in Table 3-2 image is a table of three columns and four rows mean 12 values. Even considering four bytes for each; this table would end up being 48 bytes. This is roughly how the compression engine works in xVelocity.

Compression has other levels as well, and it even compresses data more than this little bit. But this mapping table is the core of compression. The size of data is probably not the same as the data in your source database or file. You might have an Excel file that's 800MB, and when you load it into Power BI, your model becomes 8MB. The compression obviously depends on many things—cardinality of the data, number of unique values in each column, and so on.

■ **Important** If you have a column with a lot of unique values, the compression engine of Power BI suffers, and Power BI memory consumption will be huge. Examples are ID of the fact tables (not ID of dimensions that used in the fact table as a foreign key), or some created or update DateTime columns that even sometimes have millisecond information.

Where Data Is Stored in Memory

Power BI models are always loaded into the Analysis Services engine. Even if you don't have Analysis Services installed, it will be in your system when you use Power BI with the Import Data connection type.

To check this feature, go to the Task Manager of the machine that has a Power BI Desktop with Import Data connection mode opened. You'll find that the Microsoft SQL Server Analysis Services engine is running, as shown in Figure 3-5. This is where your data is stored. Analysis Services keeps it in memory.

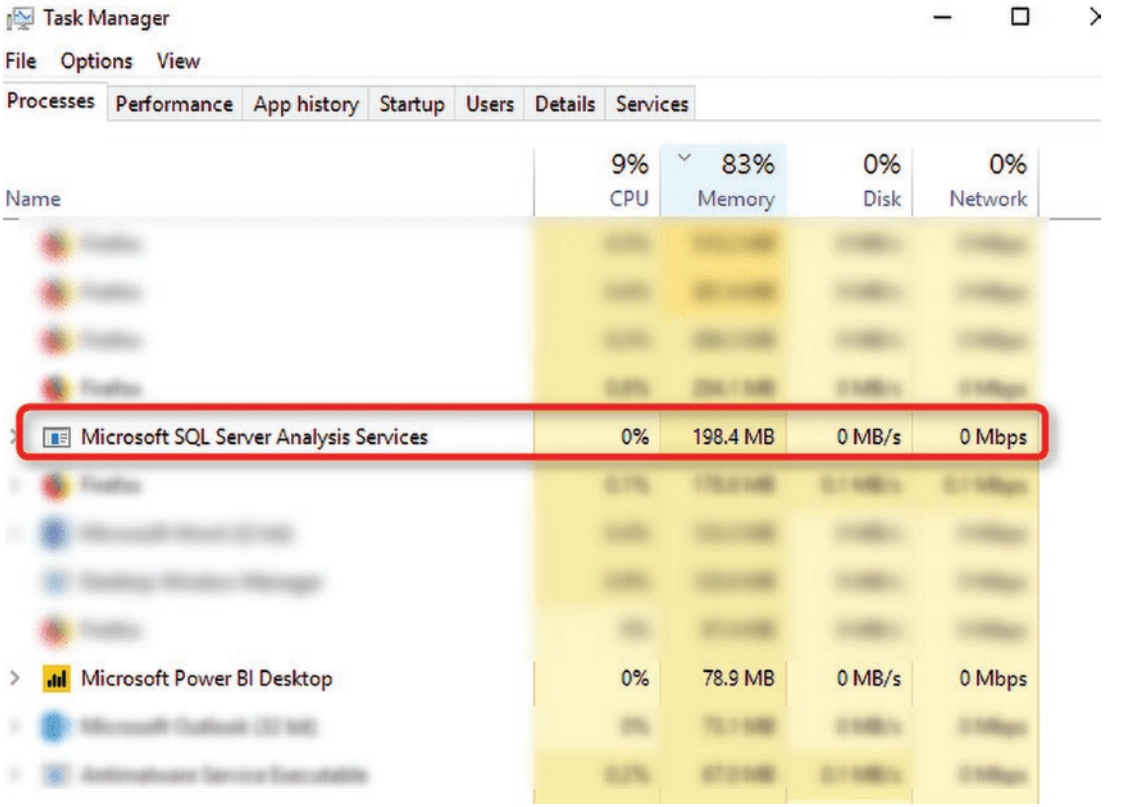


Figure 3-5. *SQL Server Analysis Services stores the data into memory*

When you save and close your Power BI file, that data will be persisted in a `*.pbix` file. The next time you open the file, data will be loaded again into Analysis Services' in-memory engine.

How About Power BI Service?

In the Power BI Service, you can also see how much memory every dataset consumes. Just click the Setting icon, and then choose Manage Personal Storage, as shown in Figure 3-6.

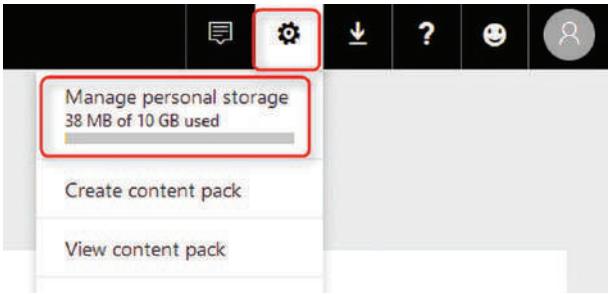


Figure 3-6. *Managing personal storage from the Power BI Service*

Figure 3-7 shows all the datasets and their sizes.

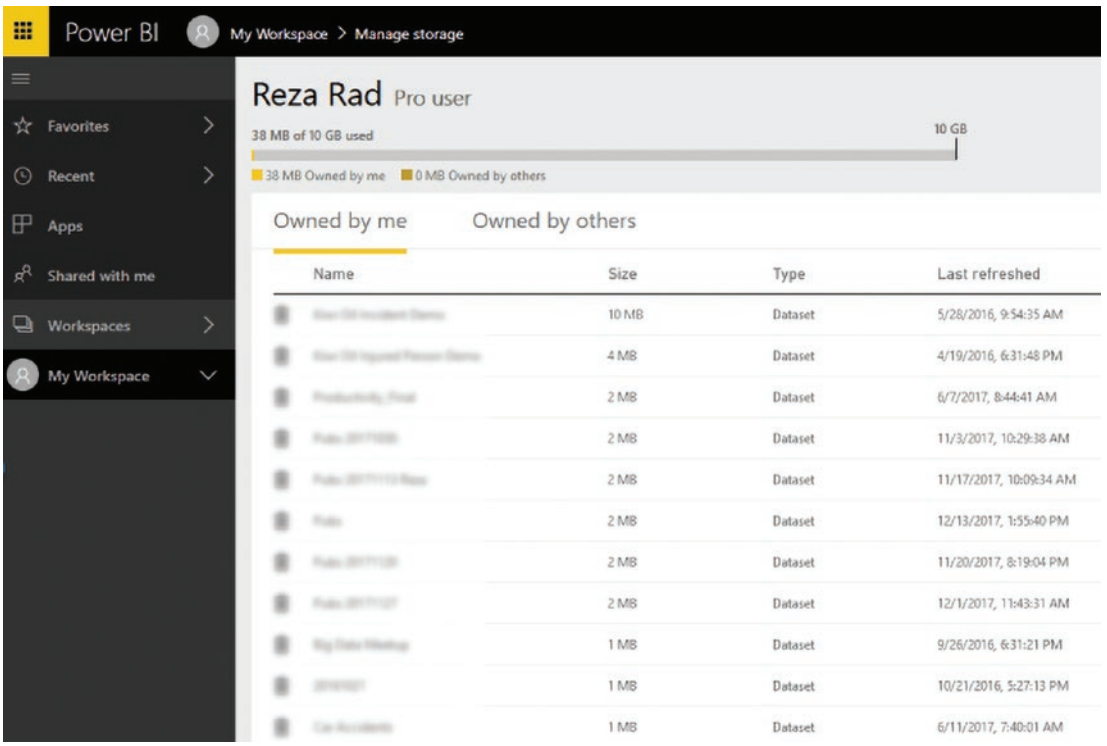


Figure 3-7. *Checking the dataset size in the Power BI Service*

Is There a Limitation on Size?

If you are developing Power BI files and then publishing them to the service, there is a limitation on the size of the files. Because the Power BI Service is a cloud-based shared hosting service for Power BI datasets, it needs to limit access so that everyone can use the service with a reasonable response time.

At the time of writing this chapter, the limit for Power BI's files is 1GB per model (or dataset in other words) if you use a Power BI Pro or Free account. You will have 10GB of space in your account, but every file cannot be more than 1GB. If you load your data into Power BI, and the file size ends up being more than 1GB, you need to think about another connection type.

■ **Important** Power BI Premium licensing offers dedicated capacity in Power BI Service. With Power BI Premium, you can have much more extensive datasets. At the time of writing this chapter, the dataset max for Power BI Premium models is 400GB. There is a per-user option for premium called Premium Per User (PPU), which gives you the dataset size of 100GB. To learn more about the licensing options in Power BI, read the chapter related to that subject that appears later in this book.

Combining Data Sources; Power Query Unleashed

You should know how the Import Data connection type works. This section looks at scenarios that this type of connection is suitable for. One of the advantages of this type of connection is the ability to combine any type of data source. Part of the data can come from Excel, and another part from SQL Server database or from a web page. With the Import Data connection type, Power Query is fully functional.

To see how this works in an example, try this. In the same Power BI file that you used earlier (which contains data from the `Pubs Descriptions.xlsx` file), click Get Data and choose from Text/CSV, as shown in Figure 3-8.

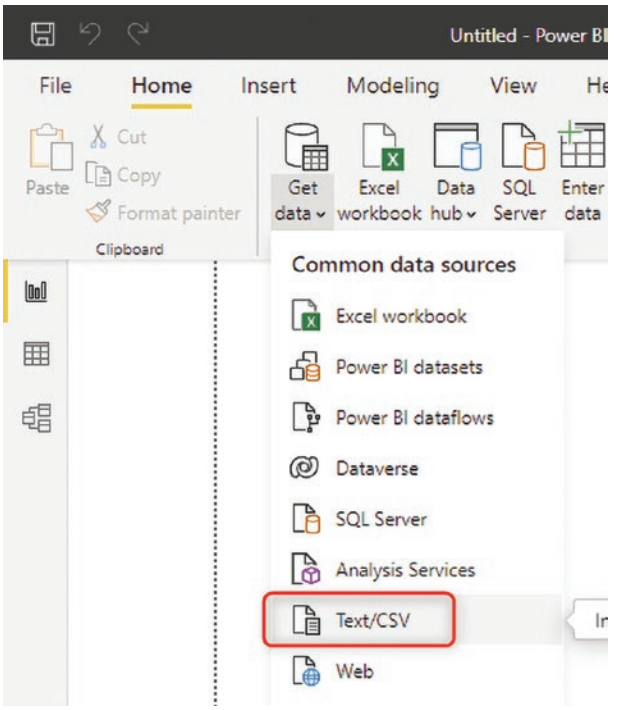


Figure 3-8. *Getting data from CSV*

Now select the `Pubs Transactions.csv` file and then click Load. Then click the Model tab. The Model tab is the bottom option on the left side panel. As shown in Figure 3-9, you will see that the tables from both data sources are connected to each other.

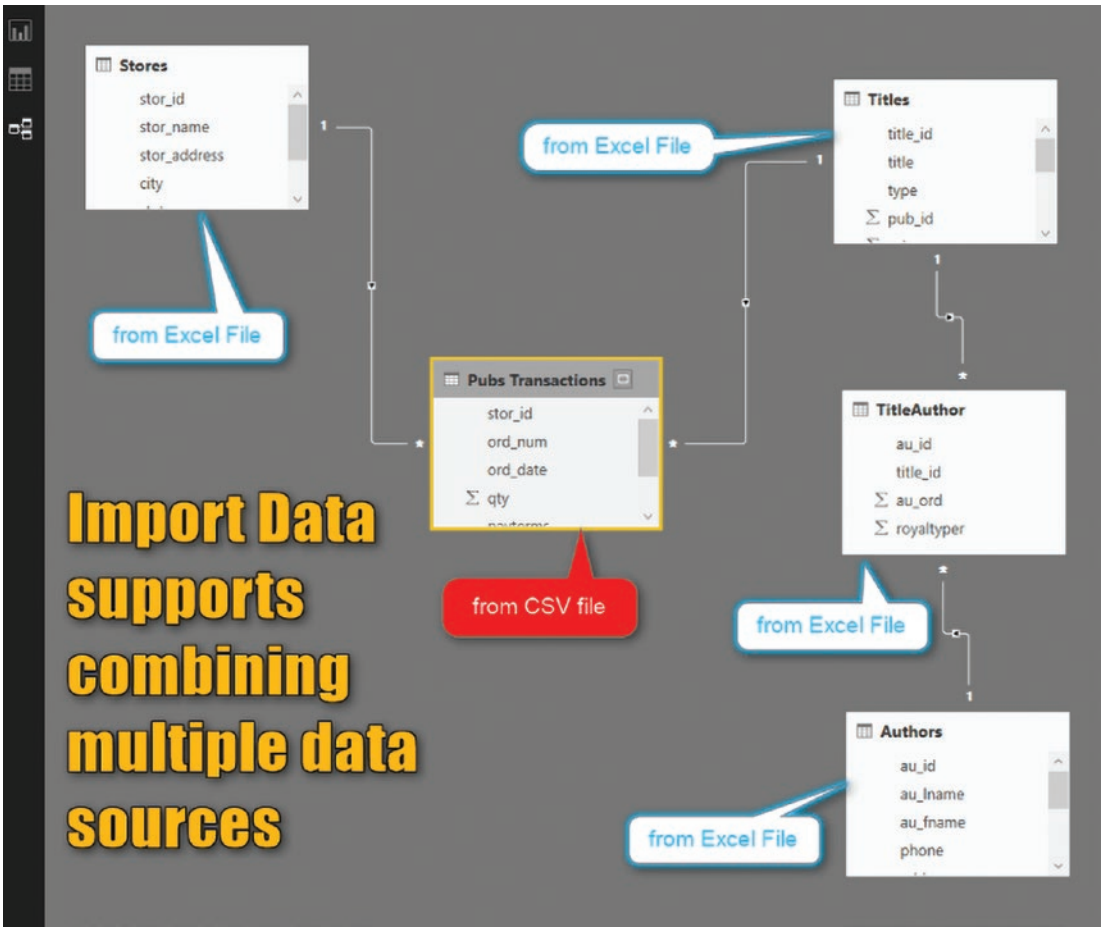


Figure 3-9. *Model diagram; Import Data supports combining multiple data sources*

One of the main benefits of Import Data is that you can bring in data from any data source and combine it with other sources. You can also leverage a fully functional Power Query with this connection type. You can do many data transformations and calculations with Power Query.

To see the Power Query transformations, click Transform Data, as shown in Figure 3-10.

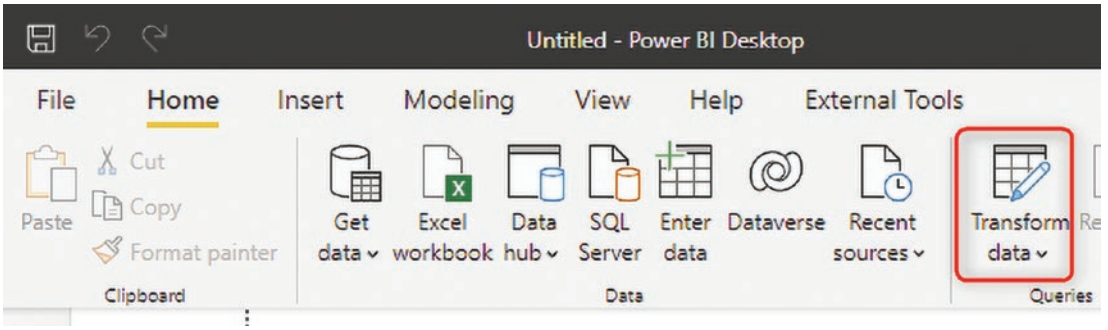


Figure 3-10. *Transforming data*

You can then see the Power Query Editor (see Figure 3-11) with many built-in transformations. To learn more about Power Query, I recommend checking out these two books that I wrote on this subject:

- *Getting started with Power Query in Power BI and Excel* (RADACAD, 2021); www.amazon.com/Getting-started-Power-Query-Excel-ebook/dp/B09DSPNZJH
- *Mastering Power Query in Power BI and Excel* (RADACAD, 2021); www.amazon.com/gp/product/B09DTLC4S2

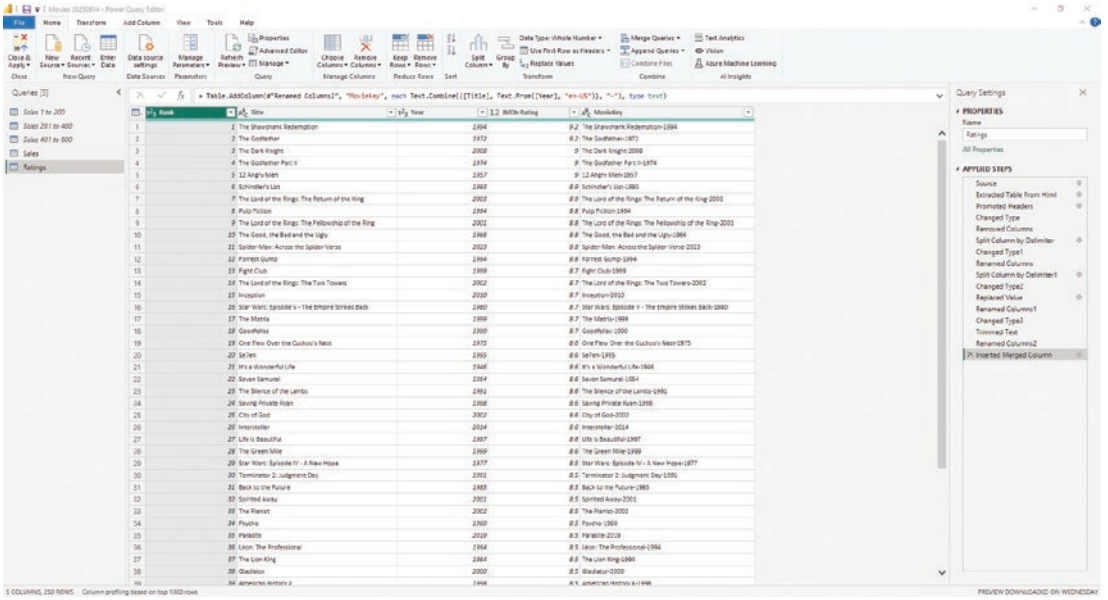


Figure 3-11. *Power Query Editor*

DAX: A Powerful Analytical Expression Language

Another powerful feature of Power BI that you have access to when using Import Data is DAX. DAX is the language that you can leverage to do analytical calculations. Calculations such as year to date, a rolling average of 12 months, and many others become super-efficient in DAX. DAX is supported by all xVelocity technologies of Microsoft (Power BI, Power Pivot, and SQL Server Analysis Services Tabular).

You can write DAX calculations in Import Data mode, such as quantity year to date, as shown in Figure 3-12.

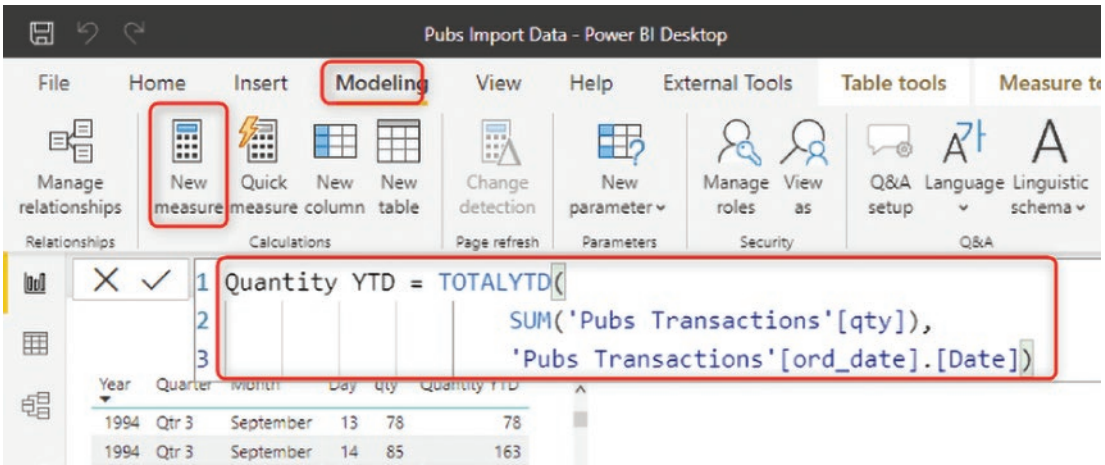
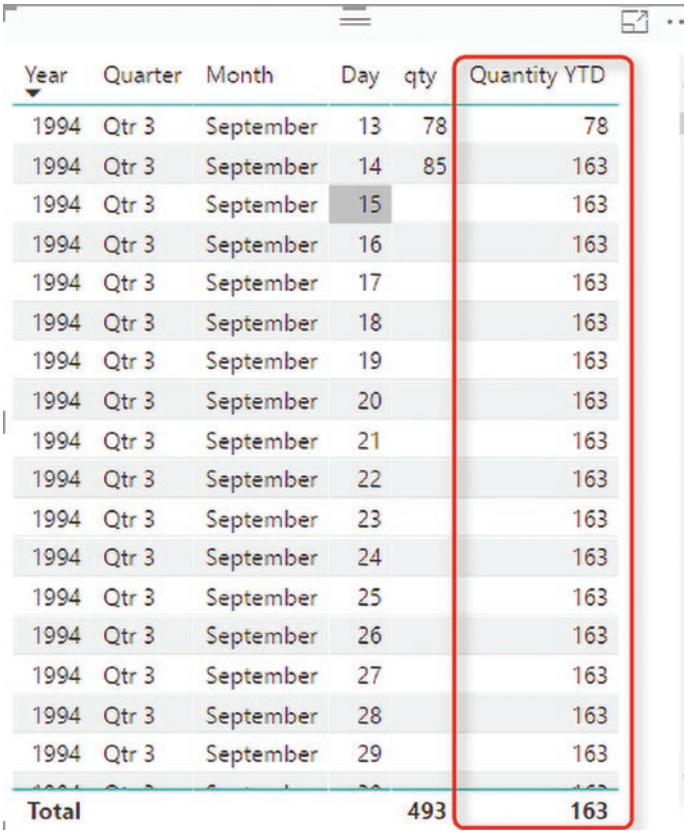


Figure 3-12. *DAX calculation for year-to-date quantity*

This calculation is defined in DAX with the following code:

```
Quantity YTD = TOTALYTD(
                    SUM('Pubs Transactions'[qty]),
                    'Pubs Transactions'[ord_date].[Date])
```

The result of that code calculates the year-to-date value of the quantity, as shown in Figure 3-13.



Year	Quarter	Month	Day	qty	Quantity YTD
1994	Qtr 3	September	13	78	78
1994	Qtr 3	September	14	85	163
1994	Qtr 3	September	15		163
1994	Qtr 3	September	16		163
1994	Qtr 3	September	17		163
1994	Qtr 3	September	18		163
1994	Qtr 3	September	19		163
1994	Qtr 3	September	20		163
1994	Qtr 3	September	21		163
1994	Qtr 3	September	22		163
1994	Qtr 3	September	23		163
1994	Qtr 3	September	24		163
1994	Qtr 3	September	25		163
1994	Qtr 3	September	26		163
1994	Qtr 3	September	27		163
1994	Qtr 3	September	28		163
1994	Qtr 3	September	29		163
Total				493	163

Figure 3-13. *Year-to-date calculation visualized in a table*

DAX is another powerful component of Power BI. It's fully functional in Import Data. DAX is a whole language to learn. I recommend reading *Power BI DAX Simplified* (Rad, 2021); `www.amazon.com/gp/product/B099SBN1XP`, which includes many DAX examples.

Publishing a Report

Publishing a report is the same regardless of the connection you use. You simply click Publish on the Home tab to publish the report to your workspace.

Gateway Configuration

For Import Data connection types that use on-premises data sources, you need to have a gateway. If your data sources are all cloud-based (such as Azure SQL database), you don't need a gateway.

Because you can easily combine multiple data sources with the Import Data connection type, it is very likely that your Power BI file will have more than one data source in it. You first need to check all the data sources in your Power BI Desktop and define them all under the gateway.

To find data sources in your Power BI Desktop, click the Home tab, and under Transform Data, select Data Source Settings, as shown in Figure 3-14.

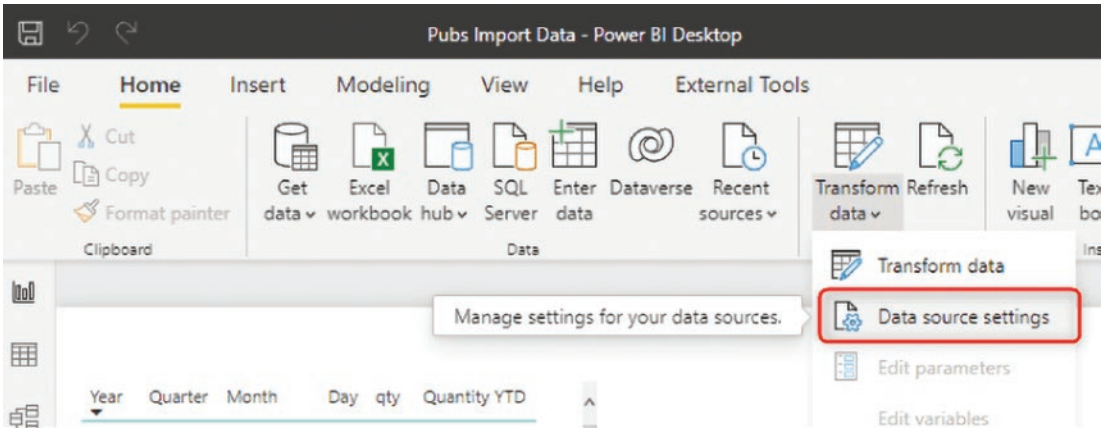


Figure 3-14. *Data source settings*

Figure 3-15 shows where you will find all the data sources.

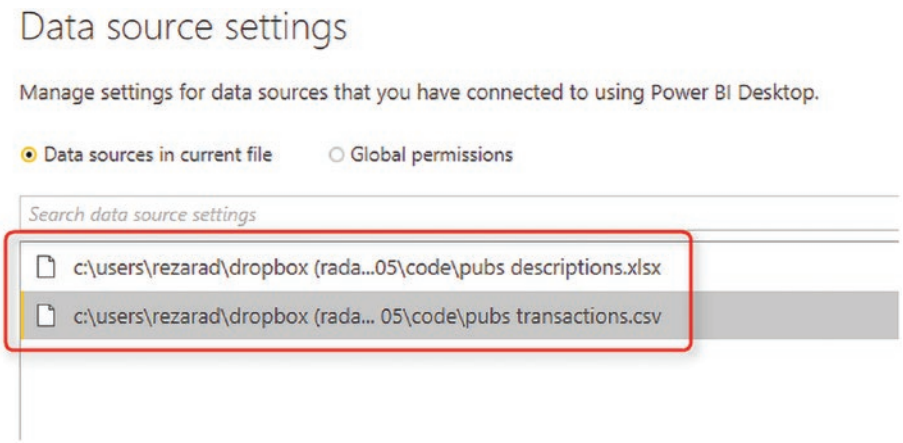


Figure 3-15. *All the data sources in the current file*

In a later chapter, I explain how to add data sources to the gateway. You need to make sure that all data sources in the `*.pbix` file are defined under the same gateway, as shown in Figure 3-16. Otherwise, you won't be able to use that gateway for the dataset. To learn how to install a gateway or add data sources to it, read the gateway chapter in this book.

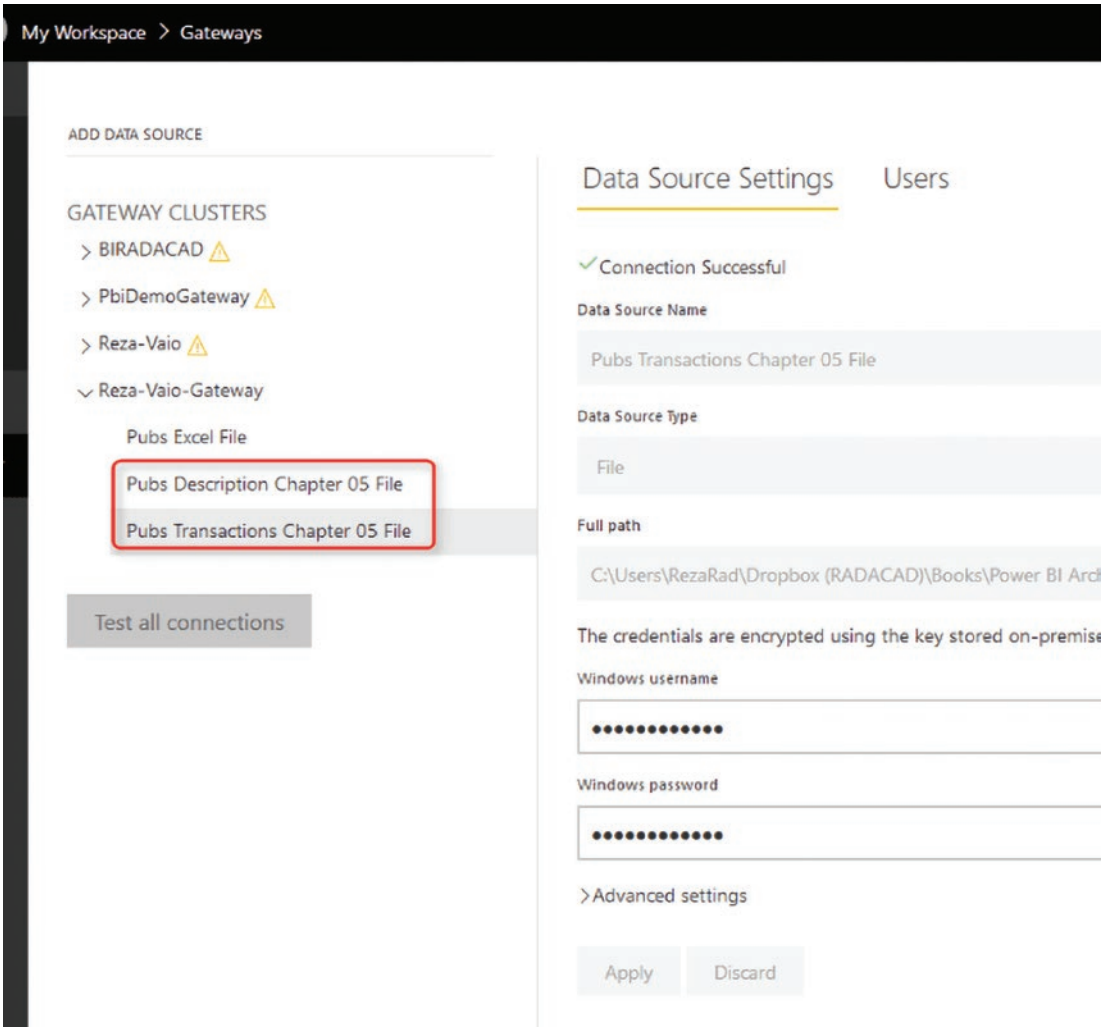


Figure 3-16. *Adding all the data sources under the gateway*

Scheduled Refresh

After adding all the sources, you can connect this gateway to the dataset within the Scheduled Refresh configuration of the dataset, as shown in Figure 3-17.

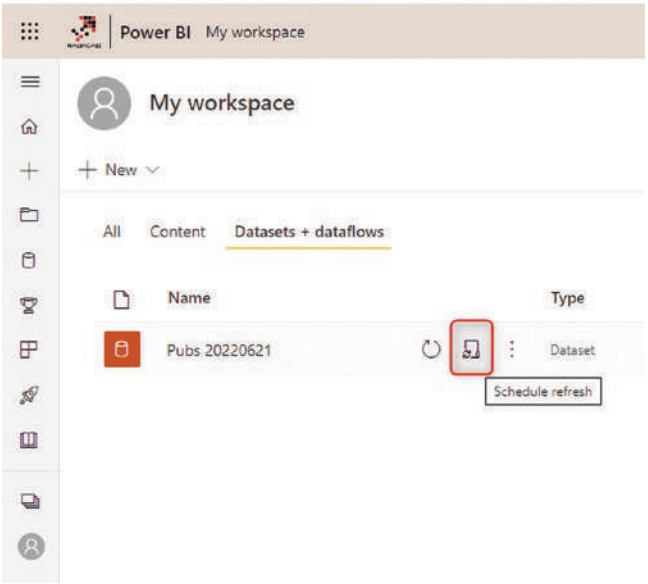


Figure 3-17. *Scheduled Refresh of the Power BI dataset in the Power BI Service*

After connecting the gateway, you can set a scheduled refresh. As shown in Figure 3-18, a refresh can be scheduled weekly or daily, with a maximum number of refreshes of eight times a day (with Power BI Premium, you can refresh it up to 48 times a day).

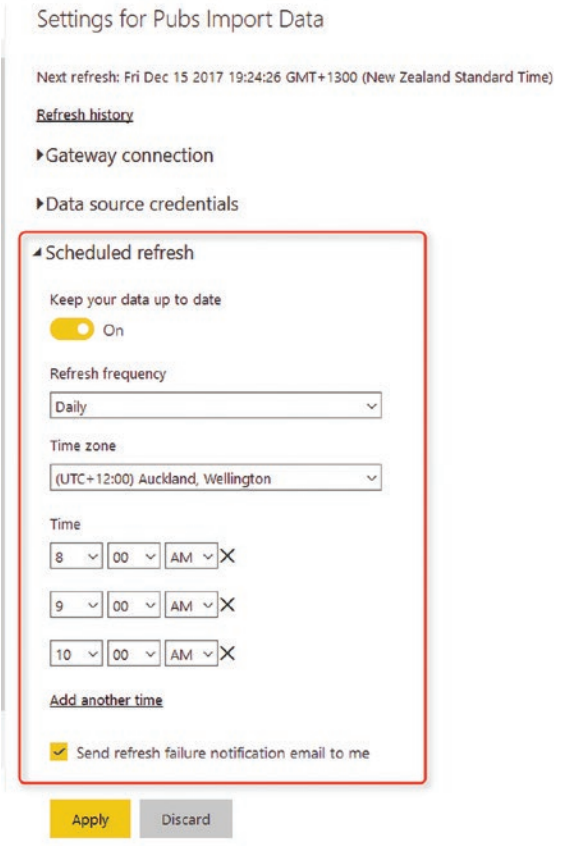


Figure 3-18. *Scheduling a refresh configuration*

After setting up the Scheduled Refresh, you can see the next refresh and the last refresh time in the dataset properties, as shown in Figure 3-19.

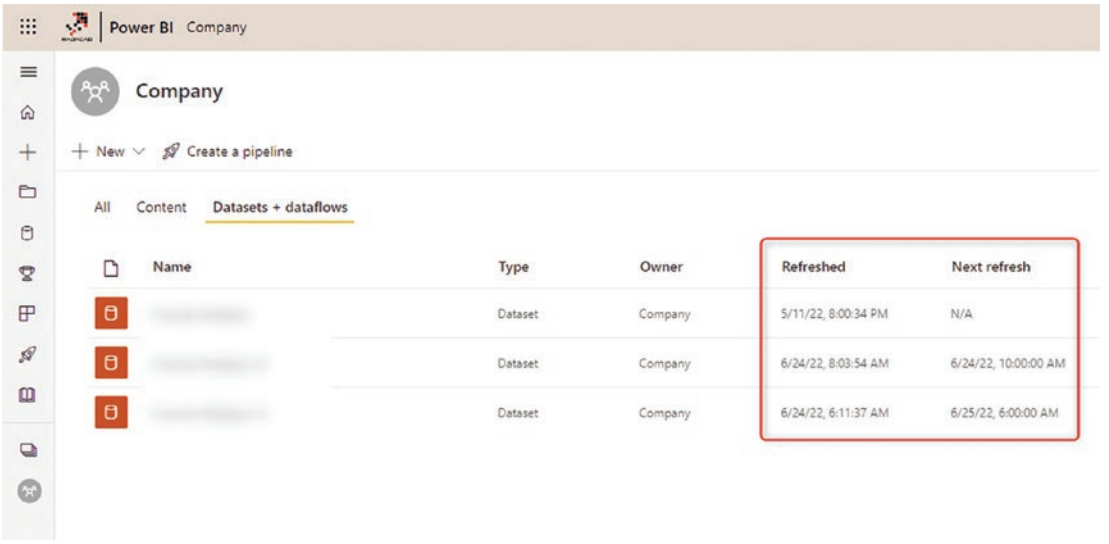


Figure 3-19. *Checking times for the next and the last refresh of Power BI datasets*

Advantages and Disadvantages of Import Data

Let's wrap things up and look at the pros and cons of Import Data are, as well as the scenarios in which this connection type is useful. This section is a wrap-up of what is explained above, just a quick review of those;

Advantages of Import Data

- **Speed: In-Memory Engine:** An advantage of the Import Data connection type is its super-fast response time. Remember that this is an in-memory technology, so querying data from memory is much faster than querying from disk. This method of connection is the fastest method of connection in Power BI.
- **Flexibility: DAX and Power Query:** With this method, Power Query and DAX are fully functional. Power Query and DAX are two dominant components of Power BI. In fact, without these two elements, Power BI is just a visualization tool. The existence of Power Query and DAX make it an extraordinary tool that can cover all analytics requirements. When you use Power BI with other types of connections, these two components aren't fully functional. Import Data gives you the flexibility to do data manipulation with Power Query and perform analytical calculations with DAX.

Disadvantages of Import Data

- **The requirement to schedule a data refresh:** In many BI scenarios, data will be refreshed overnight or on a scheduled basis. However, sometimes you need the data to be updated without delay. Import Data is not capable of doing that. With Import Data, the dataset must be refreshed and it can be scheduled to refresh up to eight times a day (or 48 times a day with Power BI Premium).

- **Very-large-scale data:** If your dataset is massive, let's say petabytes of data, and the compression engine of Power BI cannot fit it into the allowed size (which is 1GB per model for pro, and 400GB per model for premium), you must change the type of connection. Alternatively, you can choose to be part of a Power BI Premium capacity and leverage a larger dataset allowance with that option.

When Should You Use Import Data?

Import Data should be your first choice when you're choosing a connection type in Power BI. It gives you flexibility, power, and performance and avoiding it comes at big costs. Try to import your data into Power BI; if you cannot do it (maybe the size is too big, or a real-time dashboard is needed), you can try the other methods. Import Data should be the first option you try.

Summary

In this chapter, you learned about Import Data or Scheduled Refresh. With this type of connection, your data is imported into memory. However, the data will be compressed, and the copy of the data in memory is usually smaller than the actual data source size. Because this method copies the data into memory, you do need to set a scheduled refresh for this model.

Import Data or Scheduled Refresh is the fastest method, the most agile way of working with Power BI, and the most thoroughly functional connection type in Power BI. It allows you to combine multiple data sources with Power Query, write analytical calculations with DAX, and visualize the data. This method is super-fast because reading data from memory is always faster than reading it from disk.

However, the Import Data mode has a couple of limitations—the need to refresh data is one of them, and the other one is the size limitation of the Power BI files. Size limitation can be lifted when using the Power BI premium capacity, or by changing to other types of connections, which I explain in the next few chapters.

CHAPTER 4

DirectQuery

In the previous chapter, you learned about the Import Data or Scheduled Refresh connection type. In this chapter, you learn about the second type of connection, DirectQuery. This type of connection is supported by a limited number of data sources, and it mainly targets systems with huge amounts of data. DirectQuery is different from Live Connection, which is covered in the next chapter.

What Is DirectQuery?

DirectQuery is a type of connection in Power BI that does not load data into the Power BI model. If you remember from the previous chapter, Power BI loads data into memory (when Import Data or Scheduled Refresh is used as the connection type). DirectQuery doesn't consume memory because a second copy of the data is not stored. With DirectQuery, Power BI is directly connected to the data source. Anytime you see a visualization in a report with DirectQuery, the data has come straight from a query sent to the data source.

Which Data Sources Support DirectQuery?

Unlike Import Data, which is supported in all types of data sources, DirectQuery is only supported by a limited number of data sources. You cannot create a connection as a DirectQuery to an Excel File. Usually, data sources that are relational database models, or have a modeling engine, support DirectQuery mode. Here are some of the data sources supported through DirectQuery:

- Amazon Redshift
- Azure HDInsight Spark
- Azure SQL Database
- Azure SQL Data Warehouse
- Google BigQuery
- IBM Netezza
- Impala
- Oracle Database
- SAP Business Warehouse
- SAP HANA
- Snowflake

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_4

- Spark
- SQL Server
- Teradata Database
- Vertica

This list may change. With every new update of Power BI, some new data sources may be added to the DirectQuery-supported lists of Power BI.[1]

How to Use DirectQuery

To run this example, you need a SQL Server instance installed. You can download the SQL Server Developer Edition at `www.microsoft.com/en-us/sql-server/sql-server-downloads`. Then set up the `AdventureWorksDW` database on it. You can get this database from `github.com/microsoft/sql-server-samples/tree/master/samples/databases/adventure-works/data-warehouse-install-script`.

This book is not about installing and configuring SQL Server or setting up the database on it. So, I'm not going to explain how to do that. This section covers the Power BI part of the example.

Open the Power BI Desktop and, under Get Data, select SQL Server, as shown in Figure 4-1.

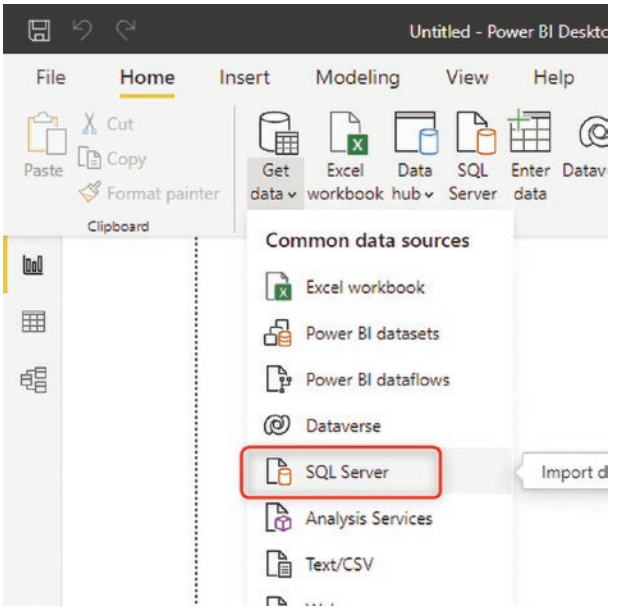


Figure 4-1. *Getting data from SQL Server*

[1] View the up-to-date list at `docs.microsoft.com/en-us/power-bi/connect-data/power-bi-data-sources`.

The very first query you get when connecting to SQL Server data source includes an option for choosing the Data Connectivity mode. Select DirectQuery, as in Figure 4-2. You also need to add your server name. If your SQL Server instance is the default instance on your machine, you can use (`.`) for the server.

Figure 4-2. *Selecting DirectQuery in Power BI*

Note that data sources that support DirectQuery also support Import Data. (All data sources support Import Data.)

In the Navigator window, you can select some tables from the `AdventureWorksDW` database, such as `DimDate` and `FactInternetSales`, as I selected in Figure 4-3.

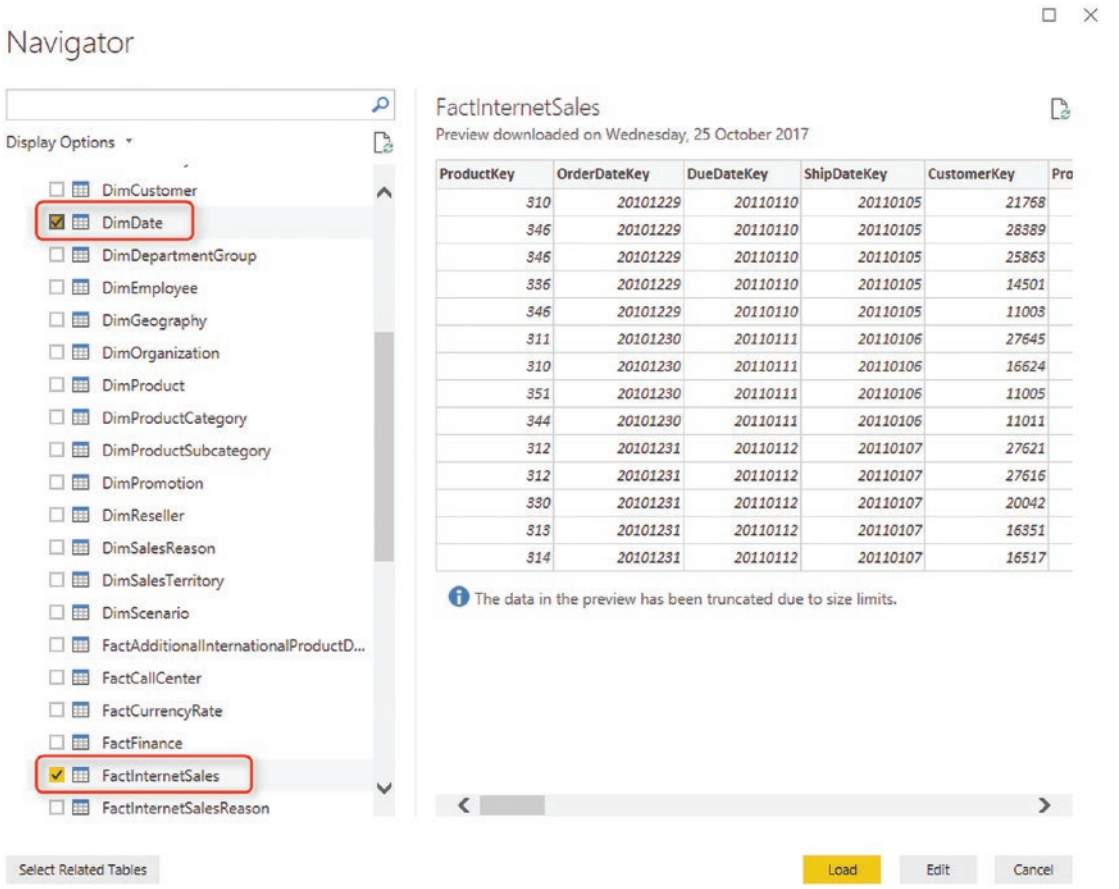


Figure 4-3. *Selecting tables in DirectQuery mode*

After selecting tables, click Load. The Data Load dialog in this connection mode is much faster because there is no need to load data into memory. This time, only metadata will be loaded into Power BI. The data remains on SQL Server.

No Data Tab in DirectQuery Mode

One of the first things you will notice in the DirectQuery mode, shown in Figure 4-4, is that there is no Data tab (the middle tab on the left side navigation of Power BI).

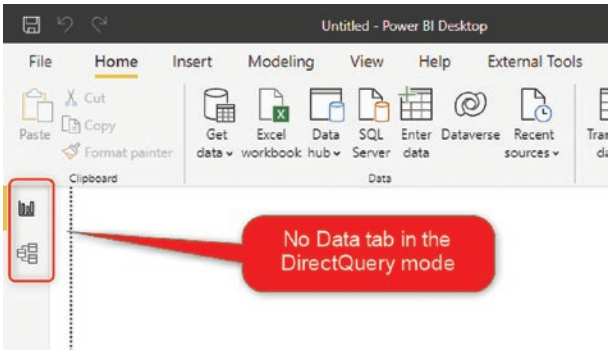


Figure 4-4. *No Data tab in the DirectQuery mode*

The Data tab shows you the data in the Power BI model. However, with DirectQuery, there is no data stored in the model. At the bottom-right side of the Power BI Desktop, shown in Figure 4-5, note that there is a note about the DirectQuery connection.

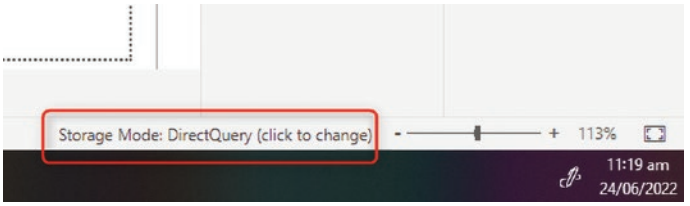


Figure 4-5. *DirectQuery is enabled*

How DirectQuery Works

With DirectQuery enabled, every time you see a visualization, Power BI sends a query to the data source, and the result of that comes back. You can check this process in SQL Profiler. SQL Profiler is a tool that you can use to capture queries sent to your SQL Server database. Figure 4-6 shows an example Power BI report on a DirectQuery model.

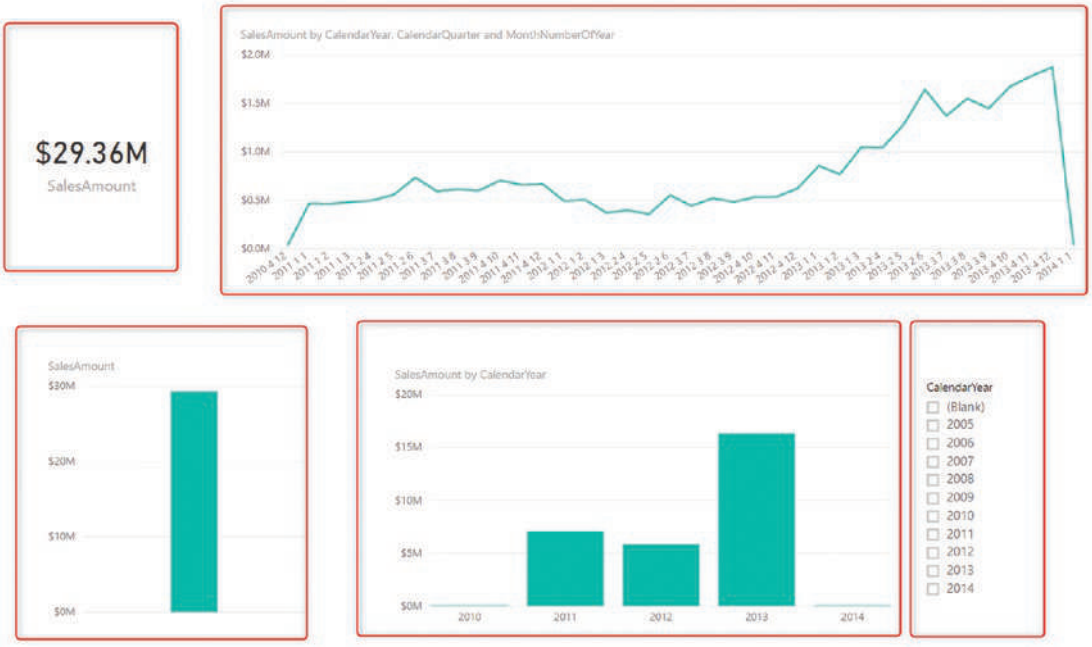


Figure 4-6. *Sample Power BI report*

Running a SQL Profiler will show you that, every time you refresh that report page or change something, there is one query for every visualization (see Figure 4-7)!

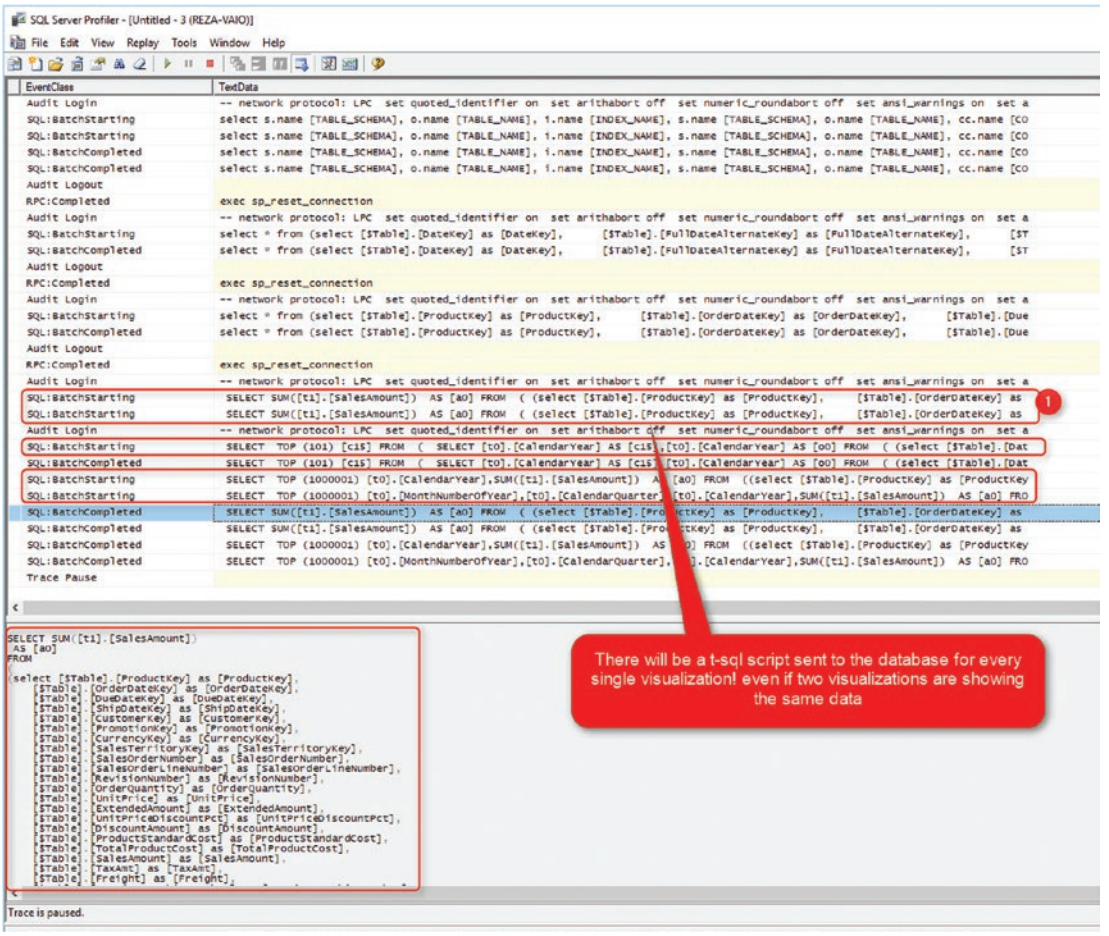


Figure 4-7. *Capturing queries sent to the database with SQL Profiler. There is a SQL Query for each visualization sent to the data source*

You can see in Figure 4-7 that in the SQL Profiler, five queries have been sent to the database. As Figure 4-7 also makes clear, even if two visualizations are showing the same thing, they send two separate queries to the database. As seen in Figure 4-8 for each visualization, one query will be produced.

Figure 4-8. *Every visualization sends a query to the database*

Performance

DirectQuery performs much slower than the Import Data option. Import Data loads data into memory. It is always faster to query data from memory (Import Data), rather than querying it from disk (DirectQuery). However, to determine how much faster Import Data is, you need to know more about the implementation. Depending on the size of data, the specification of the server that the database is running on, the network connection speed, and other factors such as whether there is any database optimization applied to the data source, the answer might be entirely different.

Performance Tuning on the Data Source

The critical point to understand is that this type of connection is the slowest, and if you decide to use it, you must immediately think about performance tuning your database.

Using DirectQuery without performance tuning the source database is a big mistake.

To understand the need for performance tuning, let's go through an example. Assume that you have a large table in a SQL Server database, a table with 48 million rows in it. You want to query the Sum of Sales column from that table.

Figure 4-9 demonstrates the performance you get if you have a normal index on the table with 48 million records.

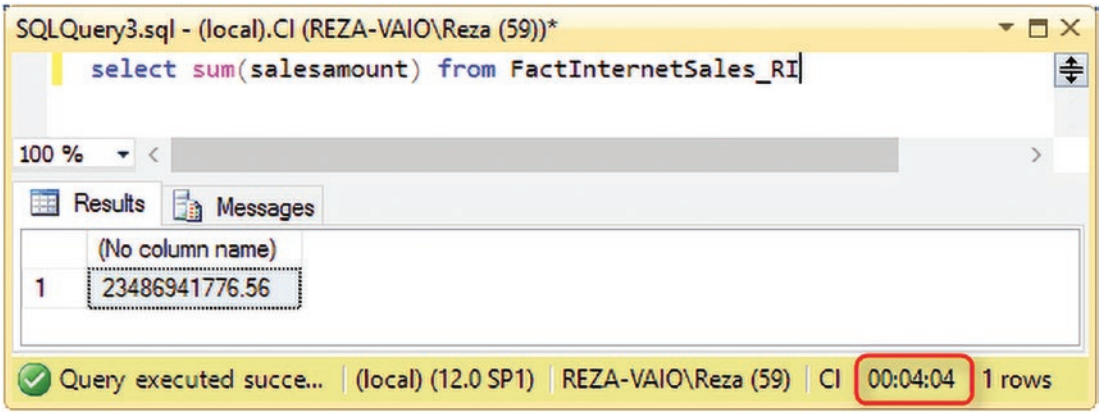


Figure 4-9. *Table with a regular index*

A regular `select sum` from this table with 48 million records takes four minutes and four seconds to run. The same query responds in less than a second when you have a Clustered Column Store index. It shows significantly improved performance when you have a Clustered Column Store index on the same table with the same amount of data rows, as shown in Figure 4-10.

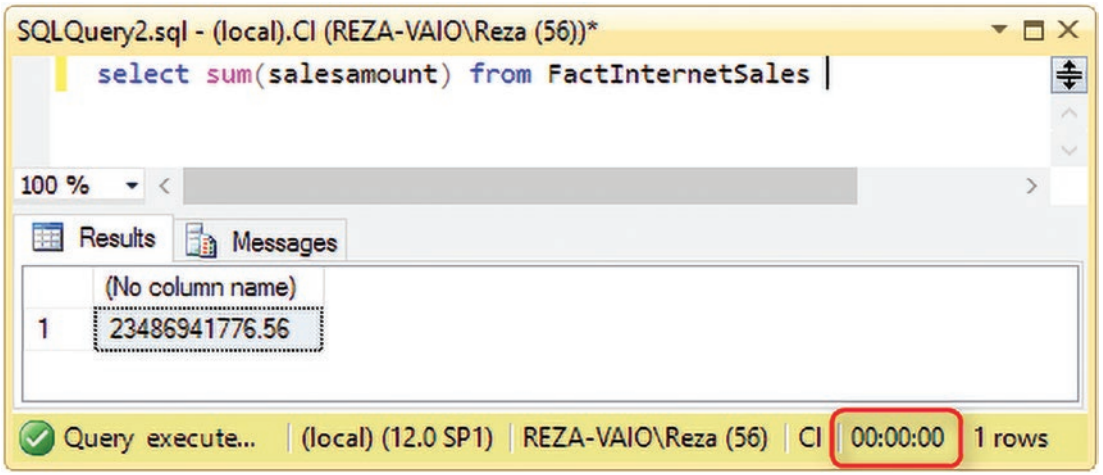


Figure 4-10. *Table with a clustered column store index*

I'm not going to teach you all performance tuning in this chapter, and I can't do it because you have to read books, blog posts, and watch videos to learn all that. That is a whole different topic on its own. The most important thing is that performance tuning is different for each data source. Performance tuning for Oracle, SQL Server, and SSAS are entirely different. Your friend for this part is Google, and the vast amount of free content available on the Internet for you to study.

Query Reduction

One of the newest features added to Power BI helps reduce the number of queries sent to the database in the DirectQuery mode. The default behavior of a slicer or filter in Power BI is to select an item in the slicer or filter, so that other visuals are filtered immediately. In the DirectQuery mode, it means it will send multiple queries to the database with every selection in filter or slicer. Sending multiple queries will reduce the performance of your report.

You may want to select multiple items, but only selecting the first item means that five queries will be sent to the database. Then, after selecting the second item, another five queries will be sent to the database. The speed, as a result, will be twice as slow. To fix this issue, you can set a property in the Options of your Power BI file.

Click the File menu in the Power BI Desktop, and then select Options and Settings. From there, choose Options (see Figure 4-11).

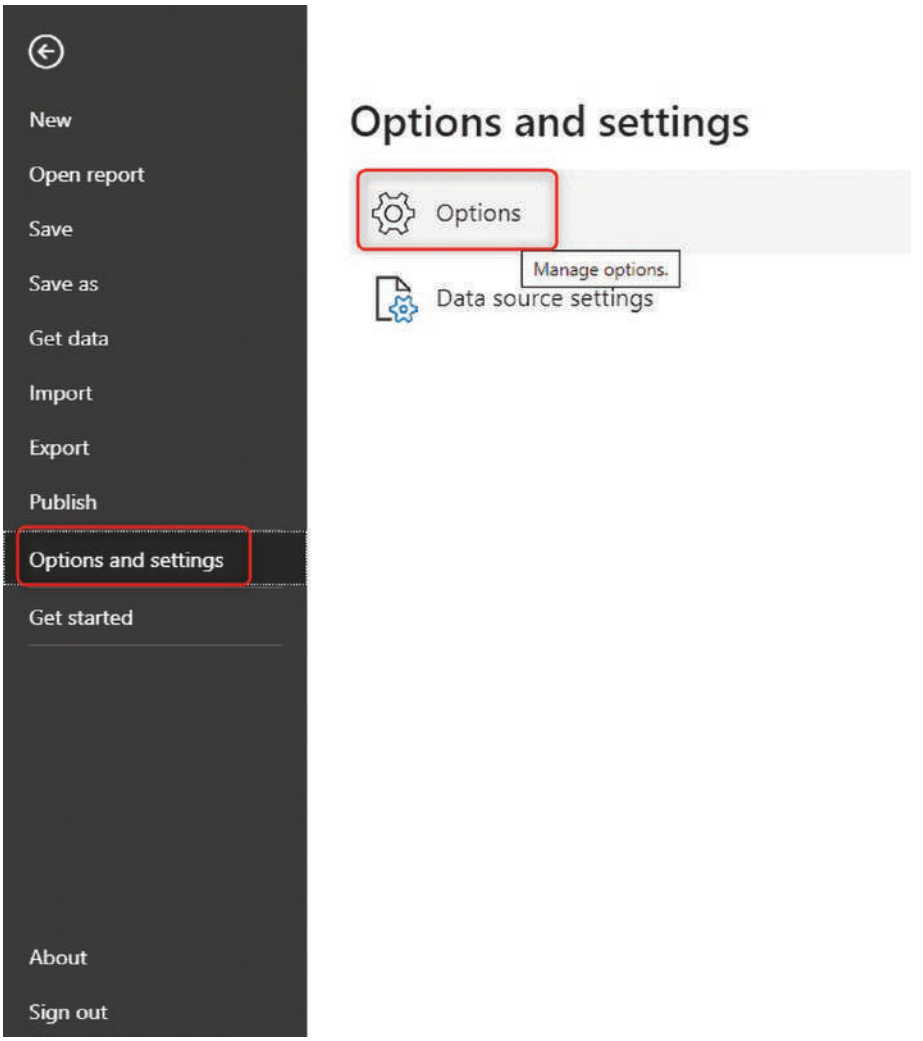


Figure 4-11. *Power BI file options*

Then click Query Reduction on the left side (the last item, as shown in Figure 4-12).

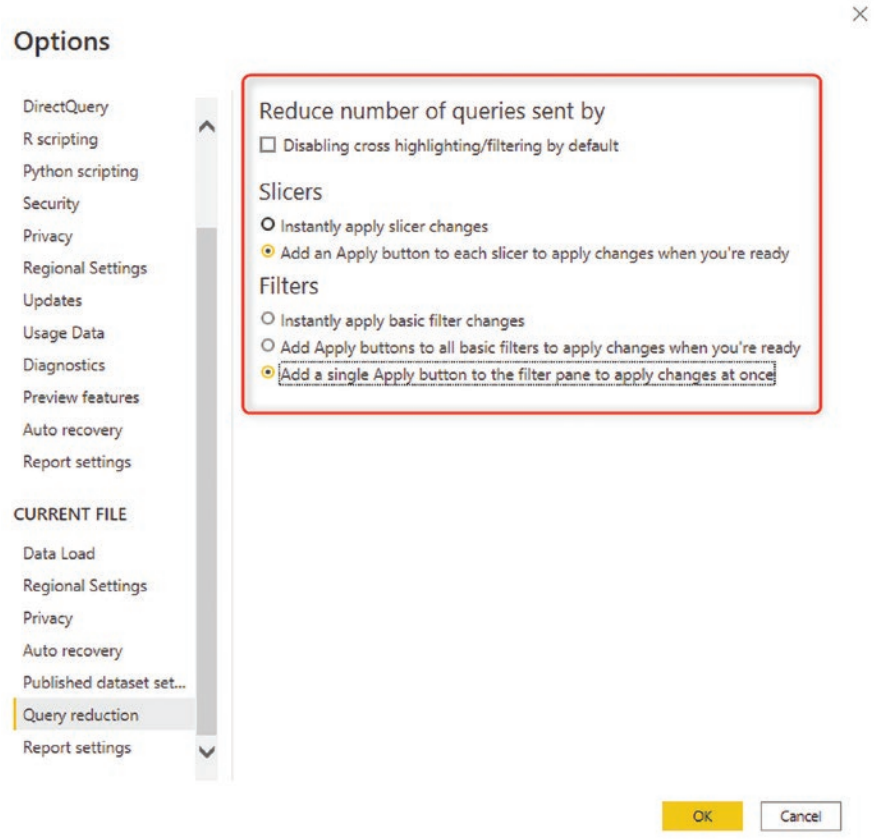


Figure 4-12. *Query reduction for DirectQuery mode*

I do not recommend using the first item in most cases. The first item will disable cross highlighting/filtering by default. When selecting the first option, the main functionality of Power BI, which is cross highlighting/filtering, will not work. Cross-highlighting/filtering means that, when you click a visual, other visuals will be either filtered or highlighted. This feature makes the Power BI an interactive visualization tool. If you have some visuals that don't need to interact with each other, enabling this option could reduce the number of queries sent to the database.

The second and third options, however, are beneficial, especially when you have multi-select slicers and filters. When you choose this option, your filters or slicers will have an Apply button on them. Changes will not apply until you click the Apply button, and then all the queries will be sent to the database. This option is highly recommended when you have a multi-selection slicer or filter. You can even choose to have one single Apply button for the entire Filter pane, which can be very helpful in query reduction. This is depicted in Figure 4-13.

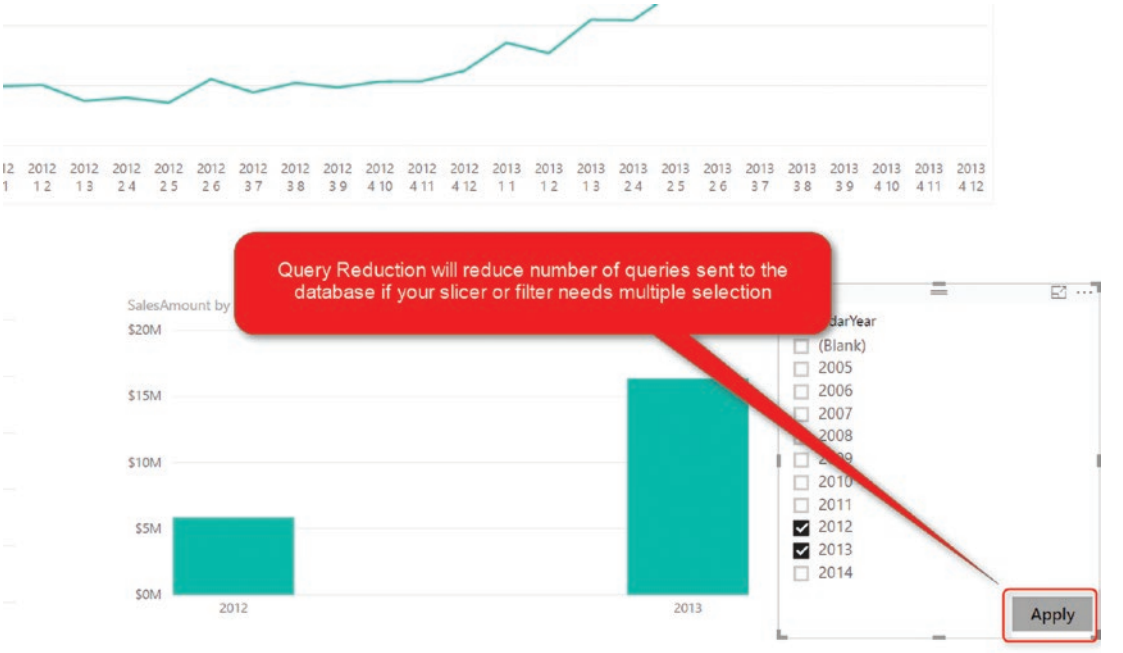


Figure 4-13. *Query reduction sends queries only once the Apply button has been clicked*

Maximum Connections Per Data Source

The number of concurrent queries sent from Power BI to the data source is important. If you allow too many concurrent queries, you are applying a very high load on the data source. On the other hand, if you have created a report page with over 15 visuals, you would need 15 concurrent queries (assuming they are not slicers, and query reduction won't reduce the number of queries).

You can set the number of concurrent queries per data source in the Power BI file options, as highlighted in Figure 4-14.

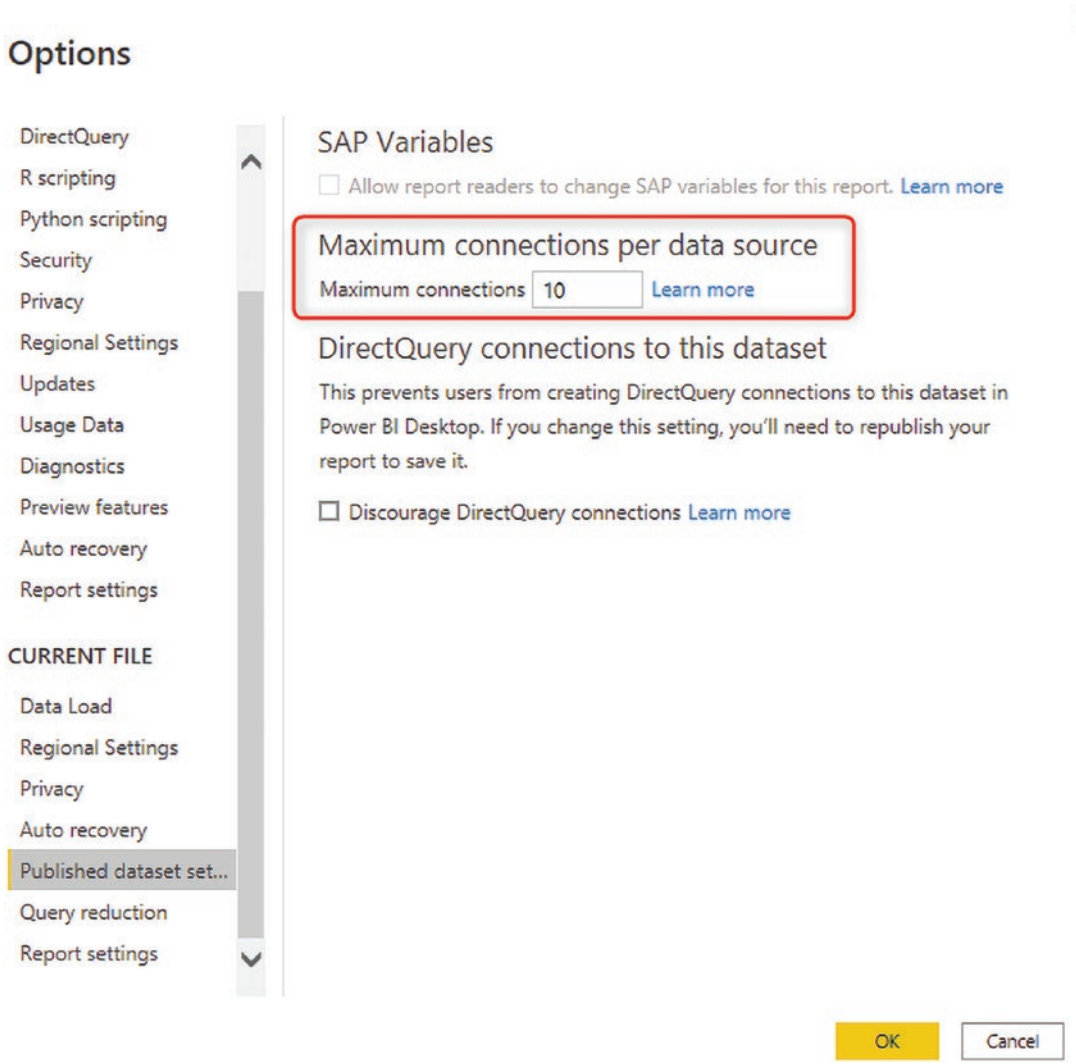


Figure 4-14. *Maximum connections per data source in the DirectQuery mode*

This setting is for the published Power BI file. A published Power BI report can be opened by multiple users. This also means increasing the concurrent connections. When you set this number, you have to take into account not only the number of visuals on the page and the performance of the data source, but also the number of concurrent report users.

Depending on your Power BI license and the environment, there are some limits to this number, as shown in Figure 4-15.

ENVIRONMENT	UPPER LIMIT (ACTIVE CONNECTIONS PER DATA SOURCE)
Power BI Pro	10
Power BI Premium	30
Power BI Report Server	10

Figure 4-15. *Maximum connections per data source setting for DirectQuery*

Adding Extra Data Sources: Composite Mode

Back in the older days of using the Power BI Desktop, if DirectQuery was the mode of the connection, it was not possible to add any other data sources. However, that changed a few years ago. You can now add extra data sources (Import, or any other connection types) to a DirectQuery-based Power BI file. This changes the file to something called *composite mode*. However, the warning shown in Figure 4-16 will be visible when you bring data in from other data sources.

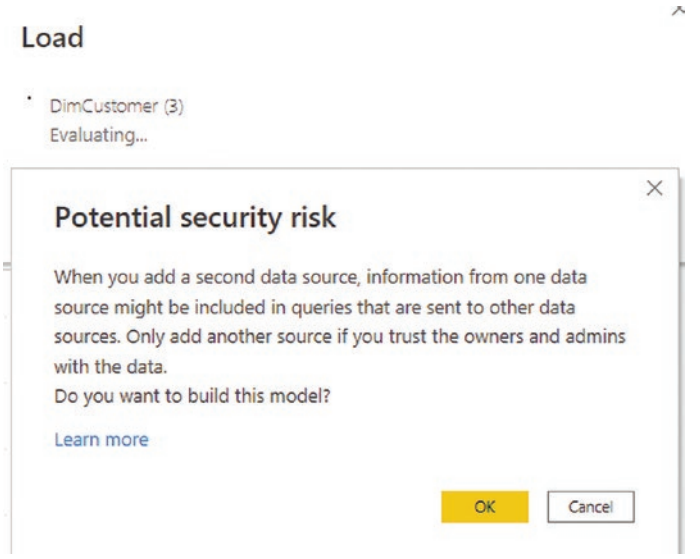


Figure 4-16. *Adding more data sources into the DirectQuery-based file*

Composite mode is a very powerful way to build Power BI files. It uses part of the data using a DirectQuery connection and other parts of the data using an Import Data connection. Often—even in large-scale data source scenarios—there are small tables for dimensions. Dimensions in such cases can

be imported while the big fact tables are used as a DirectQuery. You can learn more about composite mode in the next chapters. In Figure 4-17, `DimCustomer` is an Import Mode table, and `DimDate` and `FactInternetSales` use DirectQuery.

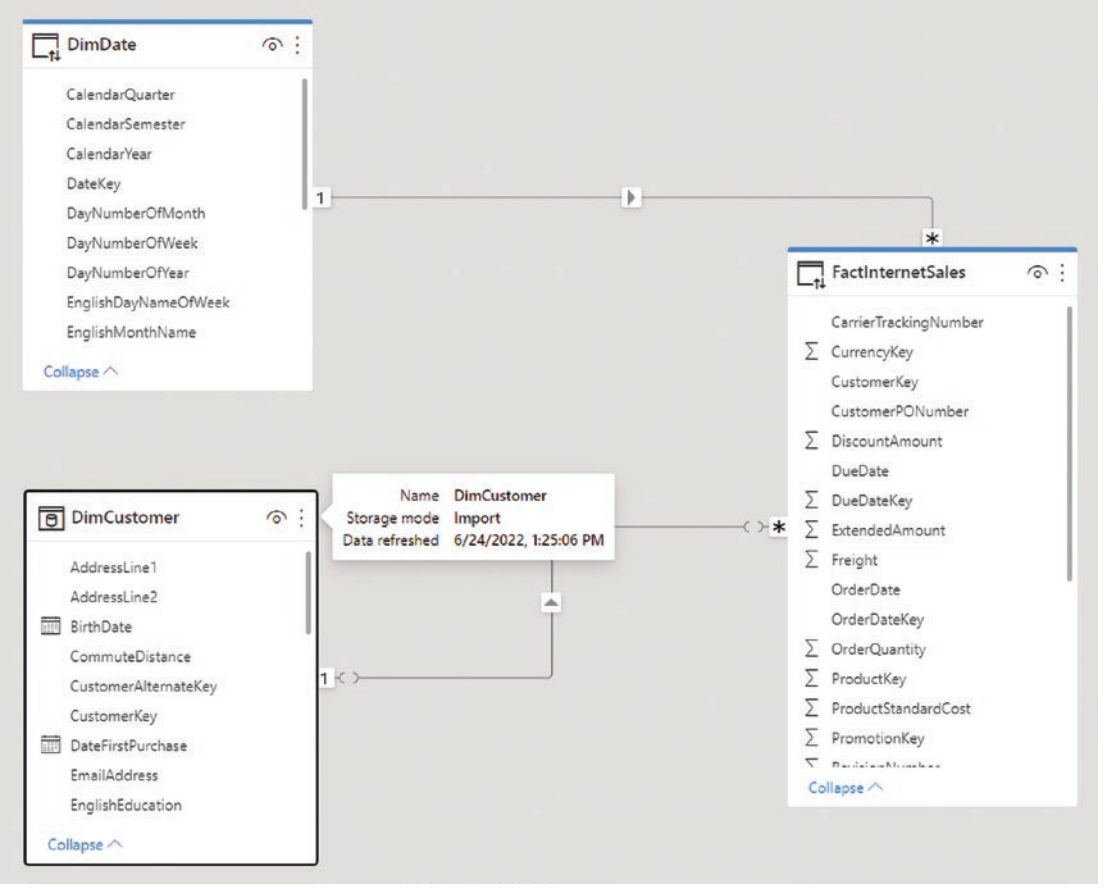


Figure 4-17. *Composite mode*

Limited Power Query

With DirectQuery, you can apply data transformations in the Query Editor window. However, not all transformations are supported. To find out which transformations are supported and which are not, you have to check the data source first. Some of the data sources don't support any transformations, such as SAP Business Warehouse. Other transformations, such as the SQL Server database, support more transformations.

If you use a transformation that is not supported, you'll get an error message that says, "This step results in a query that is not supported in DirectQuery mode." See Figure 4-18.

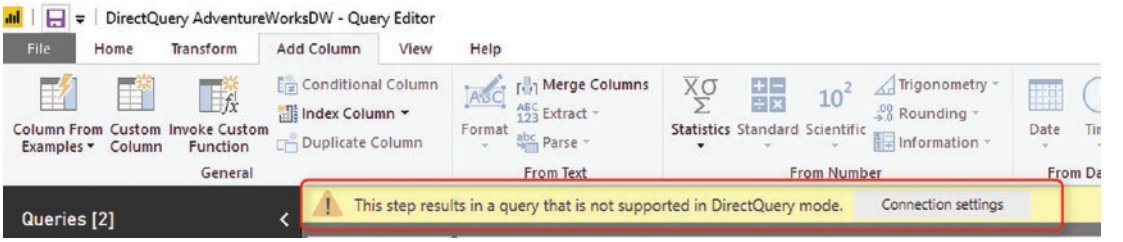


Figure 4-18. *Not all data transformations in Power Query are supported with DirectQuery data sources*

Transformations in Power Query are limited in DirectQuery mode. This depends on the data source.

As mentioned earlier, when you use a DirectQuery connection, you usually should think of another data transformation tool, such as SQL Server Integration Services, to bring data transformed into a data warehouse before connecting Power BI to it.

Limited Modeling and DAX

DAX and Modeling are also limited in DirectQuery mode. Creating calculated tables is allowed in composite mode, and the default Date hierarchy is not available. Some of the DAX functions, such as parent-child functions, are not available.

Some of the complex DAX measures might cause performance issues in DirectQuery mode. It is better to start with simple measures such as simple aggregations first, test the performance, then gradually add more complex scenarios.

No Refresh Needed

One of the advantages of DirectQuery is that there is no need for data refresh to be scheduled. Every time a user looks at the report, a query is sent to the database and the most recent data is shown. There is no need for a data refresh schedule.

When the automatic refresh happens with dashboards, the DirectQuery connection will be refreshed every 15 minutes or more. For reports, data will be updated any time a new query is sent to the database.

Large Scale Dataset

DirectQuery is best used with a massive amount of data. Because the data is not loaded into the memory, there is no limitation on the size of the data source. Any limitation comes from the data source itself. You can have petabytes of data with a DirectQuery connection. However, you need to consider performance tuning in the data source, as mentioned.

The lack of size limitation is the main reason to use DirectQuery. DirectQuery is slower and less flexible, with fewer features than Power Query and DAX. Overall, DirectQuery is good option only when the other two types of connections cannot be used.

These days, especially after composite mode became available in Power BI, it is very rare to see a pure DirectQuery-only method used in a Power BI dataset. The better approach is to use DirectQuery for large tables and use Import Data for smaller tables and then create a composite mode.

Summary

Power BI supports three types of connections. In this chapter, you learned about DirectQuery, which is one of the connection types. DirectQuery does not store a second copy of the data in memory. It keeps the data in the data source. With DirectQuery, anything you see in the report is sent to the database through T-SQL scripts. If you have ten visualizations in your report, it posts ten queries to the database and returns the result.

DirectQuery supports a huge amount of data. Because the data is not stored in the memory, the only limitation on the data size is the limitation of the data source itself. You can easily have petabytes of data in a database and connect to it from Power BI. Another advantage of DirectQuery is that there is no need for a scheduled refresh. Data is updated any time the report is refreshed with queries sent to the database, so there is no need for scheduling a refresh.

DirectQuery has many limitations and downsides. With DirectQuery, the speed of Power BI reports is much slower. DirectQuery is limited in using modeling features of Power BI such as DAX, calculated tables, built-in date tables, and Power Query functionalities. DirectQuery is less flexible and has fewer features. Using DirectQuery means you get slower reports, with less functionality on the Power BI side.

DirectQuery is only recommended if the other two connections (Import Data or Live Connection) cannot be applied. In the next chapter, I explain Live Connection. If you decide to use DirectQuery, it is better to use another integration tool and do all the data transformation before loading data into the source database. Also, it is better to take care of all the calculations in the data source itself.

DirectQuery is only recommended if the other two types of connections (Import Data or Live Connection) cannot be applied. DirectQuery is better used in combination with Import Data through a composite mode.

To close this summary, here is the list of pros and cons of DirectQuery:

Advantages

- Large scale dataset; size limitation only for the data source
- No need for data refreshes

Disadvantages

- Power Query transformations are limited
- Modeling is limited
- DAX is limited
- Slower speed of the report

CHAPTER 5

Live Connection

Live Connection is another type of connection in Power BI. This connection is similar to DirectQuery because it doesn't store data in memory. However, it is different from DirectQuery, because it includes the analytical engine of SQL Server Analysis Services Tabular. With this method, you get benefits from both worlds—large-scale model size and the analytical power of Analysis Services. In this chapter, you learn about the details of Live Connection, including things that you need to consider and how to set it up when working with a gateway.

What Is Live Connection?

Live Connection is used with three types of data sources. It does not store a second copy of the data in memory. Data is kept in the data source, and visualizations query the data source from Power BI. The types of data sources supported by this type of connection are as follows:

- Azure Analysis Services
- SQL Server Analysis Services Tabular
- SQL Server Analysis Services Multi-Dimensional
- Power BI Service Dataset

These four types are the SQL Server Analysis Services (SSAS) technology. You cannot use Live Connection with the SQL Server database engine. However, the SSAS technology can be cloud-based (Azure Analysis Services) or on-premises (SSAS on-premises).

Create a Report with Live Connection

To create a Live Connection example, you need to install the SQL Server Analysis Services Tabular model. You can download the SQL Server trial for 180 days using this link: `www.microsoft.com/en-us/sql-server/sql-server-downloads`.

After installing a SSAS Tabular instance, you can restore the `AdventureWorks` Tabular model on it. The database can be accessed and downloaded from `github.com/Microsoft/sql-server-samples/releases/download/adventureworks-analysis-services/adventure-works-tabular-model-1200-full-database-backup.zip`.

This book does not go into details of how to set up or install SQL Server or the SSAS Tabular model.

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_5

Open a Power BI Desktop and choose Get Data, Analysis Services, as shown in Figure 5-1.

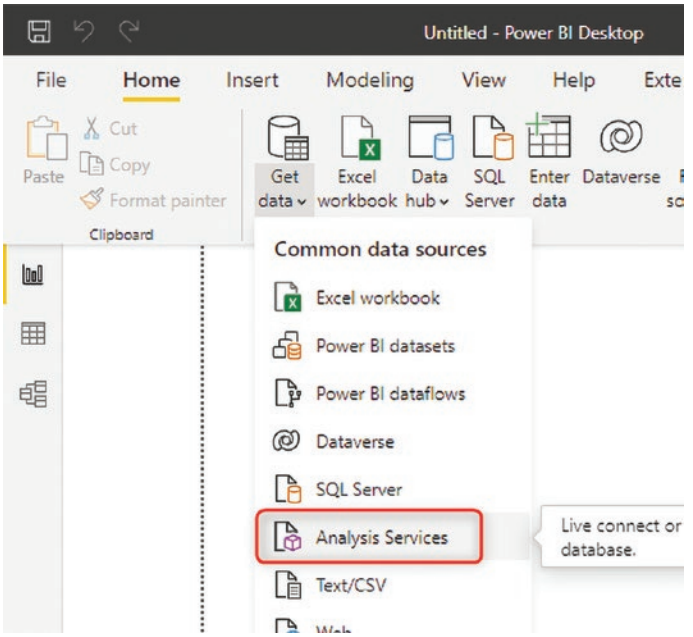


Figure 5-1. *Getting data from Analysis Services*

When connecting to a SQL Server Analysis Services database, choose Connect Live, as shown in Figure 5-2.

Figure 5-2. *Connecting live to Analysis Services*

The SSAS installed on my machine is in the tabular instance name, which is why I use the server name `.\tabular`. For your machine, the setup might be different. In the Navigator window, select the Adventure Works Internet Sales model (see Figure 5-3).

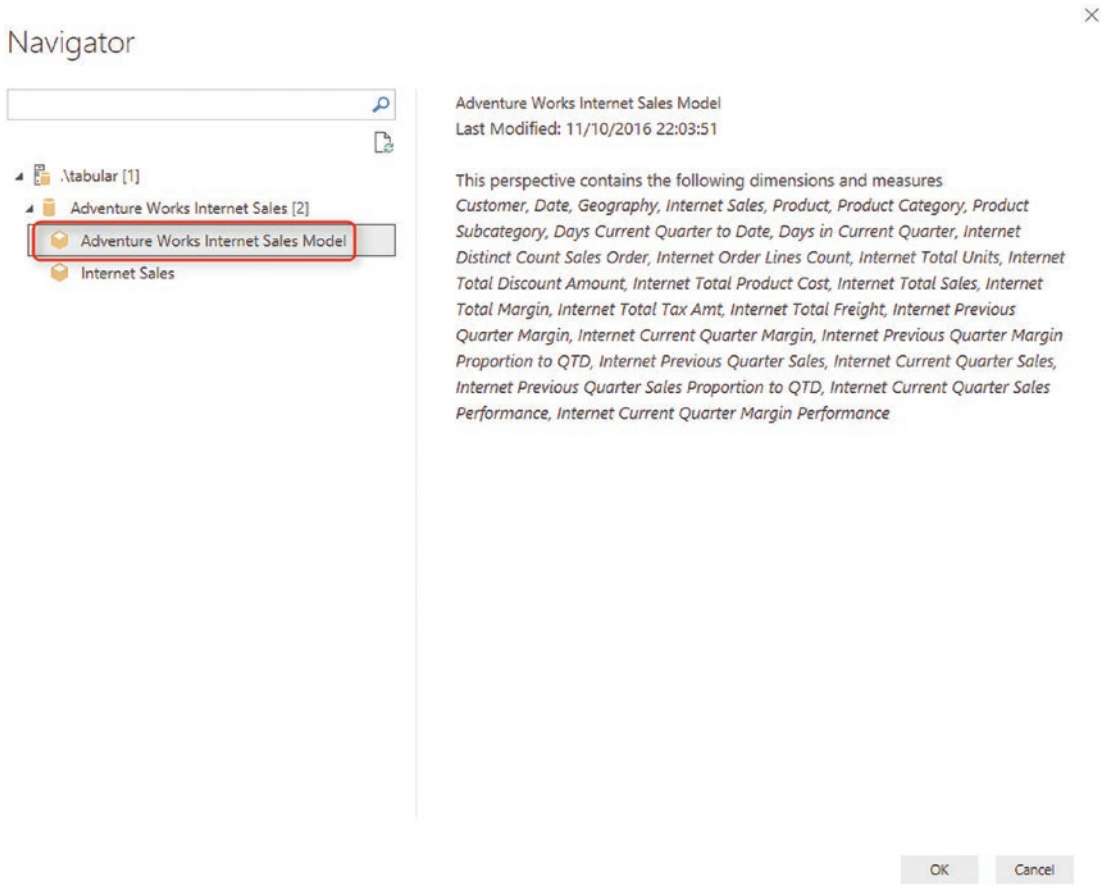


Figure 5-3. *Selecting the model from Analysis Services*

Note that you just choose a model, not tables. The model includes multiple tables, with their relationships, hierarchies, and calculations, and they will all be connected to Power BI. You can verify on the bottom-right side of the Power BI Desktop that it is a live connection. Figure 5-4 shows you what to look for.

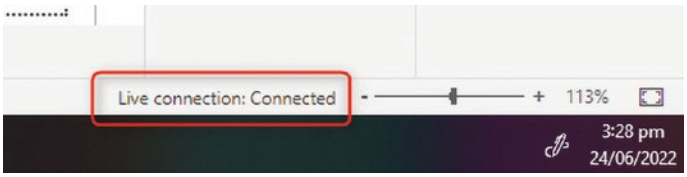


Figure 5-4. *Live Connection is enabled*

Live Connection Behind the Scenes

Consider the Power BI report shown in Figure 5-5.

Figure 5-5. *Sample Power BI report*

You can run SQL Profiler on Analysis Services to capture all the DAX queries sent to the database. Figure 5-6 features queries posted to the database for the report in Figure 5-4.

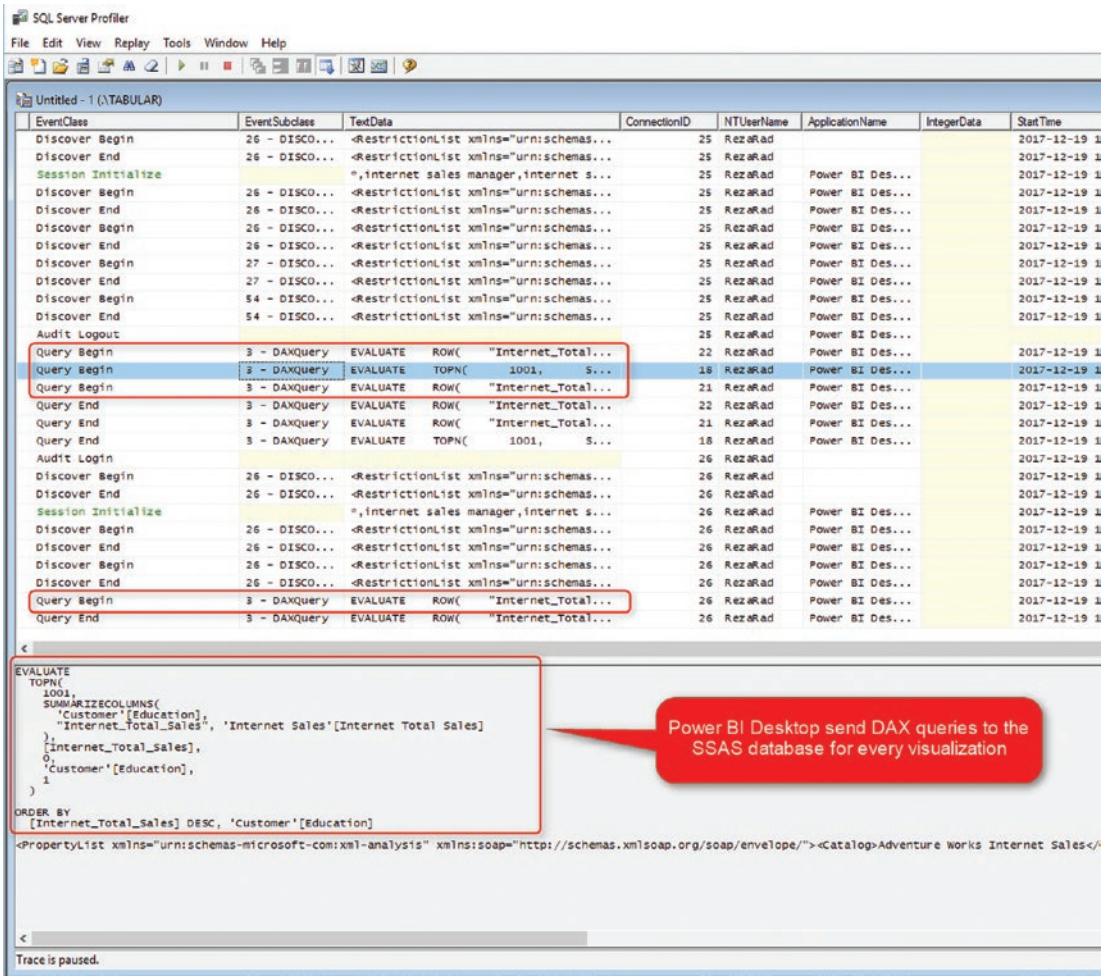


Figure 5-6. *DAX codes sent to the Analysis Services database*

As with DirectQuery, the Power BI Desktop will send a query for every visualization to the database. However, because these queries run on an analytical engine, you can typically expect faster results (this might be different depending on many factors).

Performance

Live Connection can work faster than DirectQuery in many scenarios. However, in every implementation, many factors affect the performance, and there are always exceptions. In general, because Live Connection connects to an analytical engine, calculations and analytical results are faster than a query from the database.

Using Live Connection to SSAS Tabular is usually faster because the data is stored in the memory of the machine that runs SSAS Tabular. However, this depends on network bandwidth and how calculations and model are implemented, so it might change.

Live Connection, however, is not as fast as Import Data. Import Data is the fastest possible option regarding performance in Power BI. A live connection is the second in the list, and DirectQuery is the slowest option.

Although the performance of Live Connection is usually better than DirectQuery, I highly recommend performance tuning the SSAS Model. The SSAS Model can perform fast if the server specification is at the right scale, the data model is designed well, and the calculations are written properly. If any of these items aren't tuned correctly, the performance of SSAS server and Live Connection will suffer.

Performance tuning in Live Connection is a must.

Single Data Source

Live Connection supports only one source for the connection. If you want to have more than one connection source, you can either use Import Data from the Analysis Services, or create a composite mode using DirectQuery to Analysis Services (which is explained in a later chapter in this book).

No Power Query Transformations

With Live Connection, Power Query transformations are not available. As you can see in Figure 5-7, the Edit Queries options are disabled when in Live Connection mode.

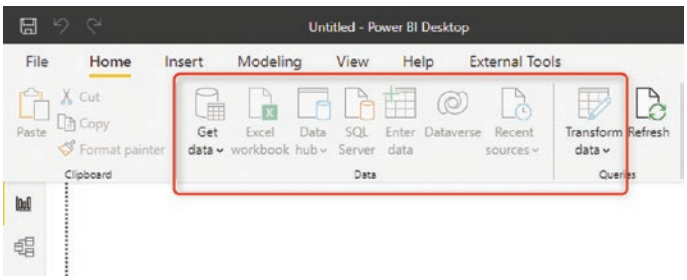


Figure 5-7. *No Power Query in Live Connection mode*

All the data transformations needs must be handled before loading data into an SSAS model. Because SSAS is not a data transformation tool, you can use SSIS to do the data transformations before loading data into the data warehouse, and then process data from the data warehouse into an SSAS data model. Figure 5-8 contains a sample scenario.

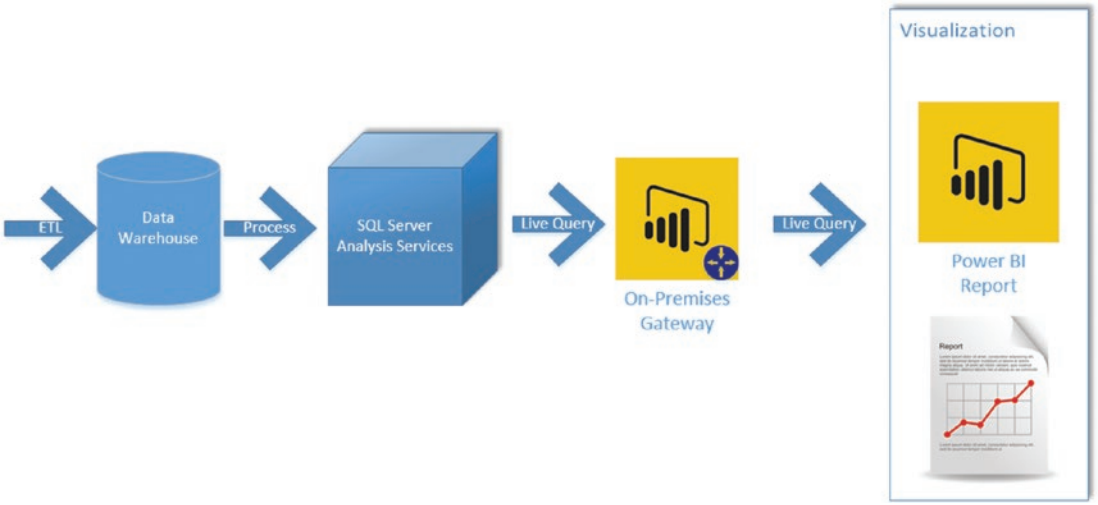


Figure 5-8. *Enterprise use of Live Connection with a data warehouse and ETL*

With Live Connection, you don't need to select each table separately. All the tables in the model will be available, as you can see in the model diagram in Figure 5-9.

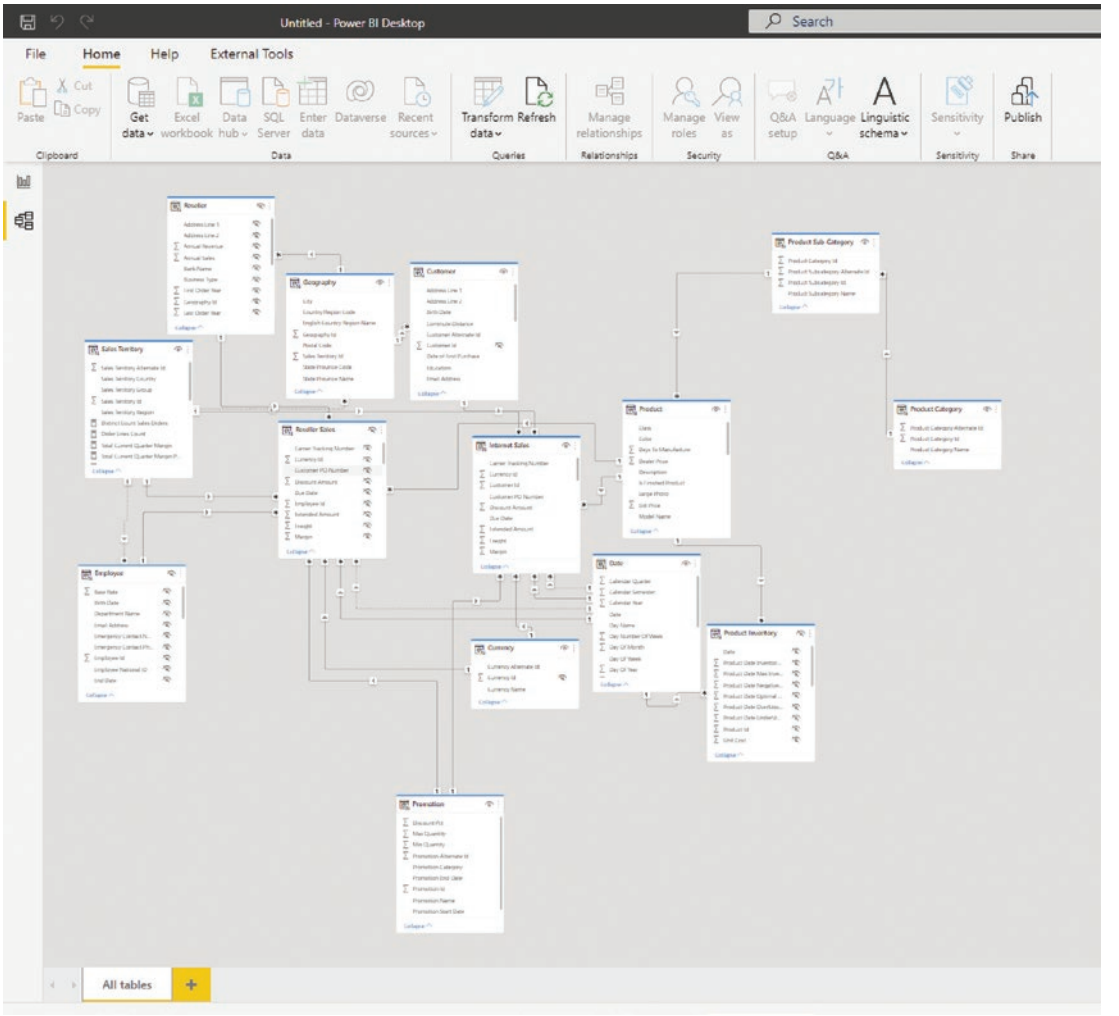


Figure 5-9. *Model diagram for Live Connection*

Modeling and Report-Level Measure

With Live Connection, modeling in Power BI is very limited. You can only create measures. The type of measure that you create a Live Connection is called a report-level measure. Report-level measures are only for this Power BI report. If you create another Power BI report connected live to the same data source, you cannot use the report-level measures that were built in the previous Power BI report.

To create a report-level measure, click Add New Measure from the Modeling tab, as shown in Figure 5-10.

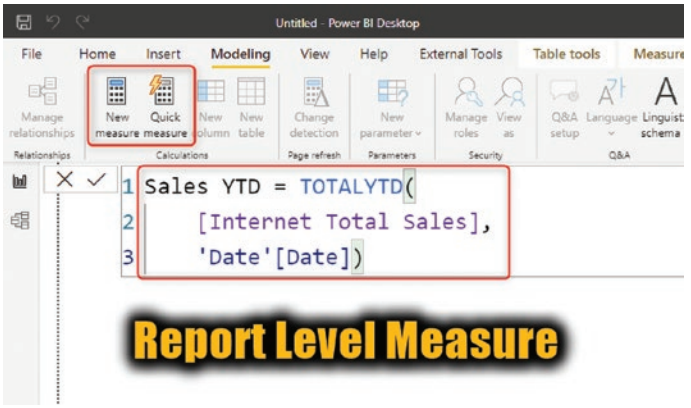


Figure 5-10. *Report-level measures*

As you can see in the example in Figure 5-10, I created a report-level measure to calculate sales year to date. The code is as follows:

```
Sales YTD = TOTALYTD(
    [Internet Total Sales],
    'Date'[Date])
```

The same code, when sent to SSAS, can be fetched with SQL Profiler, as follows:

```
DEFINE MEASURE 'Customer'[Sales YTD] =
  (/* USER DAX BEGIN */
TOTALYTD(
    [Internet Total Sales],
    'Date'[Date])
/* USER DAX END */)
EVALUATE
  TOPN(
    502,
    SUMMARIZECOLUMNS(
      ROLLUPADDISSUBTOTAL('Date'[Date], "IsGrandTotalRowTotal"),
      "Internet_Total_Sales", 'Internet Sales'[Internet Total Sales],
      "Sales YTD", 'Customer'[Sales YTD]
    ),
    [IsGrandTotalRowTotal],
    0,
    'Date'[Date],
    1
  )
ORDER BY
  [IsGrandTotalRowTotal] DESC, 'Date'[Date]
```

Report-level measures are not bound to the SSAS model. They are only for the current Power BI file; in other files they must be created again.

Report-level measures are flexible and give the users a self-service functionality. However, they reduce the governance and centralized modeling feature of Live Connection. One good use case for a report-level measure is to create conditional formatting in the visualization.

Publishing the Report and Gateway Configuration

Let's now look at how to publish and configure this report in the Power BI Service. Publishing the report here is like publishing any other report. Later in this chapter, you learn how to set up the gateway. Let's go straight to the point of adding data sources.

Create the Data Sources

Now you'll create the data sources. You might think that one gateway is enough for connecting to all the data sources in a domain. That is right. However, you still need to add a data source to that gateway for each source. The source can be a SQL Server database, and Analysis Services database, and so on. In this example, we build a data source for SQL Server Analysis Tabular on-premises. Before going through this step, I installed AW Internet Sales Tabular Model 2014 on my SSAS Tabular and want to connect to it.

To create a data source, click Add Data Source from the Manage Gateways window (you have to select the proper gateway first), as shown in Figure 5-11.

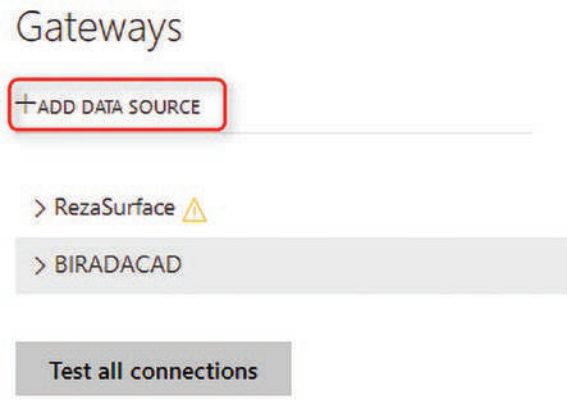


Figure 5-11. *Creating a data source under the gateway*

Then enter the data source's details, as shown in Figure 5-12. I named this data source `AW Internet Sales Tabular Model 2014`, then entered my server name and database name. Then I used Windows authentication with my domain user `<domain>\username` and the password. You should see a successful message after clicking Apply. The domain name that I used is `BIRADACAD` (my SSAS Tabular domain), and the user was `PBIgateway`, which is a user of the `BIRADACAD` domain (username: `BIRADACAD\PBIgateway`). This is an administrator for the SSAS Tabular (explained in next few paragraphs).

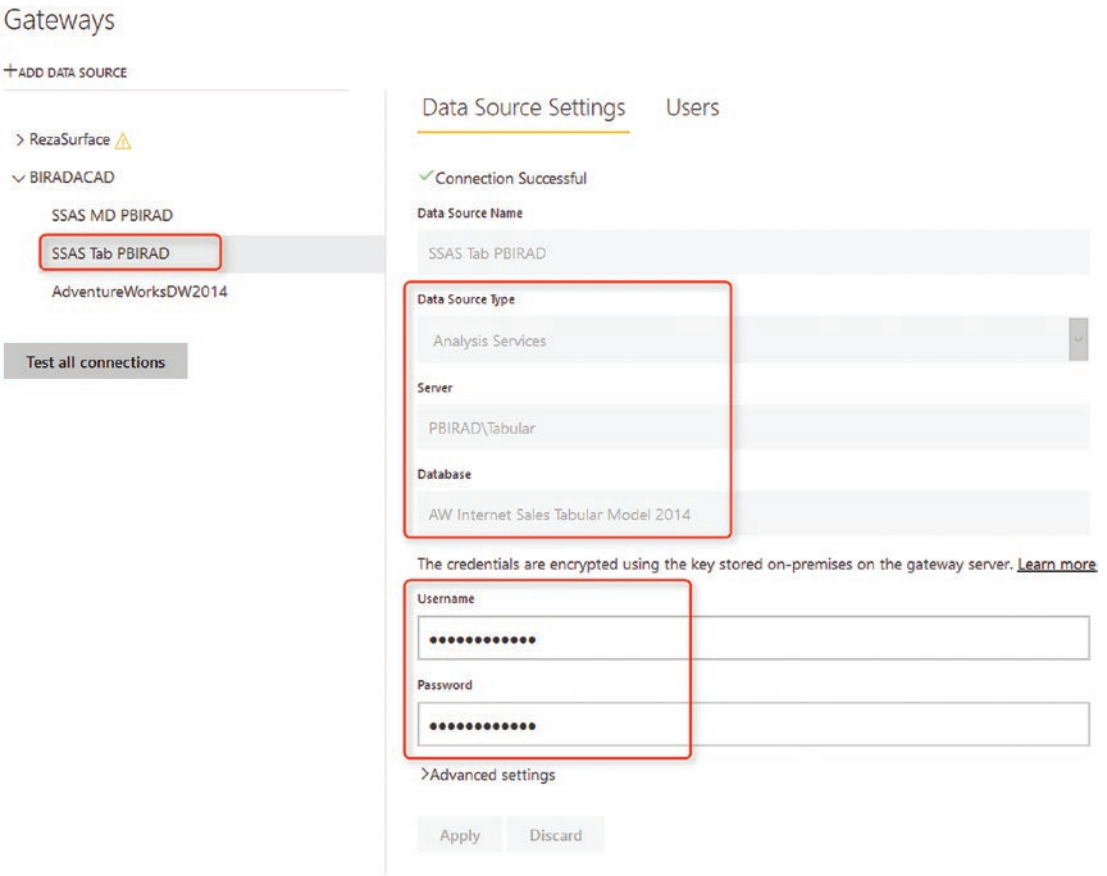


Figure 5-12. *Data source configuration*

Note that the user account that you are using should meet these conditions:

- It should be a domain user.
- The domain user should be an administrator in SSAS Tabular.

You can set the administrator for SSAS Tabular by right-clicking the SSAS Tabular instance in SSMS and then choosing the Properties option (see Figure 5-13).

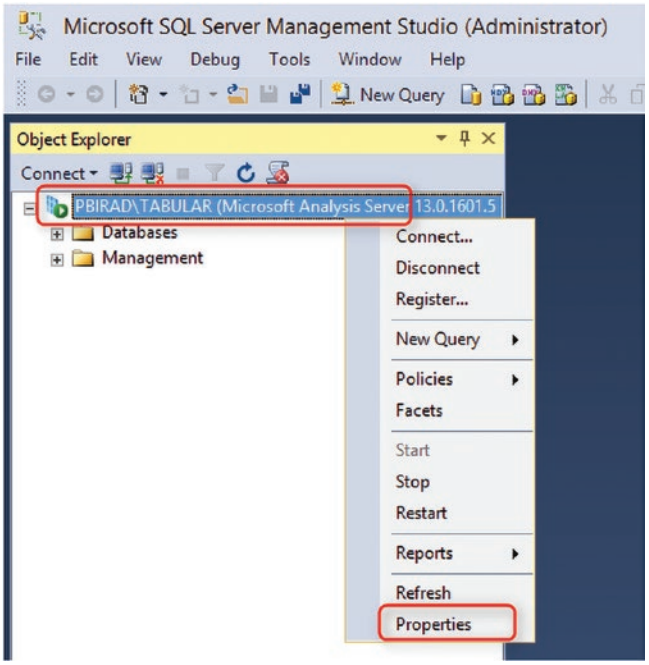


Figure 5-13. *Properties of SSAS Server*

In the Security settings tab, add the user to the administrators list, as shown in Figure 5-14.

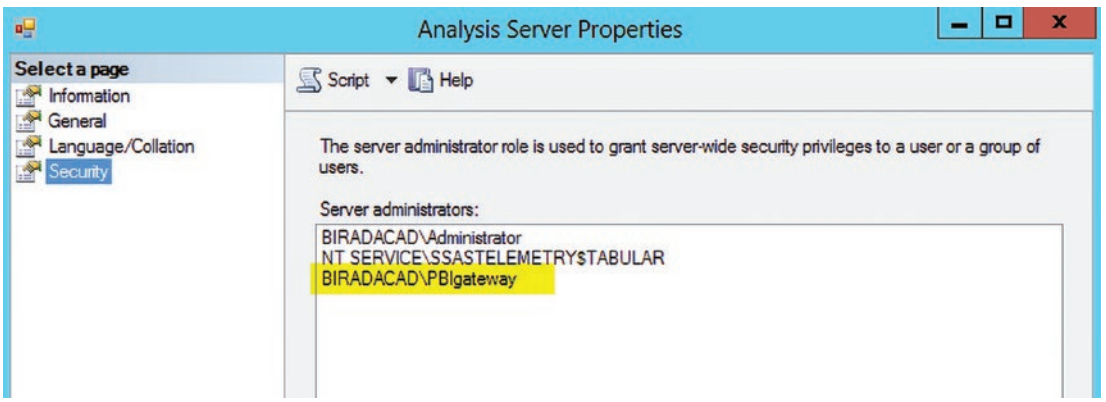


Figure 5-14. *Security settings in SSAS Server*

Using EffectiveUserName

The gateway account is used to access the Power BI cloud service from the on-premises SSAS Tabular. However, this account by itself isn't enough for data retrieval. The gateway passes the `EffectiveUserName` from Power BI to the on-premises SSAS Tabular, and the result of the query is returned based on the access of the `EffectiveUserName` account to the SSAS Tabular database and model. Figure 5-15 illustrates how this works.

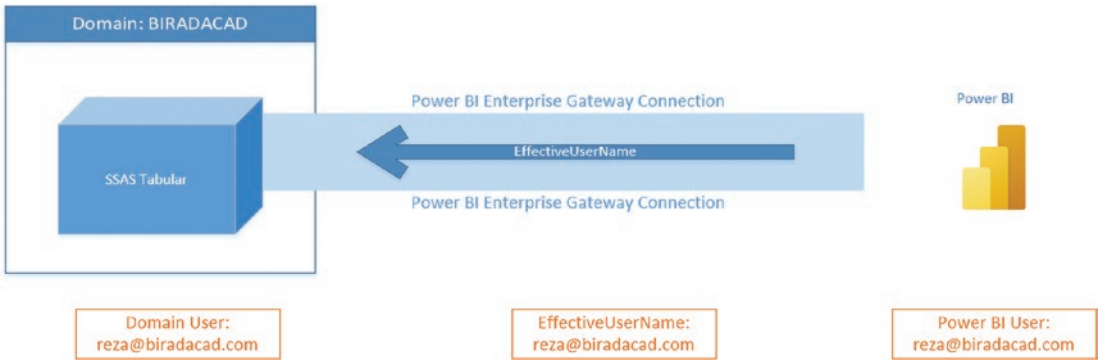


Figure 5-15. *Using EffectiveUserName*

By default, `EffectiveUserName` is the username of the user who is logged in to Power BI, or in other words, `EffectiveUserName` is the Power BI account. The Power BI account should have proper access to the SSAS Tabular database to fetch the required data. If the Power BI account is from the same domain as the SSAS Tabular, there is no problem, and security configuration can be set in the SSAS Tabular. However, if the domains are different, you have to perform UPN mapping.

UPN Mapping

Your SSAS Tabular is part of a domain (because that's how Live Connection works), and that domain might be the same domain that your Power BI user account uses. If you are using the same domain user for the Power BI account, you can skip this step. However, if the domains are different, as shown in Figure 5-16, you have to perform UPN mapping.

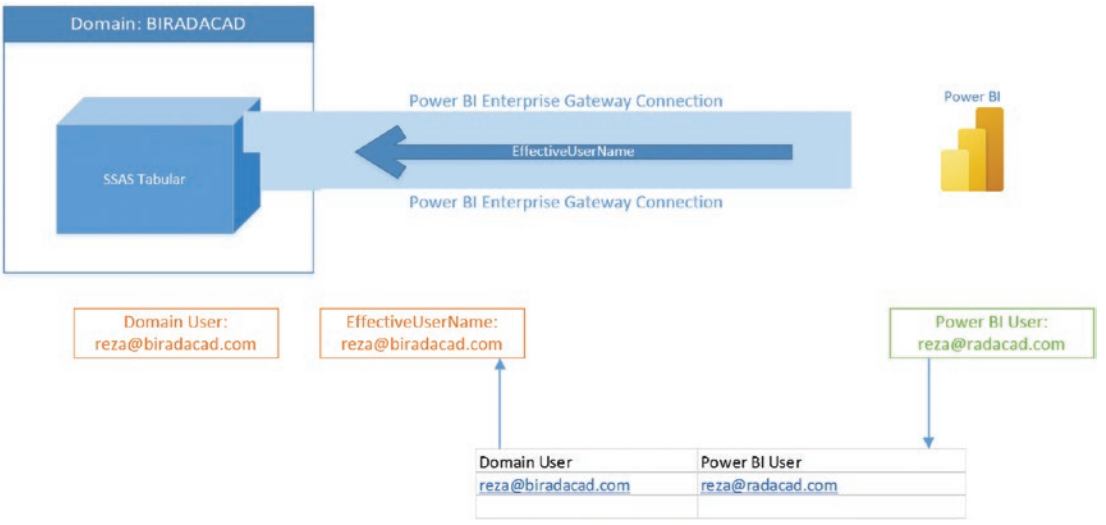


Figure 5-16. *UPN mapping table required*

UPN mapping maps the Power BI accounts to your local on-premises SSAS Tabular domain accounts. This example doesn't use the same domain account as the Power BI account, so UPN mapping must be set up, as shown in Figure 5-17.

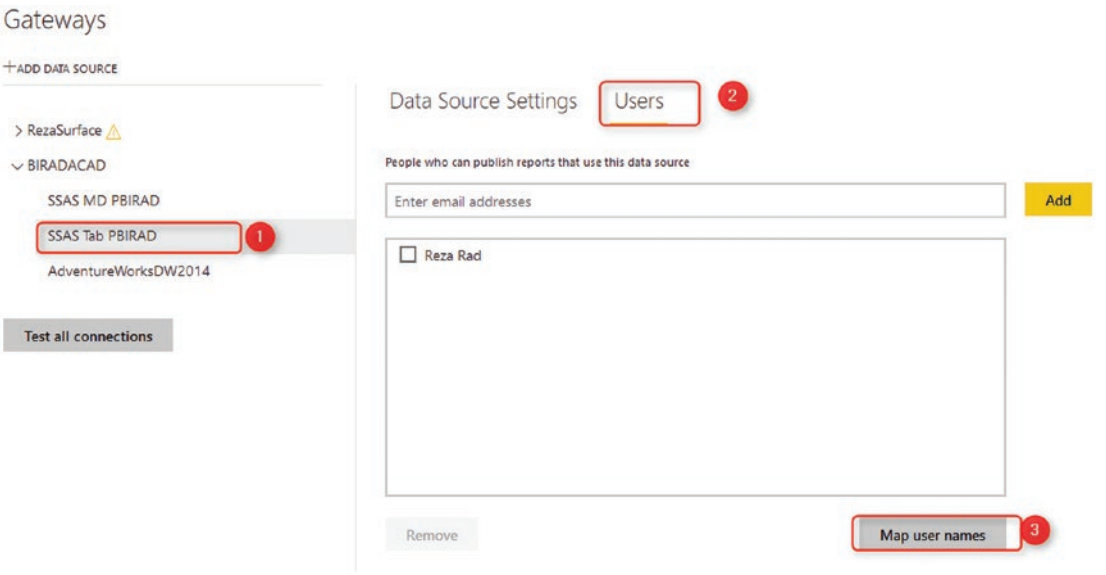


Figure 5-17. *Mapping usernames in the data source*

Then in the Mapping pane, I create a new mapping that maps my Power BI user account to `reza@biradacad.com`, which is my local domain for the SSAS Tabular server (see Figure 5-18).

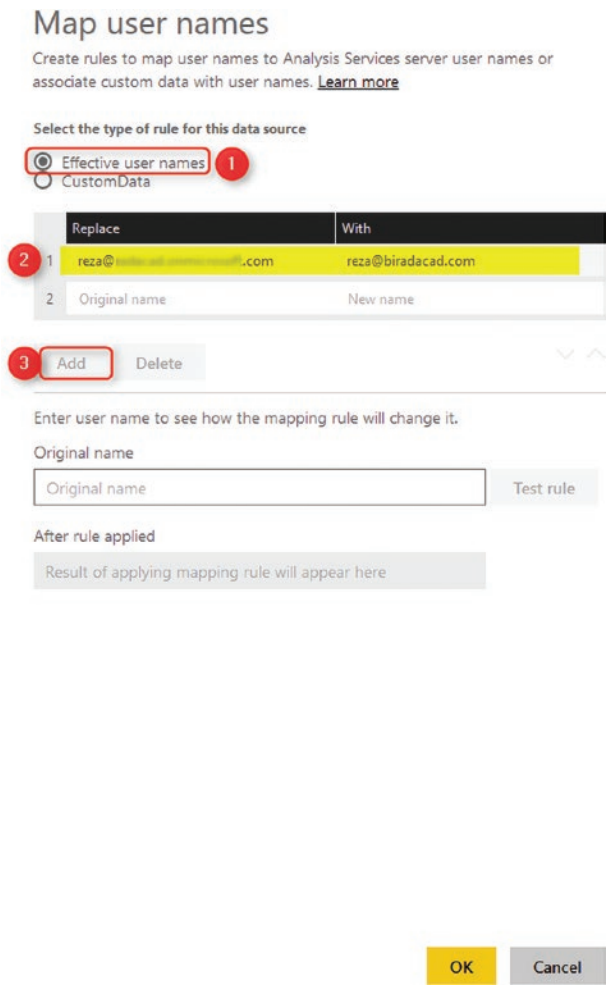


Figure 5-18. *UPN mapping setting in the Power BI Service*

With this username mapping, `reza@biradacad.com` will be passed as `EffectiveUserName` to the SSAS Tabular.

Live Connection to Power BI Service

Another type of live connection is to connect to a dataset published in the Power BI service. This dataset is then treated as an SSAS instance. The dataset in the Power BI service is hosted in a shared Azure Analysis Services (AAS) cloud environment. When you connect to it from the Power BI Desktop, it is like connecting to an instance of SSAS with a live connection.

For this option, you need to be logged in to the Power BI Desktop. You will see a list of datasets that you built, and you can choose one.

After connecting to the dataset, you will see the Live Connection message on the bottom-right side of the Power BI Desktop. This method works similarly to the connection to SSAS with Live Connection.

Connecting to another dataset in the Power BI Service with Live Connection is the recommended approach for a multi-developer environment. One user can build the model, and other users can work on various Power BI files. I explain more about this in upcoming chapters.

How Live Connection Differs from DirectQuery

One of the mistakes that many Power BI developers make is to consider DirectQuery and Live Connection the same. These two types of connections are different in many ways. Here is a list of differences between these two:

- DirectQuery is a connection mainly to non-Microsoft databases or analytical engines, or to relational databases (such as SQL Server, Teradata, Oracle, SAP Business Warehouse, and so on).
- Live Connection is a connection to four sources: SSAS Tabular, SSAS Multi-Dimensional, Azure Analysis Services, and Power BI dataset.
- DirectQuery still has limited Power Query functionality with some data sources (such as SQL Server databases).
- Live Connection has no Power Query features in it.
- You can create some simple calculated columns in DirectQuery; these will be converted to T-SQL scripts behind the scenes.
- You cannot create calculated columns in Live Connection.
- You can use report-level measures and leverage all DAX functions in a Live Connection instance.
- In DirectQuery mode, you can have limited measure abilities. For more complex measures, you must check the performance (and some of the functions such as parent-child functions are not available). Some complex measures might slow performance down significantly.
- DirectQuery mode is usually slower than Live Connection.
- A Live Connection instance is usually less flexible than DirectQuery mode.

As you see, the list explains how these two types of connections differ.

Summary

In this chapter, you learned about Live Connection. This type of connection is only available for four data sources—Azure Analysis Services, SSAS Tabular, SSAS Multi-Dimensional, and Power BI datasets. This type of connection is much more limited than DirectQuery, because you have no access to the Power Query Editor.

However, Live Connection provides a better solution than DirectQuery, because the ability to write DAX code is fully possible if the SSAS Tabular is used as a data source. With Live Connection, you get the benefits of both worlds—because data is not stored in the Power BI, the size limitation of Import Data does not apply. Also, because the SSAS Tabular can leverage DAX, and the analytical power of Power BI is based on DAX, the analytical engine of this solution is very efficient.

A Live Connection instance is faster than DirectQuery in most cases. However, it can be still slower than Import Data in the performance (depending on the SSAS server specification and the connection between the report and the server). The recommendation is to start with Import Data, and if it's not possible, use Live Connection. If neither of those can be used (for example if you are working with a huge amount of data, and the SSAS is not available because the Microsoft toolset is not used for analytics in your company), choose the last option, which is DirectQuery. However, always remember to use DirectQuery through a composite mode.

CHAPTER 6

Composite Model

In the early days of developing Power BI solutions, you could choose the DirectQuery or Import Data connection types, but not both. In 2018, Power BI added a breakthrough feature called *composite model*. Using this feature, you can have part of your Power BI dataset connected directly to a data source, and another part imported. Along with this change, you can also choose the storage mode for each entity. In this chapter, you learn about composite model—what it is and how it works.

What Is Composite Model?

A composite model in Power BI means that part of your model can be a DirectQuery connection to a data source (for example, SQL Server database), and another part can use Import Data (for example, an Excel file). Previously, when you used DirectQuery, you couldn't even add another data source into the model. See Figure 6-1.

With composite model, your very large tables can use a DirectQuery connection, without needing to import them, and you can import smaller tables so they are quickly accessible.

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_6

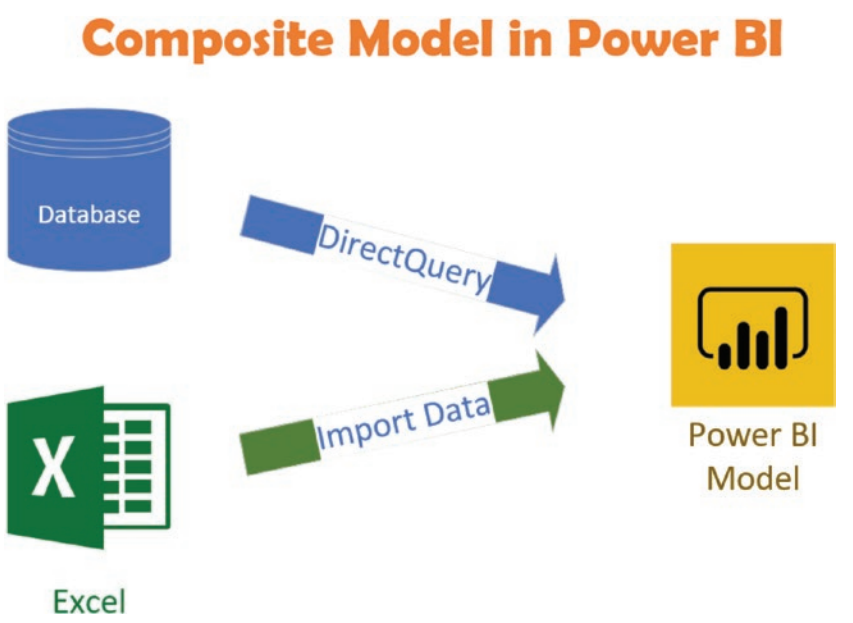


Figure 6-1. *Composite model combines DirectQuery and Import Data*

Why Use Composite Model?

To answer this question, you first need to consider the benefits of the Import Data and DirectQuery modes. Import Data is good for super-fast data analysis and is flexible, while DirectQuery is good for big data tables and data freshness. See Figure 6-2.

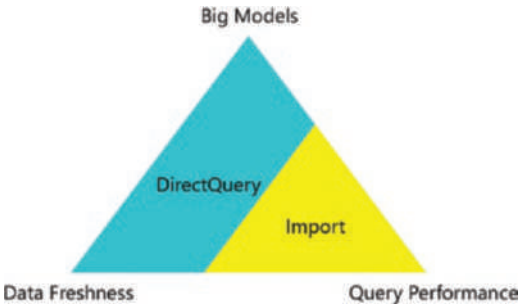


Figure 6-2. *DirectQuery and Import Data have different advantages*

Composite model combines the good things of Import and DirectQuery into one model. Using composite model, you can work with big data tables in DirectQuery, and still import smaller tables using Import Data. See Figure 6-3.

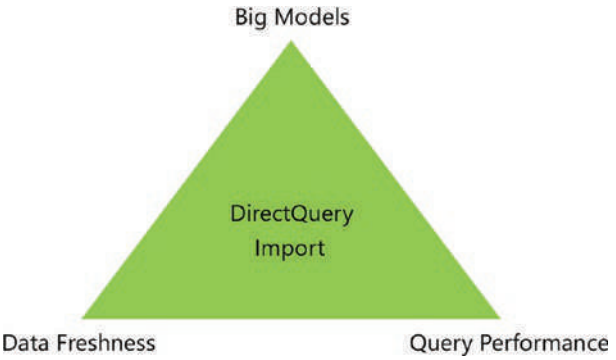


Figure 6-3. *Composite model improves performance and works with big data tables at the same time*

Using composite model, you can have tables with different connection types, some using Import Data and some using DirectQuery. See Figure 6-4.

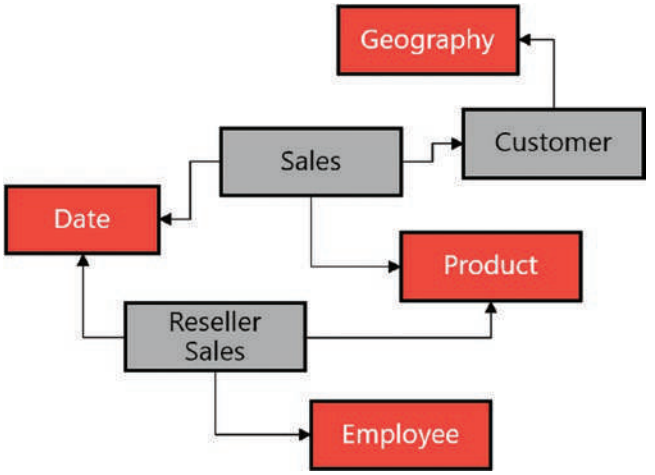


Figure 6-4. *In composite model, some of the tables can be in Import Data mode, and others can use DirectQuery (the red tables use Import Data, and the gray tables use DirectQuery)*

Having big data tables with DirectQuery and smaller tables with Import Data is much better than doing everything purely with DirectQuery, because in pure DirectQuery mode, performance is slow. In composite model, the smaller tables can be used as Import Data to improve the performance. This also comes with a new storage mode, called *Dual*, which you learn about it a bit later in this chapter.

How Does Composite Model Work?

Figure 6-5 depicts a Power BI file using a DirectQuery connection to a SQL Server database (`AdventureWorksDW` in this example).

Figure 6-5. *DirectQuery source*

In Figure 6-6, I selected the `FactInternetSales` table from this database.

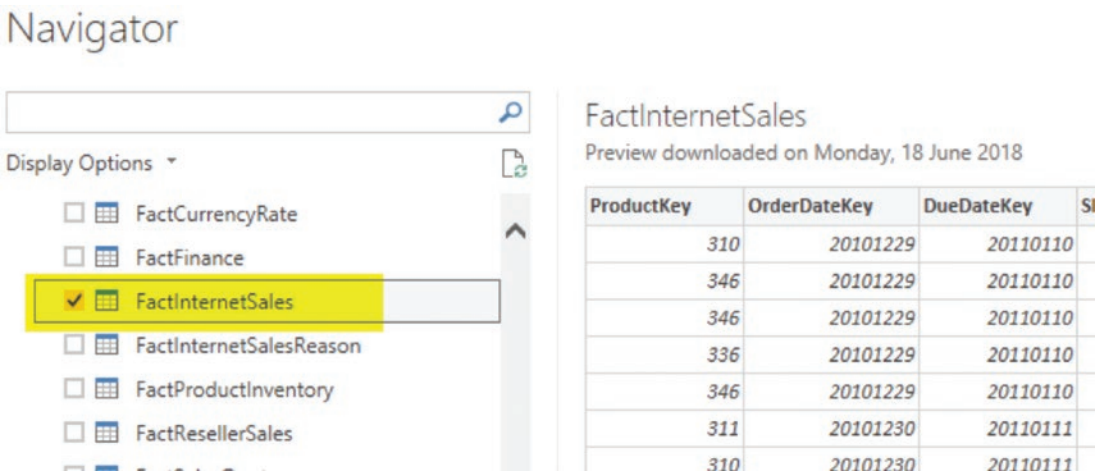


Figure 6-6. *Selecting DirectQuery tables*

In the Power BI Desktop, after loading the table, you can see (on the right-bottom corner) that it uses using the DirectQuery storage mode (see Figure 6-7).

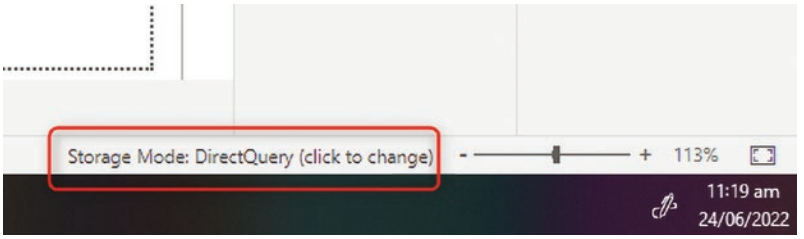


Figure 6-7. *Using DirectQuery storage mode*

The second table gets data from Excel, as shown in Figure 6-8.

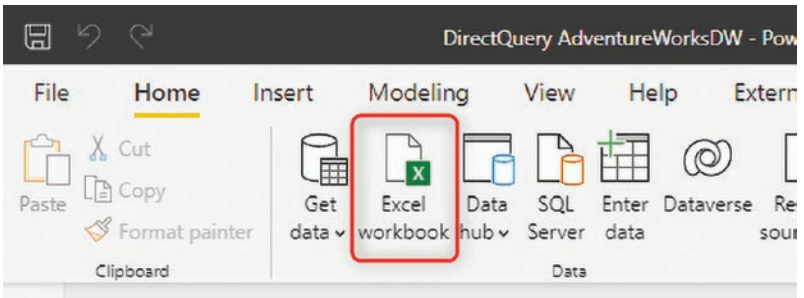


Figure 6-8. *Getting data from an Excel workbook*

Next, select `DimCustomer` from the Excel file (this Excel file is the Excel version of the `AdventureWorksDW` database). As soon as you bring a table in from another data source, you will be notified that there is a potential privacy risk, whereby the data from one data source may be used to pass parameters into another data source, as shown in Figure 6-9.

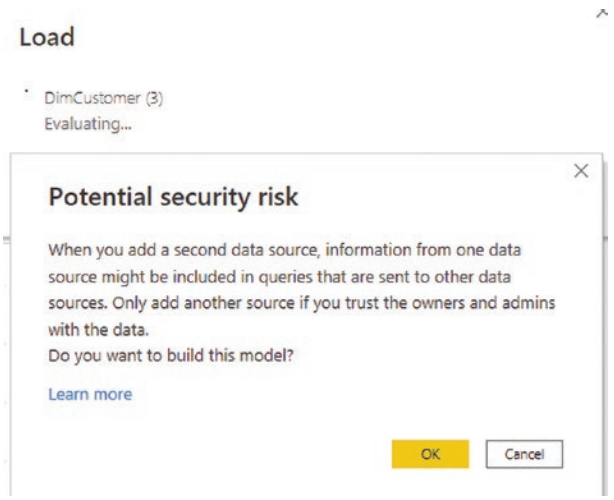


Figure 6-9. *Potential security risk warning when combining data from different sources*

After you click OK, the new data table is loaded into Power BI, and you can see in Figure 6-10 that the mode changes to Mixed.

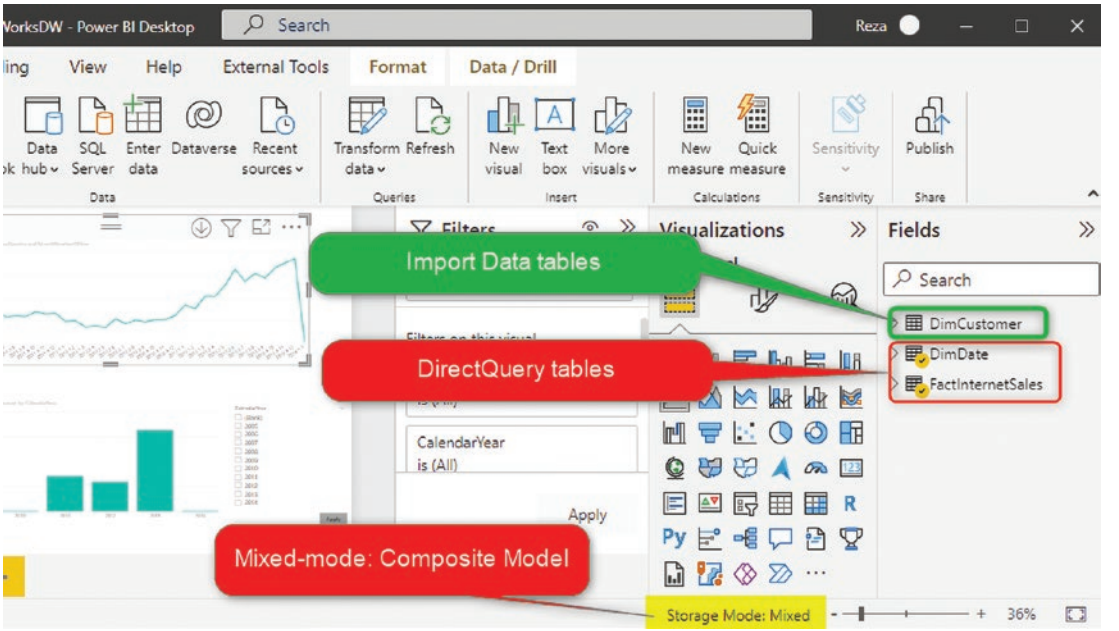


Figure 6-10. *Mixed mode*

The mixed mode of composite model indicates that the fact table is *not* loaded into Power BI. It will be queried each time from the SQL Server database table. The dimension, however, is already imported into the Power BI model and needs to be refreshed each time. If you refresh your model, you will see that the only table(s) that are refreshed are the Import Data tables, as shown in Figure 6-11.

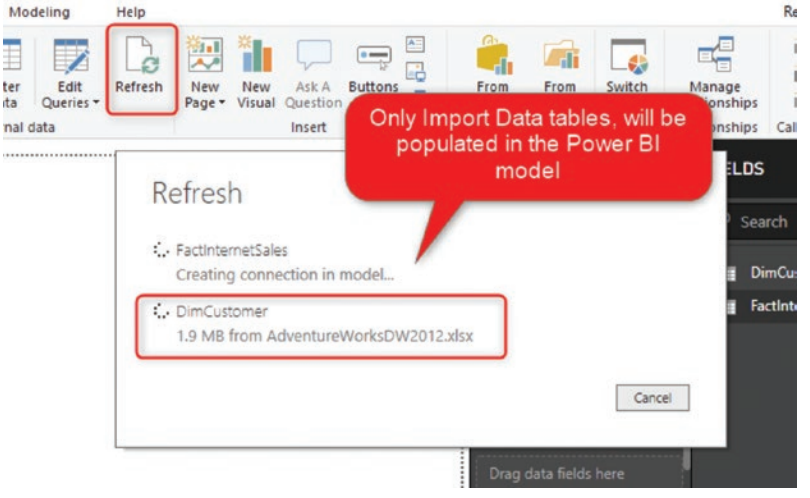


Figure 6-11. *Only Import Data tables are loaded into Power BI*

Using this option, you can see the Data tab (which you can't use in mere DirectQuery mode), and you can see only the Import Data tables there (see Figure 6-12).

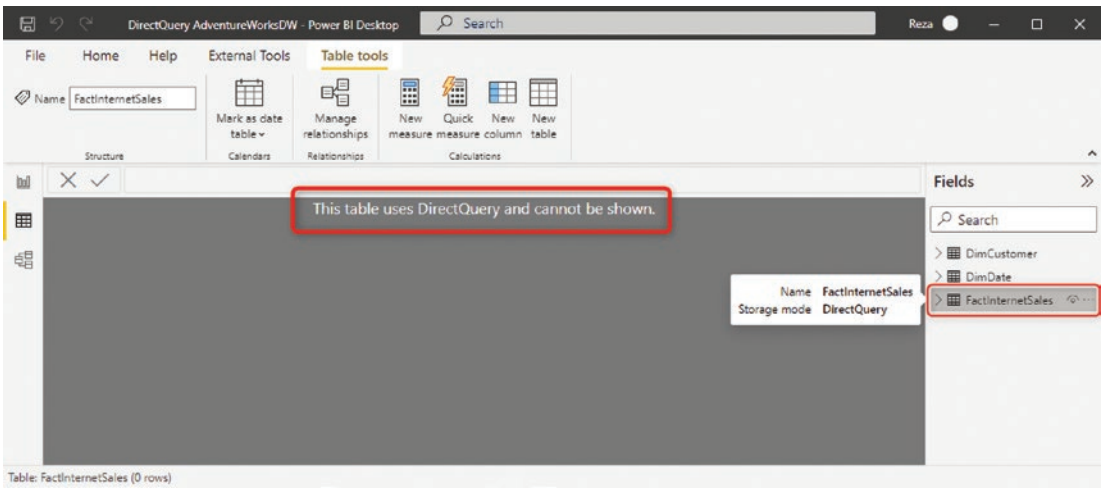


Figure 6-12. *The Data tab shows import tables only*

You can have the big data tables of the model come from a DirectQuery source, and it doesn't matter if they are billions of rows, because they are not loaded into Power BI. Small tables can be imported into Power BI for better performance.

Dual Storage Mode

With composite model, you may wonder where the data is stored in every table and how it works behind the scenes. Each table in Power BI has a storage mode. You can select a table and determine the storage mode for that table. The storage mode is Import, DirectQuery, or Dual.

If you go to the Model tab in the Power BI Desktop and click a table, you can see the Storage mode in the Properties pane, under the Advanced section. As Figure 6-13 demonstrates, there are three options: Import, DirectQuery, and Dual.

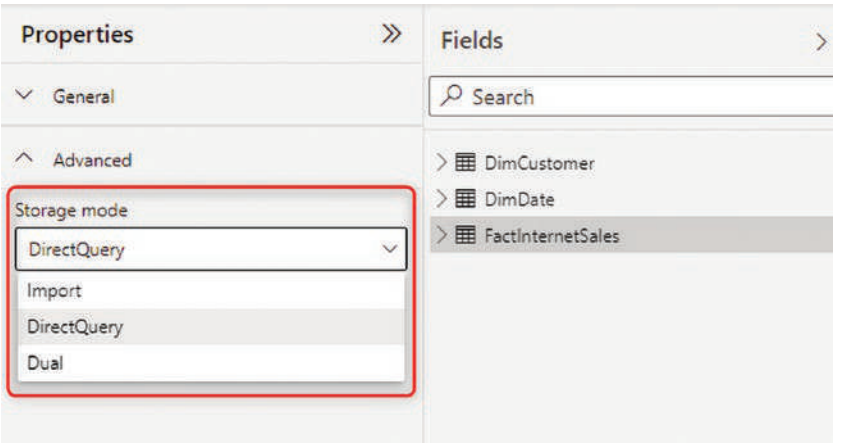


Figure 6-13. *Storage modes*

Import Data and DirectQuery are understandable enough. With Import Data, every time you refresh the data, it will be imported from the data source. DirectQuery queries directly from the source no matter what visual you use. So what is Dual mode?

If you try to change the mode of an import table to Dual, as shown in Figure 6-14, you'll see that this change is not possible.

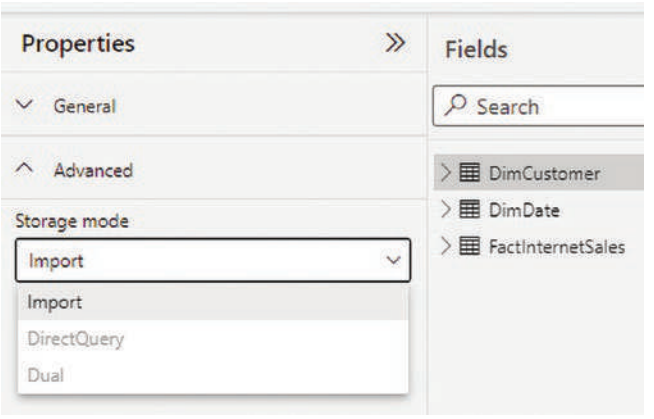


Figure 6-14. *Storage mode cannot be changed for import tables*

Dual mode is only an option for DirectQuery data sourced tables.

When you change the storage mode of a DirectQuery table to Dual (as demonstrated in Figure 6-15), it will create a copy of that table in the Power BI model. But unlike with Import Data, there is a copy in memory, and there is still the main table in the DirectQuery source.

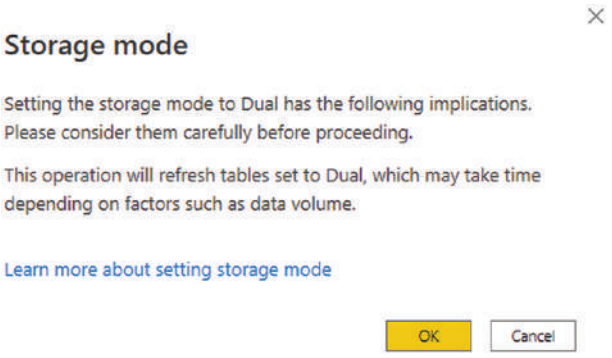


Figure 6-15. *Changing to Dual storage mode*

Dual storage mode works like DirectQuery or Import Data, depending on what other tables related to it are used in the visualization. If you have a visual that has columns from Import tables and a Dual table, the Dual table will act like an Import. If you have a visual that has columns from DirectQuery tables and a Dual table, the Dual table will act like a DirectQuery table. This ensures that you get the best performance when working with smaller tables (such as dimension tables) in Dual storage mode.

Modeling with the Power BI Composite Model

The following sections outline some points to consider when performing data modeling with a Power BI composite model.

Relationships in the Power BI Composite Model

You can create relationships between tables from different data sources and using different storage modes. Figure 6-16 shows how `DimCustomer` (Import Data) is connected to `FactInternetSales` (DirectQuery) and `DimDate` (Dual).

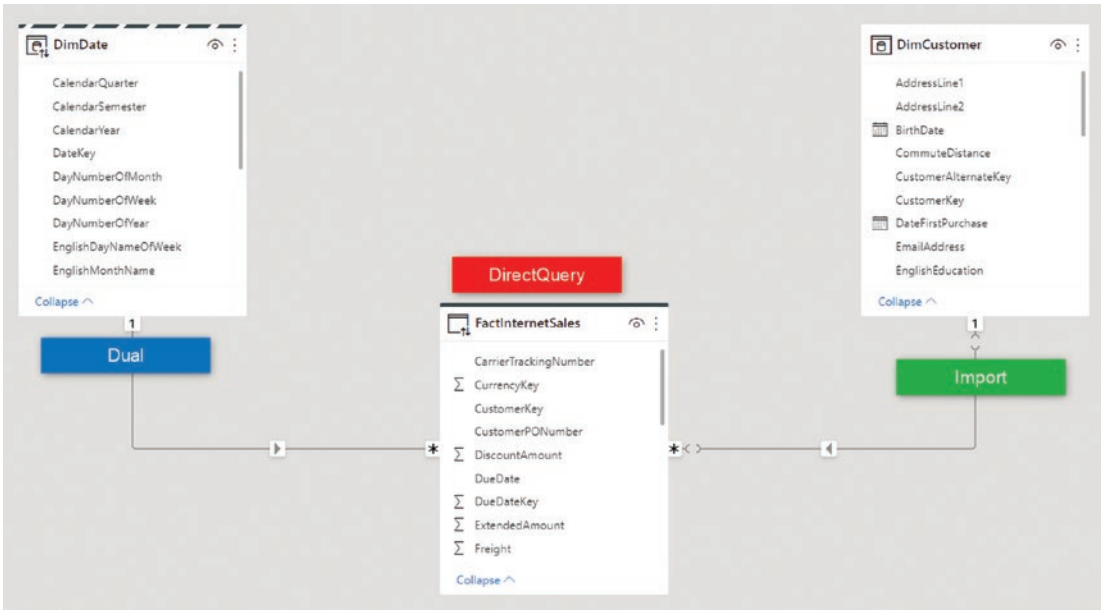


Figure 6-16. *Relationships in a Power BI composite model*

The cardinality of the relationship can be anything you want—one-to-many, many-to-one, one-to-one, and many-to-many.

Calculations and DAX

Calculated tables are available in composite model. These tables will be in Import mode, and their expression can be anything that DAX allows. You can also create calculated tables using the Power Query Editor.

Calculated tables are refreshed at the scheduled refresh time of the Power BI dataset.

Depending on where you create the calculated columns, they have some limitations. If you create a calculated column in an Import table or a calculated table, there will be no limitations. However, if you create a calculated column in a DirectQuery table, only columns from the same table can be used.

DirectQuery to Power BI Datasets

In composite model, you can create a DirectQuery connection to a Power BI dataset. This is different from a live connection to the Power BI dataset. A DirectQuery to a Power BI dataset enables you to import the Power BI dataset into the composite model, which enables you to do further data modeling on the chained dataset (see Figure 6-17). This concept is explained in a later chapter of this book.

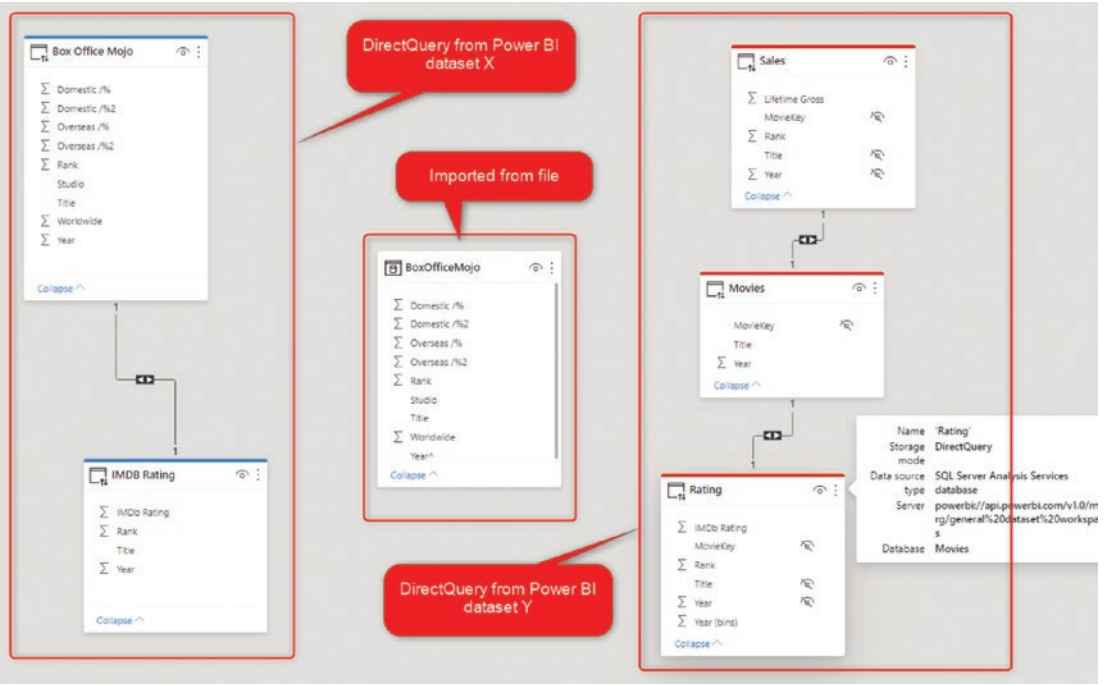


Figure 6-17. *DirectQuery to a Power BI dataset in composite model*

Summary

In summary, combining DirectQuery tables and Import Data tables in one Power BI model is called composite model or mixed mode. With this feature, you can leverage the option for using Dual storage mode to keep a copy of a DirectQuery table in memory. This is good for answering quick slicing and dicing questions.

DirectQuery was not a common method when it came to working with data in Power BI, especially because this mode couldn't originally combine multiple data sources. But now with composite model, DirectQuery is faster. If you have big data tables, I definitely recommend exploring composite model. Composite model can even go one step further in performance when aggregations are used.

Composite model is better than a pure DirectQuery approach. This option gives you performance and big data, all in one data model.

You learn more about this approach in the following chapters.

CHAPTER 7

DirectQuery for Power BI Datasets and Analysis Services

One of the recent features added to Power BI is the DirectQuery for Power BI datasets and Analysis Service. In other words, it's a composite model for Power BI datasets. This chapter explains what this means, how it's useful, and how it works in detail.

Power BI with Import Data

Most Power BI users import data from their sources into Power BI. This option provides the ultimate flexibility in modeling. If you use this approach, you can customize the tables using Power Query transformations, create relationships the way you want, write DAX calculations, and finally do visualizations. The Import Data mode of Power BI gives you untainted power and flexibility. In an earlier chapter of this book, I explained the Import Data mode in detail.

One Important Problem with Import Data

Importing data is fantastic, but nothing comes without side effects. With Import Data, one of the big problems is that every self-service user will start to import data into their own data model, and soon you end up with many data models, a lot of duplications, and silos of Power BI models everywhere. Many of those models might do the same thing over and over. See Figure 7-1.

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_7

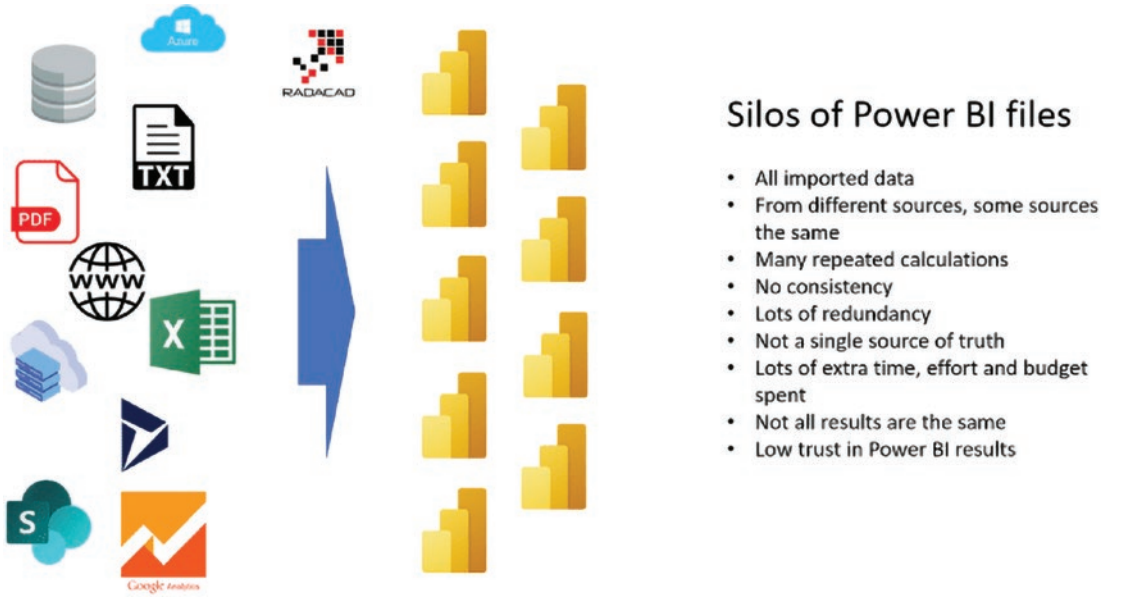


Figure 7-1. *The problem with silos of Power BI files*

Power BI Using Live Connection

An approach that comes in very handy to solve these problems is to have a Power BI report with a live connection to another dataset. This dataset can be a Power BI dataset hosted on the Power BI Service, an Azure Analysis Services dataset, or a SSAS dataset hosted on-premises. The Power BI report in this case would visualize the data from the main model.

Using Power BI as a live connection means that users cannot change the model and build their own. So there won't be a problem with inconsistency and there won't be silos of data models. There is one or a few central models only. In an earlier chapter of this book, I explained what a live connection means for Power BI.

Big Limitation of Live Connection

Live Connection is great for governing the Power BI implementation and avoiding silos. However, it is not a useful option for self-service users, because they cannot import other data sources and build variations on top of it. The most they can do is create report-level measures. A chapter later in this book explains how to import data from a Power BI dataset so that these types of users can still leverage some of the calculations done in the central model.

DirectQuery for Power BI Datasets or Analysis Services Datasets

Now that you have the background information, let's talk about the main topic. This feature, which was first announced in December 2020, allows you to create a new dataset that uses the central dataset. You can also bring your variations into it. See Figure 7-2.

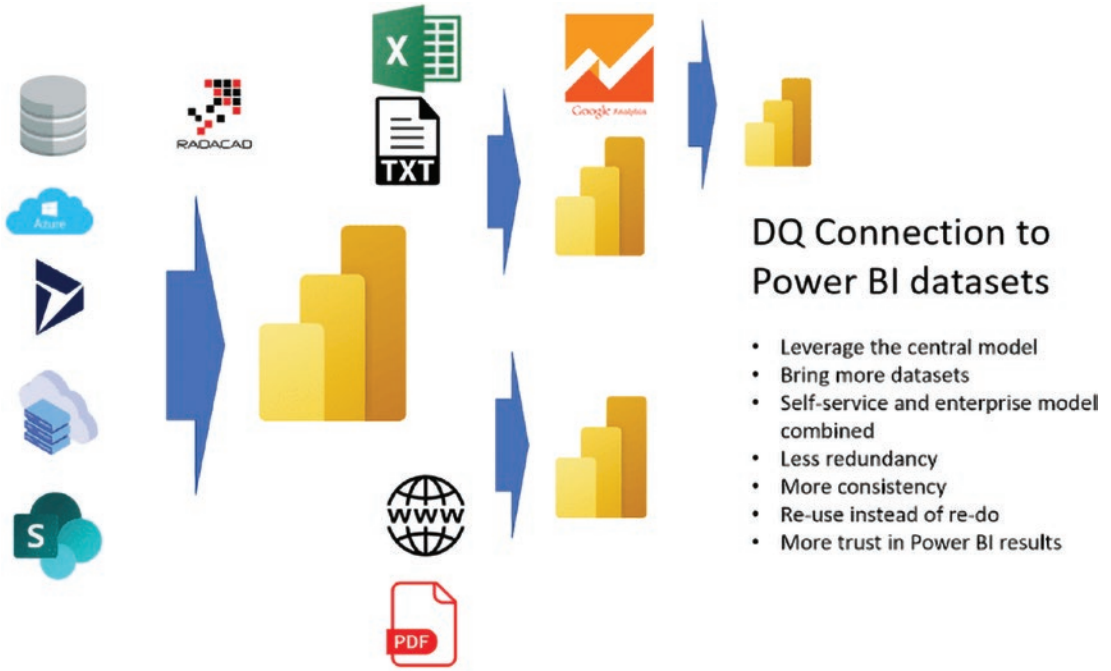


Figure 7-2. *DirectQuery for Power BI Datasets and Analysis Services*

This feature allows self-service users to reuse the existing model and add more options to it in a new model. Instead of reinventing the wheel, they add more components to the existing model. This is where the two worlds of self-service and Enterprise BI come together.

As an example, say a developer on the BI team builds a data model that has some entities about Sales from the SQL Server database, Dynamics 365, Azure blob storage, SharePoint, and a few other sources. This model can be used directly to create a report, or it can be used by a self-service user who wants to use other data from Excel and CSV files and have a more complete visualization for a specific use.

The model built by that self-service user can also be customized by another self-service user. As you see in Figure 7-2, this feature provides layers of implementation and development, rather than having to redo everything.

Why Is This Feature a Big Deal?

This feature allows you to:

- Leverage the central model
- Bring in more datasets

- Combine the self-service and enterprise models
- Have less redundancy
- Have more consistency
- Reuse instead of redo
- Have more trust in the Power BI results

This feature is more than a feature—this is where the two worlds of Enterprise BI and self-service BI come together, as illustrated in Figure 7-3. Instead of blocking each other's way and going their own ways, now these two help each other reach a common goal.

Figure 7-3. *Self-service BI and Enterprise BI come together with DirectQuery for Power BI Datasets and Analysis Services*

How It Works Under the Hood

Using this feature, you'll have a new copy of the data model that gets part of the data as a DirectQuery source from the Power BI dataset connection, and another part of the data from the data sources you import. There are limitations to this function at the time of writing this chapter. However, these limitations will be lifted one by one, gradually.

Not from My Workspace

The first thing you need to consider is that this feature doesn't work for datasets located in My Workspace, as shown in Figure 7-4. Right now, if you host a dataset in My Workspace, a connection to that dataset will be a normal live connection and you can't combine other data sources with it.

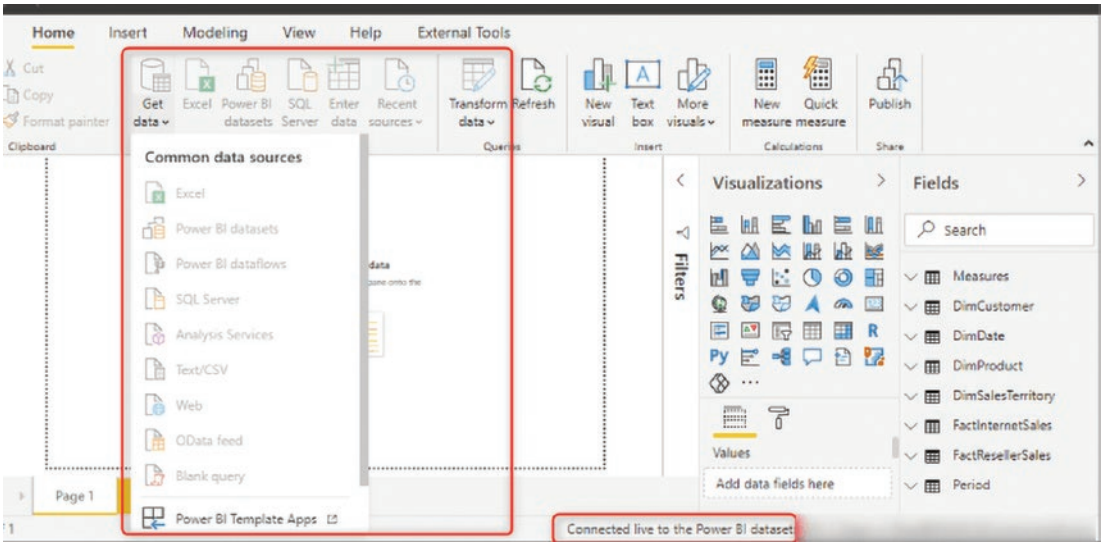


Figure 7-4. *DirectQuery to Power BI Dataset is not supported for datasets hosted in My Workspace*

This limitation will likely be lifted in the future, based on mentions in Microsoft's documentation. However, even if this is not a limitation when you read this book, I strongly advise you not to use it.

Imagine someone built a dataset and published it to My Workspace. Then others used that dataset to build something on top of it. If that person leaves the organization, it will be hard to get a hold of the original dataset. Having a dataset that is used as a source of other datasets in My Workspace is not recommended.

Creating a DirectQuery to Power BI Dataset

Creating a DirectQuery connection to a Power BI dataset is simple. From Get Data, choose Power BI Datasets, as shown in Figure 7-5.

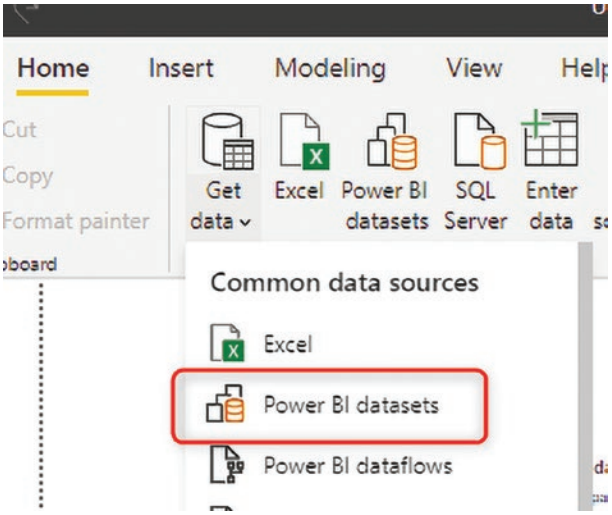


Figure 7-5. *Getting data from a Power BI dataset*

This will give you a normal live connection to the Power BI dataset, as shown in Figure 7-6.

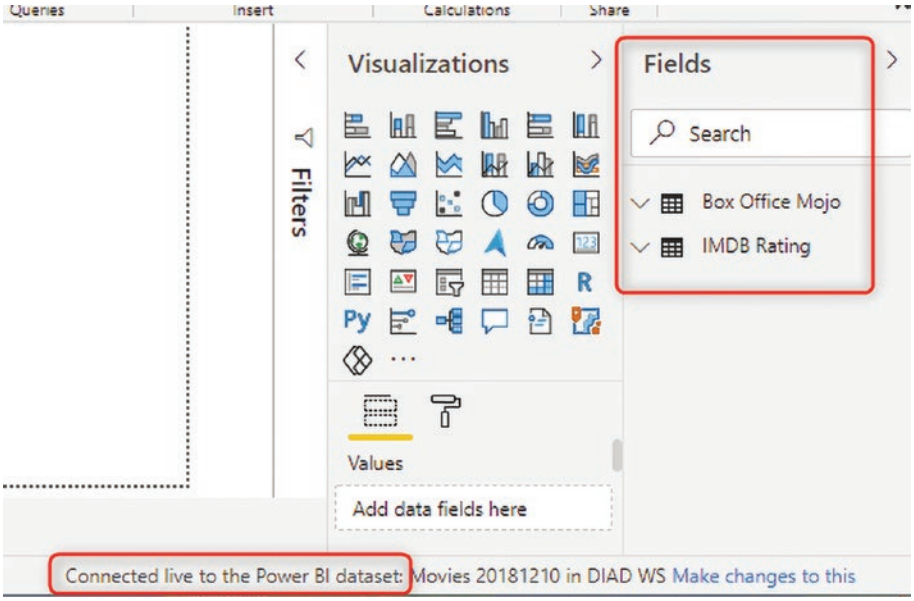


Figure 7-6. *Live connection to the Power BI dataset*

You can now get data from another dataset, as shown in Figure 7-7.

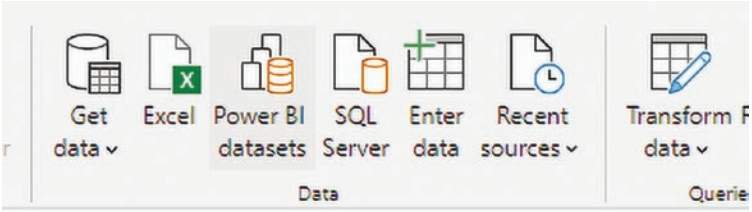


Figure 7-7. *Getting data from another dataset*

Or you can click Make Changes to this Model, as shown in Figure 7-8.

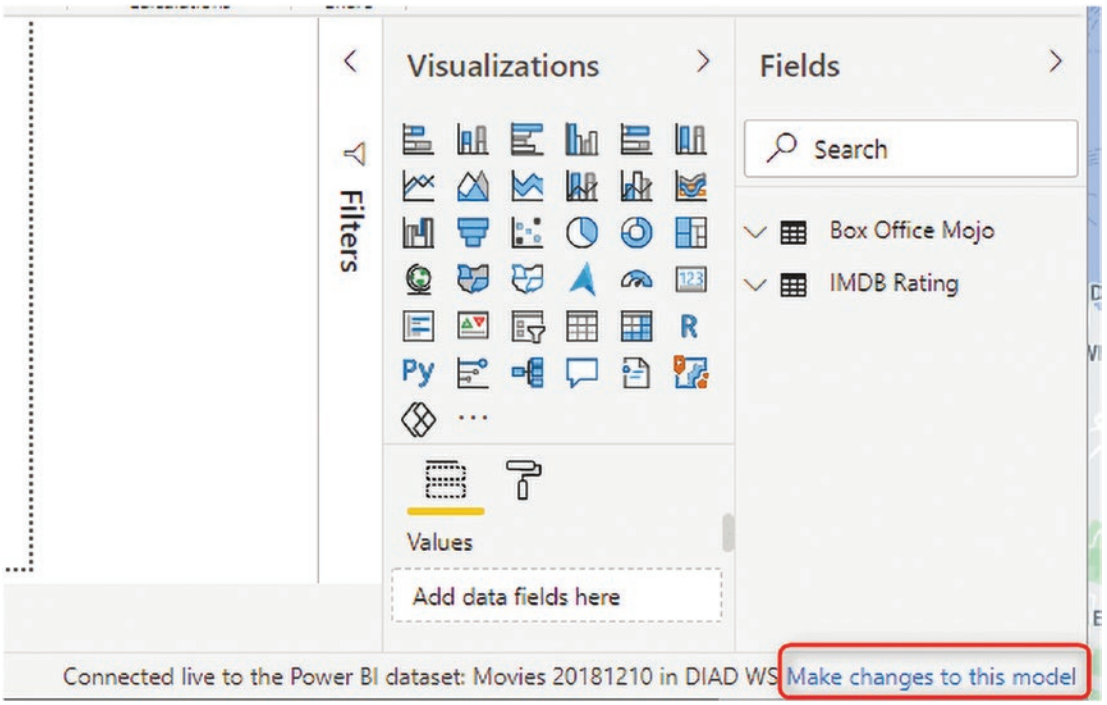


Figure 7-8. *Making changes to the live Power BI dataset connection model*

Using either of these options will show you the message that this operation will create a local dataset copy with the ability to change, as Figure 7-9 indicates.

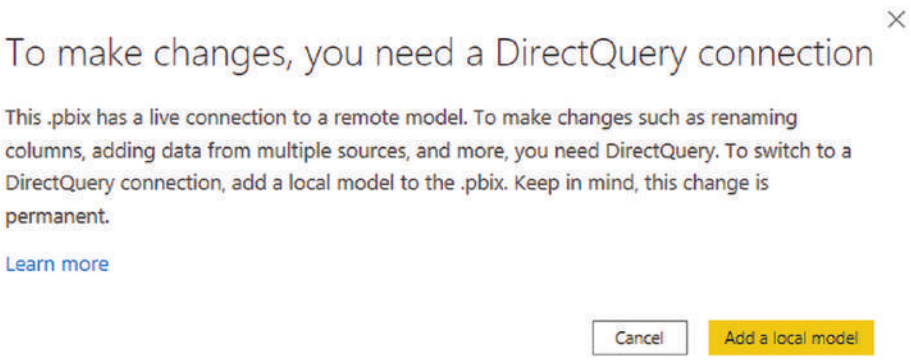


Figure 7-9. *Adding a local model (DirectQuery to Power BI dataset)*

This action will change the storage mode of your Power BI dataset tables to DirectQuery, as shown in Figure 7-10.

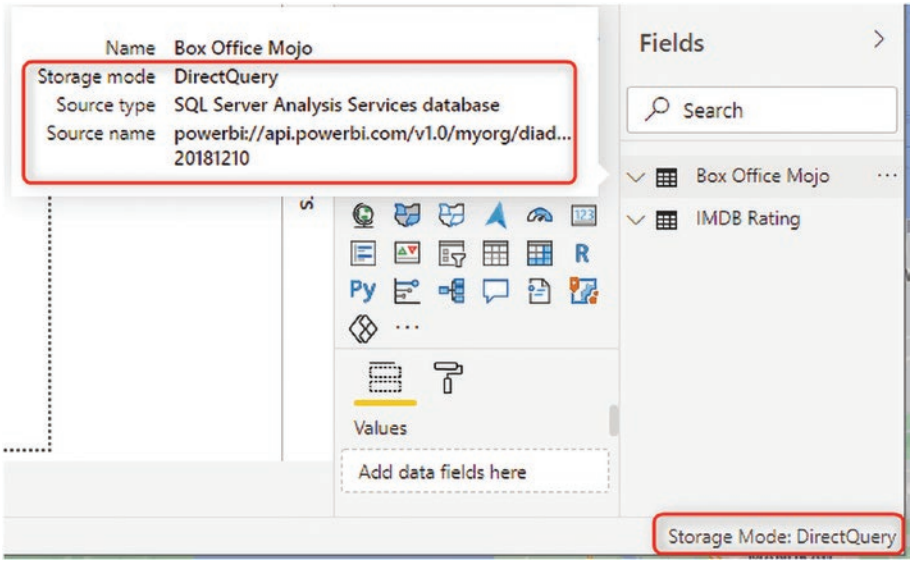


Figure 7-10. *DirectQuery to Power BI dataset*

DirectQuery to Power BI Dataset vs Live Connection to Power BI Dataset

With Live Connection to Power BI dataset, the only thing you can do is create report-level measures. Anything else has to be done in the original model.

With DirectQuery to Power BI dataset, you can add other data sources, columns, measures, tables, and so on, as shown in Figure 7-11. In other words, you can build a new model on top of the existing one with the changes you want.

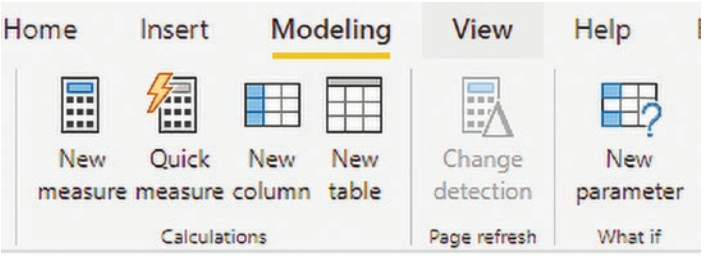


Figure 7-11. *Changes in the model using DirectQuery to Power BI dataset*

Multiple DirectQuery Connections Are Supported

You can have part of the model coming from one Power BI dataset, and another part coming from another dataset, as shown in Figure 7-12. You can also import some other tables. The new model view of Power BI shows these very nicely, with different color headings for each category.

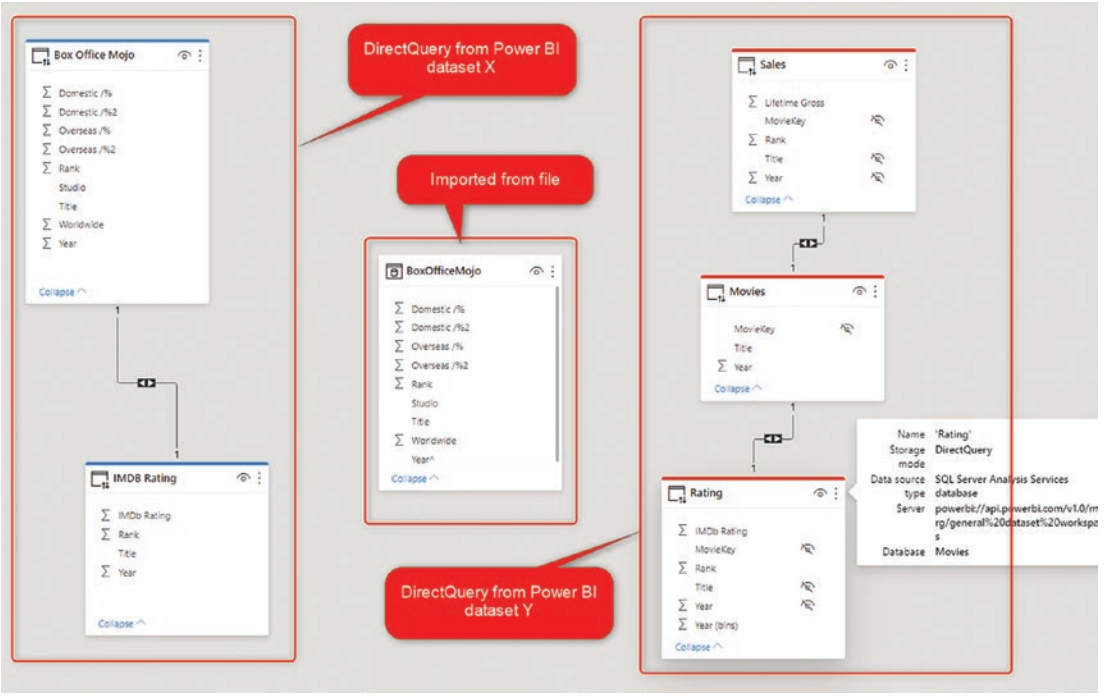


Figure 7-12. *Composite model using DirectQuery to Power BI dataset*

Creating a Relationship Between Different Data Sources

As you can see in Figure 7-13, you can easily create relationships between tables from different sources.

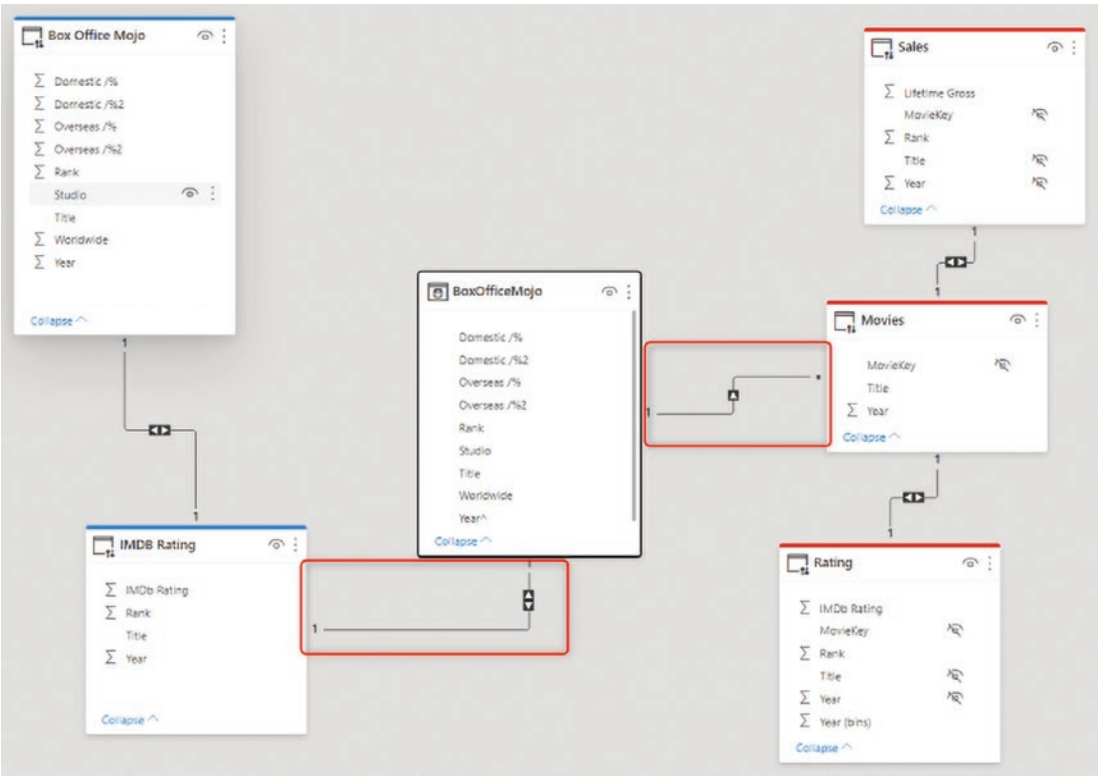


Figure 7-13. *Creating relationship in composite model using DirectQuery to Power BI dataset*

Publish to Service

The new data model, when published to service, requires some steps compared to other types of Power BI datasets. The following sections explain these steps.

Error: There Is No Gateway

For the visuals that have anything from the DirectQuery Power BI dataset, you may see an error saying, "There is no gateway," as highlighted in Figure 7-14.

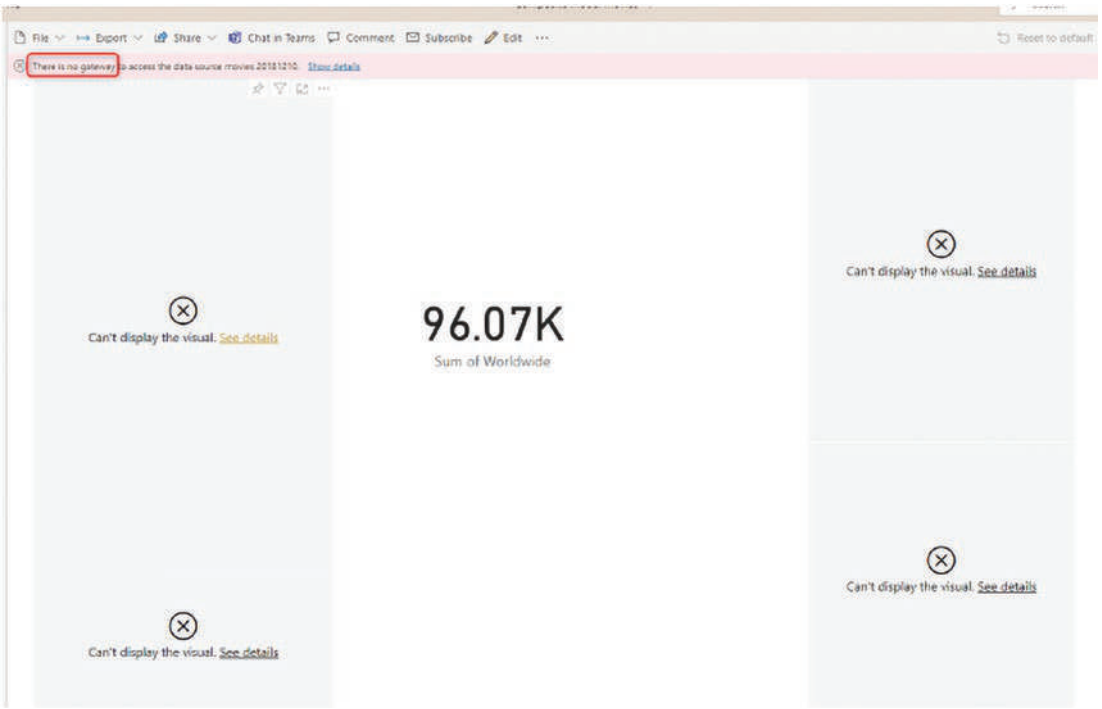


Figure 7-14. *The "there is no gateway" error for DirectQuery to Power BI datasets*

To solve this error, go to the dataset's setting, as shown in Figure 7-15.

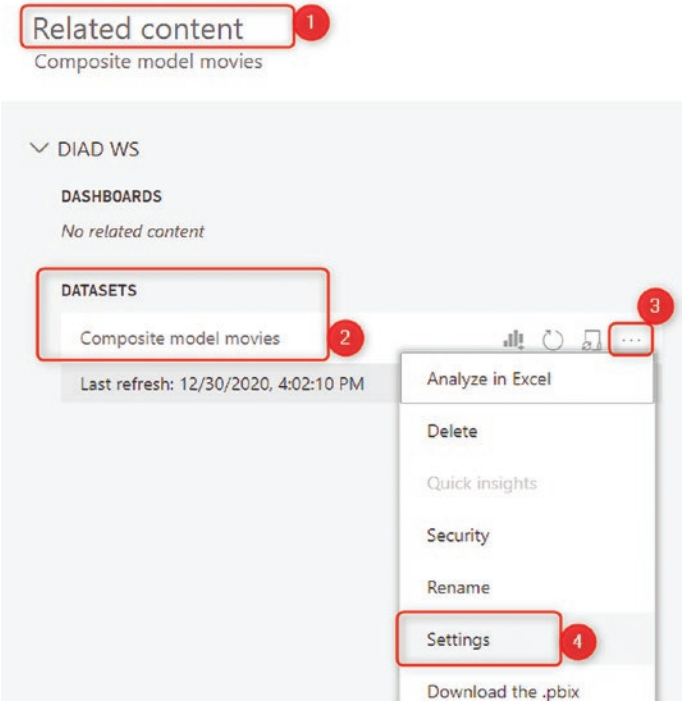


Figure 7-15. *Dataset's settings*

Remember that you need a gateway for any data source located on-premises and imported, as indicated in Figure 7-16. Select that in the gateway. Do not select a gateway option for your Power BI datasets.

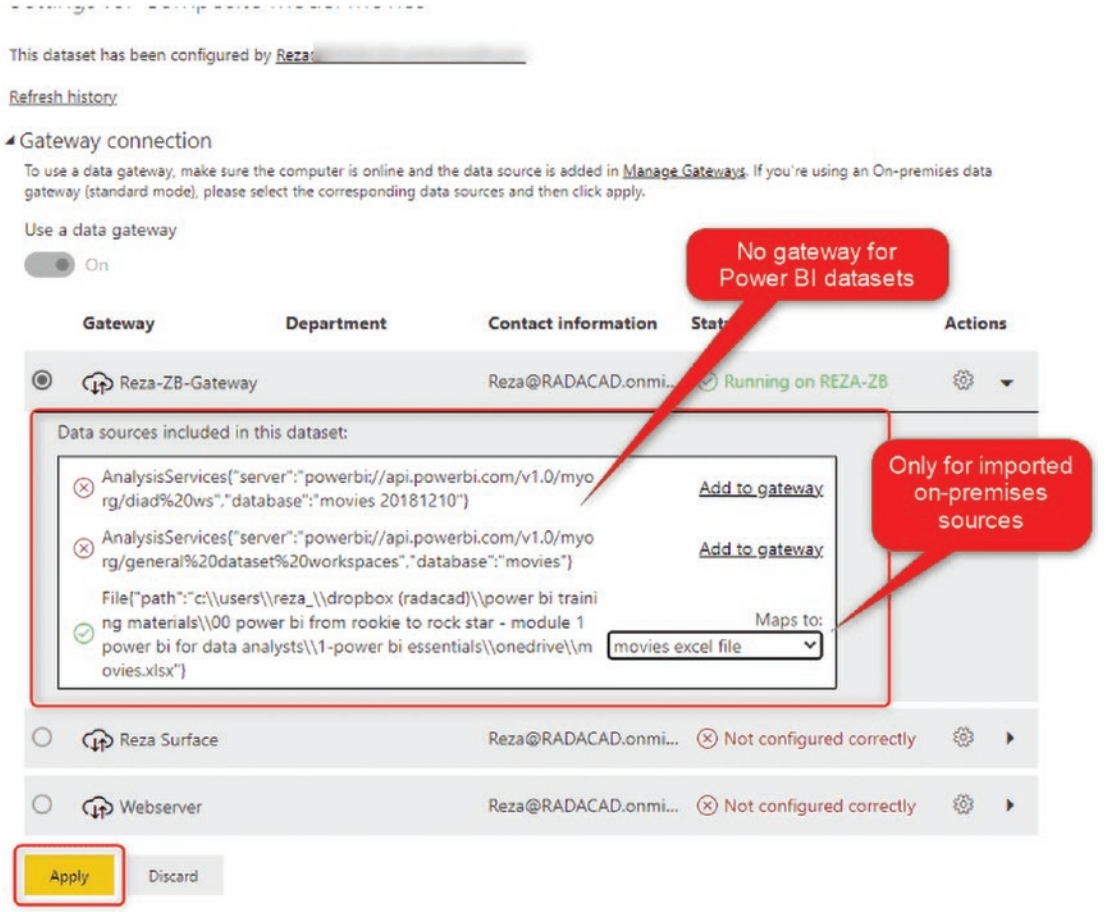


Figure 7-16. *Fixing the gateway error for DirectQuery to Power BI dataset*

If you are new to the gateway, there is a chapter about Power BI Gateway later in this book.

Data Source Credentials

After setting up the gateway, you need to set the credentials for the Power BI datasets, as shown in Figure 7-17.

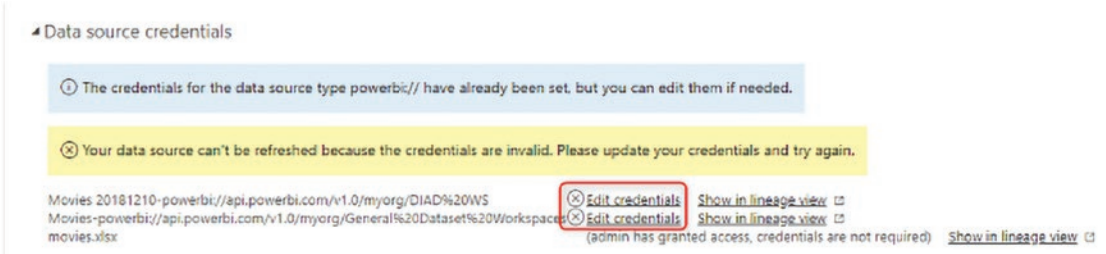


Figure 7-17. *Setting up data source credentials for Power BI datasets*

Select OAuth2 and enter the Power BI account credentials that have access to the datasets, as shown in Figure 7-18.

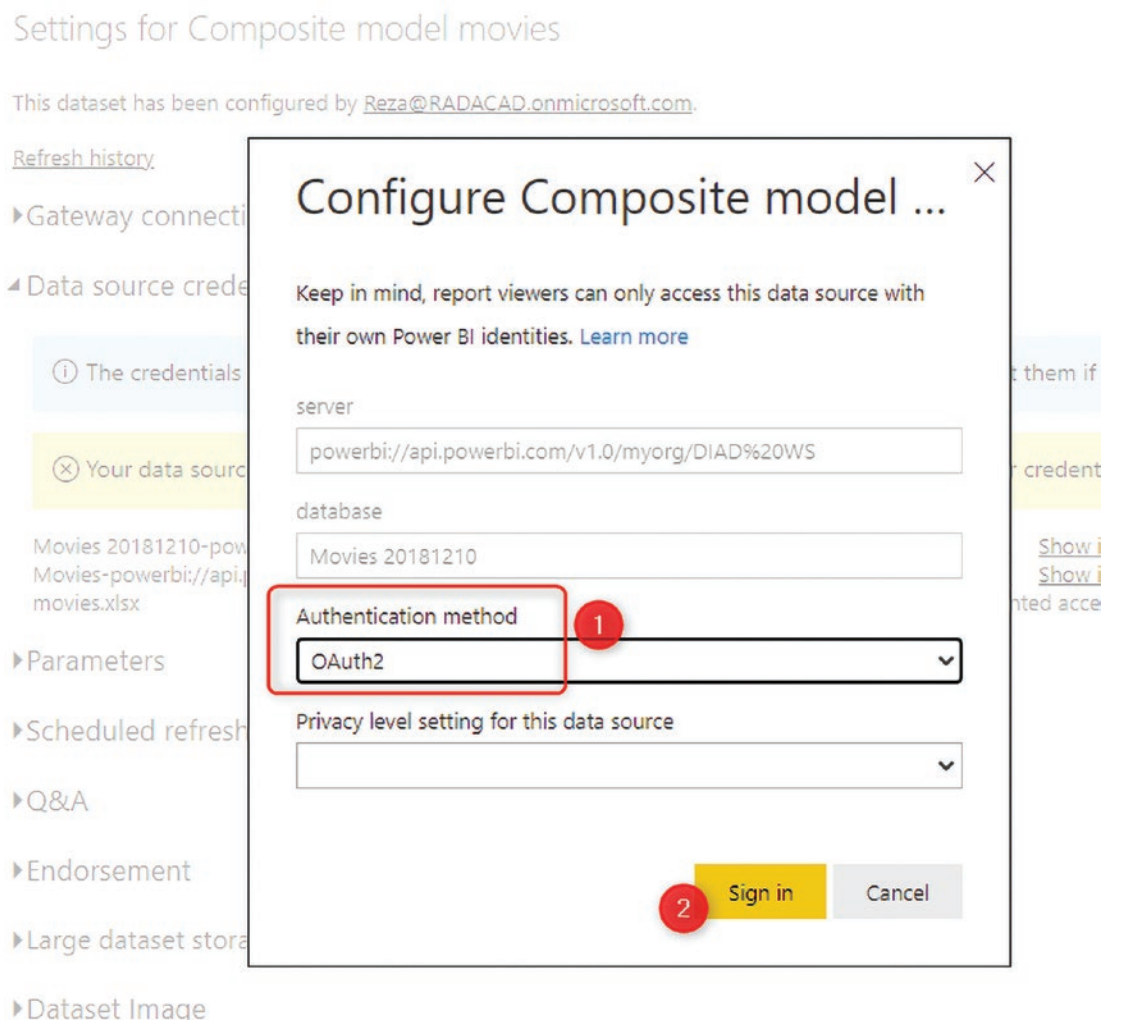


Figure 7-18. *Entering credentials for DirectQuery to Power BI dataset*

The credentials simply create the connection. The access to the data is based on the logged-in user's access to the data.

Lineage View

You can also see lineage view, which shows you all the data sources in your composite Power BI model. See Figure 7-19.

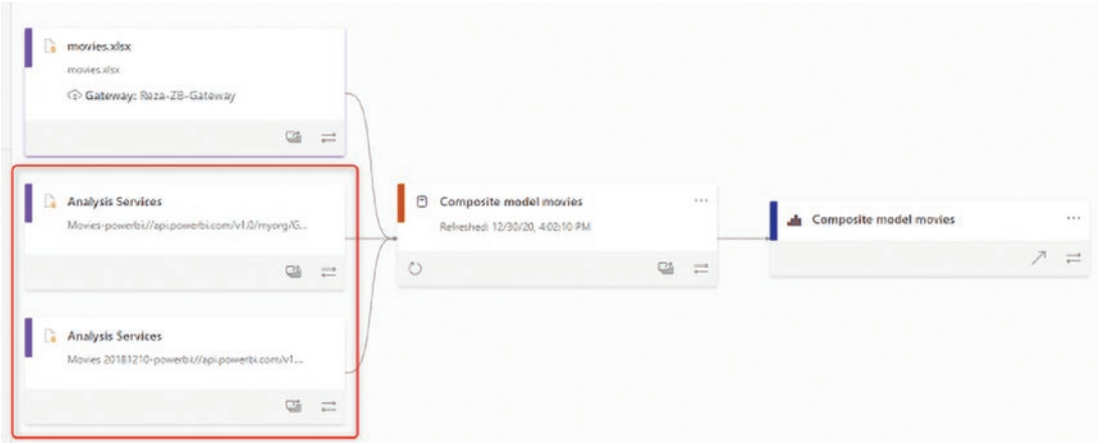


Figure 7-19. *Lineage view for a Power BI dataset*

After following these steps, the report will appear. See Figure 7-20 as an example.

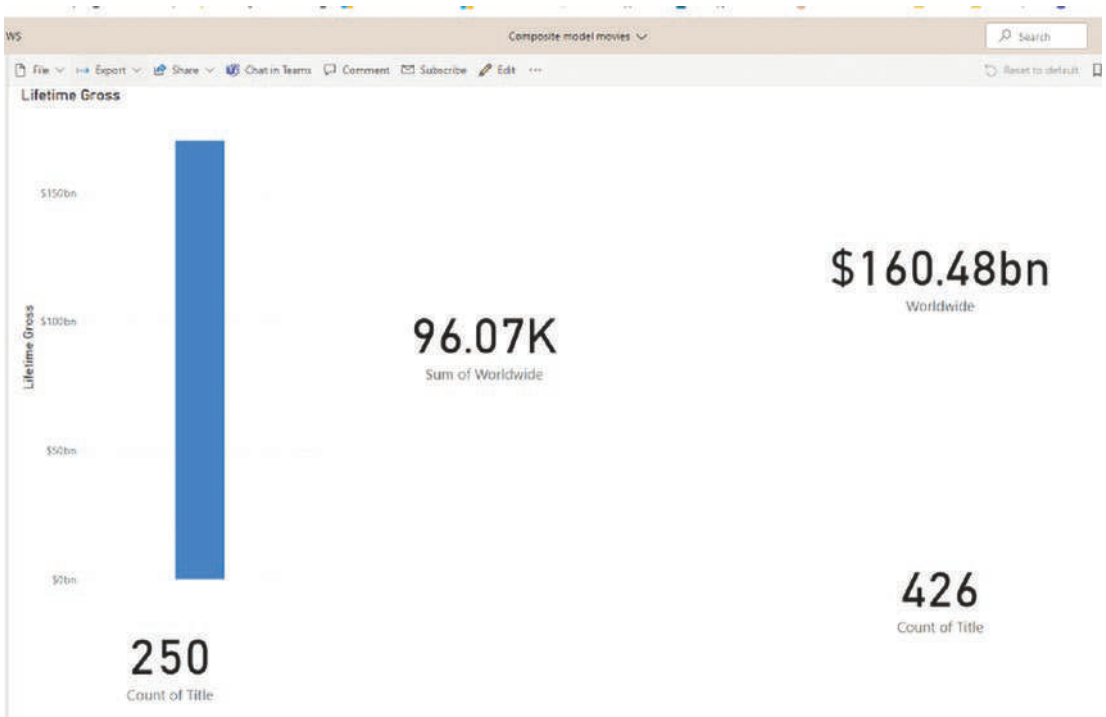


Figure 7-20. *Composite model report appears without errors*

Sharing the Report

This type of report can be shared like any other. However, sometimes, depends on the sharing configurations, the user gets the output shown in Figure 7-21.

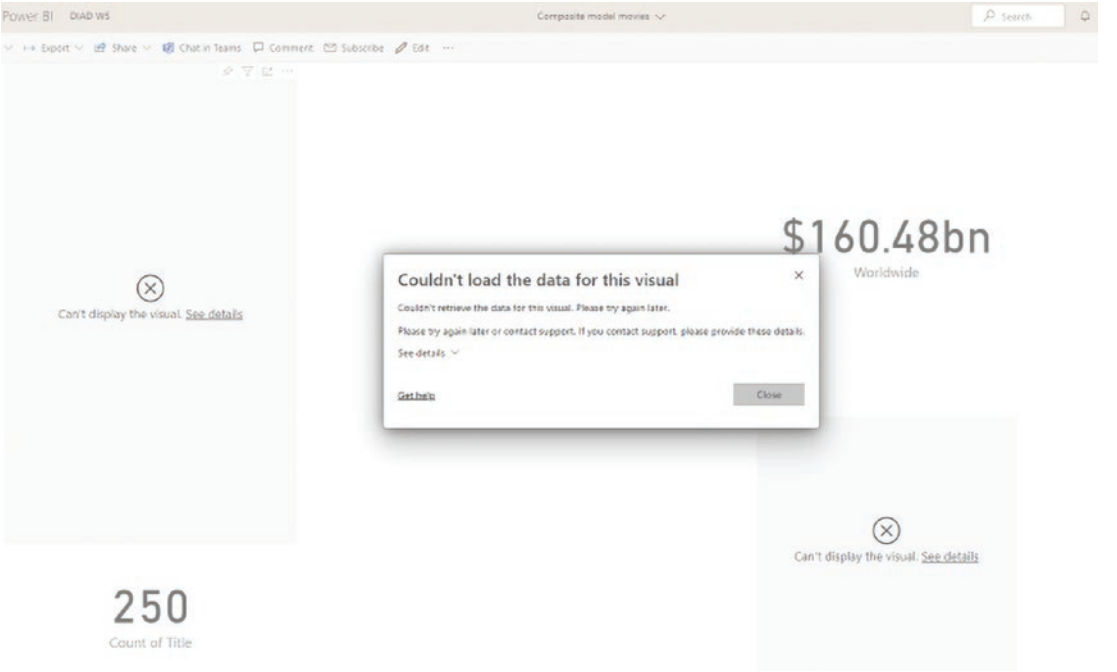


Figure 7-21. *Error when sharing a composite model with a DirectQuery to Power BI dataset connection*

Building Access to the Source Power BI Datasets

This happens often when the user who the report is shared with doesn't have access to one of the Power BI datasets. As shown in Figure 7-22, you can go to each dataset one by one and choose Manage Permission for that dataset.

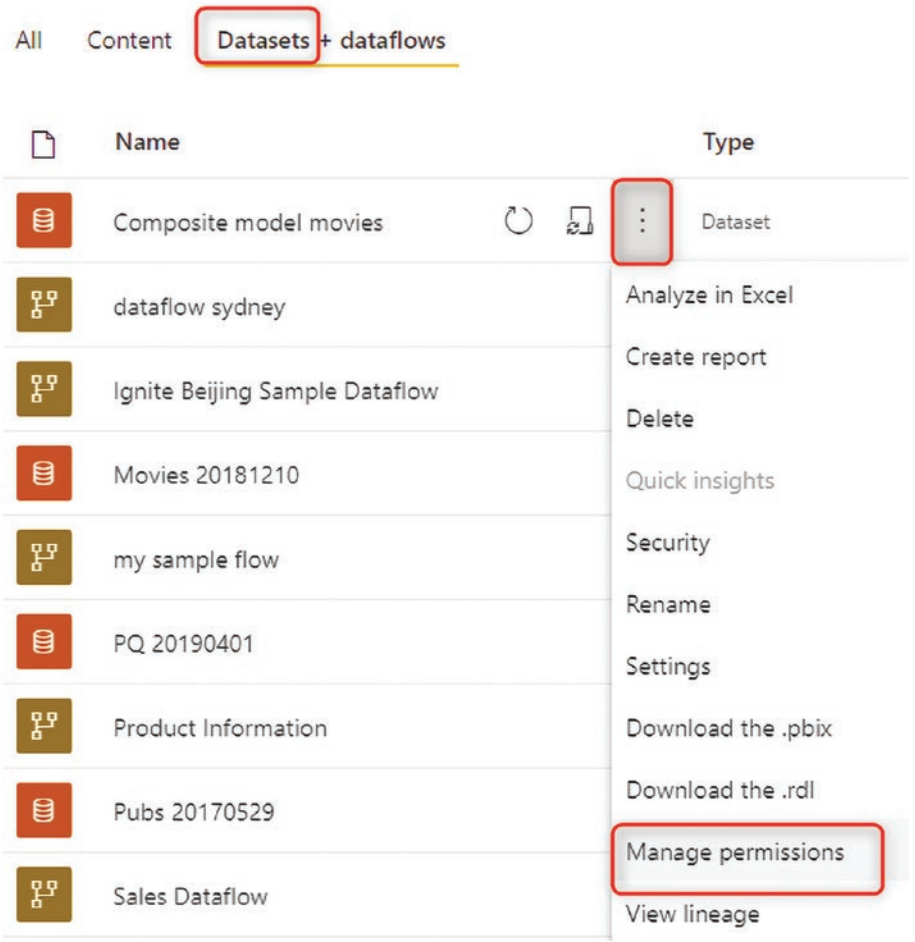


Figure 7-22. *Managing permissions of the sources of the Power BI datasets*

If the user doesn't have build access, enable it for them, as demonstrated in Figure 7-23. They should then be able to access the dataset just fine.

Figure 7-23. *Adding build access to the Power BI dataset*

With build access, the user can see the report properly, as shown in Figure 7-24.

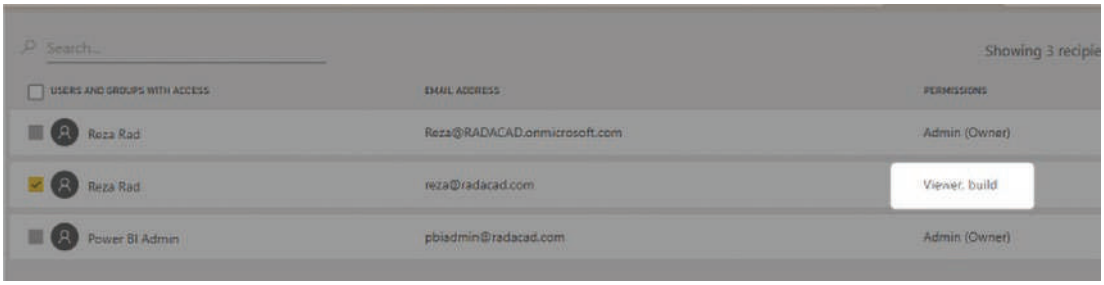


Figure 7-24. *User has build access to the Power BI dataset*

Summary

This feature is a major milestone in developing Power BI solutions in any organization. This feature can help you build less redundant code, have better consistency across your analysis, reuse whatever has been done so far, and customize on top of it. This feature significantly changes the way Power BI is used in a development environment.

In the next chapter, you learn about the differences between all the connection types in Power BI.

CHAPTER 8

Choosing the Right Connection Type: DirectQuery, Live, Import, or Composite Model

Power BI supports different methods for connecting data. That is why the decision to choose the right method is always a tough one. You have learned about of DirectQuery, Live Connection, Import Data, and composite model. Which situations would you choose one over the other? What are the pros and cons of each? What are the best practices for choosing each? Which one is faster? Which one is more flexible? I always get a lot of questions like these in my courses, conference talks, and blog posts. In this chapter, you get answers to all of these questions and more. This chapter will help you choose the right data connection methodology and architecture for your Power BI solution.

Why Is This a Tough Decision?

If Power BI had only one way to connect to data sources, you choice would be easy. You would never need to choose between methods and find the right one. However, Power BI supports multiple methods for connecting to data: DirectQuery, Live Connection, Import Data (or some call it Scheduled Refresh), and composite model. Many of you might still think that DirectQuery and Live Connection are the same, however, they are different. You learn about their differences in this chapter. Each method has benefits and drawbacks. Depending on the scenario that you are implementing, you might choose one over the others. Changing from one method to another can be a time-consuming task once you're in the implementation process. So the best approach is to choose the best method from the beginning.

Choosing the right method is an important step for your Power BI Solution Architecture. You need to make this decision in the early phases, before starting the implementation process. In this chapter, I explain in detail every method and answer the following questions:

- What is Import Data/Schedule Refresh?
- What is DirectQuery?
- What is Live Connection?
- What is the difference between Live Connection and DirectQuery?
- Pros and cons of each method
- Which method performs the fastest?

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_8

- Which method is the most flexible?
- Which method is the most scalable?
- What are the architecture scenarios to use for each method?
- What is the role of the gateway?

What Is Import Data (Scheduled Refresh)?

This method has two names—some call it Import Data and some call it Scheduled Refresh. With this method, data from the source is loaded into Power BI. Loading data in Power BI consumes memory and disk space. If you are developing Power BI on your machine with the Power BI Desktop, it is the memory and disk space of your machine. If you are publishing the report on a website, it is the memory and disk space of the Power BI cloud machines.

If you have 1 million rows in a source table, and you load that data into Power BI with no filtering, you end up having the same amount of data in Power BI. If you have a database with 1,000 tables, however, you only load ten of those tables in Power BI, you get memory consumption for only those ten tables. The bottom line is that you spend memory and disc space equal to how much data you load into Power BI.

Compression Engine of xVelocity

The first assumption that you might come to after reading this explanation about Import Data is that, if you have a database with 100GB and import it into Power BI, you will have a 100GB file in Power BI. This is not true. Power BI leverages the compression engine of xVelocity and works on a column-store in-memory technology. The column-store in-memory technology compresses data and stores it in a compressed format. You could have a 1GB Excel file and when you import it into Power BI, your Power BI file ends up being 10MB. This is mainly because of the compression engine of Power BI. However, the compression rate varies. It depends on many things—the number of unique values in the column, the data types, and many other situations. A later chapter explains the compression engine in detail.

The short read for this part is this: Power BI stores compressed data. The size of the data in Power BI will normally be much smaller than the size in the data source.

Important Pros and Cons of This Method

Power BI Is Fully Functional

With this method, you get full functionality of Power BI. You can use Power Query to combine data from multiple sources, or DAX to write advanced time intelligence expressions and visualizations. There is no limitation on the functionality of Power BI with this method. You can use all its components.

Size Limitation

With this method, you have a limitation on the size of the model. Your Power BI model (or let's say the file) cannot be more than 1GB (if you are not using Power BI Premium capacity or Premium Per User licenses). You usually have up to 10GB in your account; however, the files can be up to 1GB. Power BI Premium allows you to have up to 400GB loaded on the Power BI website (Premium Per User allows up to 100GB). If you are using Power BI Report Server, the size limitation is 2GB. However, remember that 1GB in the Power BI file is not equal to 1GB of data in the source. (As mentioned in the compression engine section.)

This Is the fastest Method

This connection method is the fastest option possible. Data is loaded into the memory of the server and report queries are evaluated from the data loaded into memory. There are no lags or slowness with this method (as long as you designed your Power BI model with no performance issues).

What Is DirectQuery?

DirectQuery is a direct connection to the data source. Data is not stored in the Power BI model. Power BI will be a visualization layer, which queries the data from the data source every time. Power BI only stores the metadata of tables (table names, column names, relationships…), not the data. The Power BI file size is much smaller, and you will likely never hit the size limitation because there is no data stored in the model.

DirectQuery is possible using only a few data sources. Some of the data sources that support DirectQuery from Power BI include:

- Amazon Redshift
- Azure HDInsight Spark
- Azure SQL Database
- Azure SQL Data Warehouse
- IBM Netezza
- Impala
- Oracle Database
- SAP Business Warehouse
- SAP HANA
- Snowflake
- Spark
- SQL Server
- Teradata Database

Important Pros and Cons of This Method

Scalability: The Main Advantage

This method does not have a size limitation. Mainly because no data is stored in the Power BI file, so you never get an issue with the size of data. You can have data sources with petabytes of data in SQL Server, Oracle, or any other supported data sources and connect to them from Power BI.

Limited Functionality: Few Power Query Operations, Mainly Visualization

This method does not have the full functionality of Power BI. With this method, you have only two tabs in Power BI Desktop—Report and Relationship. You can change the relationship in this mode.

In Power Query, you are limited to the number of operations. The majority of the operations that cannot be folded cannot be used. To learn more about query folding, visit `radacad.com/not-folding-the-black-hole-of-power-query-performance`. With this mode, however, you have full visualization support.

Slow Connection

A big disadvantage of this method is that the connection is slower than with other types of connections. Note that every visual sends a query to the data source and the data comes back. You usually have more than one visual in your report, and when slicing and dicing, you are sending queries to the data source. Performance tuning the data source is a must when using this model.

To show a small example of performance tuning, Figure 8-1 shows the performance I get when I have a normal index on my table with 48 million records.

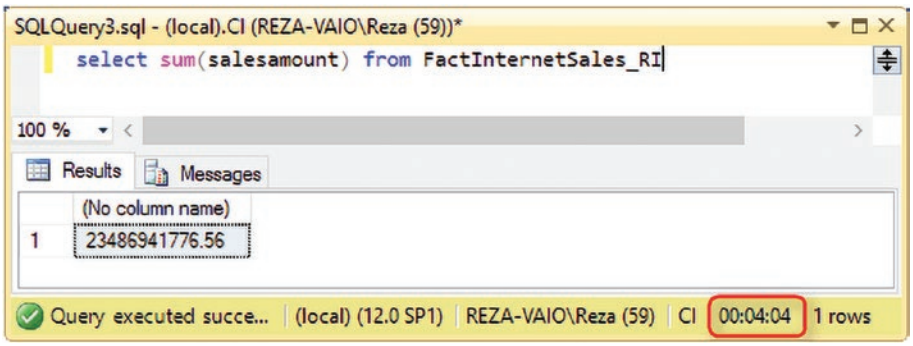


Figure 8-1. *A regular index*

A regular `select sum` from my table with 48 million records takes four minutes and four seconds to run. The same query responds in less than a second when I have a Clustered Column Store index, as you can see in Figure 8-2. You'll see significantly improved performance when you have a Clustered Column Store index on the same table with the same amount of data rows.

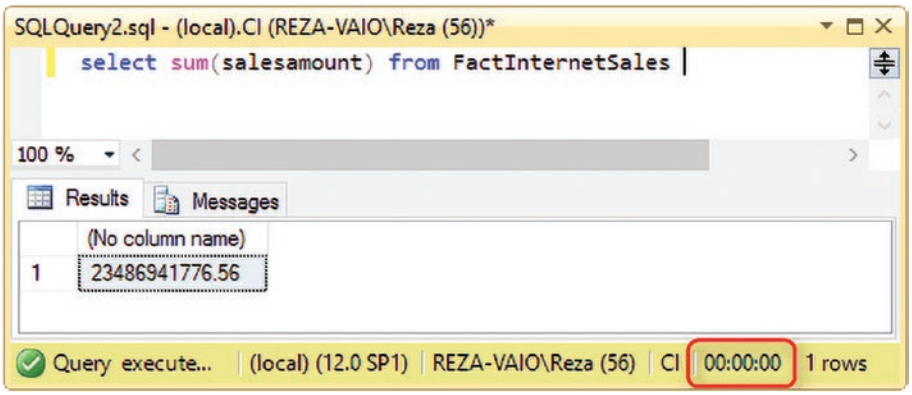


Figure 8-2. *Performance boost with a Clustered Column Store index*

This chapter doesn't teach you about performance tuning; you'll have to read books, blog posts, and watch videos to learn about it. That is a whole different topic on its own. The most important thing to remember is that performance tuning is different for each data source. Performance tuning for Oracle, SQL Server, and SSAS is totally different. Your friend for this part is Google, and the vast amount of free content available on the Internet for you to study.

What Is Live Connection?

Live Connection is similar to DirectQuery in the way that it works with the data source. It does not store data in Power BI, and it queries the data source every time. However, it is different from DirectQuery. Live Connection is only supported for these data sources:

- Azure Analysis Services
- SQL Server Analysis Services (SSAS) Tabular
- SQL Server Analysis Services (SSAS) Multi-Dimensional
- Power BI Dataset in the Service

Because these data sources are modeling engines themselves, Power BI connects to these and fetches all model metadata (measure names, attribute names, relationships…). With this method, you need to handle all your modeling requirements in the data source; Power BI just surfaces that data through visualization.

Important Pros and Cons of This Method

Big Model Size with OLAP or Tabular Engine

The main benefit of this model is that you can have a big-sized data model, and you can leverage the modeling layer of SSAS. SSAS Tabular will give you DAX, and Multi-Dimensional will give you MDX. With either of these two languages, you can cover all your calculations and modeling needs. This method has better modeling features than DirectQuery. In DirectQuery, there is no DAX or MDX. All the calculations need to be done on the database side. Doing calculations on the database side is often much more complex than doing them in the analytical expression language.

No Power Query, Just Visualization

The big disadvantage of this method is that you do not have access to Power Query.

Report-Level Measures

You get report-level measures with this type of connection, which give you the ability to write DAX measures. However, you might want to keep them in the data source to keep your model consistent.

Report-level measures are a great feature because users can create measures without having to call the BI developer. However, these measures are not added to the dataset. These are just for the report. So for consistency of your model, you might want to keep measure creation as part of your SSAS data source model.

Composite Model: The Best of Both Worlds

Power BI enables you to combine DirectQuery sources and Import Data sources into one dataset, as Figure 8-3 illustrates. Compositing gives you the performance and flexibility of Import Data, and the scalability and large data size of DirectQuery.

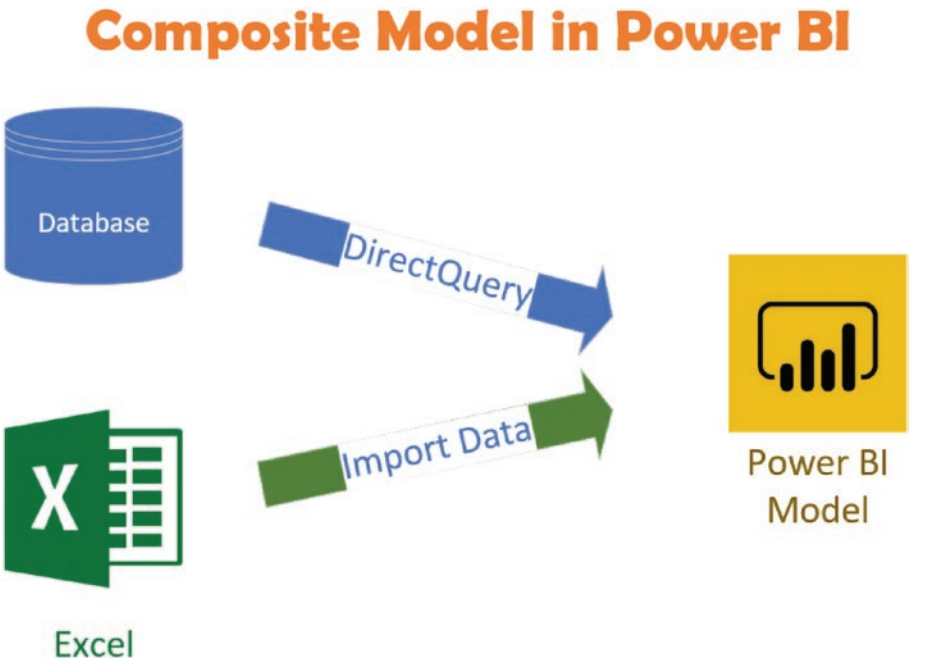


Figure 8-3. *Composite model*

Composite model not only supports DirectQuery to data sources such as SQL Server and other DirectQuery sources, it also enables you to use DirectQuery to the Power BI Dataset. This enables you to create a chained dataset from the main Power BI dataset. See Figure 8-4.

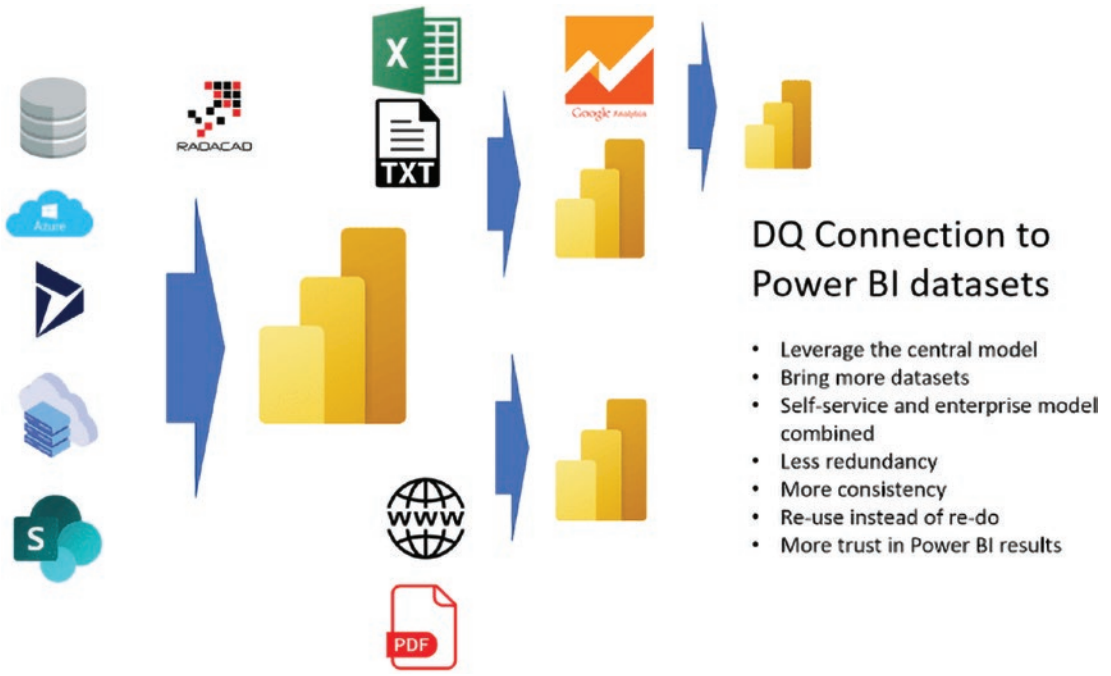


Figure 8-4. *DirectQuery to Power BI Dataset*

Composite model has many useful features, and it is highly recommended that you build a composite model for any of your DirectQuery data sources. This can be as simple as keeping the smaller tables in Import Data mode (such as dimension tables). This also enables you to create aggregated (and imported) versions of your DirectQuery tables for better performance.

Differences Between Live Connection and DirectQuery

Now that you know about all different types of connections, let's look at the differences between Live Connection and DirectQuery:

- DirectQuery is a direct connection to data sources such as SQL Server, Oracle, IBM, and so on.
- Live Connection is a direct connection to the Analysis Services model (Azure AS, SSAS Tabular, Multi-Dimensional, or a Power BI Report published service).

Relationship Configuration

With DirectQuery, you can still configure relationships in some cases. With Live Connection, you have no relationship ability. This should be handled in the data source. Because the Analysis Services is a modeling engine, you can build more than just a relationship there. Things such as hierarchies, measures, and columns can be created in the data source for the Live Connection and then be used in Power BI.

Report-Level Measures

With some types of SSAS live connections (to tabular model or Power BI Service), you get report-level measures.

No Power Query in Live Connection

In DirectQuery, you still can do simple Power Query transformations. However, in Live Connection, Power Query is not available. All you can do is change the source data model to another model or another server.

Pros and Cons of Each Method

I have already explained the main pros and cons in each section. This section reviews them all in handy lists.

Import Data or Scheduled Refresh

Advantages

- Fastest possible connection
- Power BI is fully functional
- Combines data from different sources
- Full DAX expressions
- Full Power Query transformations

Disadvantages

- Power BI file size limitation (this is different for Premium and Pro)

DirectQuery

Advantages

- Large-scale data sources are supported. No size limitation
- Pre-built models in some data sources can be used instantly

Disadvantages

- Very limited Power Query functionality
- Slower connection type: Performance tuning in the data source is must

Live Connection

Advantages

- Large-scale data sources supported. No size limitation as far as SSAS supports
- Many organizations already have SSAS models, so they can use them as a Live Connection without the need to replicate into Power BI

- Report-level measures
- MDX or DAX analytical engines in the data source of SSAS can be a great asset for modeling compared to DirectQuery

Disadvantages

- No Power Query
- Cannot combine data from multiple sources
- Slower connection type: Performance tuning in the data source is must

Which Method Is the Best and Fastest?

Import Data is the fastest possible option. Data is loaded into the memory of the server and all queries are resolved immediately. Live Connection is the next option in this list, especially if the SSAS Tabular or Power BI Service is used, because these two are in-memory technologies and perform faster than the Multi-Dimensional option.

DirectQuery is the slowest type of connection. You have to consider the performance tuning of your data source. However, DirectQuery can be combined with Import Data to boost the performance of your reports. I recommend you use aggregated import tables to even make things faster when querying big data tables.

The winner in this case is Import Data. The Import Data parts of the composite model go into the same category.

Which Method Is More Flexible?

With Import Data, you get the full functionality of Power BI, including full power query transformations, DAX measures, and visualizations.

Direct Query and Live Connection are the next on this list because they both give you something. DirectQuery provides a few Power Query options. Live Connection provides report-level measures.

The winner here is again Import Data. However, a composite model can have part of the data imported. So the flexibility on that part of the data is exactly the same as Import Data.

Which Method Is More Scalable?

The Import Data method has a size limitation. If you aren't using Power BI Premium, this method is not scalable. With DirectQuery and Live Connection, you get better scalability. Data sources support a large amount of data. The winners here are Live Connection and DirectQuery.

Which Architecture Scenarios Are Best for Each Method?

Import Data for Agility and Performance

Import Data has a fully functional Power BI and great performance. If your dataset is not huge, you can easily use this method and produce reports very quickly.

Live Connection for an Enterprise Solution

Many enterprises already have pre-built models in the SSAS Tabular or Multi-Dimensional options. These models can easily be used in a Power BI Live Connection.

Even if your company hasn't started the AAS (Azure Analysis Services) or SSAS solution, and you are dealing with a huge dataset, this option is better than Direct Query. In SSAS, you have the analytical expression languages of MDX or DAX to cope with lots of calculations and modeling challenges. Figure 8-5 shows a sample architecture that can be used with this method.

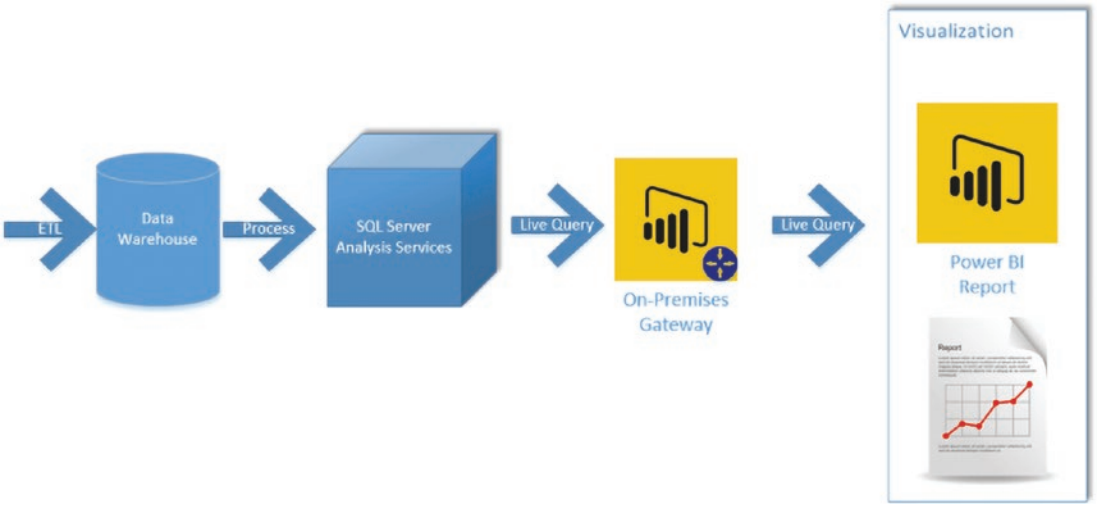


Figure 8-5. *Enterprise use of a Power BI live connection to Analysis Services*

Live Connection for Team Development

Even if you are not working in an enterprise environment with a dedicated BI team, you may find there are many benefits to using Live Connection. If you are working on an analytics project with multiple report authors and team members, you will enjoy a lot of benefits by sharing a dataset with a live connection. The dataset can then be consumed by report authors using the Get Data command from a Power BI Dataset and a live connection.

Direct Query for Non-Microsoft Sources

The DirectQuery connection is not used very much in Microsoft solution architecture settings. The main reason is that if you have a Microsoft-based solution architecture, you will probably use SSAS to leverage its analytical engine. DirectQuery mode is used mainly in non-Microsoft architectures, such as Oracle, IBM, or SAP HANA systems.

Even when you decide to use DirectQuery, I strongly recommend not using pure DirectQuery. Always combine DirectQuery tables with smaller import tables (dimensions or aggregated tables). This normally results in better performance when querying data in the reports.

What Is the Role of the Gateway?

Regardless of the type of connection you use (Import Data, DirectQuery, or Live Connection), if the data source is located on-premises, you need a gateway for it. Otherwise, you do not. To learn more about setting up a gateway, see the related chapter later in this book.

Summary

You've learned about the different types of connections—Live Connection, Import Data, DirectQuery, and Composite Model. You've learned their differences, pros and cons, and scenarios in which each should be used. There are still a lot of details for each method. In this chapter, I tried to explain everything in general to give you a holistic view. For more detailed information about each connection, read the previous chapters about each connection type. Now that you know about the connection types, the following chapters cover the important elements of the Power BI architecture—datasets, dataflows, and datamarts.

CHAPTER 9

Dataflows

Dataflows are an important component of the Power BI architecture. Using them can enhance the development and maintenance of your Power BI solution significantly. However, there are many Power BI implementations that don't use this functionality, even though more than three years have passed since it was released. In this chapter, you learn what a dataflow is and why you should use it.

What Is a Dataflow?

A dataflow is the data transformation service that runs on the cloud *independent* of the Power BI dataset or solution. This data transformation service leverages the Power Query engine and uses the Power Query online and UI to do the data transformation, illustrated in Figure 9-1.

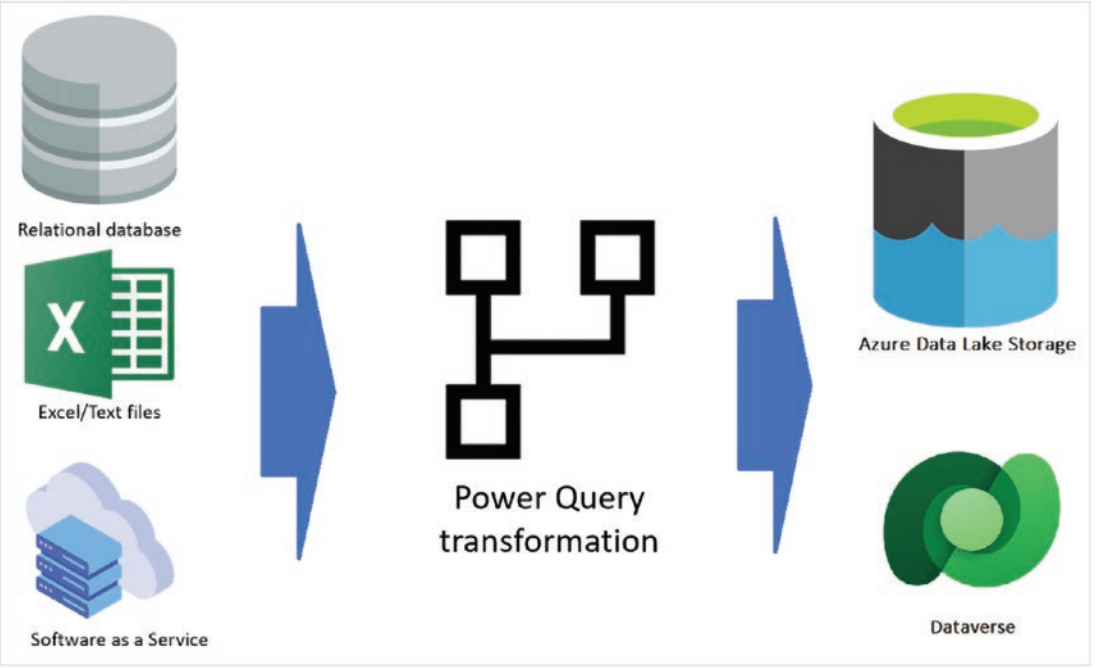


Figure 9-1. *A dataflow is a data transformation in the cloud that's independent of the Power BI dataset*

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_9

One might say that the Power BI dataset, when published to the service, also runs the data transformation online. Would it then be a dataflow? The answer is no. Because Power Query, which is part of the Power BI dataset, is loading data into the dataset directly. With a dataflow, the destination is not a dataset. It can be Azure Data Lake Storage or a dataverse (or some other storage types explained later). This makes the dataflow the independent data transformation component of Power BI.

A Dataflow Is Service-Only (Cloud-Only) Object

You cannot author or create a dataflow using the desktop tools such as the Power BI Desktop. The only way to create a dataflow is to do it in the cloud. In the Power BI Service, you can do it in a workspace. A dataflow created in the service (shown in Figure 9-2) can be used in the desktop tools (to connect and get data).

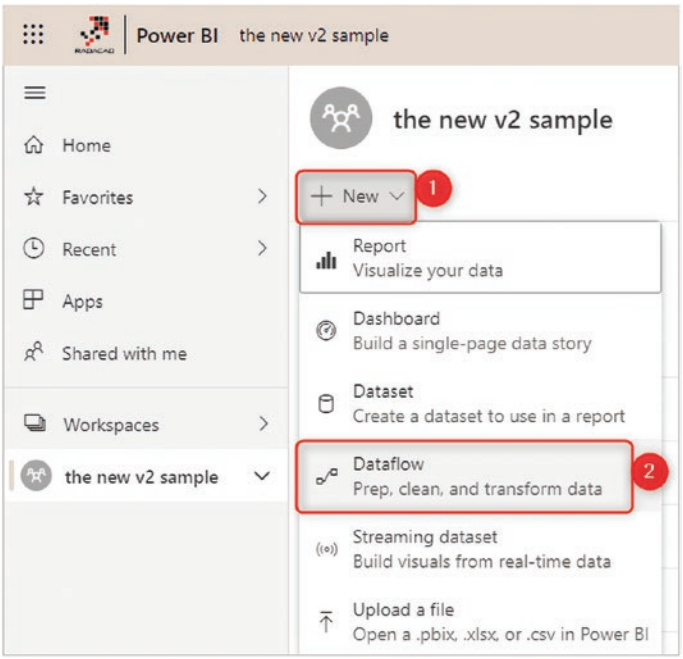


Figure 9-2. *Creating a dataflow*

A Dataflow Is Not Just for Power BI

In the Power BI world, we call them Power BI dataflows. However, dataflows are not just for Power BI. You can create dataflows in Power Apps (called Power Platform Dataflows). This process is illustrated in Figure 9-3.

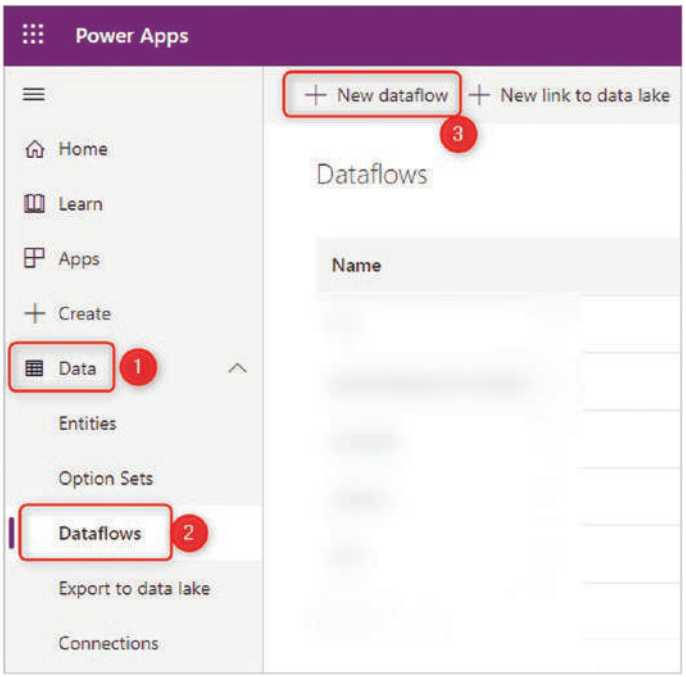


Figure 9-3. *Dataflows created in Power Apps*

Calling the dataflows by the platform in which you create them is not a common method. Instead, dataflows have two categories—Standard and Analytical.

Where Do Dataflows Store Data?

Dataflows do not store the data in the Power BI dataset. As you read earlier, the dataflow acts independently of the Power BI dataset. There must be another storage for the dataflow objects. The storage for a dataflow can be Azure Data Lake Storage or dataverses (There are other storage options available in Dataflow Gen2 as part of the release wave plan for Microsoft Fabric).[1]

You don't need an extra license to get the data storage option for a dataflow. If you use the Power BI Pro license, there will be an internal Azure Data Lake storage available for your dataflow tables. If you use Power Apps licenses, you will have storage available in the dataverse to use for your dataflows.

Other Microsoft services, such as Power Platform, can then connect to the dataflow, as shown in Figure 9-4.

[1] docs.microsoft.com/en-us/power-platform-release-plan/2022wave2/data-integration/load-azure-sql-database-dataflows

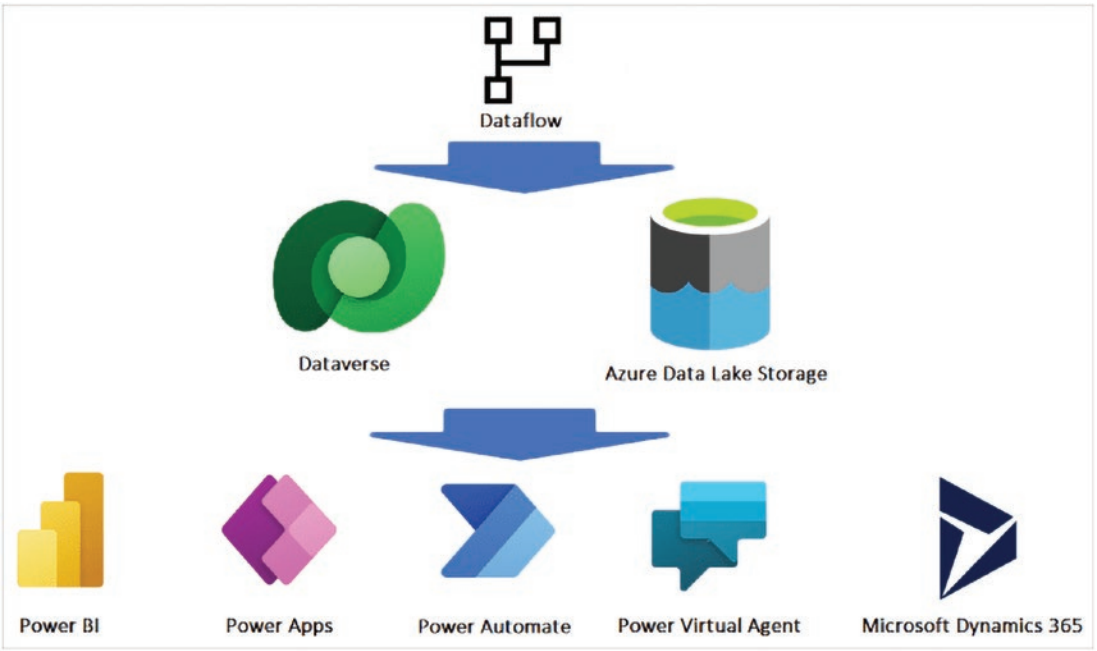


Figure 9-4. *A dataflow stores data in Azure Data Lake storage or dataverses*

Standard vs. Analytical Dataflows

Dataflows are characterized by Standard or Analytical. This categorization is not only based on the storage option of the dataflow but also on some of the functionalities available in each option. Analytical dataflows give you more analytical power, such as Computed entity[2] and AI functions in the dataflow[3]. The Standard dataflow store the data in dataverses only. There are a few other differences, too, but they are beyond the scope of this book.

Table 9-1 is a summary of these dataflows.

[2] To learn more about a computed entity, visit `radacad.com/linked-entities-and-computed-entities-dataflows-in-power-bi-part-4`

[3] To learn more about dataflow AI functions, visit `radacad.com/ai-in-dataflow-power-bi-webservice-auto-azure-ml-part2`

Table 9-1. *Standard vs. Analytical Dataflows*

Operation	Standard	Analytical
How to create	Power Platform dataflows	Power BI dataflowsPower Platform dataflows by selecting the Analytical Entity checkbox when creating the dataflow
Storage options	Dataverse	Power BI provided Azure Data Lake Storage for Power BI dataflows, Dataverse provided Azure Data Lake Storage for Power Platform dataflows, or customer provided Azure Data Lake storage
Power Query transformations	Yes	Yes
AI functions	No	Yes
Computed entity	No	Yes
Can be used in other applications	Yes, through the dataverse	Power BI dataflows: Only in Power BIPower Platform dataflows or Power BI external dataflows: Yes, through Azure Data Lake Storage
Mapping to standard Entity	Yes	Yes
Incremental load	Default incremental-loadPossible to change using the Delete rows that no longer exist in the query output checkbox at the load settings	Default full-loadPossible to set up incremental refresh by setting up the incremental refresh in the dataflow settings
Scheduled Refresh	Yes	Yes, the possibility of notifying the dataflow owners upon failure

Dataflows Are Powered by Power Query Online

Dataflows are powered by Power Query Online. Every transformation is completed by the dataflow engine, and the UI provided for doing the transformation is the Power Query Editor Online. There are more than 80 data sources available using dataflows, some of which are shown in Figure 9-5.

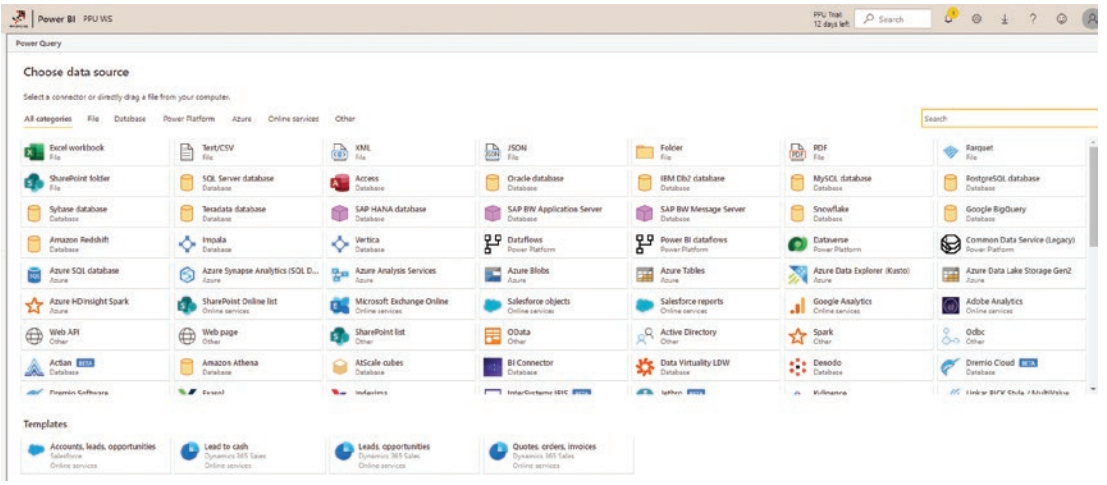


Figure 9-5. *Data sources for dataflows*

The online Power Query Editor enables you to do most of the transformations through the GUI. However, you can also use the Advanced Editor and work with the M script directly. See Figure 9-6.

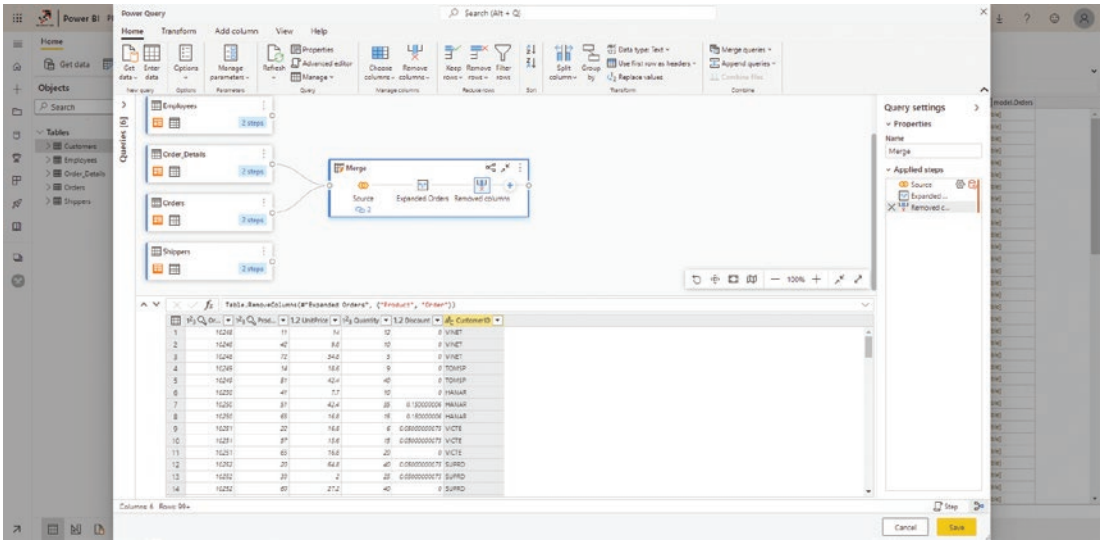


Figure 9-6. *Power Query Editor online*

Dataflows Can Be Used in Power BI, Excel, and Other Services

Depending on the type of dataflow, you can get data from it in a Power BI Desktop (or a Power BI Dataset), as in Figure 9-7, in Excel, and in some other services. This makes the dataflow a fully-independent component on its own.

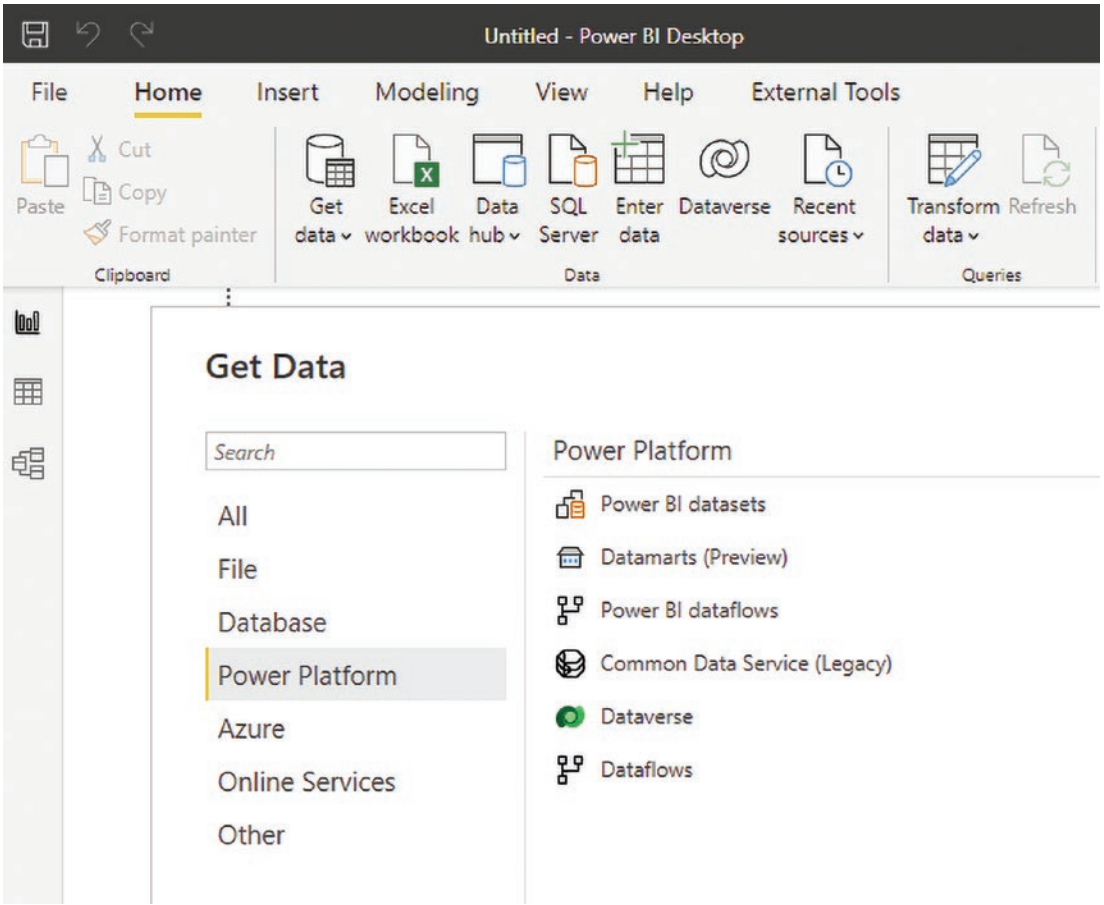


Figure 9-7. *Getting data from a dataflow in the Power BI Desktop*

In Excel, the same thing is available, as shown in Figure 9-8.

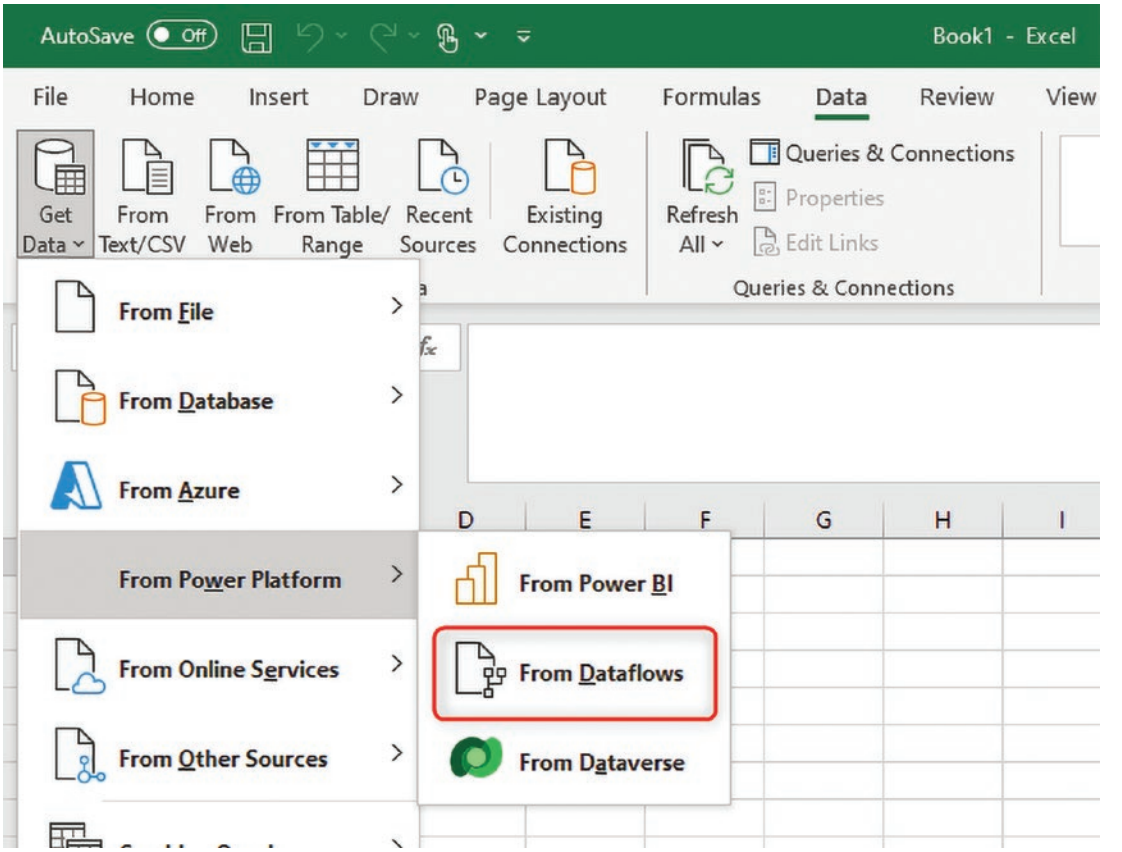


Figure 9-8. *Getting data from a dataflow in Excel*

The next sections discuss the use cases of dataflows.

Example Dataflow Scenarios

Now comes the big question of why you should use a dataflow. What is the point of it? I find it best to explain using examples.

Using One Power Query Table in Multiple Power BI Reports

Have you ever had the need to use one Power Query table in multiple Power BI reports? Of course you did. If you worked with Power BI for some time, you know that tables generated through Power Query are only part of one Power BI file. If you want to use the same table in another file, with a combination of some other tables that is not in the first file, you need to replicate the Power Query transformations (or copy and paste the M script) into the new `*.pbix` file. You may think, no I don't, but Figure 9-9 shows an example—Date Dimension!

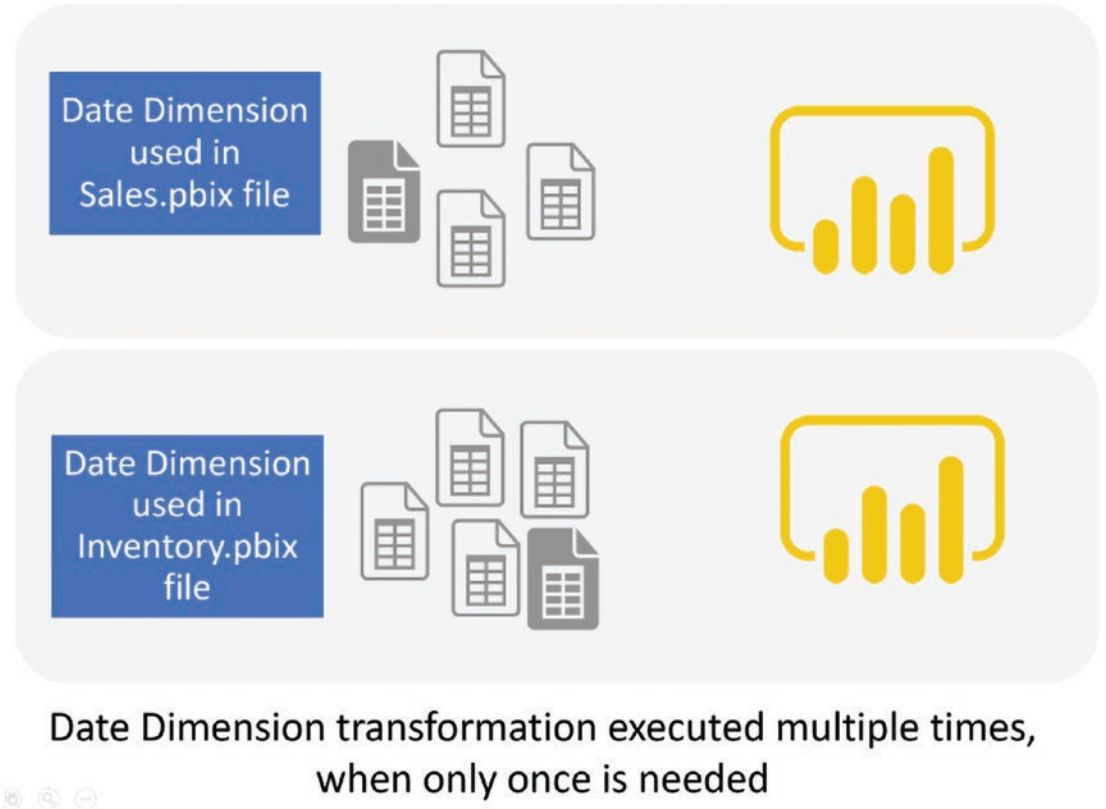


Figure 9-9. *A table that is needed in multiple Power BI files*

Date dimension is a table that you use in a `*.pbix`, let's say for Sales Analysis, and in another `*.pbix` for inventory reporting, and another `*.pbix` for HR data analysis. What do you do in these situations? Copy the script of Date Dimension in all of these files? What if after a year, you decide to add a transformation or a column to the date dimension? Then you need to replicate this change in all `*.pbix` files that you have, otherwise, your code will become inconsistent. It would have been much better if you did the transformation once, stored the output somewhere, and then reused it. As Figure 9-10 demonstrates, this is exactly what dataflows can do for you!

Figure 9-10. *Processing the common table in the dataflow and reusing it in multiple PBIX files*

Reusable tables or queries across multiple Power BI files are one of the best candidates for dataflows.

Different Data Sources with Different Refresh Schedules

What do you do if you have a dataset that includes two tables with different schedule options? For example, the Sales transactions table comes from the SQL Server database changes every day, and you need to refresh this data every day. However, the mapping table used for some of the products and maintained by the product team changes every quarter. If you have both of these queries in one `*.pbix` file, you have no other choice but to refresh at the maximum frequency needed, which is once a day.

However, what if there were a mechanism that could refresh the mapping table every quarter, apply all needed transformations, and store it in a table. Then every day you just need to read it. Dataflows can do that for you; with one query running the data transformation script and loading it into a destination table. This can be scheduled for whatever plan you need. Figure 9-11 illustrates this scheduling feature.

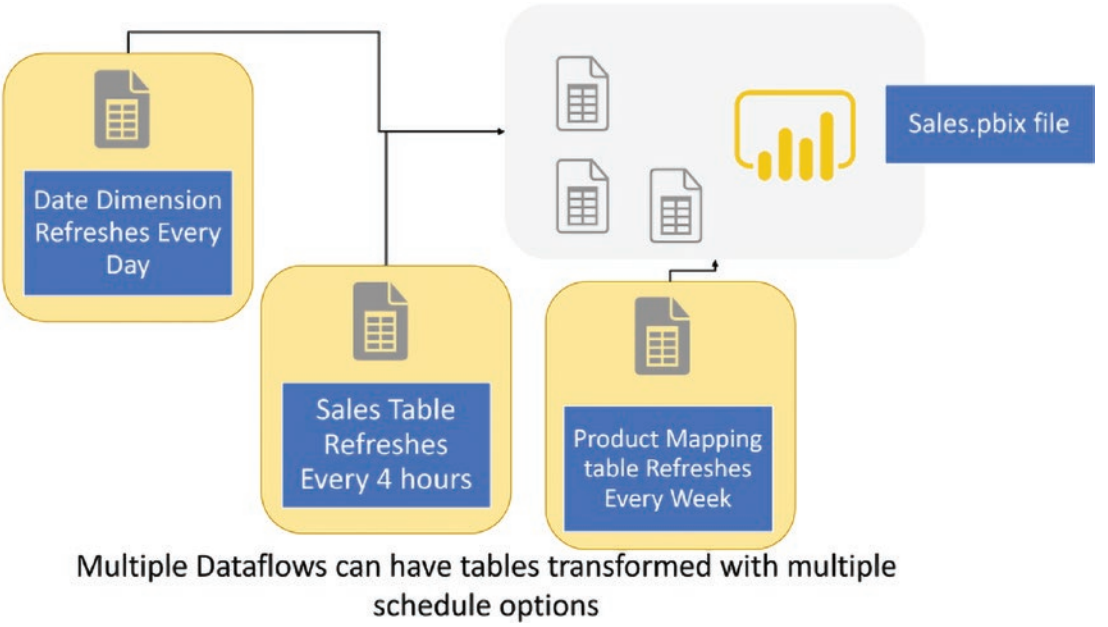


Figure 9-11. *You can have multiple dataflows for different scheduled refresh settings*

Dataflows can run the extract, transformation, and load (ETL) process on a different schedule for every query (or table).

Centralized Data Warehouse

With the evolution of Power BI and other self-service technologies, many companies started to implement a BI system without having a data warehouse. However, if the number of BI systems increases, the need for a centralized data warehouse appears quickly. A data warehouse is a specifically designed database that stores data in the format needed for reporting. In traditional BI systems, one of the phases of building a BI system, and one of the most important phases, is to create a centralized data warehouse. The ETL process will extract data from the data sources and load it into the centralized data warehouse. All reporting solutions then use the data warehouse as the single source of truth.

Dataflows can be an important part of building a centralized data warehouse for your Power BI solution. You can build the structure you want through Power Query scripts in a dataflow. Dataflows then run those scripts and store the data in output tables. Output tables of the dataflow can act as a centralized data warehouse for your `*.pbix` files. Alternatively, you can have your own Azure Data Lake storage and configure it the way that you want, with the structure of tables that you want, and get a dataflow to load data into those tables. See Figure 9-12.

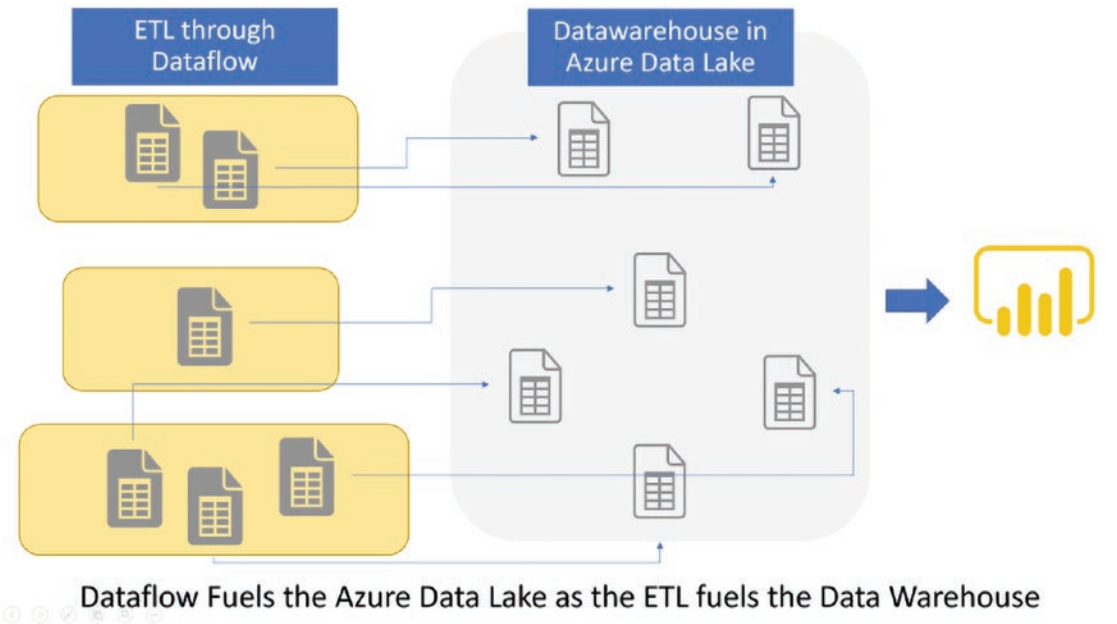


Figure 9-12. *Dataflows can feed data into your data warehouse*

Dataflows can be the ETL engine that fuels the centralized data warehouse in Azure Data Lake storage.

Power BI datamarts are a new component of Power BI, and they can be a much better replacement for dataflows when used for Data Warehouse purposes. I suggest strongly reading about datamarts in future chapters of this book.

Getting Started with Dataflows in Power BI

This section provides hands-on experience with dataflows and explains how dataflows work. First, you'll see how to create a dataflow. The first thing you need to know is that the dataflow creation and maintenance process is happening in the Power BI Service, not in the desktop, because dataflows are not part of any report or `*.pbix` file.

Developing and editing dataflows is possible through the Power BI service (not the desktop).

The second important thing you need to know is that dataflows can be created only in an app workspace. You cannot create a dataflow in My Workspace. So either create one or navigate to an app workspace. As you can see in Figure 9-13, I am in an app workspace called `dataflow`. The name of the app workspace can be anything you want.

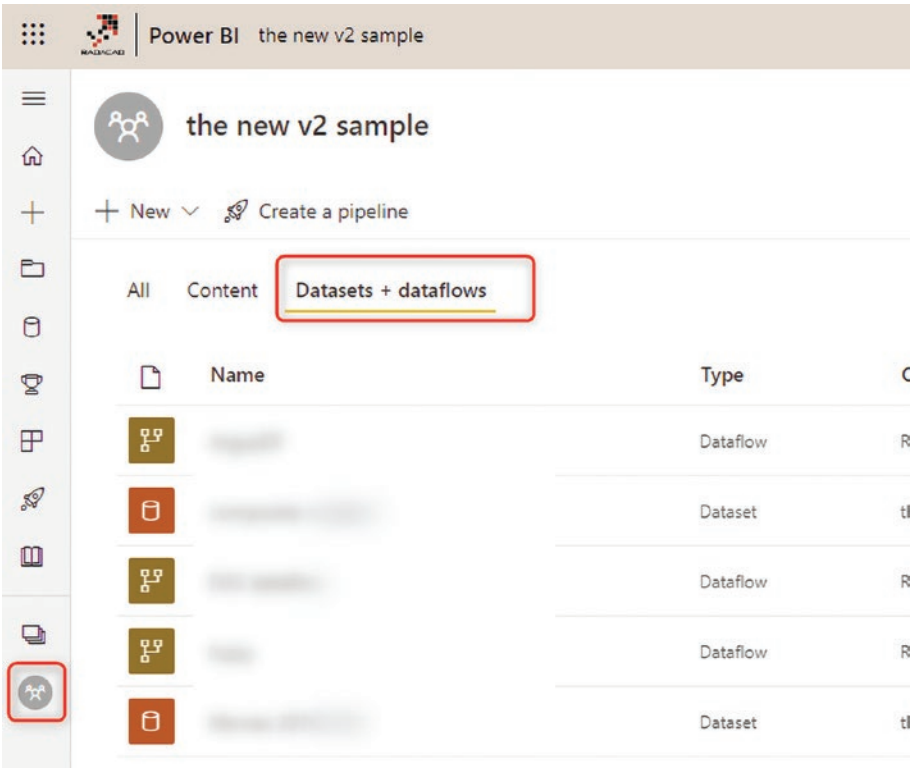


Figure 9-13. *Dataflows should be created in an organizational workspace*

Dataflows are only available in an app workspace (not in My Workspace).

In the Datasets + Dataflows tab, you can see any dataflows you created. If you don't see the Dataflow tab, even in an app workspace, then there is something else you need to consider—the administrator's control of the dataflow.

Administrator's Control

Power BI administrators can turn off and on the creating and use of the dataflow for users in the tenant. So if you don't see the Dataflows option, it is probably because the Power BI administrator has turned that feature off. Contact your administrator to get that enabled. At the time of writing this book, this option can be only turned on or off for the entire organization (like many other options in the tenant setting at the very first few months of their appearance), but I believe this feature is available for a select group of people (like many other options in the tenant settings in the future).

In the Admin Portal of Power BI service, under Tenant Settings, there is a configuration option for dataflows, as shown in Figure 9-14.

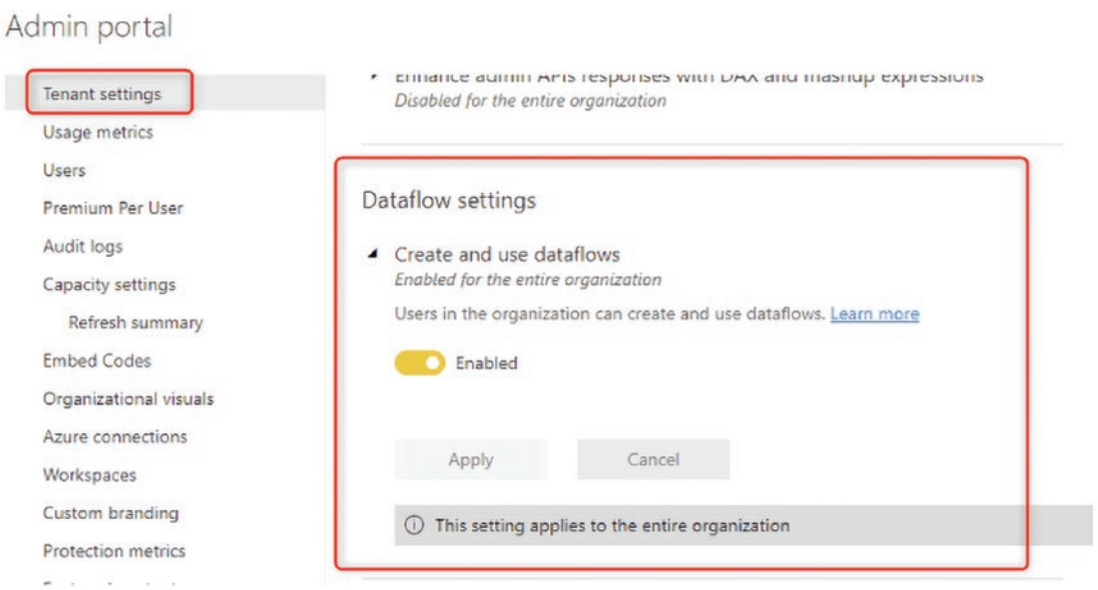


Figure 9-14. *Tenant settings for dataflows*

Creating Your First Dataflow

Now that you are ready, you can start building your first dataflow. Start by clicking the Create option and choosing Dataflow, as shown in Figure 9-15.

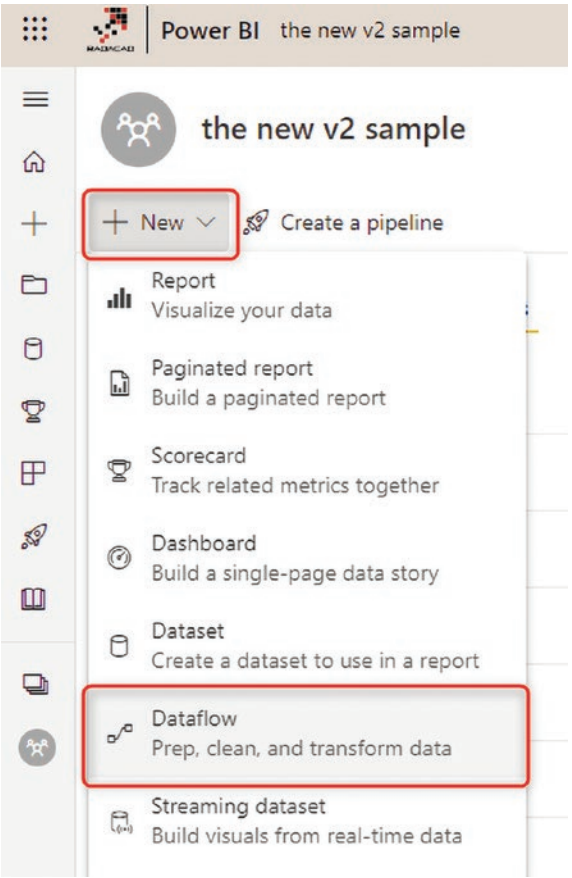


Figure 9-15. *Creating a dataflow in the workspace*

Each dataflow acts like a job schedule process. It has one or more transformations in it and can be scheduled. These transformations can write data into entities and tables. In Figure 9-16, you can see that there is an option to Define New Tables. Let's start with that.

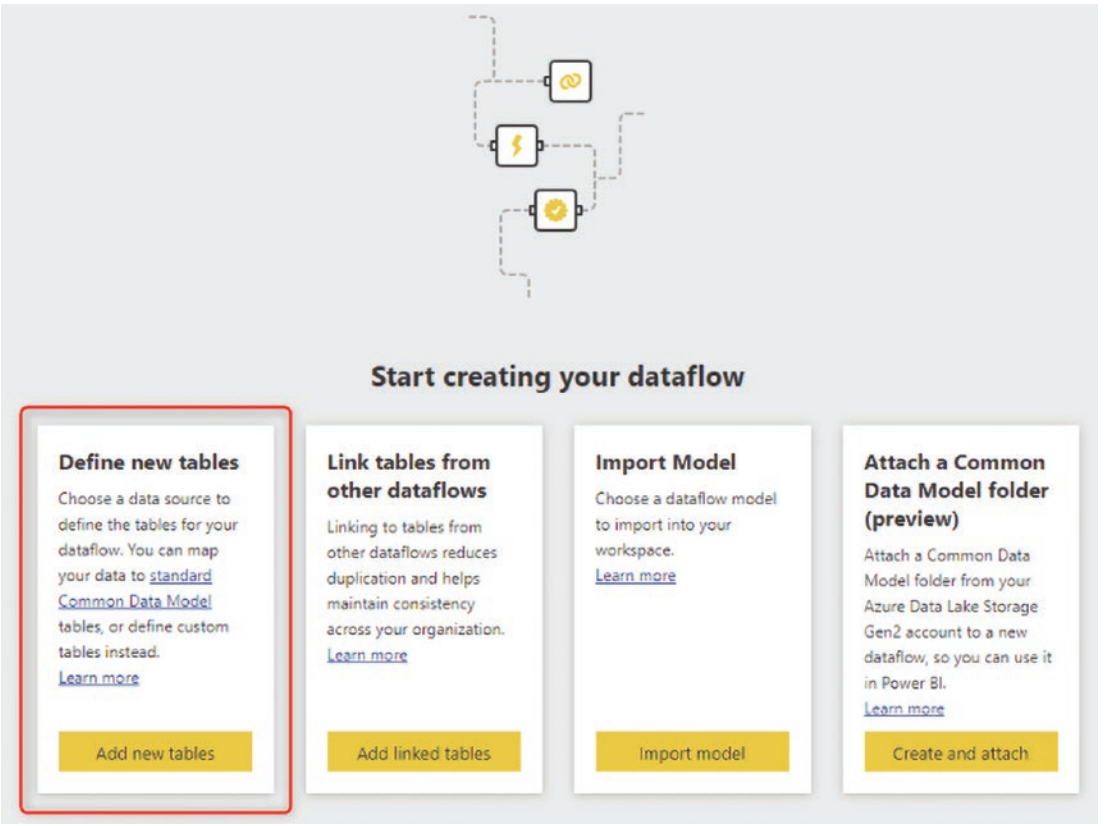


Figure 9-16. *Defining new tables in a dataflow*

Click Add New Tables. You can see a list of all supported data sources. (Recently in an update, many more data sources were added to the list). As you can see in Figure 9-17, the interface is very similar to the Get Data interface of the Power BI Desktop.

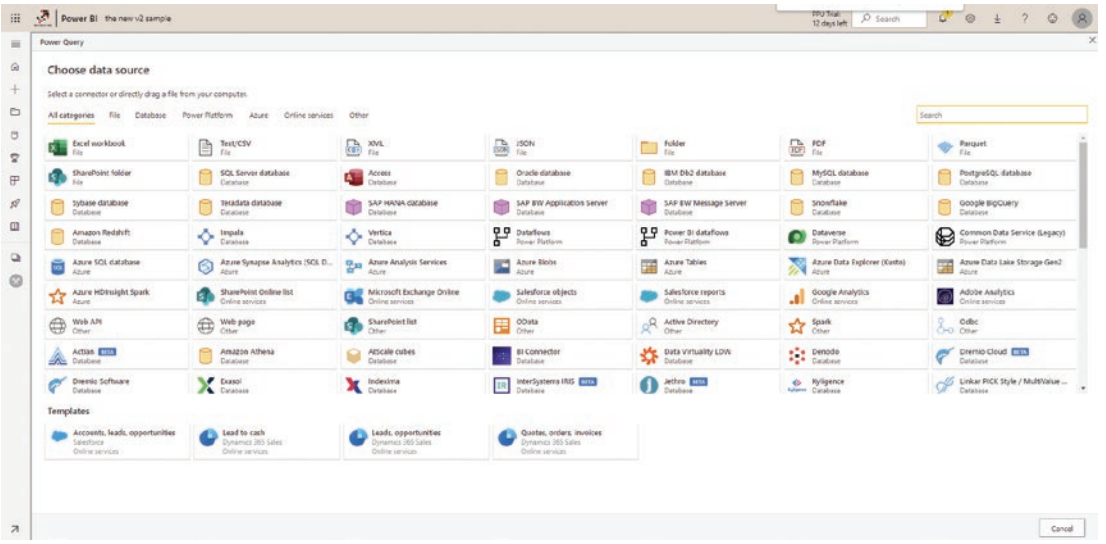


Figure 9-17. *Data sources available for dataflows*

Sample Datasets

For this example, you will be using an OData source, because it doesn't need a gateway set up or need additional setup requirements. Select OData[4] as the data source type, and in the URL part, enter the following address—`services.odata.org/V3/Northwind/Northwind.svc/`—as shown in Figure 9-18.

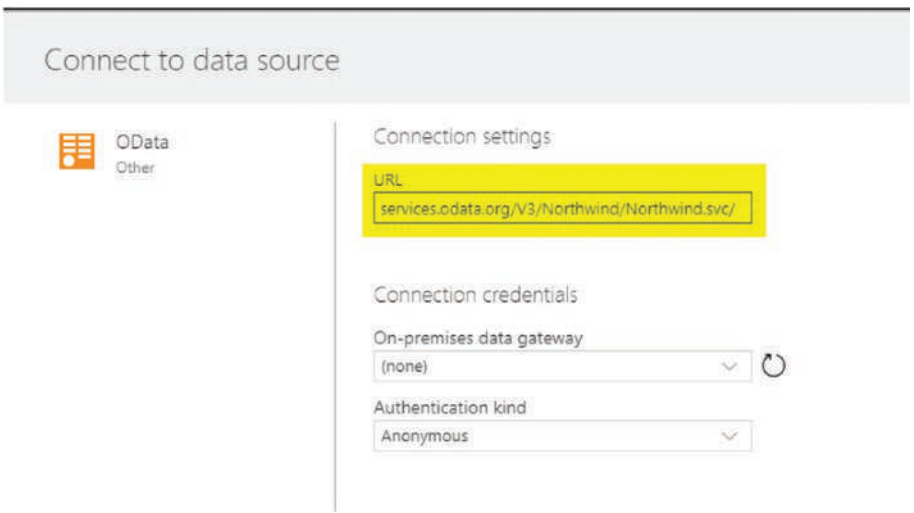


Figure 9-18. *Getting data from OData*

[4] OData is a dataset that is available through API, and its output can include one or more tables. Many data transformation tools can read data from OData. Power Query (or in this case, a dataflow) also has an OData data source connection.

After this step, you should see a screen that is very similar to the Navigator window in the Power Query Editor. On this screen, you will see a list of all tables and can start exploring them—see Figure 9-19.

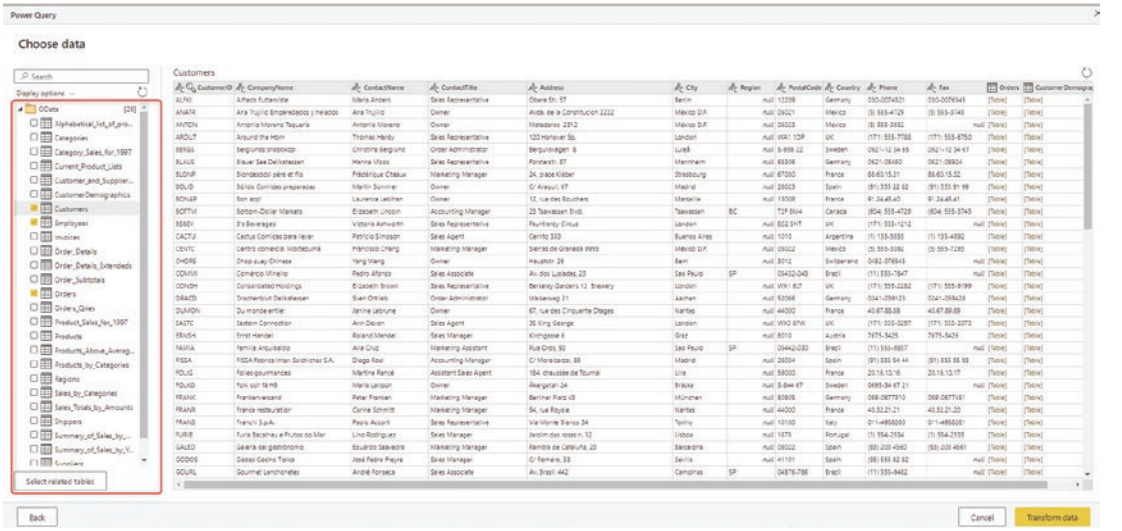


Figure 9-19. *Selecting tables for transformation*

In this example, select the Customers, Employees, and Orders tables and then click Transform Data.

Power Query Editor Online

As you can see in Figure 9-20, after selecting tables, you will see a screen that is very similar to the Power Query Editor, but an online version of it. You can see the Queries pane, the Steps (Query Settings) pane, the Data Preview pane, and the Data Transformation pane. There have been recent updates to the graphical interface of the Power Query Editor online.

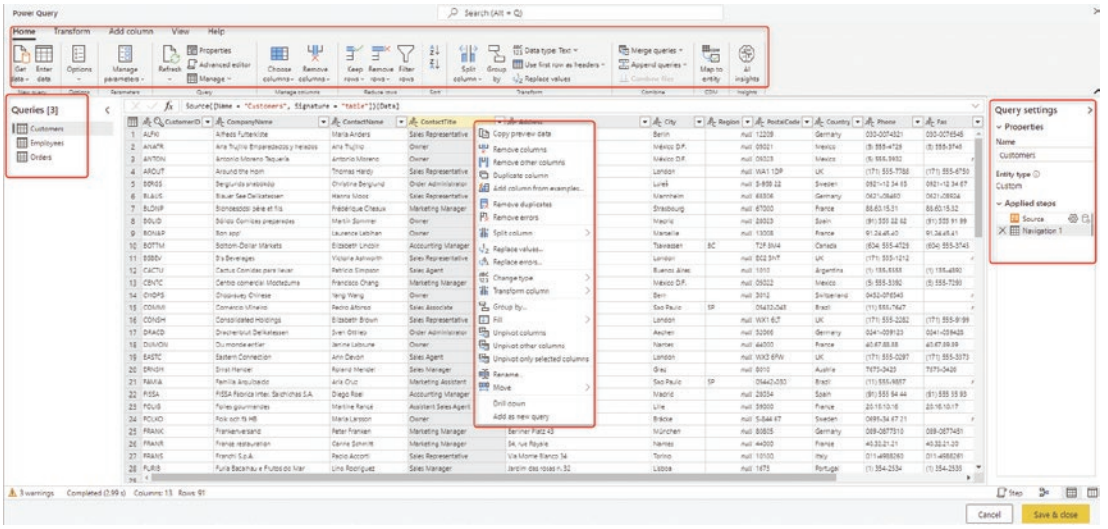


Figure 9-20. *Power Query Editor online*

In this example, I selected the three tables with no transformation and clicked Done. You can see all the tables in this dataflow. In dataflows, these are called entities. For every entity, you can specify Incremental Refresh if you want. You can then save your dataflow, as shown in Figure 9-21.

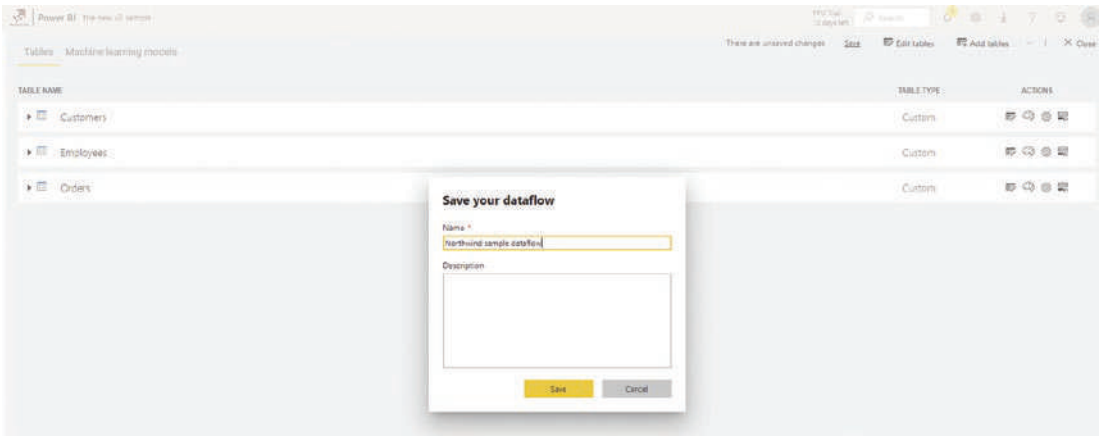


Figure 9-21. *Saving your dataflow*

Setting Up a Gateway

If your data is sourced from an on-premises (local domain) source, you need to have a gateway set up, as shown in Figure 9-22. You can read information about the gateway in a later chapter in this book. The difference here with Power Query in the Power BI Desktop is that, because you are developing the file locally, you can connect to an on-premises data source in the Power BI Desktop and start developing your solution, and then after publishing it to the service, you can set up the gateway. However, in a dataflow,

because everything is happening in the service, you need to have a gateway set up if you are connecting to an on-premises data source. Otherwise, you cannot pass the first step. If you are connecting to an online data source (such as Azure SQL Database), you don't need a gateway.

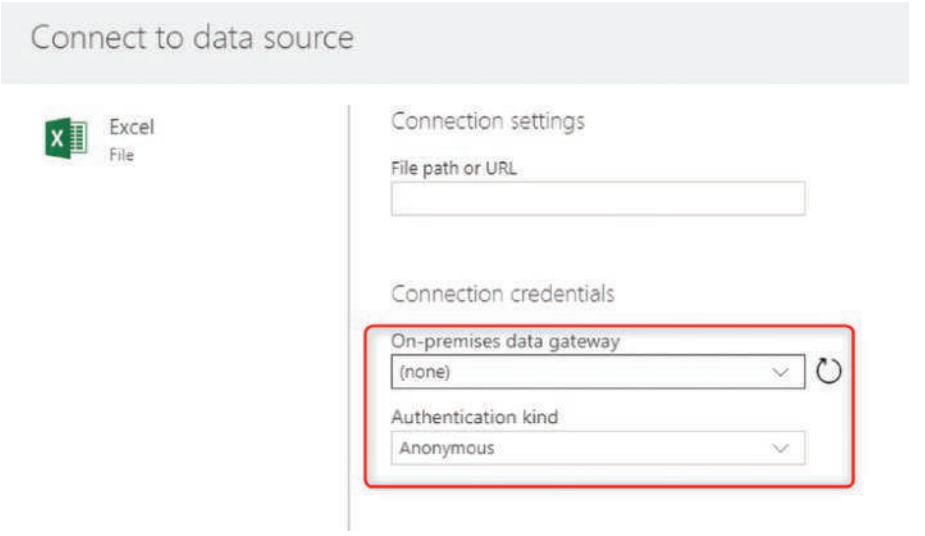


Figure 9-22. *Gateway setup for the dataflow tables*

Blank Query as the Data Source

One way to transfer your Power Query scripts to dataflow is to select () Blank Query as the data source, as shown in Figure 9-23, and copy and paste the M script from your Power Query tables to the dataflow. However, be mindful that not all Power Query transformations are supported yet. You may need to make some changes.

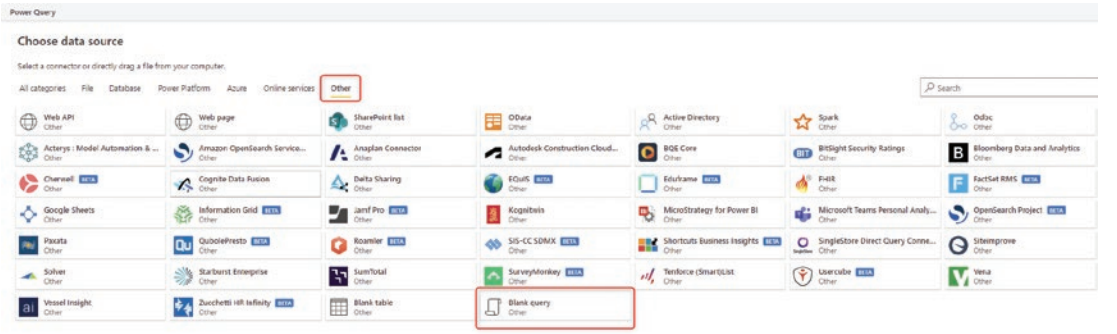


Figure 9-23. *Startting the dataflow table with a Blank Query*

This script (see `radacad.com/download/13519/`) is creating a basic date dimension. (You can learn more about it at `radacad.com/all-in-one-script-to-create-date-dimension-in-power-bi-using-power-query`.) You can copy and paste it into the blank query in the dataflow. See Figure 9-24.

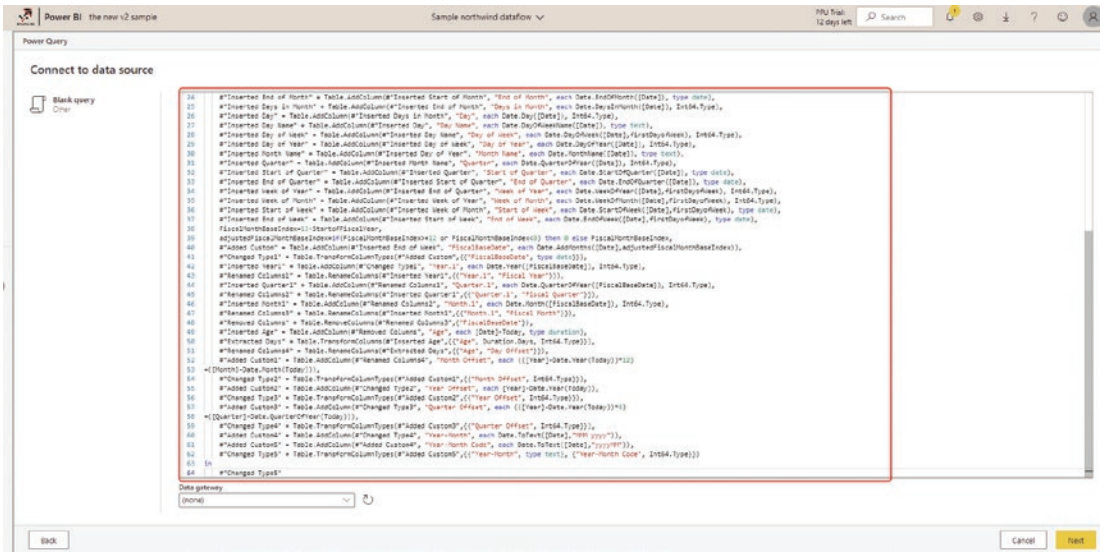


Figure 9-24. *Copying and pasting the M script into the blank query*

Dataflow should be able to show you a list of steps based on the script, although this depends on the functions used in the script and whether they are already supported in dataflow. See Figure 9-25.

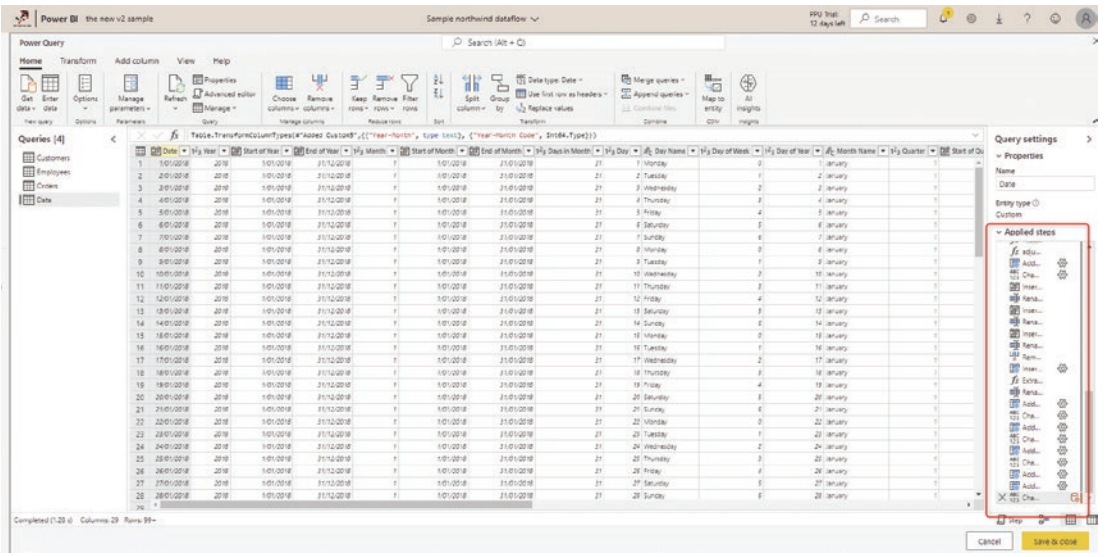


Figure 9-25. *Applied steps in the online Power Query Editor*

You can also see that there is a very useful diagram view in the Power Query Editor online that demonstrates all the steps and the data at each step. See Figure 9-26.

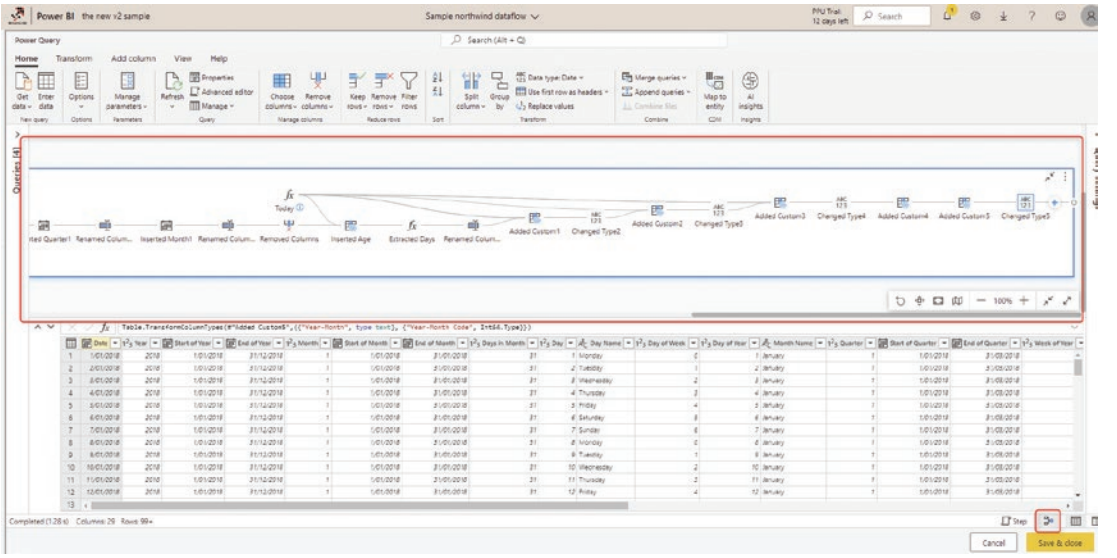


Figure 9-26. *Diagram view in the online Power Query Editor*

Scheduled Refreshes

You can configure dataflow refreshes at each dataflow level (not at the entity level), as shown in Figure 9-27.

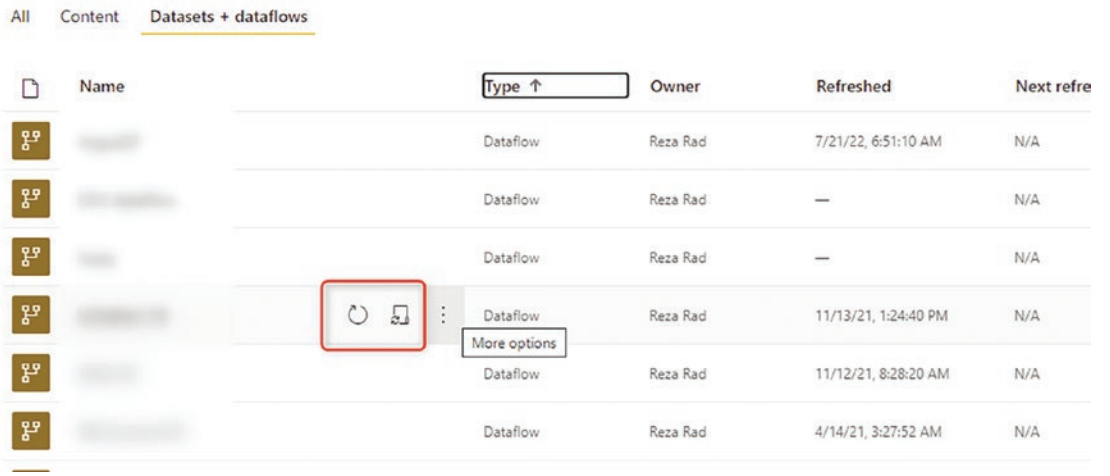


Figure 9-27. *Scheduled refresh of dataflows*

Get Data from Dataflow in the Power BI Desktop

In the Power BI Desktop, you can get data from the dataflows under the common data sources, as shown in Figure 9-28.

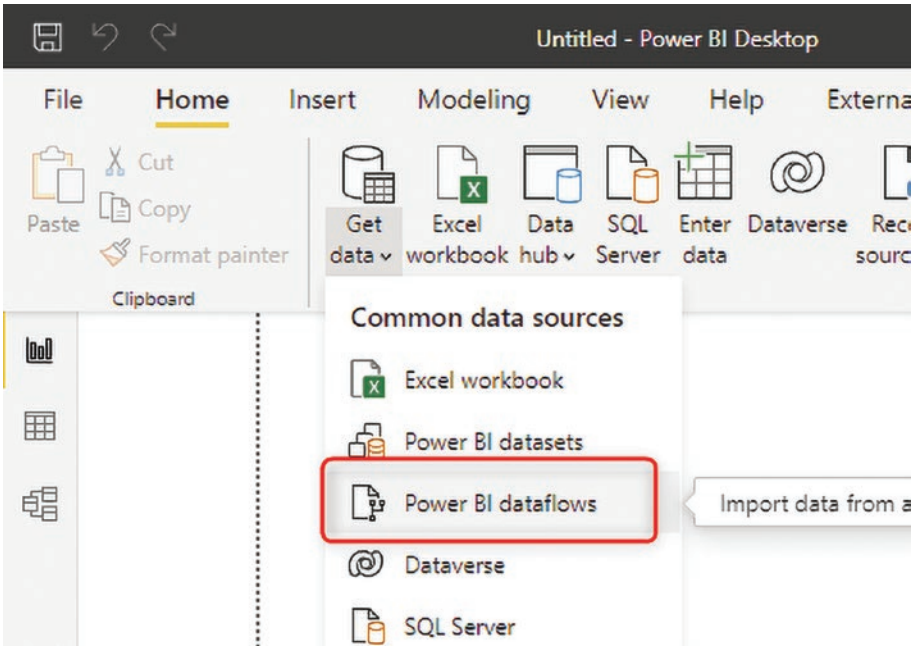


Figure 9-28. *Getting data from Power BI dataflows under the common data sources*

Under the Power Platform, you can choose Power BI dataflows. Note that there is also an option for dataflows in general, which includes Power Platform dataflows, as shown in Figure 9-29.

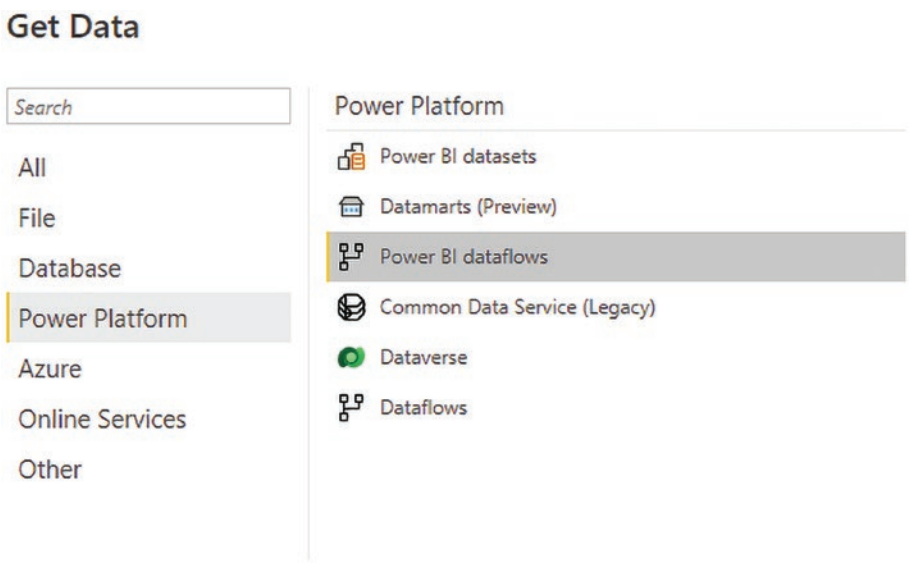


Figure 9-29. *Getting data from dataflows using the Power Platform menu option*

After entering your Power BI credentials, you can then see all the workspaces with dataflows under it, and you can select the tables you want from each dataflow, as shown in Figure 9-30.

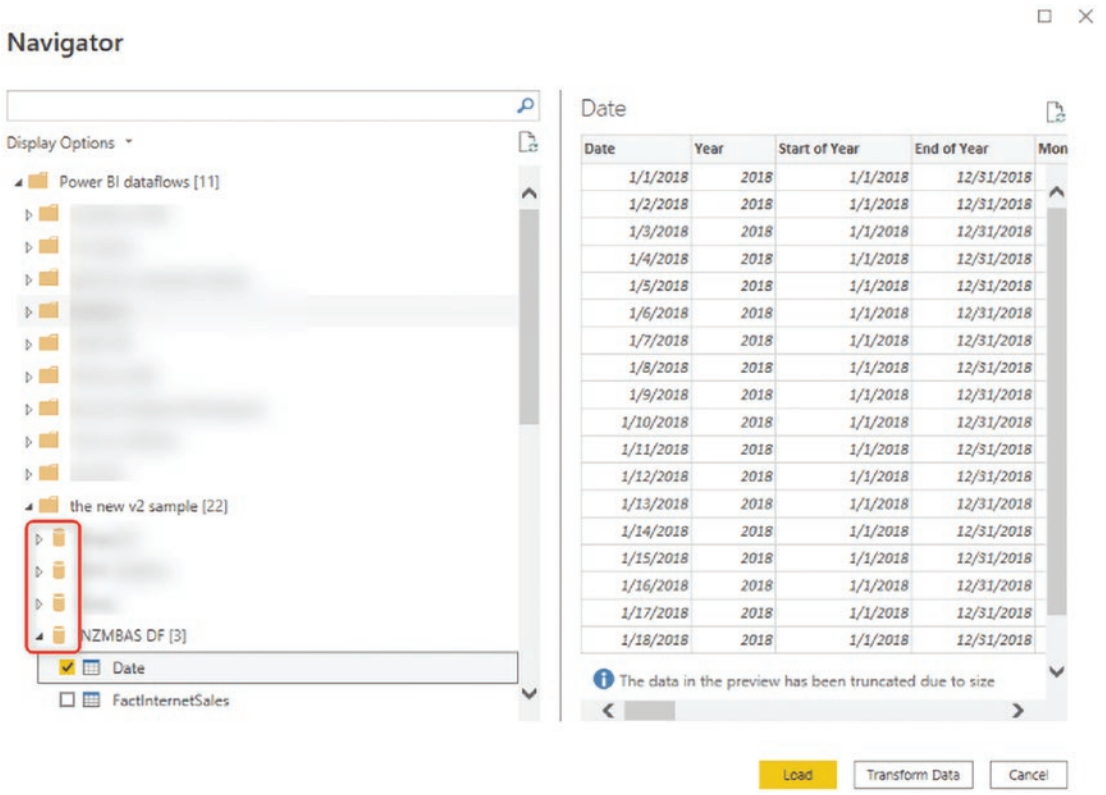


Figure 9-30. *Selecting the dataflow to get the data from the Navigator window*

Each dataflow is like a database, and you can have multiple tables in it. Once you get the data from the dataflow, the Power Query in the Power BI Desktop will simply import the data (but not the transformations) from the dataflow table, as Figure 9-31 shows.

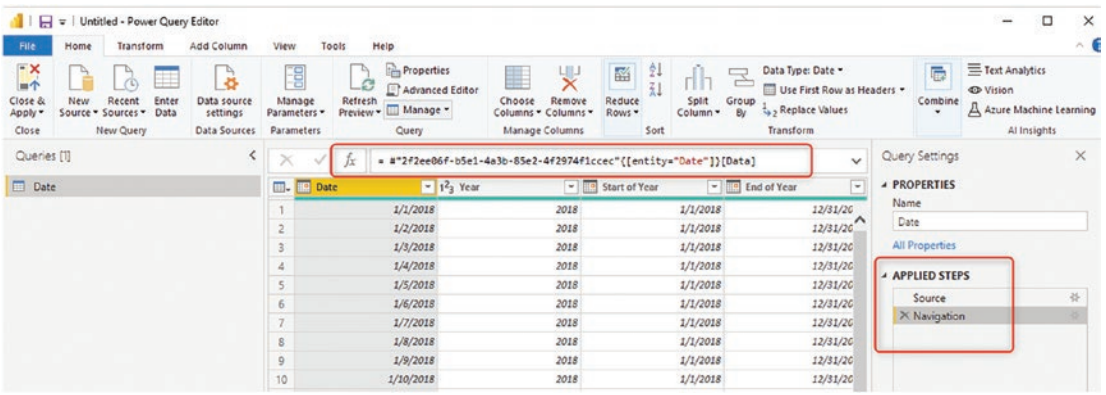


Figure 9-31. *Importing data from the dataflows*

It is possible to have a DirectQuery connection to the Dataflow tables as well, but that requires a Premium license.

Get Data from Dataflows Using Excel

You can also use Excel to get data from a dataflow table. This option gives you access to both Power BI and Power Platform dataflows in one place. See Figure 9-32.

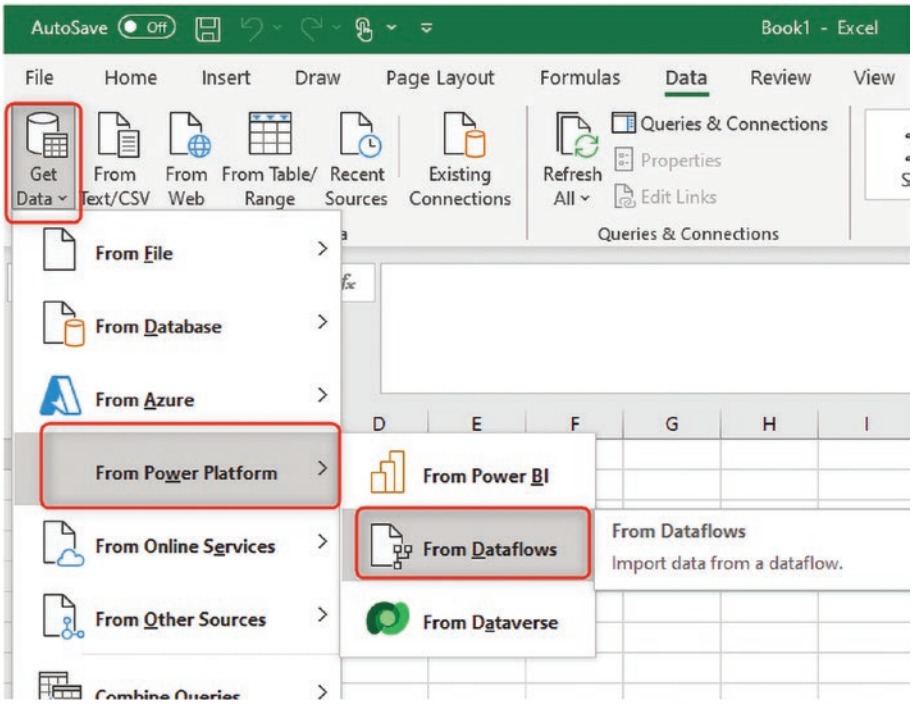


Figure 9-32. *Getting data from dataflows in Excel*

After entering the credentials, you can choose the dataflow from Environments (for Power Platform Dataflows) or Workspaces (for Power BI Dataflows). See Figure 9-33.

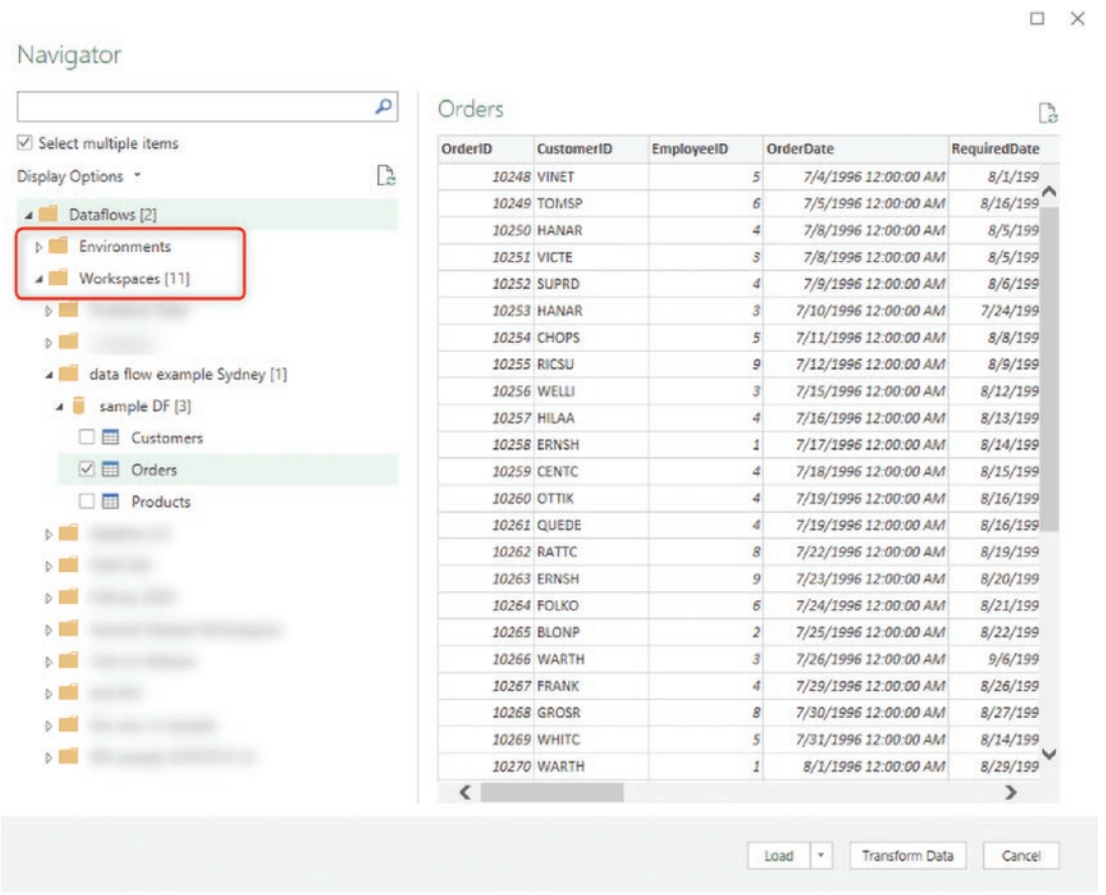


Figure 9-33. *Choosing from the dataflows in Excel*

Summary

Dataflows provide the data transformation engine of Power BI, which is independent of other Power BI objects. Dataflows are powered by the Power Query engine and the Power Query Editor online. Using dataflows, you can separate the ETL layer of the Power BI implementation from the rest of the work. Using dataflows is highly recommended because you can reuse your existing tables in multiple files. Dataflows are not just for Power BI, then can be used in Excel too, and they can be created in the Power Platform. Dataflows come in two categories—Standard and Analytical. In this chapter, you learned how to create a dataflow.

CHAPTER 10

Shared Datasets

Have you ever wanted to reuse part of a model in another report? Imagine two report visualizers on your team who want to create Power BI report visualizations from your data model. You have already done some modeling and calculations. How can this be done the best way without high maintenance costs? The answer is a shared dataset in Power BI. In this chapter, you learn about the following:

- What a shared dataset is in Power BI
- How a shared dataset can help with Power BI development
- Where the shared dataset is in the Power BI architecture
- How the shared dataset works behind the scenes in the Power BI service
- What certified and promoted datasets are

What Is the Dataset in Power BI?

When you create a Power BI report (a `*.PBIX` file) and the data connection mode is Import Data, the report has two components—a report and a dataset. When you are in the Power BI Desktop environment, you can't see the separation that easily unless you go to the task manager and see the dataset running behind the scenes under the Power BI Desktop task threads, as shown in Figure 10-1.

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_10

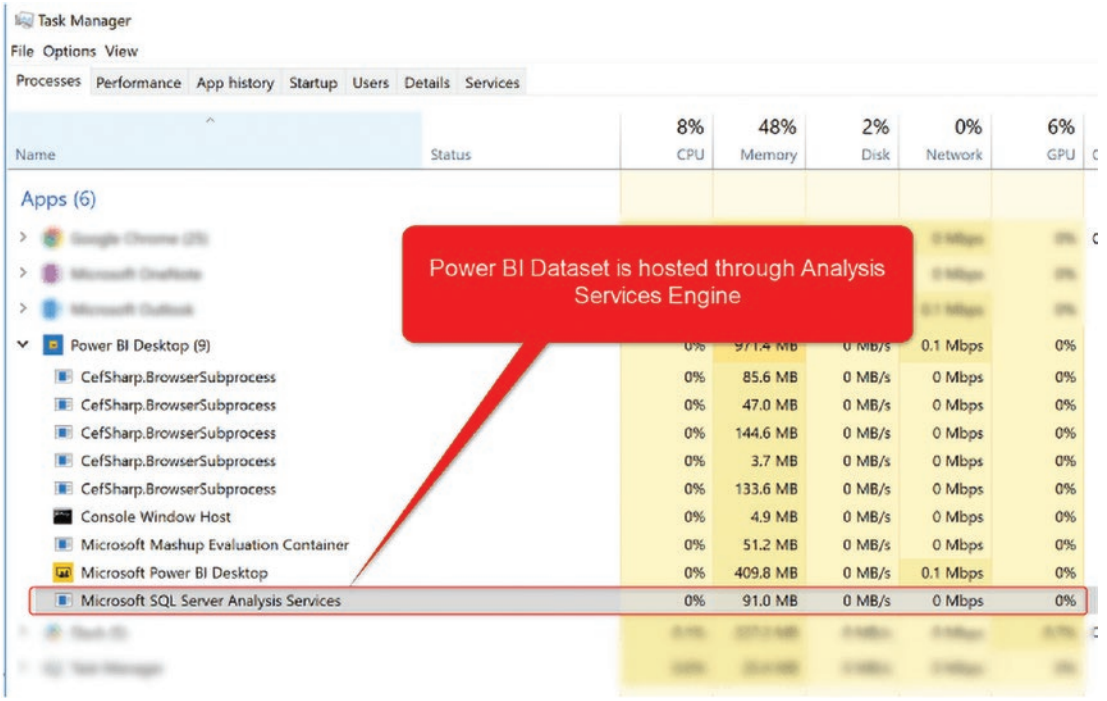


Figure 10-1. *The Task Manager showing the Microsoft SQL Server Analysis Service running behind the scenes of the Power BI Desktop*

However, when you publish the PBIX file into the service (the Power BI website), you can easily see that there are two objects—a report and a dataset.

- The *report* is the visualization layer of your Power BI implementation
- The *dataset* includes the data, tables, relationships, calculations, and connections to the data source (see Figure 10-2).

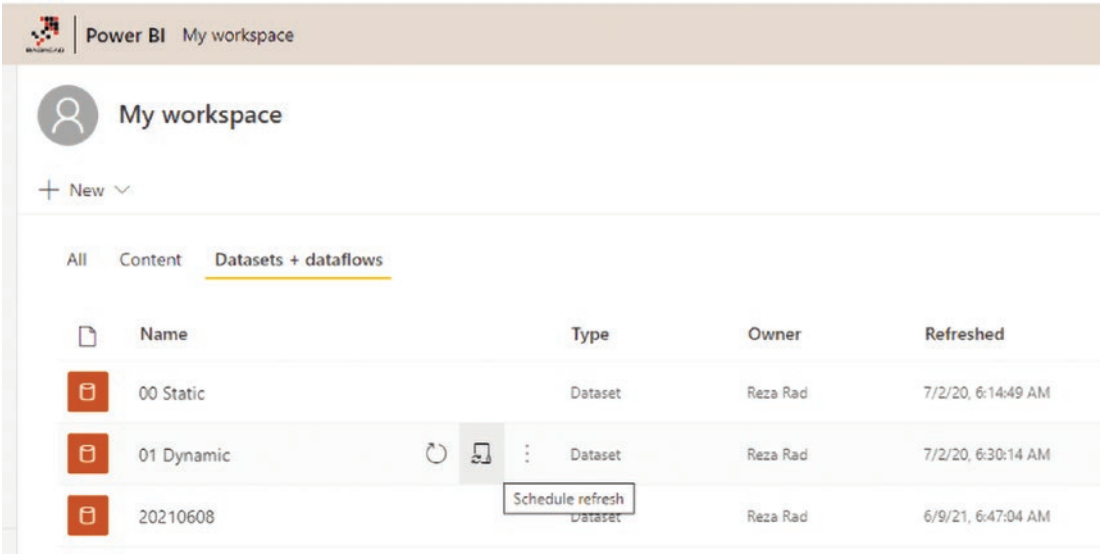


Figure 10-2. *Datasets in the Power BI Service*

You can schedule the refresh for the dataset and connect to on-premises sources (through a gateway) or cloud-based sources.

What Is Included in the Dataset?

So far, you know that a dataset is a separate object from the report. However, precisely which parts of the development are in the dataset? Here are some of the components that are part of the dataset:

- The connection to the data source
- Tables and their data
- Calculated columns, tables, and measures
- Hierarchies
- Formatting and settings of the fields (visibility, formatting, display folders, sort by column, data category, and so on)
- Relationships

Anything that somehow is related to the data is part of the dataset.

What Is a Shared Dataset?

Now that you know about the dataset, let's talk about the shared dataset. A shared dataset is a dataset that's shared between multiple reports. You can create a new report from an existing dataset through the Power BI website, as demonstrated in Figure 10-3. This will create a report without a dataset. In fact, the dataset of that report is the dataset that you are creating the report from. This type of report is also called a *thin* report.

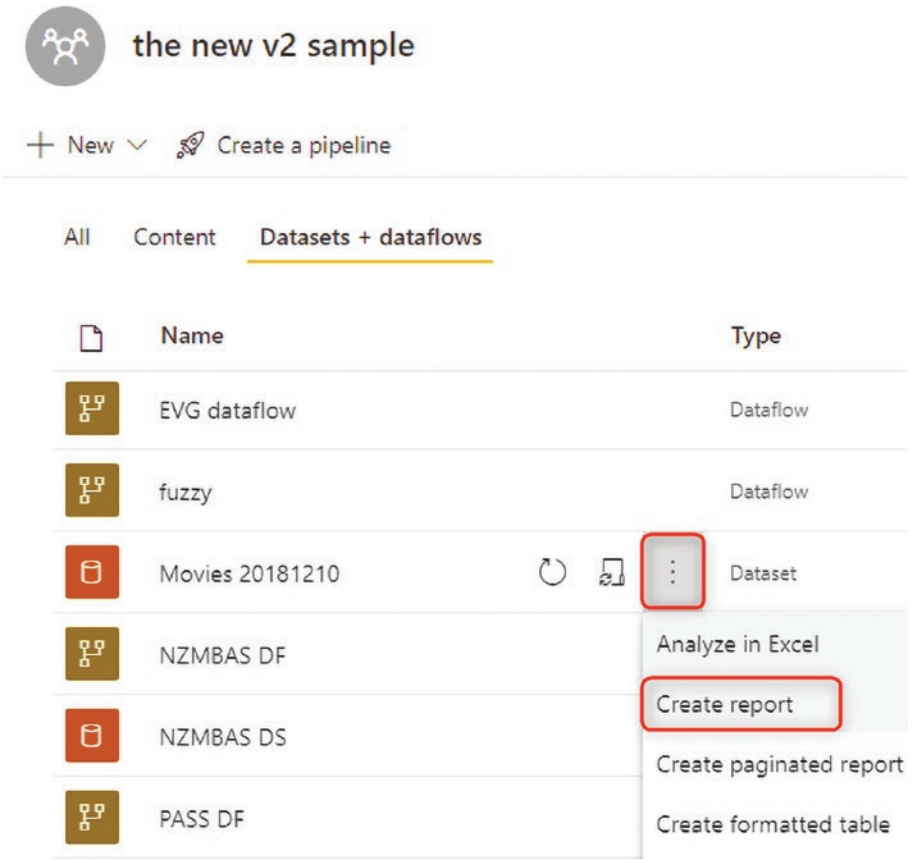


Figure 10-3. *Creating thin reports from the Power BI dataset*

A thin report is a report without a dataset. This type of report is usually connected live to an existing dataset.

You can also create thin reports from the Power BI Desktop. To do this, you can choose Power BI Dataset under the Data Hub, as shown in Figure 10-4.

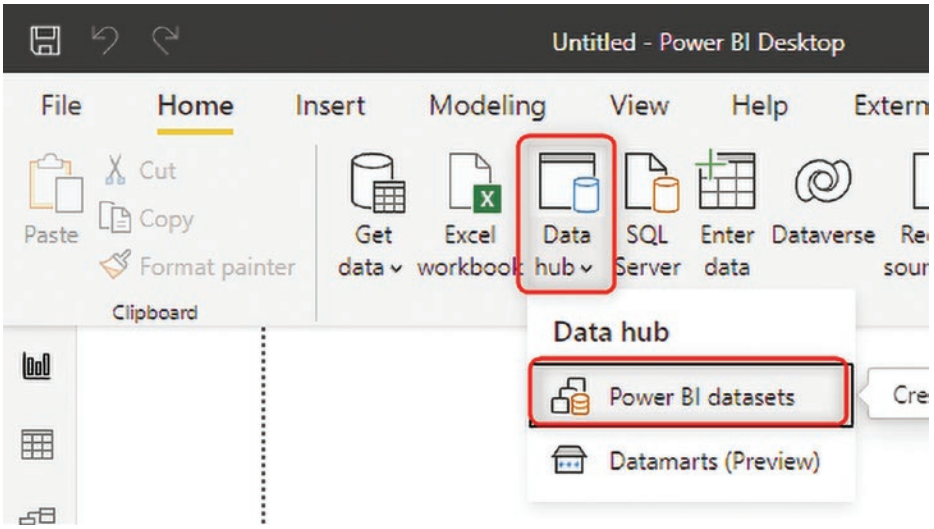


Figure 10-4. *Creating thin reports from the Power BI Desktop*

Thin reports have the same attributes as the Power BI reports connected using Live Connection. They give you the ability to create report-level measures, but beyond that, the modeling is limited, unless you create a composite model using DirectQuery to a Power BI dataset.

When a shared dataset is refreshed, all of the associated reports will have new data. A shared dataset is one step closer to the multi-developer tenant in the Power BI environment. Figure 10-5 shows an example of a shared dataset.

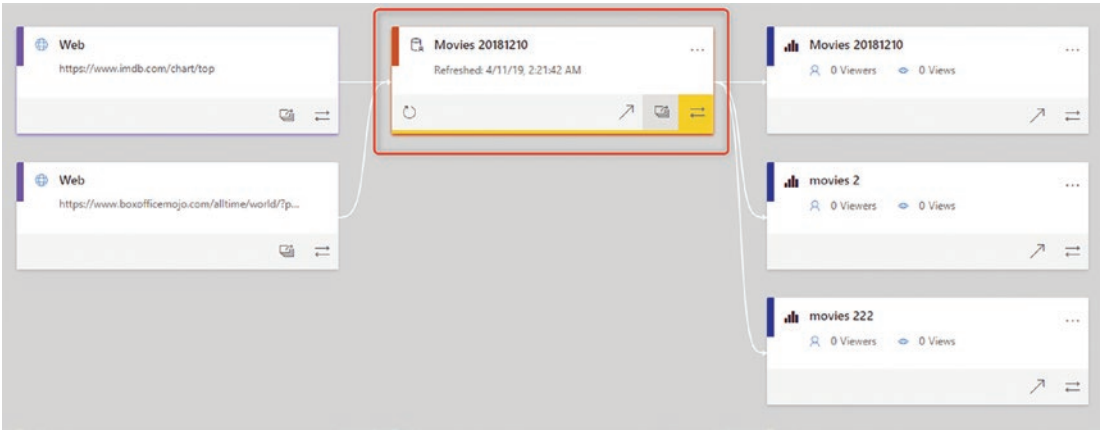


Figure 10-5. *Power BI Shared dataset*

How Do You Create a Shared Dataset?

Any Power BI dataset can be a shared dataset. To use it as a shared dataset, first you need to publish your PBIX file to the Power BI Service. After publication, you will have a dataset and a report. The dataset can then be used to create other reports.

If you want to create a dataset without a report, at the time of writing this book, it is not possible to do that using Power BI Desktop. You can create a Power BI file (which includes the dataset and report in one), and then publish it to the service. After publishing it to the service, you can delete the report associated with that dataset. The challenge with this method, however, is that any future updates to the Power BI dataset and republishing will also republish the report again, and you need to then remove it. It might be easier to ignore the report associated with the dataset. Or simply just use that report to troubleshoot your dataset.

Sharing Datasets Across Multiple Workspaces

For a long time, sharing datasets was only possible inside a workspace. You could not use a dataset from one workspace as the source for a report in another workspace. However, the feature became available a few years ago, and you can now share the dataset across multiple workspaces. When you get data from a Power BI dataset through the Power BI Desktop, as shown in Figure 10-6, you can select which dataset you want to get the data from.

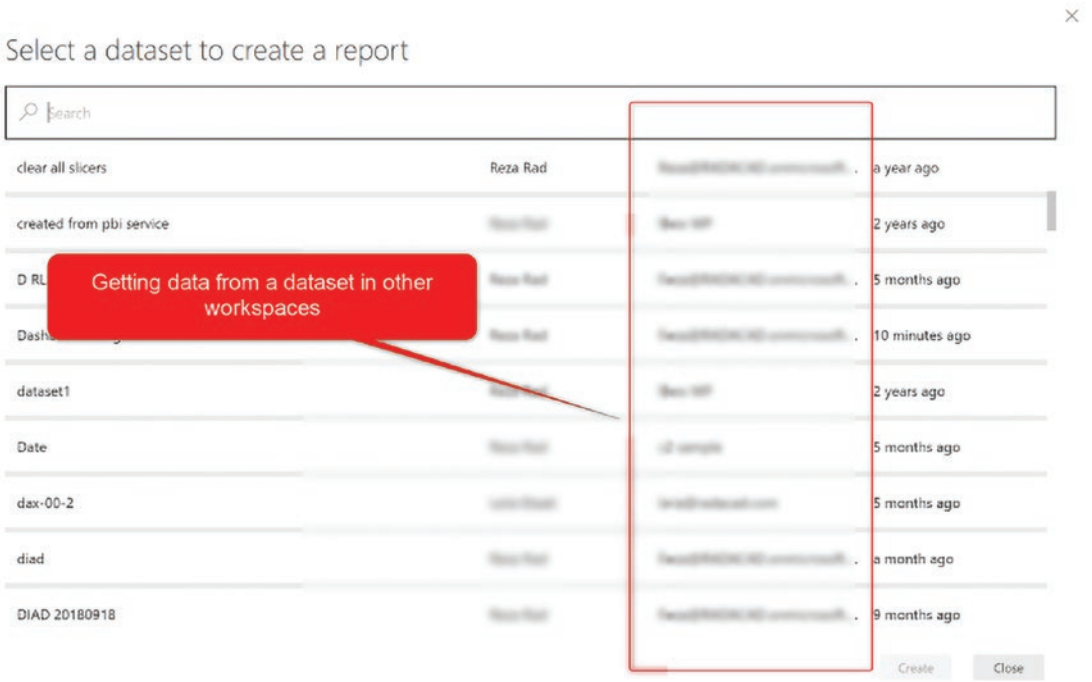


Figure 10-6. *Getting data from a Power BI Dataset*

External Dataset: How Does Shared Dataset Work Behind the Scenes?

When you get data from a Power BI dataset in workspace 1, and then save your report in workspace 2, you will see something called an *external* dataset (it was previously called a linked dataset). The fact is that what you see is just a link. Power BI will bring the link to that dataset into the new workspace. This link helps you understand when the dataset gets refreshed.

Figure 10-7 demonstrates what an external dataset looks like in Lineage View.

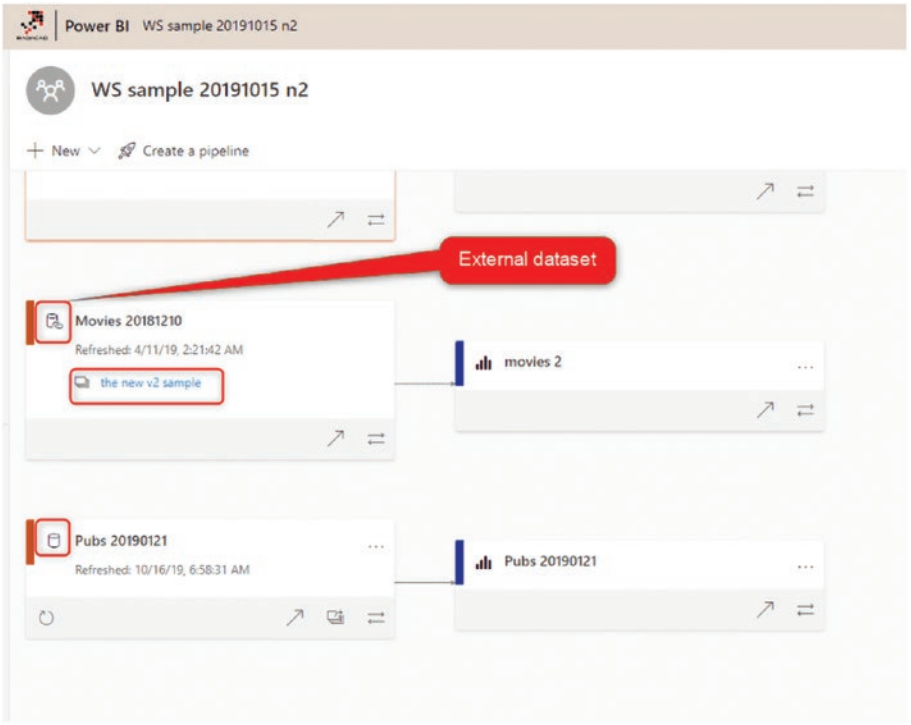


Figure 10-7. *An external dataset in Power BI*

An external dataset is not a copy; it is a link to the original dataset.

Why Use Shared Datasets?

Now the million-dollar question is why should you use a shared dataset? What is good about it? What are its main benefits? Let's answer these questions through an example.

Let's assume you are the Power BI developer of the `Sales.PBIX` file on your team. Your team recently hired a data analyst with good visualization skills named Maggie. Maggie wants to build some visualization on your `Sales.PBIX` file, However, you want to do some modeling (writing calculations, bringing more tables, adding relationships, and so on) at the same time. How can you do this?

Suppose you give Maggie a copy of your `Sales.PBIX` file and call it `Maggie's Sales.PBIX`. She can build new visualizations, but now her version will be different from your `Sales.PBIX` file. What if you wanted to merge your changes (new calculations and tables) into her file? This brings lots of headaches because you're managing two versions of the same file.

Instead of copying files, you can create a shared dataset, and Maggie can create a Power BI thin report connected to the same dataset. This way, maintaining the solution will be much easier in the future. Whenever you update your dataset, Maggie needs to refresh the file to get the new changes. A shared dataset separates the modeling layer of your Power BI solution from the rest of it. It's like a dataflow, which separates the ETL layer.

Shared Datasets in the Power BI Architecture

In a later chapter, I explain how dataflow and shared datasets can play an important role in the multi-developer tenant of Power BI implementation.

In a nutshell, using the dataflow ensures that you can bring the data well prepared in a central area, which you can call a centralized data warehouse in the Azure Data Lake. Using the shared datasets, you can build data models that multiple reports can use. Figure 10-8 shows how the architecture works in diagram view.

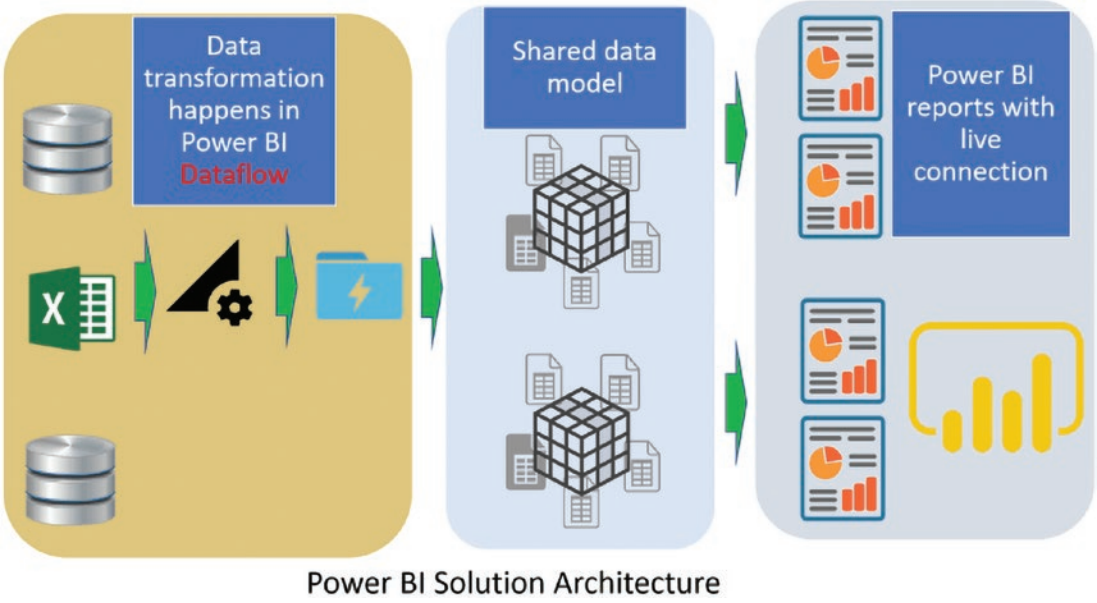


Figure 10-8. *Power BI Solution Architecture diagram*

Instead of having silos of Power BI reports and files everywhere, you can build an architecture that works best with multiple developers, has less data, code, and logic redundancy, and is easier to maintain. I highly recommend reading this chapter to learn more about this architecture and learn how the shared dataset located in this architecture is a key element.

Endorsement: Certified and Promoted Datasets

When Power BI developers get data from a Power BI dataset, they see all the datasets from all workspaces that they have access to. This might be a bit confusing. There might be many datasets shared in the environment. The developer might wonder which of those they can use.

A labeling system is added to the Power BI datasets, which helps in this scenario. You can mark some of the datasets as certified or promoted. To certify a dataset, an approval process can ensure the dataset has passed some of the tests. You can clarify through this labeling system which datasets are good to be used as the source and which are not. You can build the concept of Gold, Silver, and Bronze datasets. Gold datasets are fully tested, and reconciled, whereas a Bronze dataset hasn't been through any testing yet.

To use this labeling system, the creator of the dataset can go to the dataset's settings area, as Figure 10-9 demonstrates.

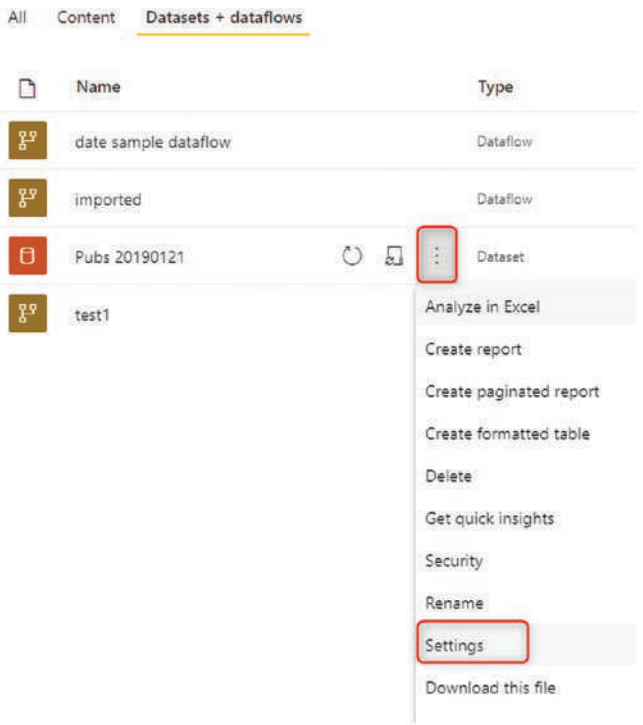


Figure 10-9. *Dataset settings*

In the settings, you can set the Endorsement level, as shown in Figure 10-10.

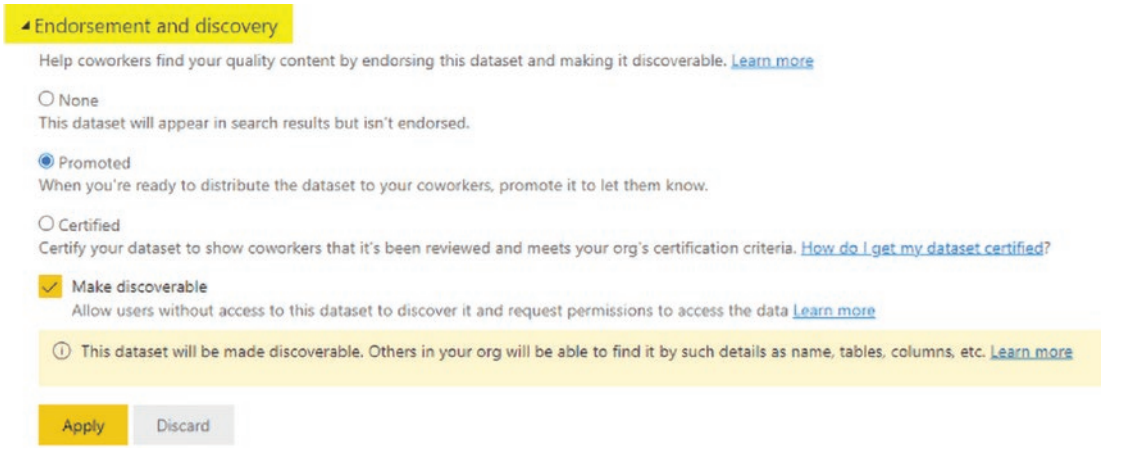


Figure 10-10. *Power BI dataset endorsement*

As you can see, the Certified option might not be available. The Power BI tenant administrator has the authority to enable labeling and give out access in the Tenant Settings. Take a look at how to enable access in Figure 10-11.

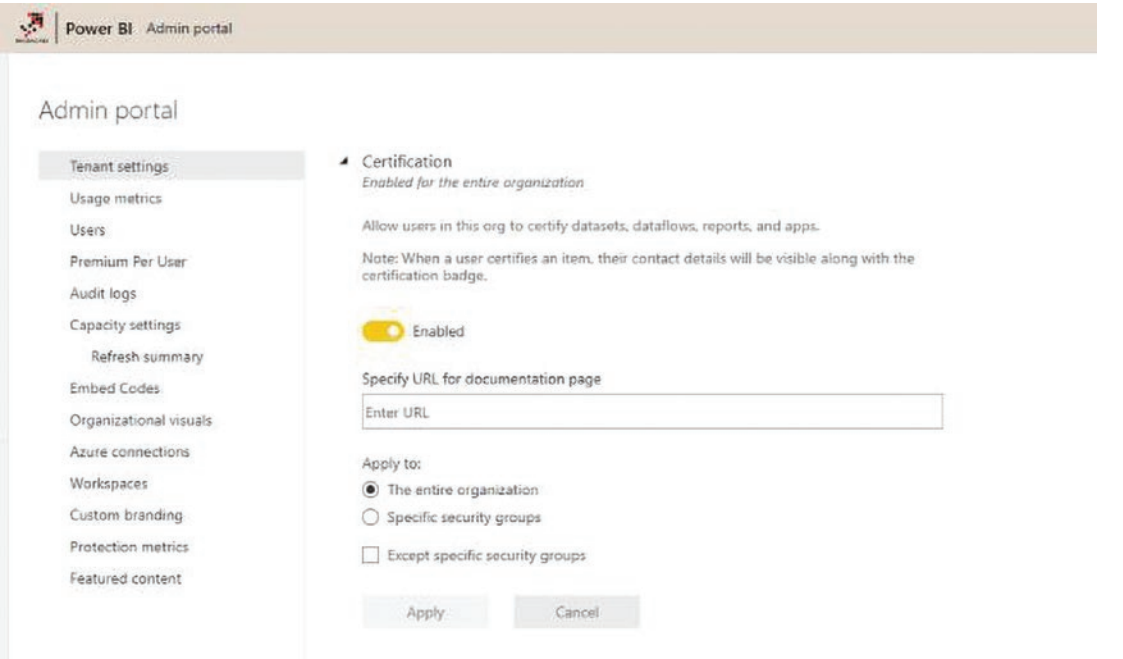


Figure 10-11. *Admin setting determining who can certify datasets in the Power BI Service*

You can also determine if promoted or certified content is discoverable throughout the tenant under the same Tenant Settings of the Admin Portal, as shown in Figure 10-12.

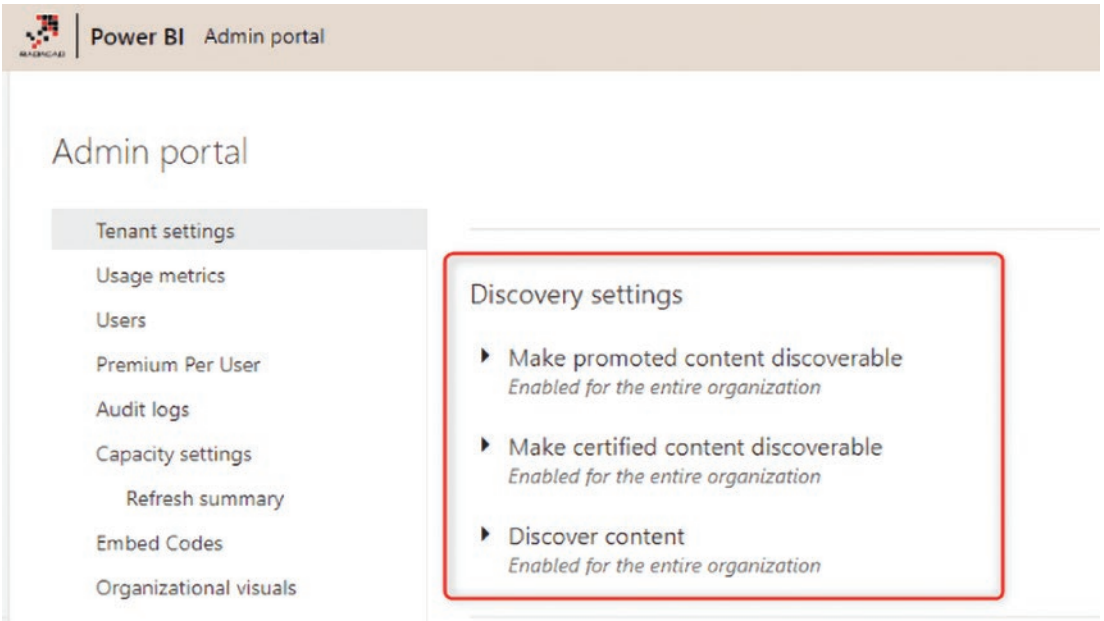


Figure 10-12. *Discovery settings in the Power BI admin portal*

The endorsement labeling system, shown in Figure 10-13, helps Power BI developers determine what is the level of certification that a dataset must be used as a shared dataset, and then can select based on that; see Figure 10-13.

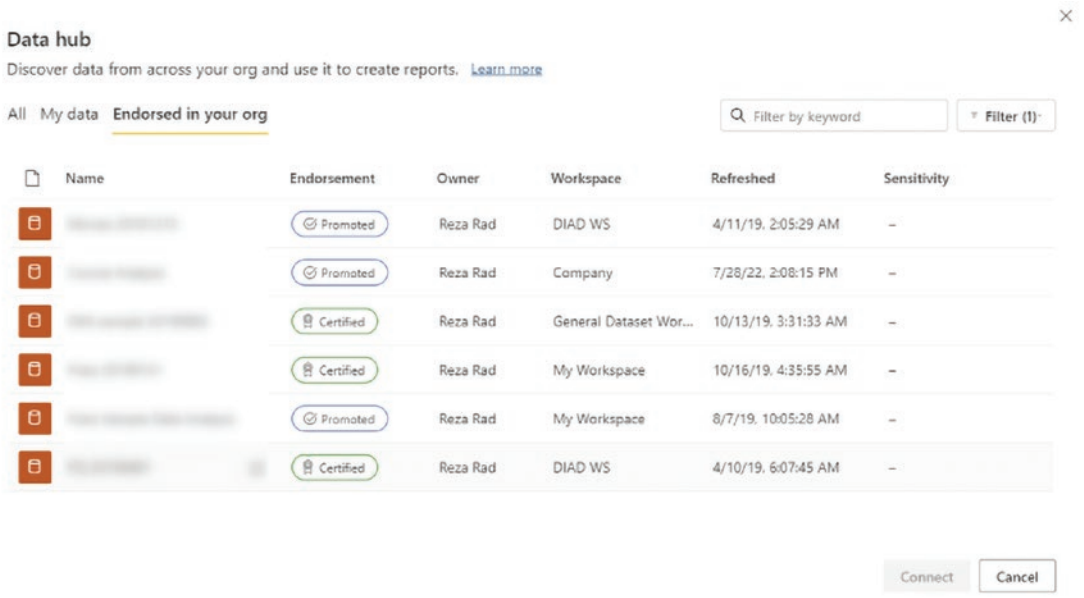


Figure 10-13. *Filtering the datasets by the endorsement in the Power BI Desktop*

Summary

You can use a shared dataset to create centralized data models serving multiple reports. You can reduce the maintenance time, the redundancy of the code, and the data using this approach. Having a labeling system of the certified or promoted dataset is also a great way to put some process and governance in place and ensure that the shared datasets have been through some process of testing and reconciling.

CHAPTER 11

Datamarts

One of the newest additions to Power BI is the datamart. Power BI datamarts are more than just another feature; they are a major milestone where the development of Power BI solutions are revolutionized. They help both citizen data analysts and developers. This chapter explains what datamarts are and how they help you with your Power BI implementations.

Power BI for the Citizen Data Analyst

Power BI came to the market in 2015 with the promise of being a tool for citizen data analysts. A citizen data analyst is someone who does not have a developer background but understands the business and the data related to that business. Power BI (and many other self-service tools) target this type of audience.

You don't need to be a developer to use the Power BI Desktop. It's so easy and straightforward that even by just opening the tool and clicking here and there you can easily learn how to use it. If you read a few guides, you can build your first report and dashboard using Power BI. That is exactly the promise that Microsoft offered with Power BI.

Governance and Reusability

As time goes by in your Power BI development cycle and you build more Power BI files, you may realize that you need something else. Building everything in a Power BI file is simple, but maintenance is problematic. What if you want to reuse a table in another Power BI file? What if you want to reuse a measure or expression in another report?

That is why Power BI offers separate components that build the full architecture of Power BI development, components, features, and technologies, such as thin reports (reports that don't have a dataset and connect live to another dataset), shared datasets (datasets that can be used to feed data into multiple reports), dataflows (the data transformation engine in the cloud), composite model (combining a shared dataset with additional data sources), and so on. All of these technologies create a better development lifecycle for Power BI developers. Figure 11-1 shows this Power BI multi-layered architecture.

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_11

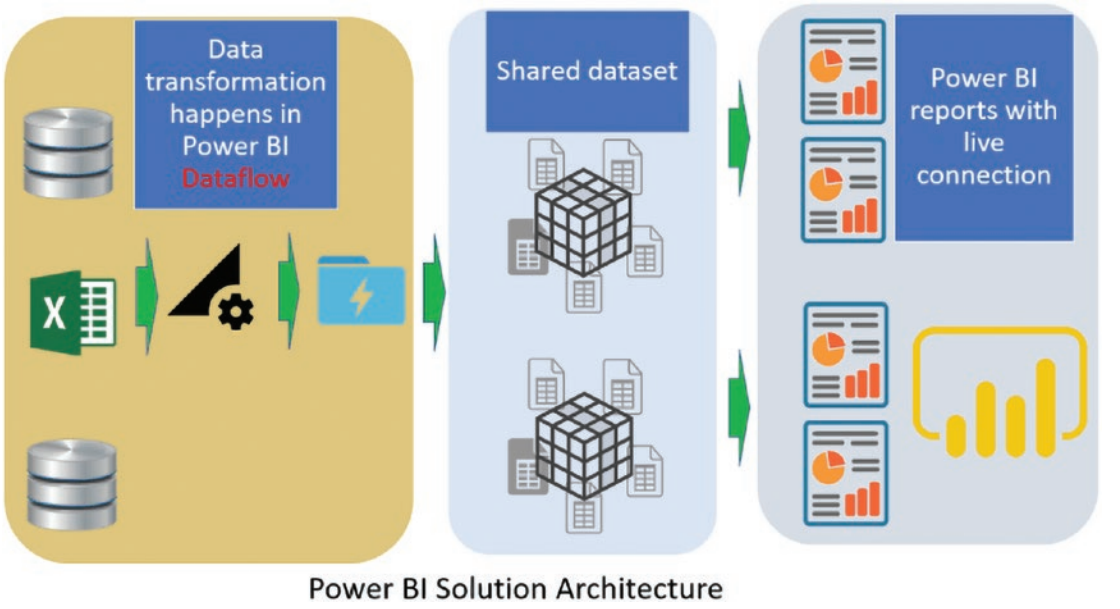


Figure 11-1. *Power BI Solution architecture with separate layers*

Although all these components are fantastic features in the Power BI ecosystem, there is still a need for a database or a data warehouse as a central repository of the data. The need for this repository comes from many different aspects—keeping the integrated data structured in a relational database, having a central database with all the data from other source systems in it, creating views to cover particular needs for reports, and more.

What are Datamarts in Power BI?

Datamarts are used to close the database gap in the Power BI ecosystem, but they are much more than that. If you want just a database, you can design it in an Azure SQL database or other platforms. The problem is that you need to build the database in a tool such as SSMS (SQL Server Management Studio), then have an ETL process (such as dataflows, ADF, or SSIS) to feed data into that database, and then the Power BI dataset using the Power BI Desktop. As you can see in Figure 11-2, datamarts give you a single unified platform to build all of these without needing another tool, license, or service. Datamarts make Power BI enough for you to do all your BI requirements, but with less control over how the database is implemented.

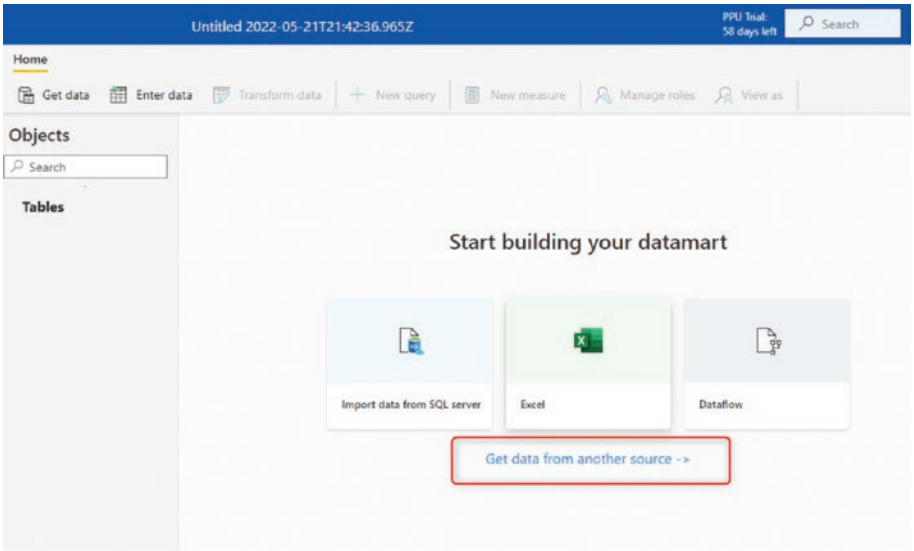


Figure 11-2. *Building a datamart*

A Power BI datamart is a combined set of dataflows, an Azure SQL database, a Power BI dataset, and a Web UI that manages and builds all these functionalities, as depicted in Figure 11-3.

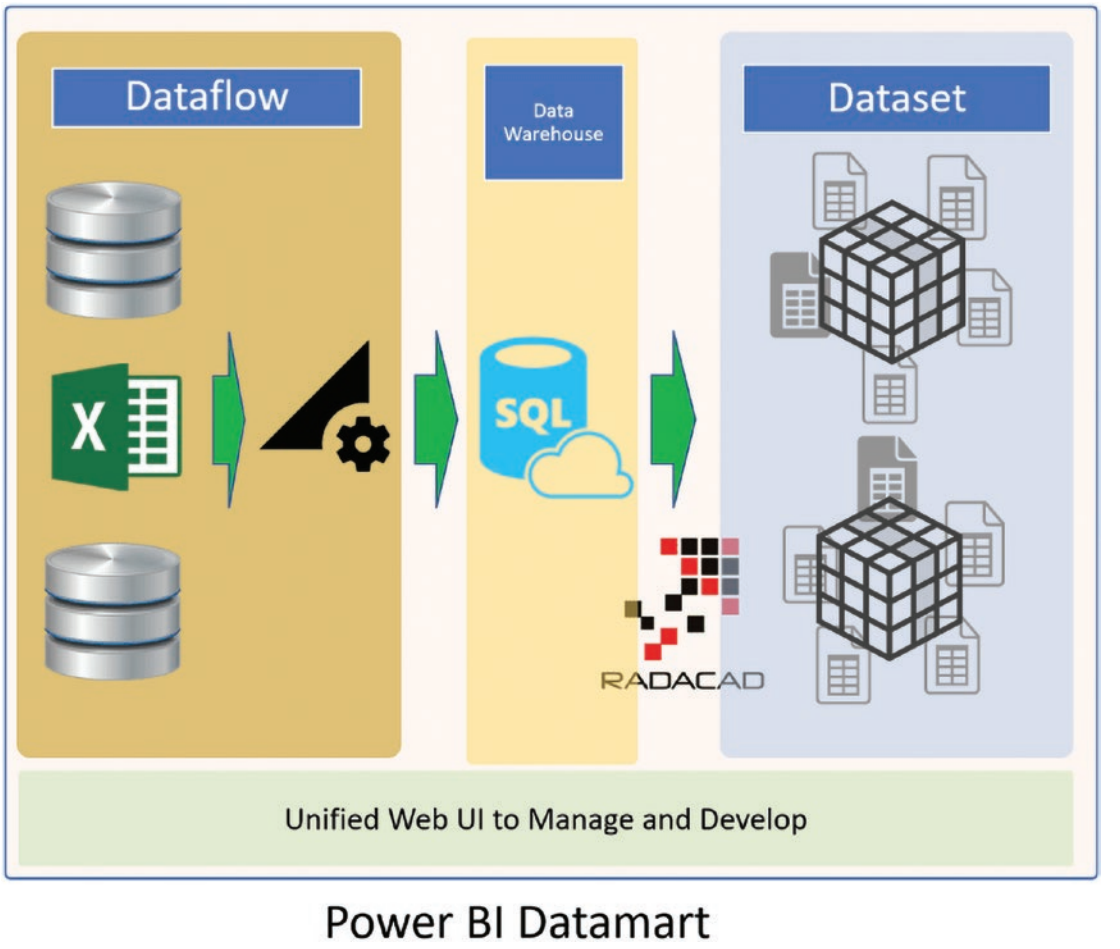


Figure 11-3. *Datamart components*

The data warehouse term I use here sometimes causes confusion. Some will use the term data warehouse for huge databases that need to scale with technologies such as Azure Synapse. However, the term data warehouse here means the database or repository where you store the star-schema-designed tables of dimension and fact tables for the BI model. These tables can be big or small.

Who Are Datamarts for?

If you are wondering who datamarts are for, or who can use them, the following sections explain a few scenarios.

Power BI Datamart for the Citizen Data Analyst

Daniel is a data analyst at a small to mid-size company. His background is not in development. He knows the business though. He understands how the business operates and he understands the data related to the business. He wants to build dashboards and reports in Power BI. If he does all that in the Power BI Desktop, soon he will realize that there isn't good governance around such a structure. His company doesn't have a data warehouse as such, and no BI team to build such a thing. He can use a Power BI datamart to have a fully governed architecture with dataflows (transformation and ETL layer), an Azure SQL Database (data warehouse or dimensional model), a Power BI dataset (the analytical data model), and reports. All of these can be developed using the UI of the Power BI Service. Daniel does not need to open any other tool or service; he does not need to learn SQL Server database technology or any other technologies except the Power BI itself. This is an example of a datamart empowering a citizen data analyst to build a Power BI solution that is scalable, governed, and self-service at the same time.

Power BI Datamart for Enterprises

Arwen is a data analyst at a large enterprise with a data warehouse and BI team. However, every time Arwen asks for a change in the centralized data model from the BI team, it takes months if not years to get the results (because of the bottleneck of requests). Now using datamart, Arwen can build her data warehouse with the data transformation layer and everything in a way that can be consumable for future projects or by colleagues using Power BI. The solution will be governed by the Power BI service, the BI team can implement a process for certifying datamarts and, as a result, Arwen can not only implement quicker but also can help the BI team reduce their backlog. Power BI datamarts empower both Arwen and the BI team to implement faster Power BI solutions in a fully-governed structure.

Power BI Datamart for Developers

Peter is a BI developer. He knows how to work with databases and write T-SQL queries. He can use the Web UI of the datamart to write T-SQL queries to the Azure SQL database. Or he can use the database connection and connect to the database using a tool such as SSMS. He can also connect to the dataset built by the datamarts using the XMLA endpoint using SSMS, Tabular Editor, or any other tools to enhance the data model and take it to the next level. Power BI datamarts empower Peter in his development work throughout his Power BI implementation.

What Are the Features of Datamarts that Empower Power BI Development?

Throughout this chapter thus far, you have learned about some of the features of datamarts that empower the Power BI developers. The following sections explain these features separately.

Datamarts Complete the BI Ecosystem

Datamarts use the dataflows for data transformation, an Azure SQL database for the data warehouse (or dimensional model), and the Power BI dataset for the analytical data model. Finally, the Power BI report can connect to the dataset. This builds a complete four-layer implementation in Power BI, as shown in Figure 11-4.

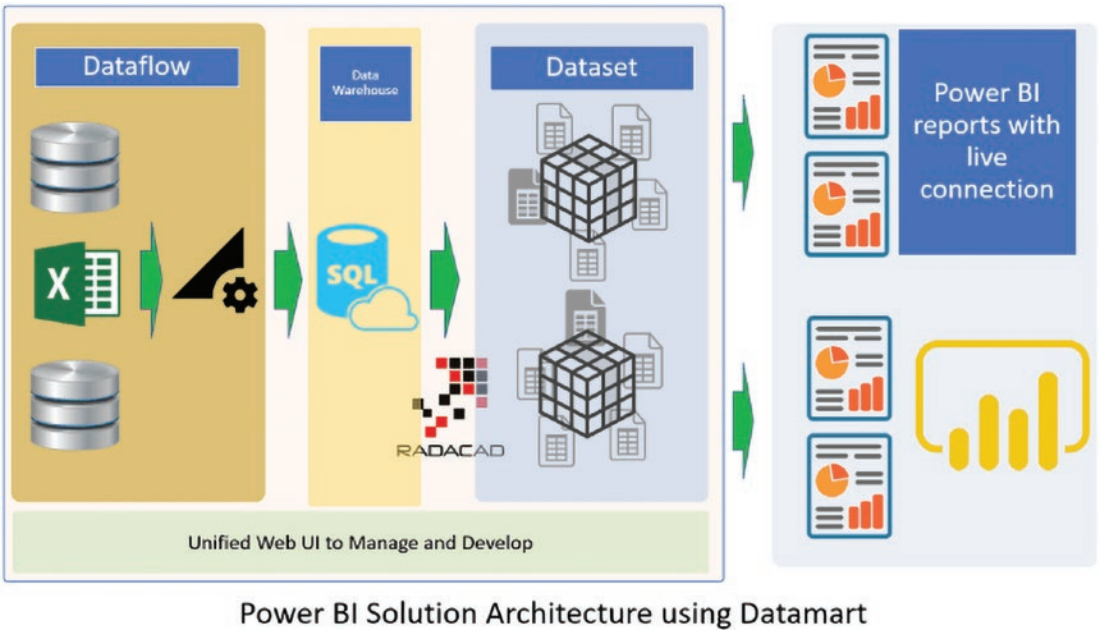


Figure 11-4. *Datamart in the Power BI architecture*

One Place to Manage and Build, No Other Tools Needed

The user interface to build datamarts is web-based. You don't even need to install the Power BI Desktop. You can use any operating system (Mac, Windows, or even a tablet). You build the entire Power BI solution, from getting data from data sources all the way to building the reports, using the same UI in the Power BI Service. You don't need SSMS, Visual Studio, Power BI Desktop, and so on. See Figure 11-5.

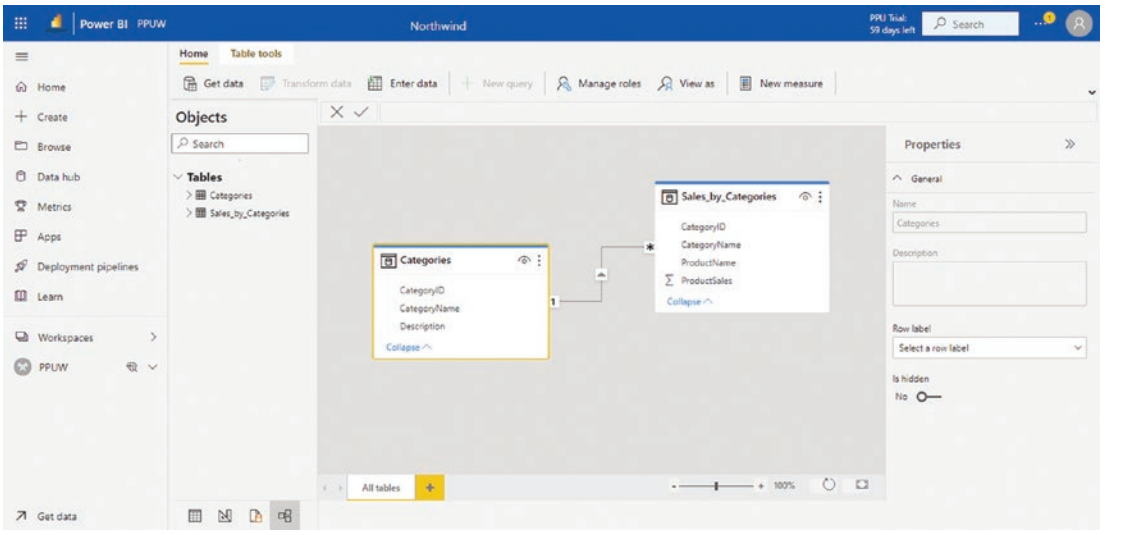


Figure 11-5. *Single unified UI to manage everything: datamarts*

Although at the early stages of building datamarts, there are some functionalities that are not yet % possible using the Web UI, this will improve in the near future.

One License Is Enough

Datamarts build an Azure SQL database for you, but you don't need to purchase a separate license from Azure Portal. You don't even need to have an Azure subscription. Power BI datamarts give you all of that using the Power BI Premium capacity, or Premium Per User license. The database, the Dataflow, and the dataset are all part of your Power BI license.

The Vast Horizon and the Future of Datamarts

Datamart is just the beginning of many wonderful features to come. Imagine features such as Slowly Changing Dimension (SCD) and inferred dimension members handling implementations. You can think about monitoring the dataflow processes in a way that the incremental refresh data that is processed every night is stored in log tables and you can troubleshoot any potential problems easily.

Think about what features can be enabled now that there is a single Web UI enabled for the developers, version control, and the ability for team members to work on the same Power BI project simultaneously. This is all on the horizon. These features do not exist yet in datamarts. However, datamarts can be the base on which all these amazing features can be built.

Governance

Like many other objects in the Power BI workspace, datamarts can have governance aspects such as endorsements and sensitivity labels. If the datamart is marked with specific organizational sensitivity labels, and the link is somehow sent by mistake to someone who isn't part of the organization and should not see the data, that is all covered by the sensitivity labels and configurations of Microsoft Azure behind the scenes.

Getting Started with Power BI Datamarts

So far, you learned what a datamart is and the use cases in a Power BI implementation. In this section, you experiment with datamarts and learn through an example how they work. The interesting thing in all the steps that follow is that you only need a web browser to build a datamart.

Premium Workspace

Power BI datamarts are only accessible through a Premium workspace. You either need to have a Premium capacity workspace or create a workspace using a Premium Per User (PPU) account, as you can see in Figure 11-6. If you don't have a PPU account, you can easily apply for a 60 day trial through the Power BI Service.

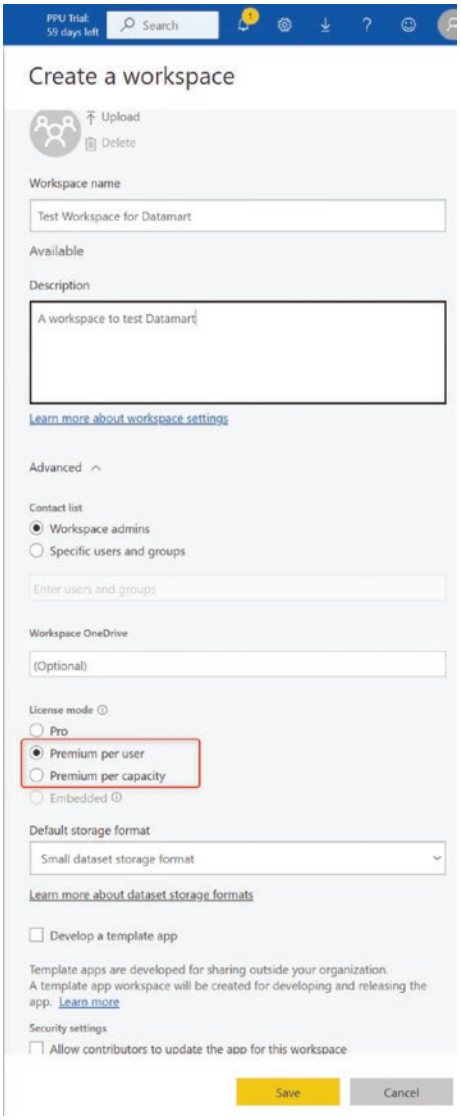


Figure 11-6. *Creating a workspace to test a Power BI datamart*

A workspace with Premium settings usually has an icon representing this fact, as shown Figure 11-7.

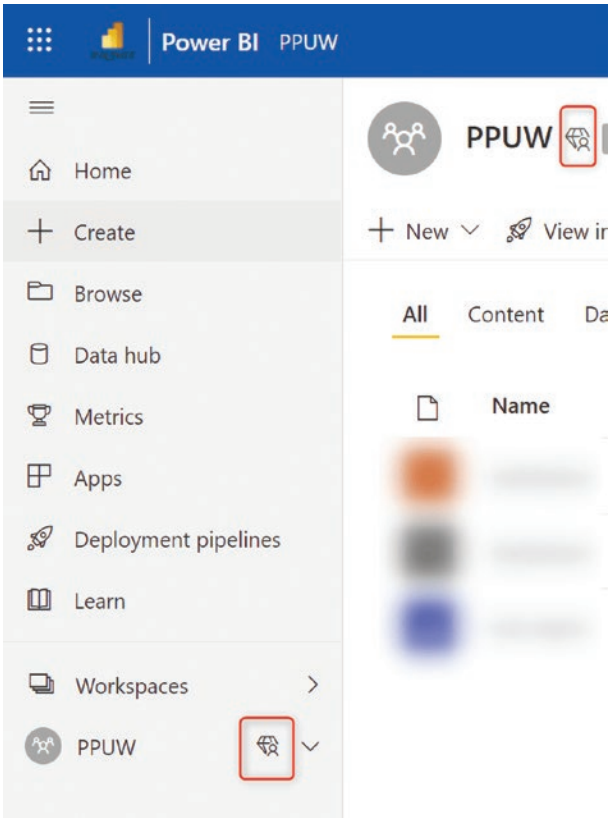


Figure 11-7. *The Power BI premium workspace icon*

Creating a Datamart

Click New, and from the list, select Datamart, as shown in Figure 11-8.

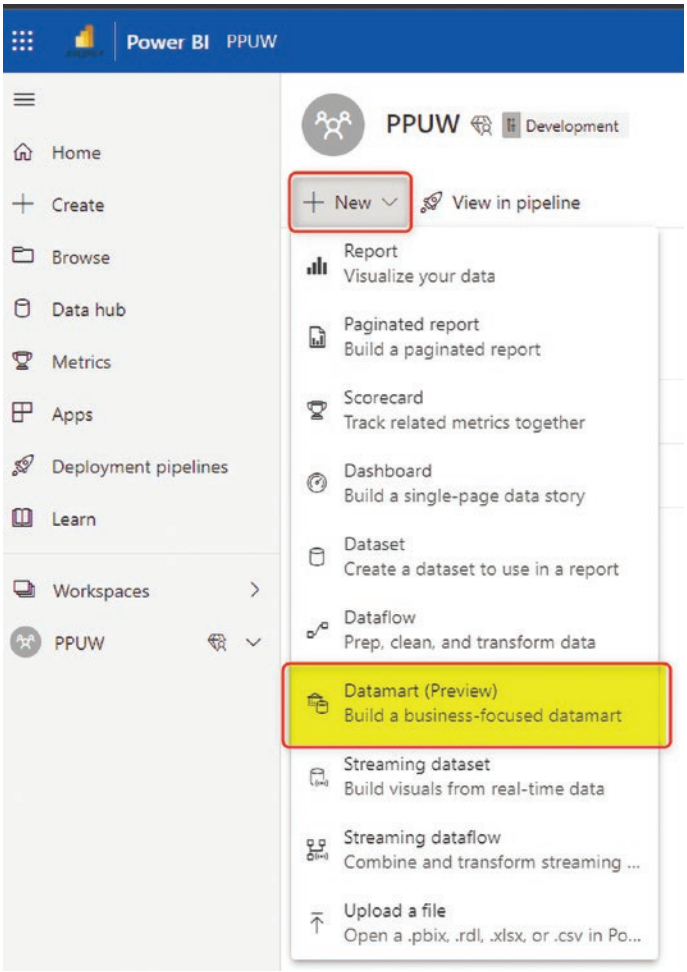


Figure 11-8. *Creating a Power BI datamart*

Note that the ability to create datamarts can be enabled or disabled through the Tenant Settings of the Power BI admin portal, as shown in Figure 11-9.

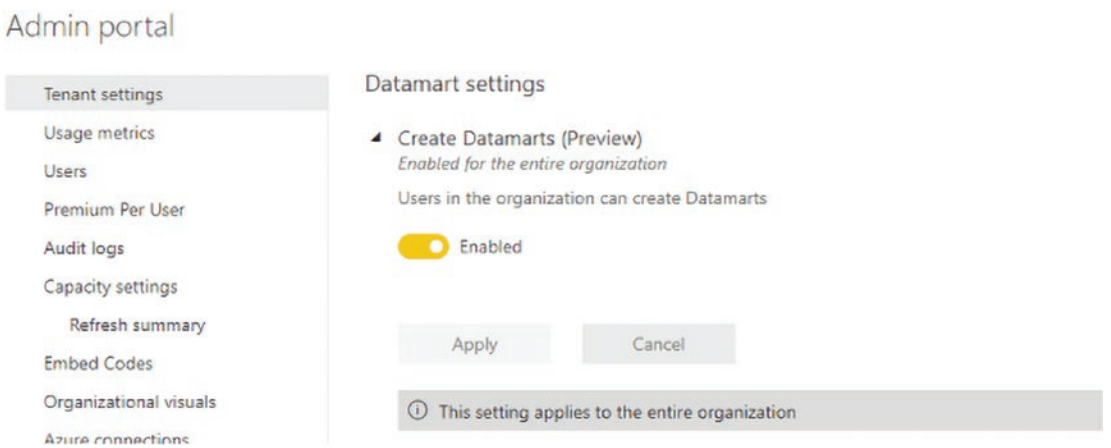


Figure 11-9. *Enabling datamart creation in the Power BI Tenant*

Get Data: Power Query Online

When you create a datamart, the first thing you do is to choose where to get the data. You can start by getting data from a dataflow, Excel, SQL Server, or pretty much any other data source that Power BI offers a connector to (more than 150 sources). To do this, select the Get Data from Another Source.

This will open dataflow's familiar Get Data window, where you select the source, as depicted in Figure 11-10.

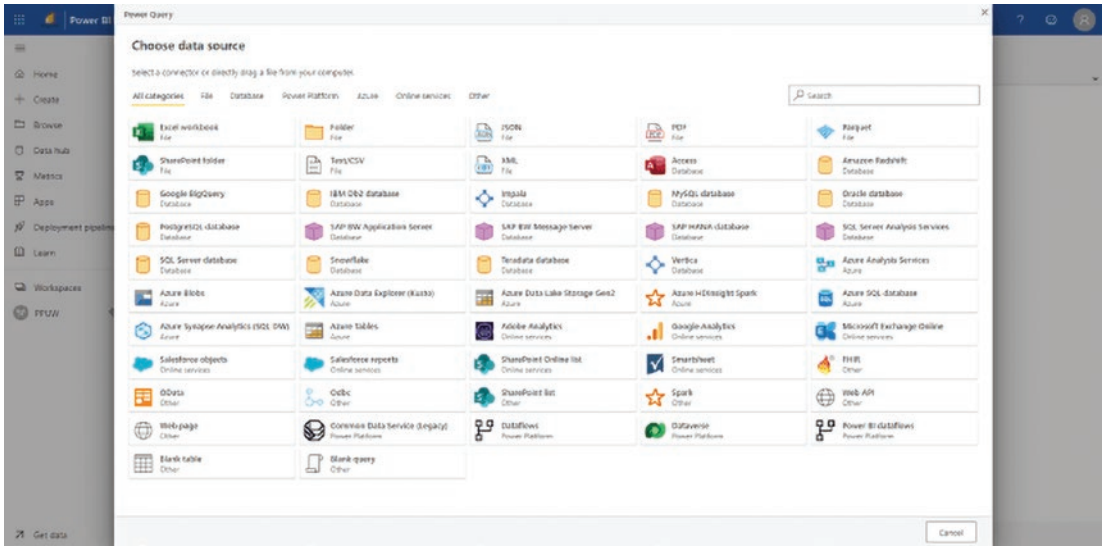


Figure 11-10. *Data source connectors for Power BI datamarts*

For this example, I use OData as a source and this URL: `services.odata.org/Northwind/Northwind.svc`. Enter it, as shown in Figure 11-11.

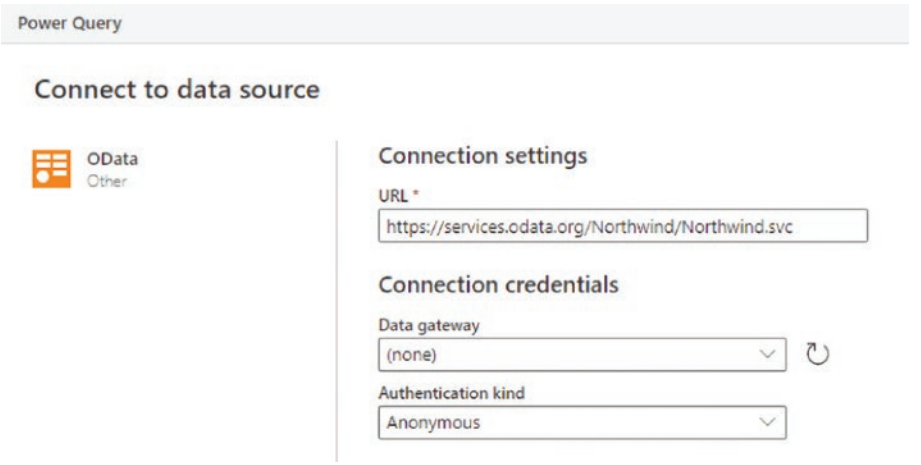


Figure 11-11. *OData source for a Power BI datamart*

Choose the data tables you need. I am using the Orders tables and some related tables; see Figure 11-12.

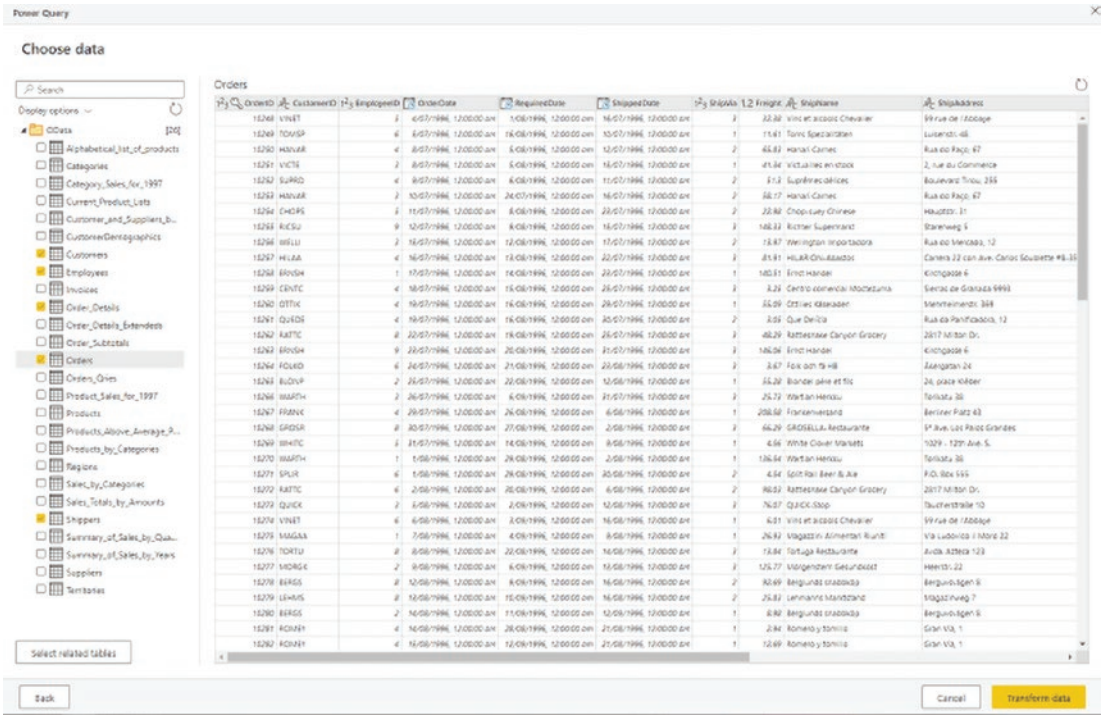


Figure 11-12. *Power Query Navigator window*

As you can see in Figure 11-13, the next step will reveal the Power Query Editor online version, which enables to do all the transformations you need.

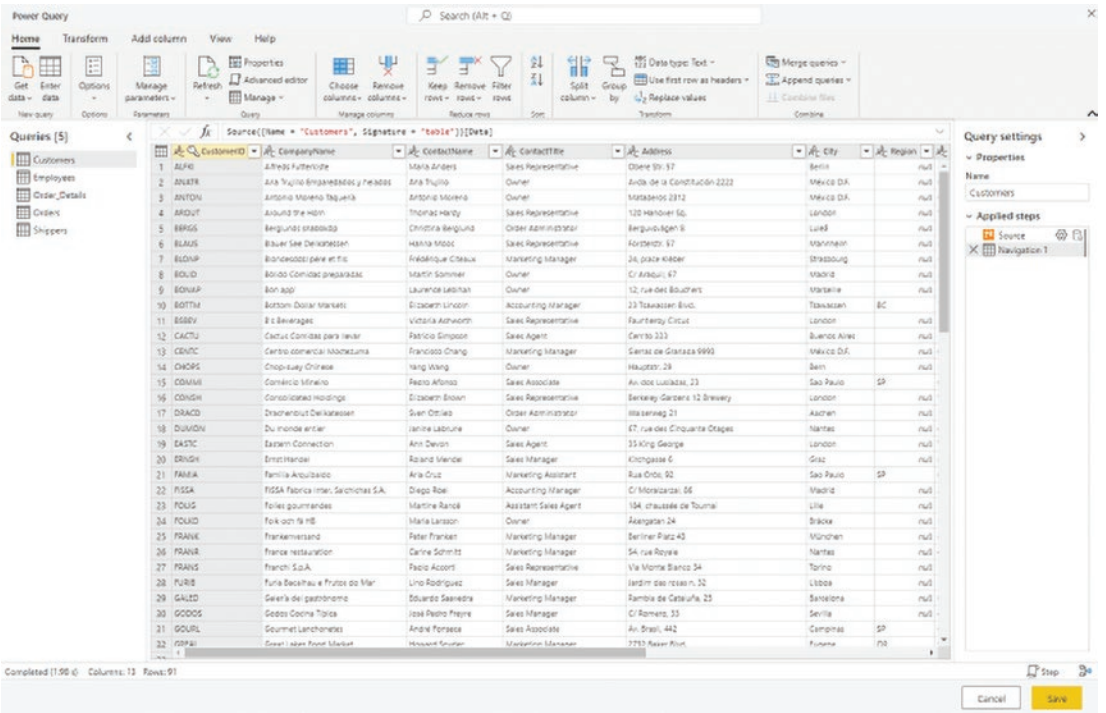


Figure 11-13. *Power Query Editor online version*

The data will then be loaded into the datamart, as shown in Figure 11-14.

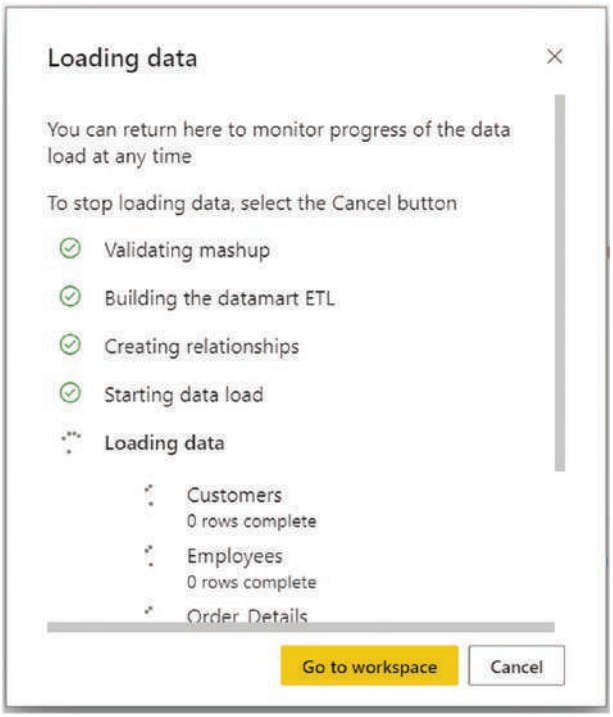


Figure 11-14. *Loading data into a Power BI datamart*

The Datamart Editor

Once the data is loaded into the datamart, you can see the Power BI Datamart Editor (see Figure 11-15). This editor will improve a lot in the future I believe. However, even right now, the editor has some mind-blowing features and capabilities. I explain these features as the chapter unfolds.

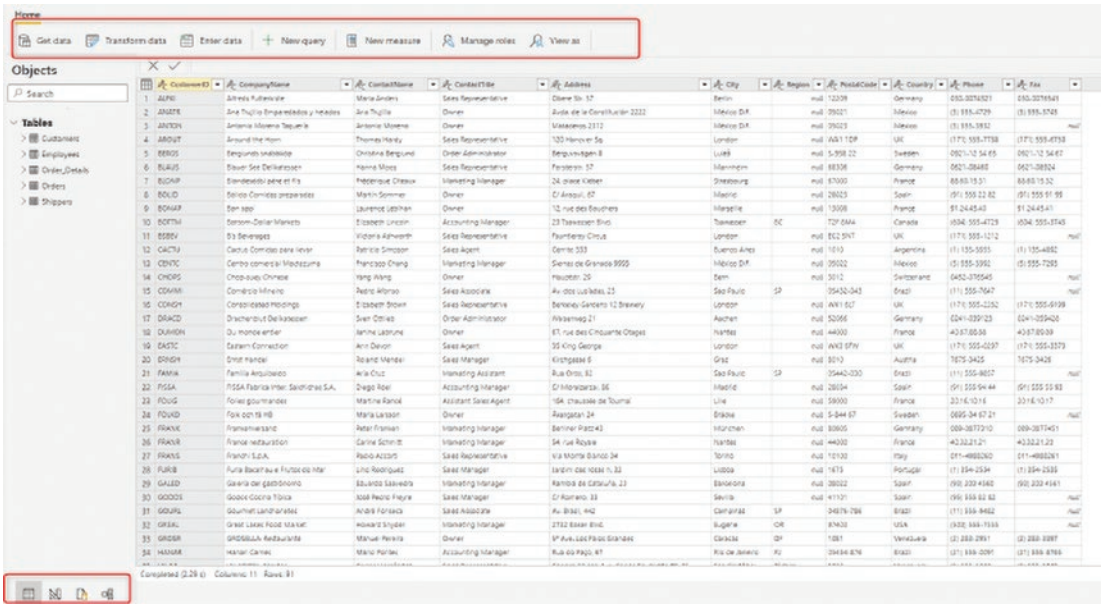


Figure 11-15. *The Power BI Datamart Editor*

The Data View Tab

The first tab is the Data View, which is like the Data tab in the Power BI Desktop. This is the place where you can see the data rows in each table. That is the view you see in Figure 11-15.

You can add data to this datamart using the Get Data option, or use Transform Data to get back to the Power Query Editor. You can also enter data directly as a table if you want.[1] These options are shown in Figure 11-16.

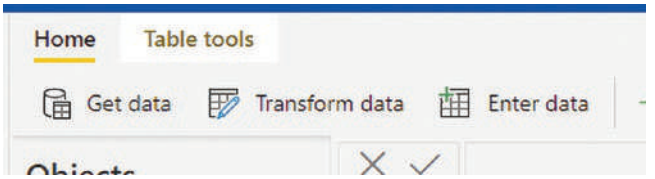


Figure 11-16. *The Get Data and Transform Data options for Power BI datamarts*

Creating a New Query

An interesting capability of datamarts is creating new queries. A new query is like a new table generated through the Power Query process. It might be a combined version of a table or a transformed version of one table (see Figure 11-17). You can of course do that already in Power BI Desktop using the Power Query Editor. However, the reason it is called New Query here is that, behind the scenes, it is created as a view in the Azure SQL database.

[1] Learn more at `radacad.com/create-a-table-in-power-bi-using-enter-data-and-how-to-edit-it`

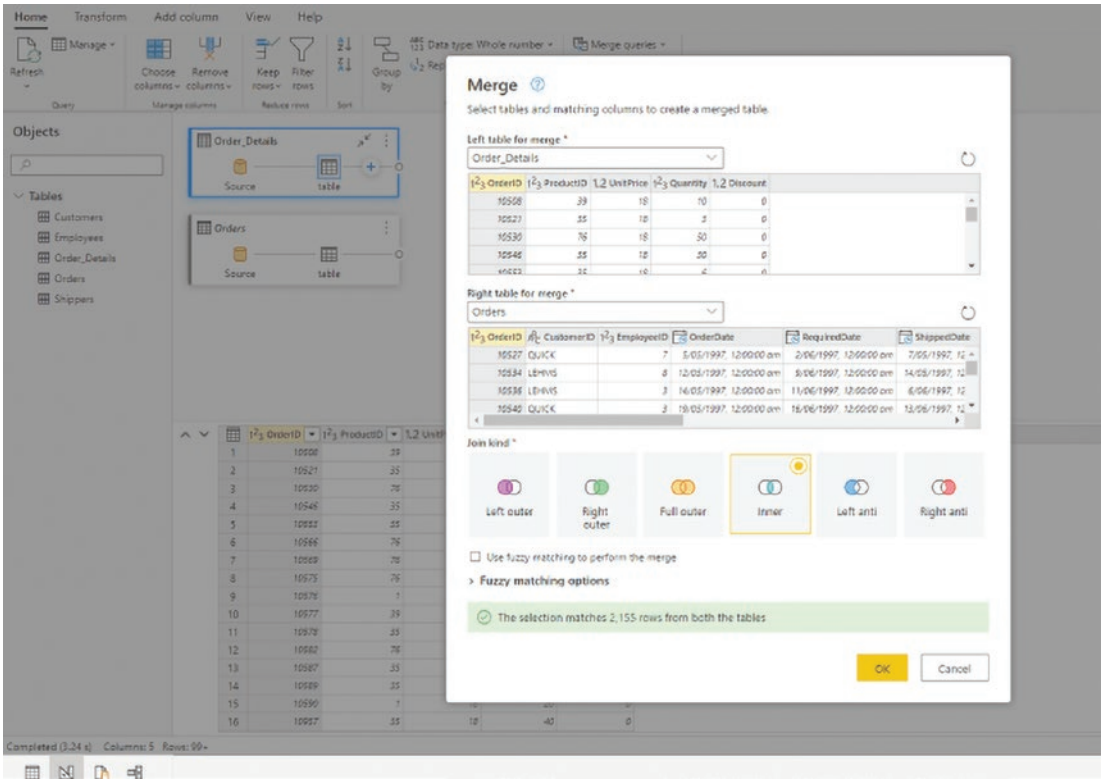


Figure 11-17. *Data transformation using Power Query Editor online to create a new query in a Power BI datamart*

Figure 11-18 shows what you see in the second tab.

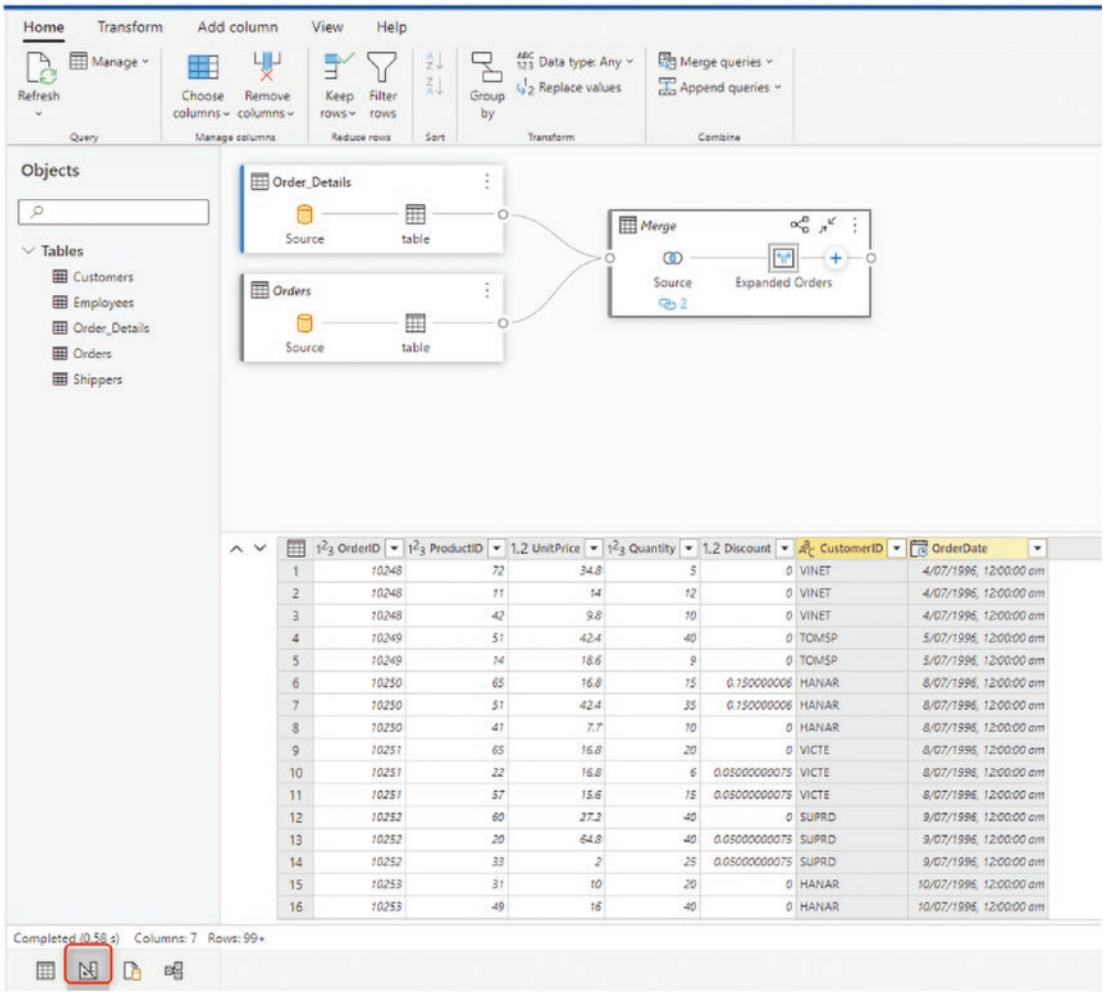


Figure 11-18. *Creating a new query in a Power BI datamart*

Writing T-SQL Queries

Power BI datamarts create an Azure SQL database behind the scenes. You can write queries in T-SQL statements in the third tab, shown in Figure 11-19.

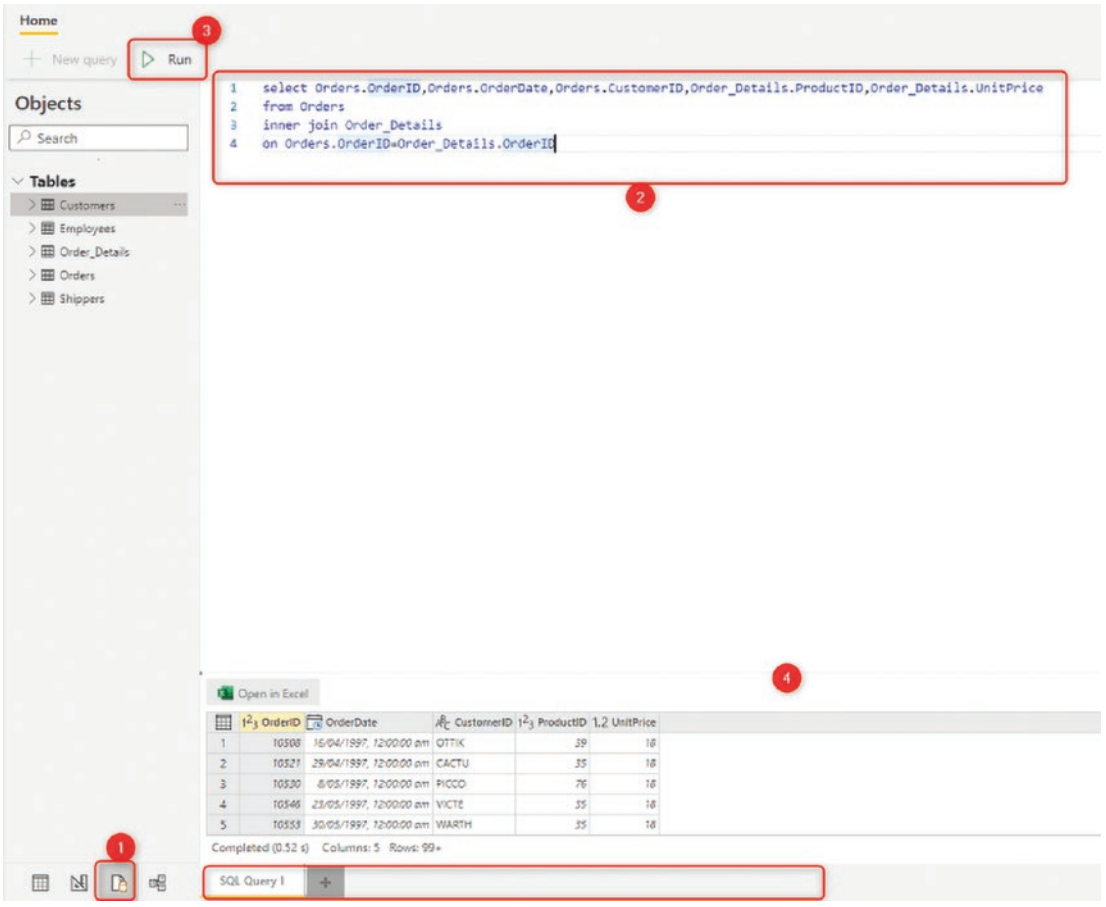


Figure 11-19. *Writing T-SQL queries on Power BI datamarts*

Power BI enables citizen data analysts and developers to use this component.

Model Diagram and Relationship Editor

The unified UI for datamarts enables you to write T-SQL queries, see the data, and transform data. It also enables you to define the relationship and build the diagram of the model, adding configurations for fields and tables all in one Web UI. Figure 11-20 shows the scope of what datamarts offer.

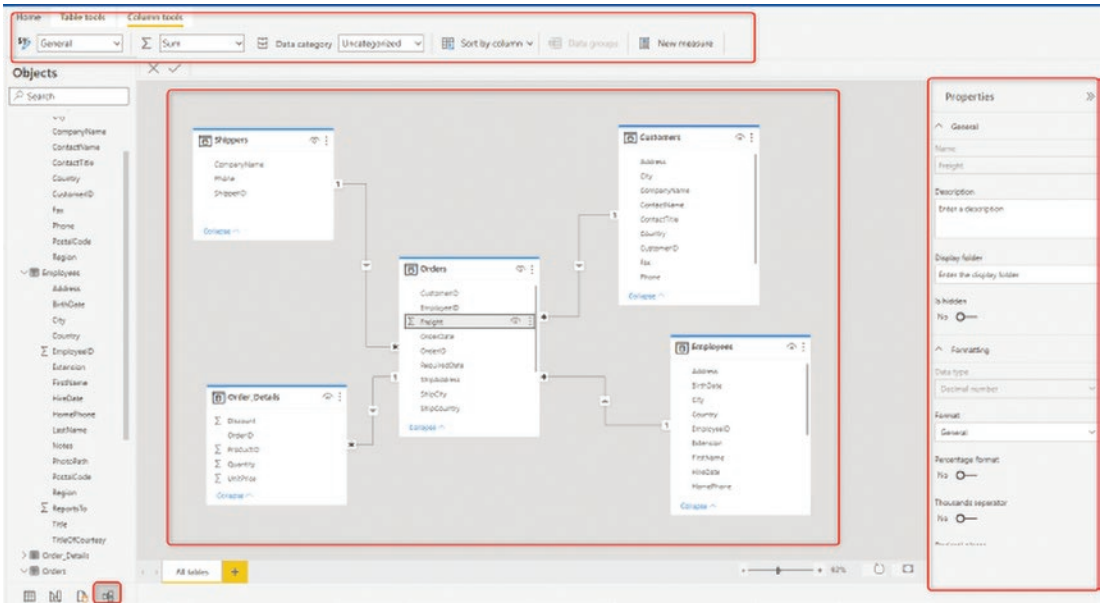


Figure 11-20. *The Model tab in the Power BI Datamart Editor*

The interesting fact here is that you create relationships only once, but the datamart behind the scenes will create it in both the Azure SQL database and in the subsequent Power BI dataset.

Creating Measures and Writing DAX Expressions

You can create measures using the Datamart Editor, as shown in Figure 11-21, and the Expression Editor will help you with the DAX function library and the IntelliSense to write whatever you want.

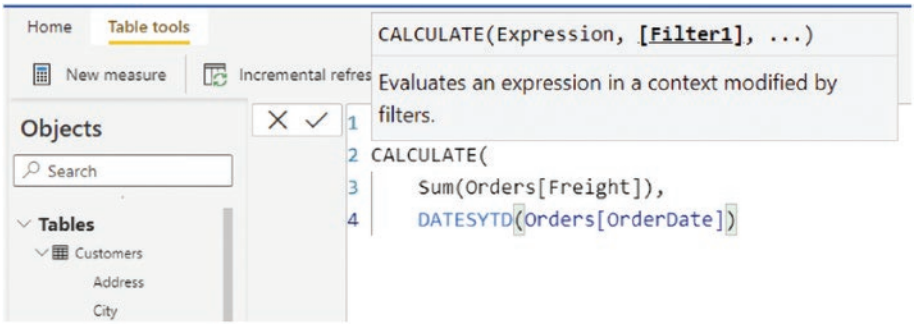


Figure 11-21. *Writing DAX measures in Power BI datamarts*

Incremental Refresh: One Setup for All

Incremental refresh enables you to set up a delta load rather than loading the entire data. When you use dataflows and datasets separately, you need to set incremental refreshes in them one by one. However, using the Datamart Editor, you can set the Incremental Refresh once, and it sets it everywhere for you. You can see what this process setup looks like in action in Figure 11-22.

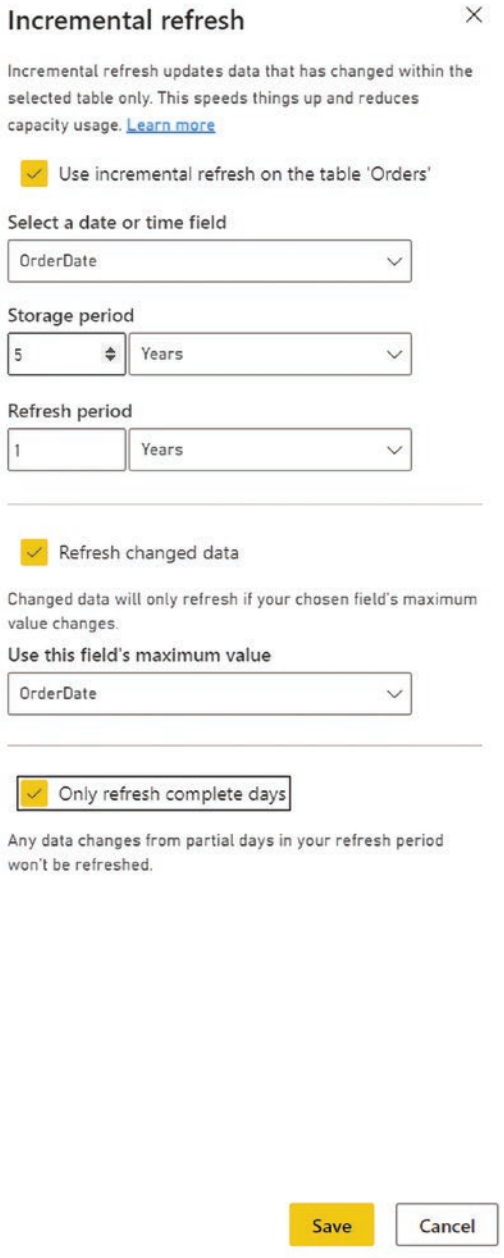


Figure 11-22. *Incremental refresh setup for Power BI datamarts*

Row-Level Security: In Databases and Datasets

Another interesting feature of datamarts is that you don't need to set up Row-level security in two different places (in the database and the dataset). As shown in Figure 11-23, you set it once in the datamart, and it will implement it in both places for you.

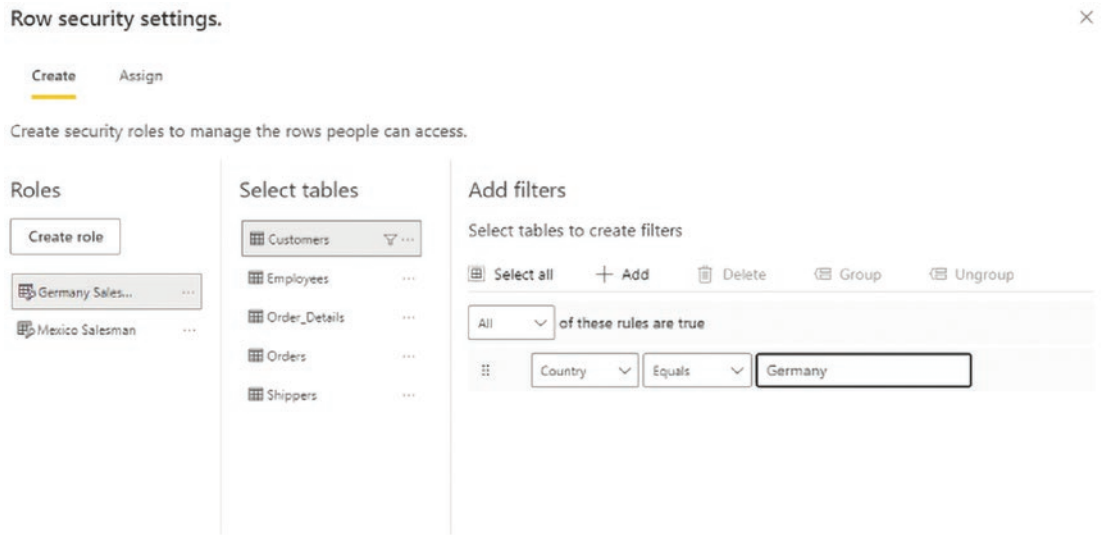


Figure 11-23. *Row-level security setup process in datamarts*

The process of assigning roles to users all happens in one place. Unlike using the Power BI Desktop for role creation and then the Power BI service for assignment. However, the difference is that you don't use DAX expressions for RLS. This is because these RLS settings are originally for the Azure SQL database and the associated dataset follows that.

You can test and view a specific role if you want, as shown in Figure 11-24.

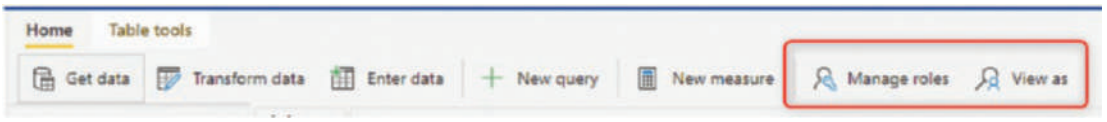


Figure 11-24. *Creating roles and testing them using the Power BI Datamart Editor*

You can rename the datamart by clicking the current title and changing it, as shown in Figure 11-25.

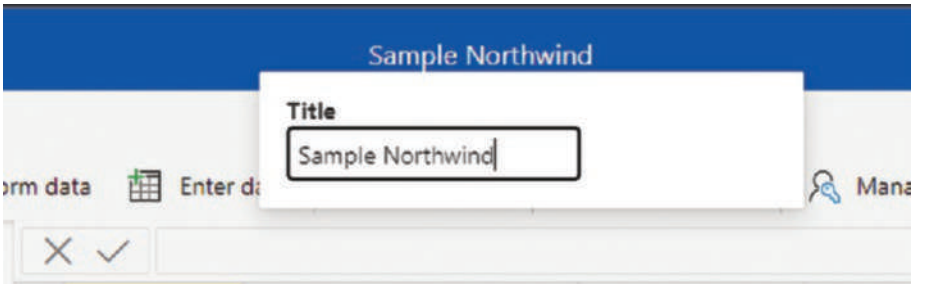


Figure 11-25. *Renaming a datamart*

Congratulations, you've built your first Power BI datamart. It will show up as shown in Figure 11-26.

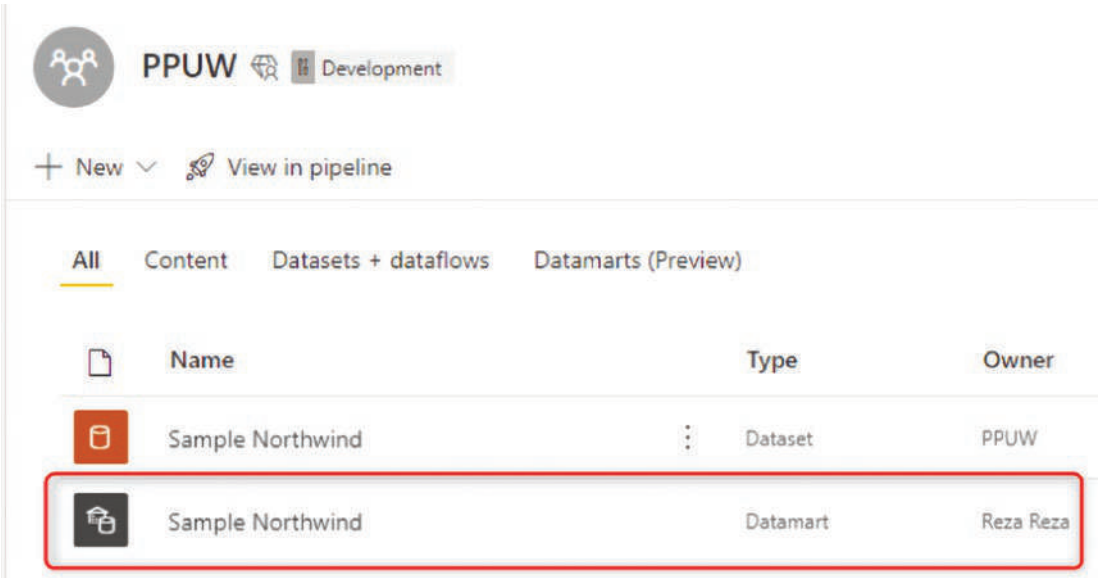


Figure 11-26. *The Power BI datamart in the service*

Power BI Datamart Components

What is a Power BI datamart underneath? Can you connect to the database generated by the datamart? How can the dataset that's associated with the datamart be used? Is there a linage view? This section of this chapter answers these questions and you will learn about datamart's components.

Power BI Datamarts Under the Hood

When you create a datamart, three objects are created behind the scenes: a dataflow, an Azure SQL database, and a dataset (see Figure 11-27). Some of these components are exposed as individual components, and some are hidden.

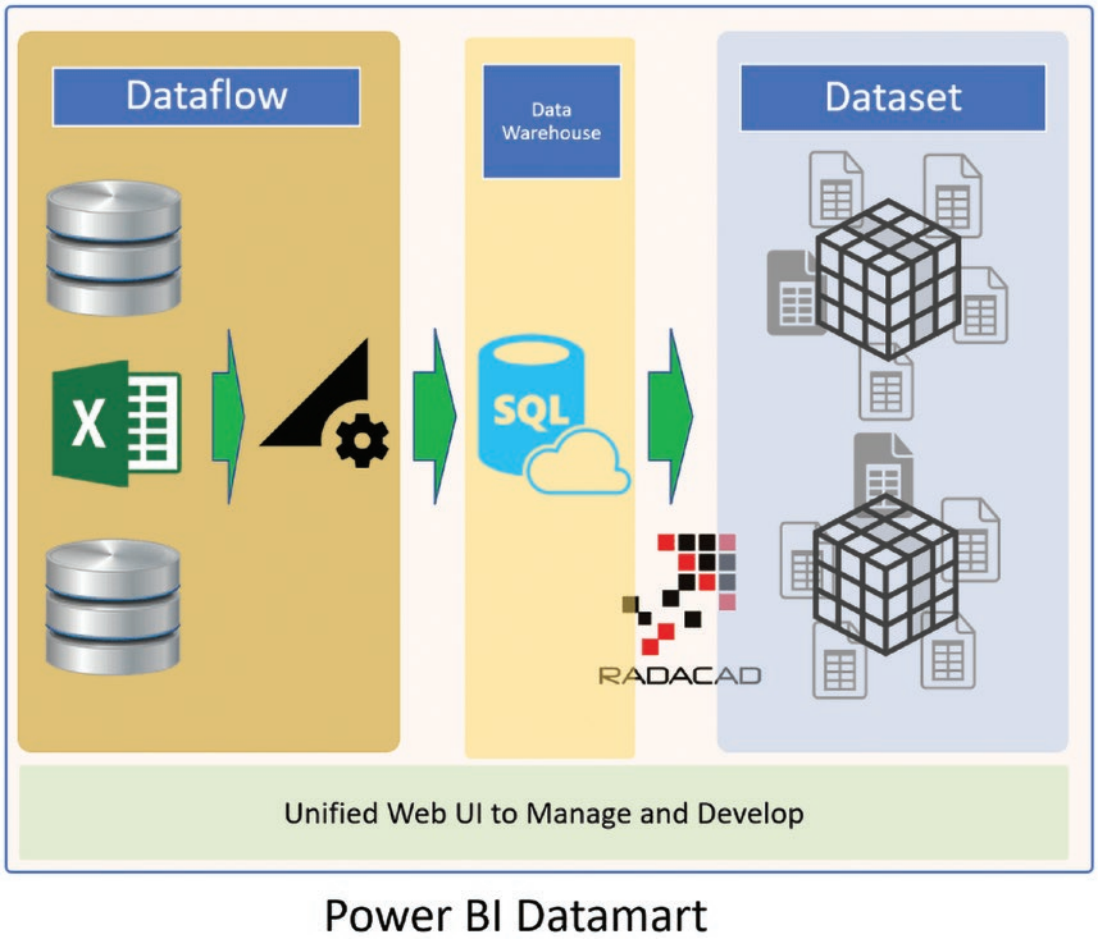


Figure 11-27. *Power BI datamart's components*

The Power BI Dataset

When you create a datamart, the most obvious component of it can be seen separately, and that is the Power BI dataset, as shown in Figure 11-28.

Figure 11-28. *The dataset associated with the datamart*

You can also connect to the dataset using the Power BI Desktop (like a normal dataset), or using the XMLA endpoint, which can be found under the Datasets setting, as shown in Figure 11-29.

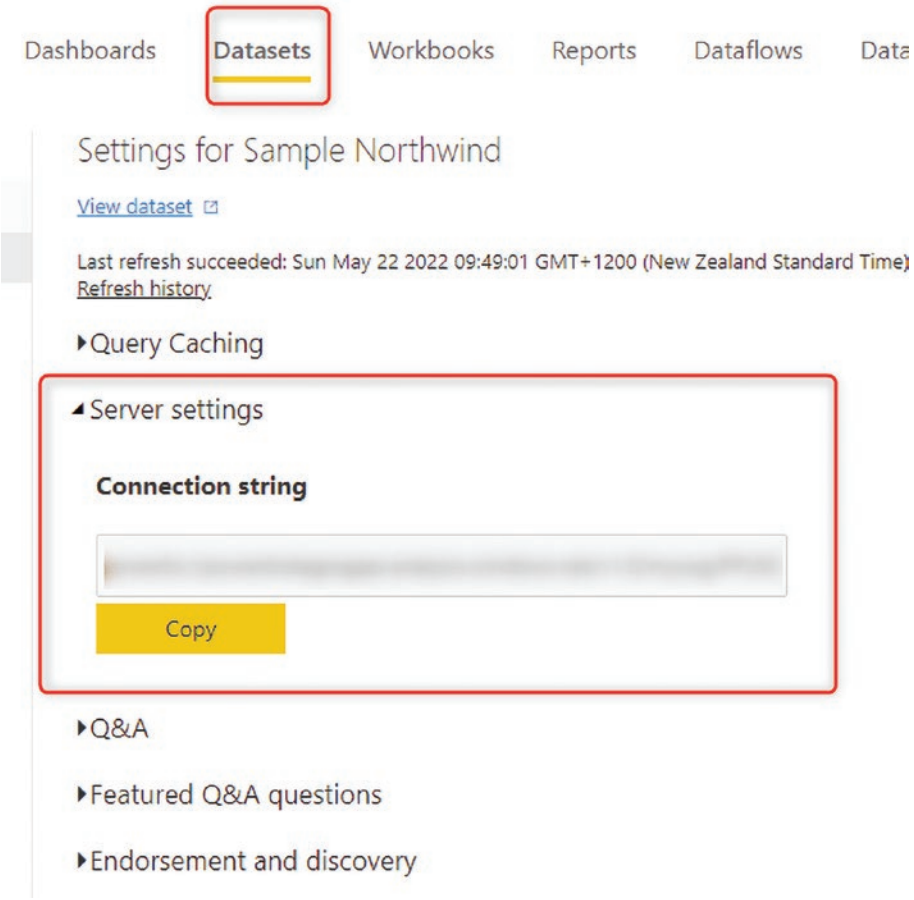


Figure 11-29. *Power BI dataset settings*

This connection string then can be used with SSMS, Visual Studio, Tabular Editor, Power BI Helper, and many other tools to connect to the Power BI dataset.

The dataflow and the Azure SQL database are not that easy to find inside the Power BI service (apart from the Datamart Editor). But there are ways to connect to them.

Connecting to the Azure SQL Database

To find the connection to the Azure SQL database, click the more options button on the Datamart and select Settings, as shown in Figure 11-30.

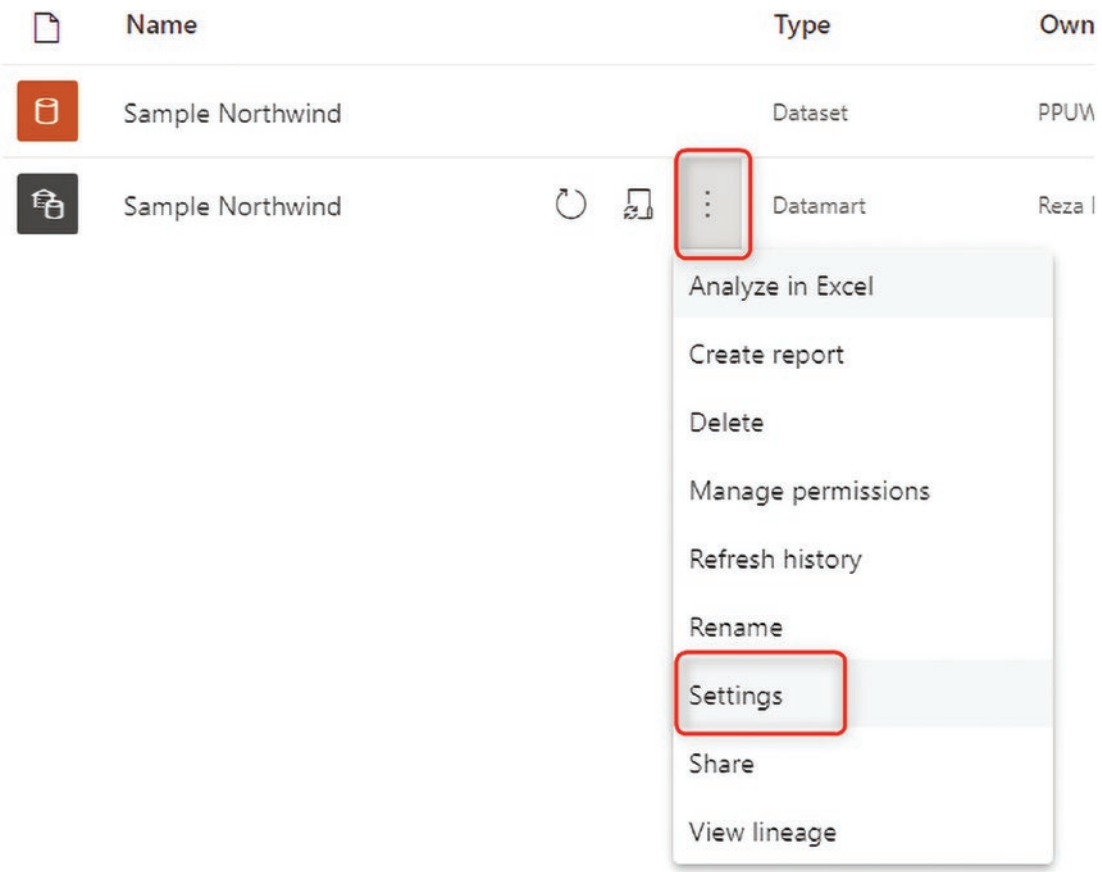


Figure 11-30. *Power BI datamart settings*

Then under Server Settings, you'll find the Connection string, as shown in Figure 11-31.

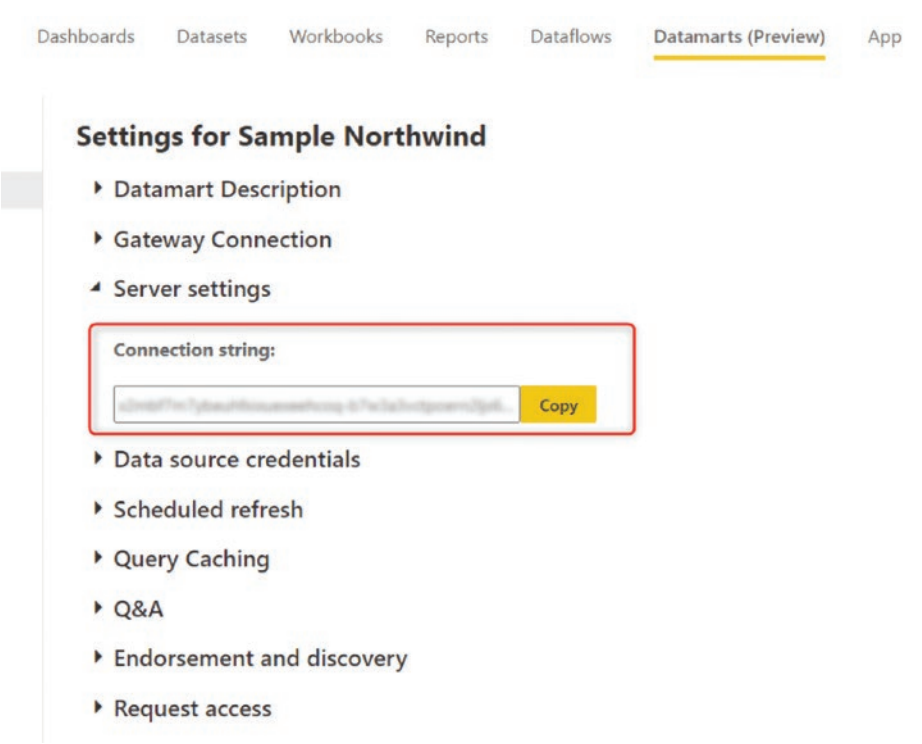


Figure 11-31. *Finding the Azure SQL database connection from the Power BI datamart*

This connection can be used with other tools, such as SSMS (SQL Server Management Studio). Note that you should be using Azure Active Directory–Universal with MFA for authentication and your Power BI account as the username, as shown in Figure 11-32.

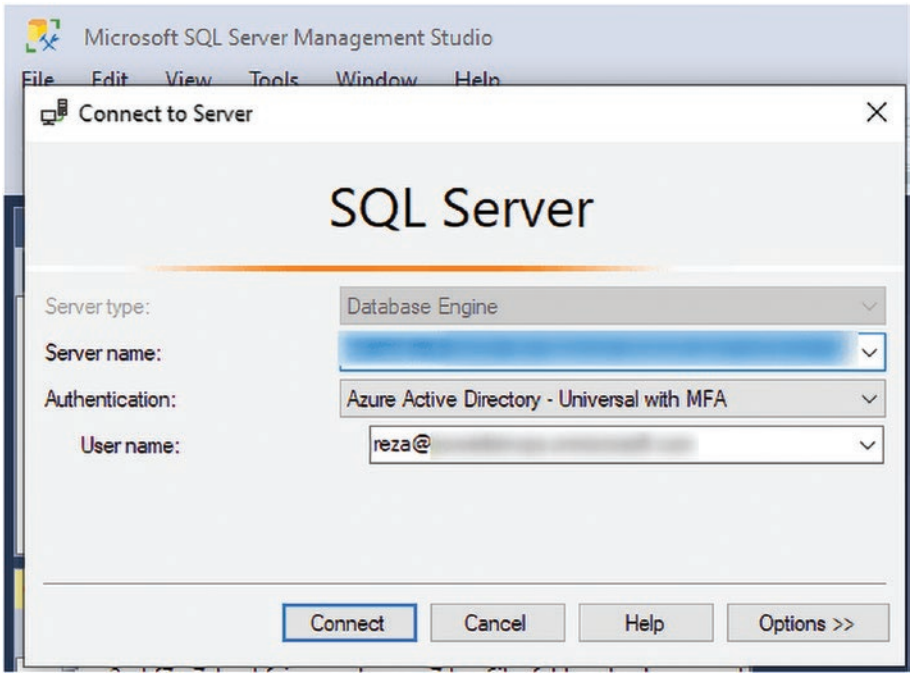


Figure 11-32. *Connecting to the Azure SQL database of a Power BI datamart*

You can then see the database with the objects in it. The tables are hidden at first as an extra safety measure. However, there is a view per table that you can see, plus other metadata tables such as relationships. See Figure 11-33.

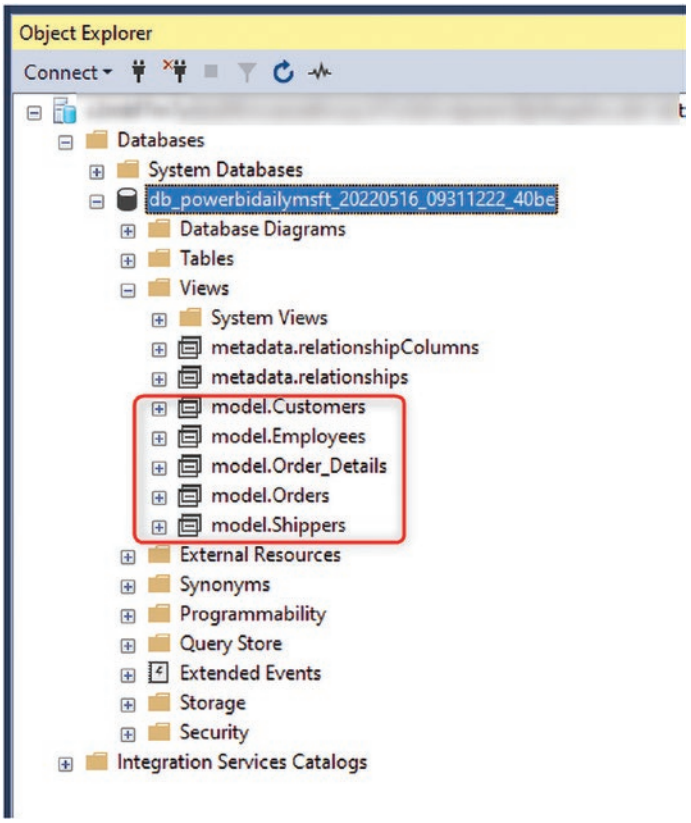


Figure 11-33. *The datamart's database structure*

Similar to any Azure SQL database, you can view the script of the views, such as the `model.Customers`. You can see in Figure 11-34 that even in the script, the row-level security implementation can be visible.

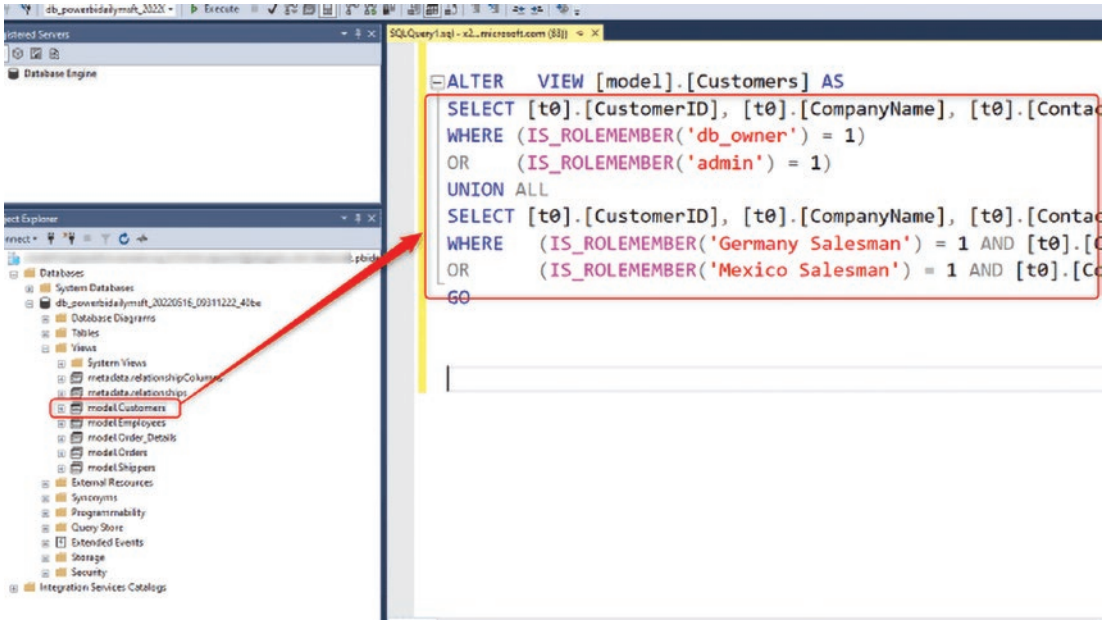


Figure 11-34. *Views in an Azure SQL database of the datamart*

There are limitations that ensure that you don't change anything in this structure that would cause the datamart to fail.

Building Reports from Datamarts

There are multiple ways to create reports from a datamart. You can go to the Data Hub and click the datamart, as shown in Figure 11-35.

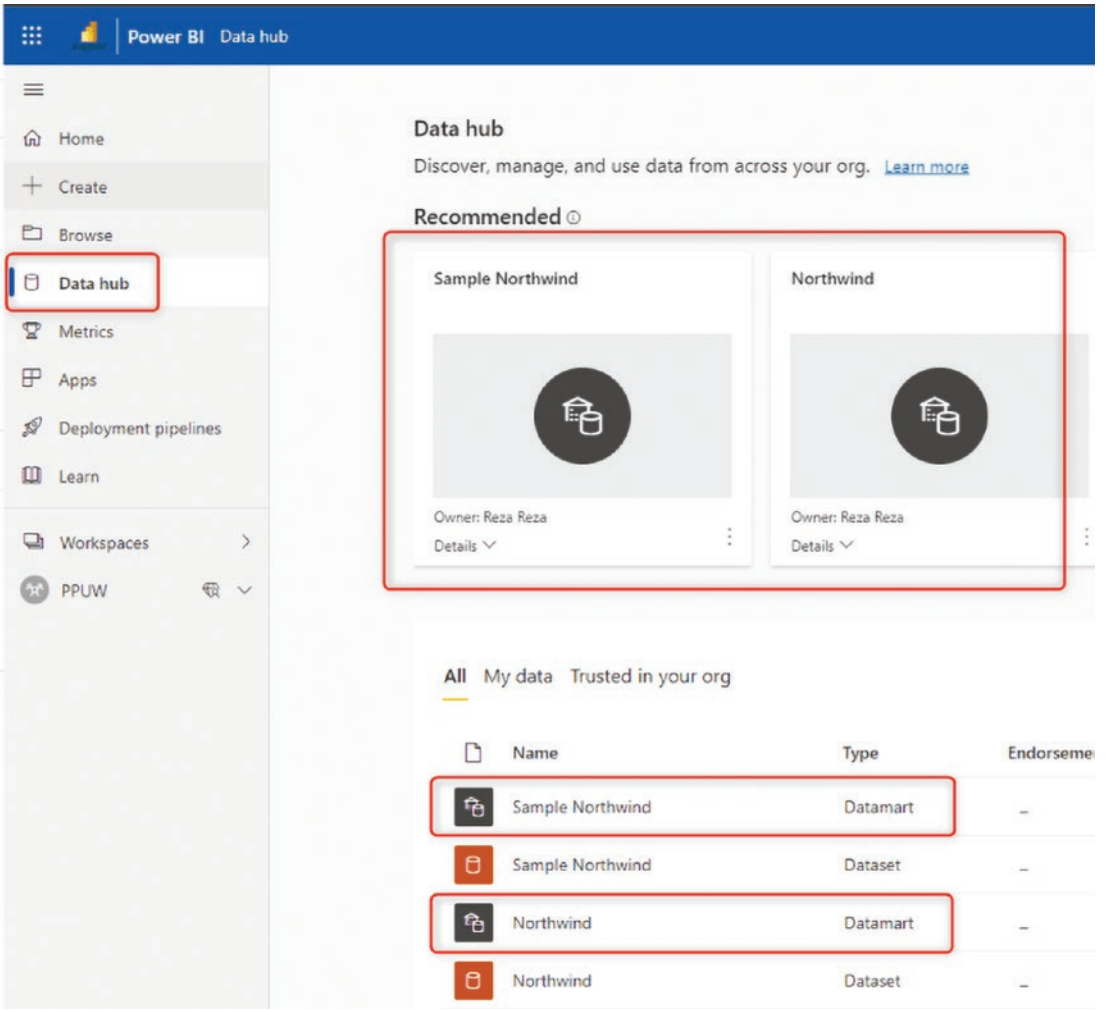


Figure 11-35. *Datamarts in the Power BI Data Hub*

In the datamart's details, you can see the SQL connection string and create a report from scratch, which will be connected to this datamart, as shown in Figure 11-36.

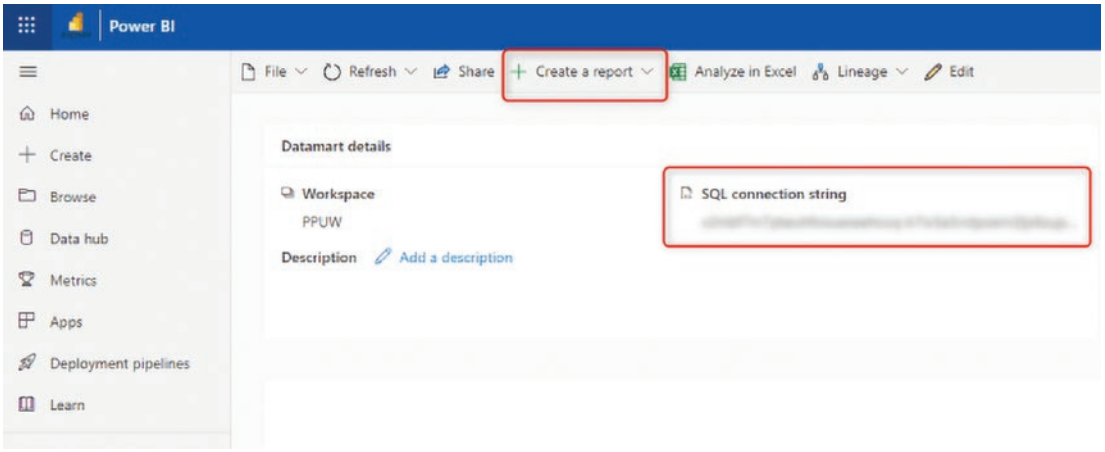


Figure 11-36. *Creating a report from a Power BI datamart*

The report will automatically connect to the tables in the dataset, as shown in Figure 11-37.

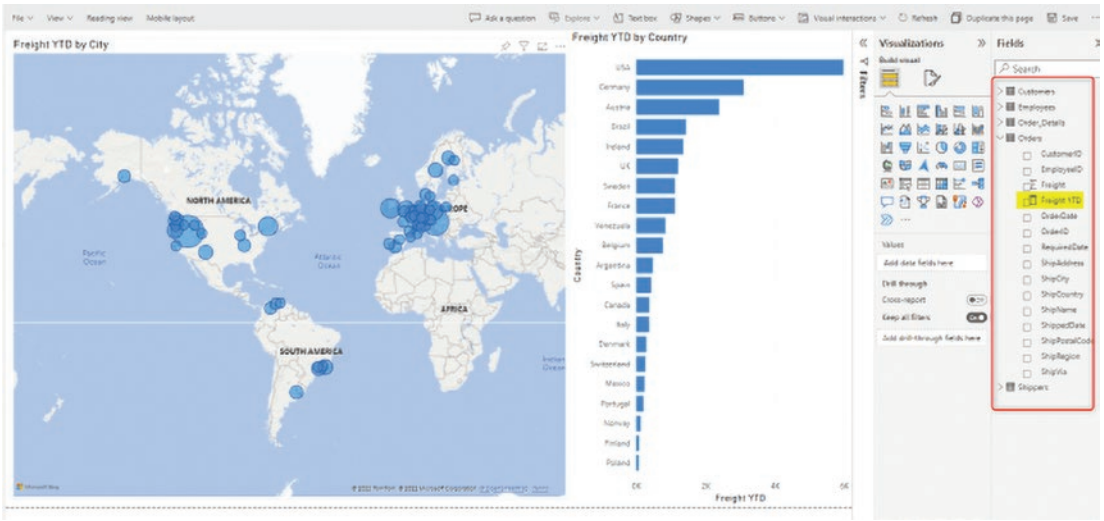


Figure 11-37. *Creating a Power BI report in the Power BI Service from a datamart*

The report can be also created from the dataset associated with the datamart, or from the Power BI Desktop by choosing Get Data from Power BI dataset, as you see in Figure 11-38.

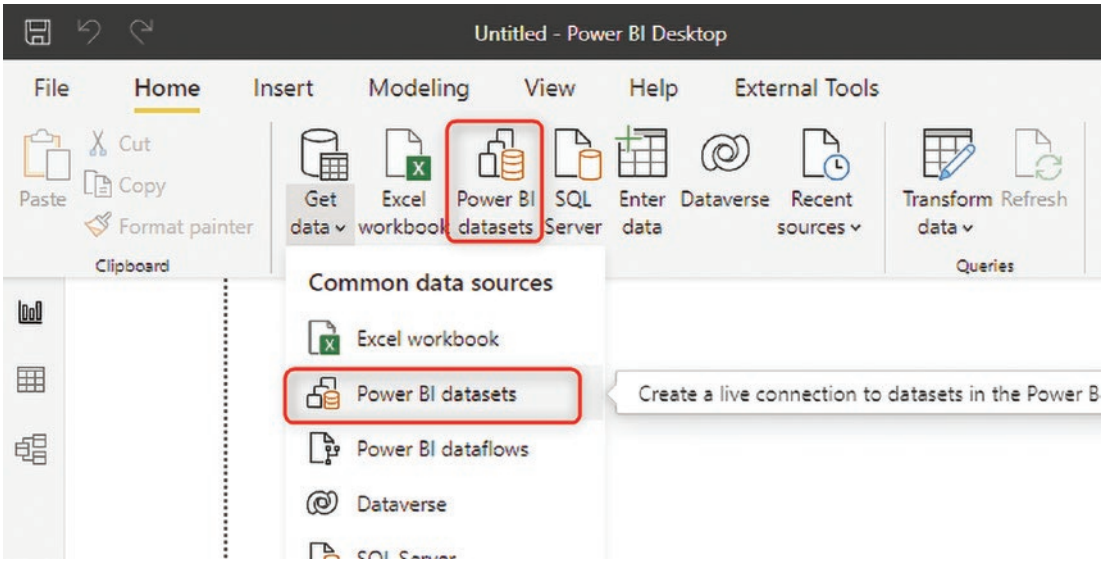


Figure 11-38. *Getting data from a Power BI dataset*

Lineage View

If you are on the datamart, on the dataset associated with it, or on any of the reports connected to that dataset, you can see the Lineage view, as shown in Figure 11-39.

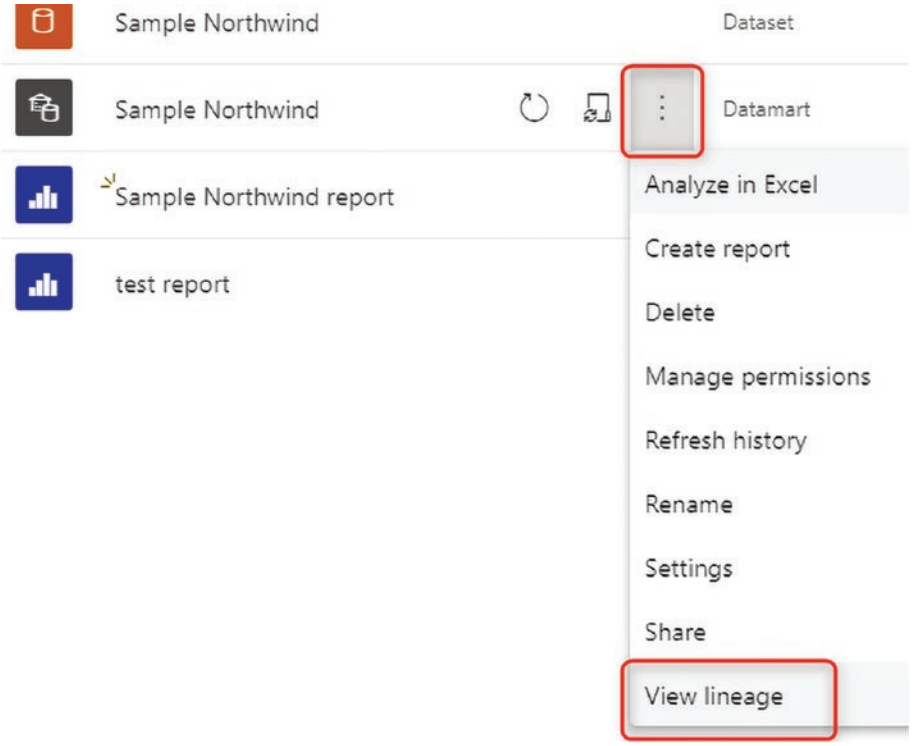


Figure 11-39. *Lineage view of a Power BI datamart*

In Figure 11-40, you can see that the datamart is populating data from an OData source and feeding it into a dataset. There are three reports generated from that dataset. The dataflow part of the datamart and the Azure SQL database is hidden from this diagram (they are parts of the datamart).

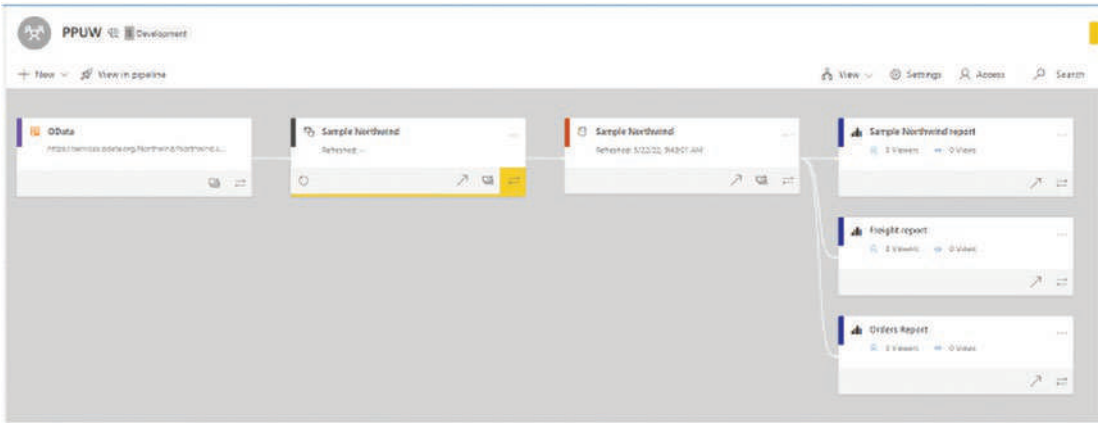


Figure 11-40. *Power BI Lineage view*

Also in the Data Hub, when you click the datamart, you can see all the reports generated from it (see Figure 11-41).

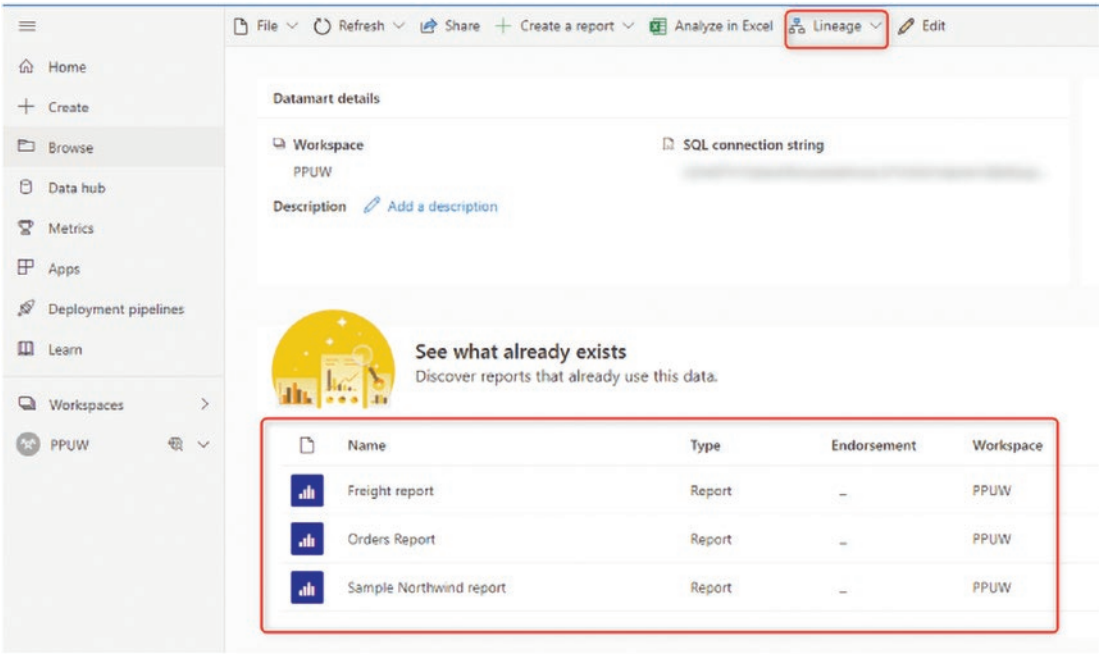


Figure 11-41. *The Datamart details in the Power BI Data Hub*

Summary

This chapter was about Power BI datamarts. It explained what a datamart is, what features it includes, and who should use it. I strongly believe that datamarts revolutionize the way we develop Power BI solutions. Datamarts are the future of building Power BI solutions in a better way.

In the second part of the chapter, you learned how to create a Power BI datamart and about the features inside the Datamart Editor. As you saw from an example, datamarts provide a single unified editor experience, from getting data and transforming it, to writing queries, creating relationships, and event managing things such as row-level security.

Finally, you've learned what is under the hood of Power BI datamarts: a dataflow, an Azure SQL database, and a dataset. You can connect to some of these components using other tools. In this chapter, you learned some methods for doing that. You can also build a report that connects to the dataset created by the datamart, and see the entire Lineage view.

CHAPTER 12

The Multi-Layer Architecture

Using Power BI is simple. However, using it properly requires a good architecture. In a multi-developer environment, the architecture of a Power BI implementation should allow multiple developers to work on the same solution at the same time. On the other hand, the Power BI architecture should be designed so that different layers can be decoupled for better integration. This chapter explains how dataflows, datamarts, and shared datasets can help build such an architecture in Power BI.

Challenges of Using a Single PBIX File for Everything

Before I start explaining the architecture, it is important to understand the challenge and think about how to solve it. The default usage of Power BI involves importing data into the Power BI data model and then visualizing it. Although there are other modes and other connection types, Import Data is the most popular option. However, there are challenges when using a PBIX file with everything in one file, as illustrated in Figure 12-1. Here are some:

- Multiple developers cannot work on the same PBIX file at the same time. (Multi-developer issue.)
- Integrating the single PBIX file with another application or dataset would be very hard. (Ongoing maintenance issue.)
- All data transformations happen inside the model, and the refresh time is slower. (Performance issue.)
- The only way to expand visualization is by adding pages to the model, and you would end up with hundreds of pages. (Complexity issue.)
- Every change, even a small change in the visualization, means deploying the entire model. (Deployment issue.)
- Creating a separate Power BI file with some parts referencing the model would not be possible; as a result, you would create a lot of duplication. (Ongoing maintenance issues.)
- If you wanted to reuse some of the tables and calculations in other files in the future, it won't be easy to maintain when everything is in one file. (Lack of flexibility issue.)
- And many others.

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_12

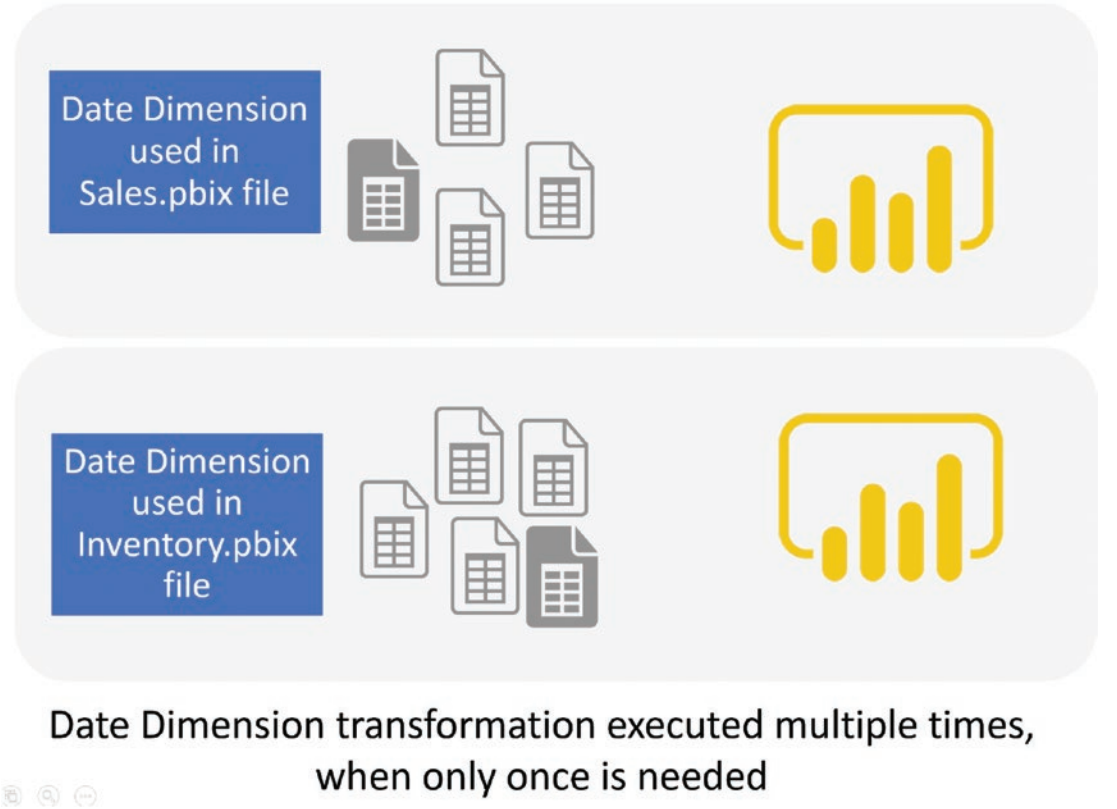


Figure 12-1. *An example of one table needed in multiple Power BI files*

Suppose you are the only Power BI developer in your organization and are working on a small model that won't grow into multiple Power BI files in the future. In that case, you can keep everything in one file. However, I do not recommend keeping everything in one file if the scenario is different.

Using Dataflows to Decouple the Data Preparation Layer

Power Query is used for data preparation in the world of Power BI. Based on my experience and what I have seen from other implementations, you will spend 70 percent of your time on data preparation for a proper data model. That means if you are doing a Power BI implementation for ten months, then seven months will be spent on data preparation with Power Query! You don't want that time to be wasted and duplicated for another work. You want to maintain the efforts you have made for your data preparation.

Instead of doing your data preparation as part of a Power BI file, which will link that work only to that file, you can do it through a Power BI dataflow, as shown in Figure 12-2. Power BI dataflows run Power Query independently from Power BI files. The independence of dataflows from a Power BI file means that you can do data preparation for one entity only once and use it multiple times! In other words, you can use the dataflow to decouple the Power BI system's data preparation (or ETL) layer from the Power BI file. After doing the data preparation using the dataflow, the Power BI files can get data from it using the Get Data option from the dataflow.

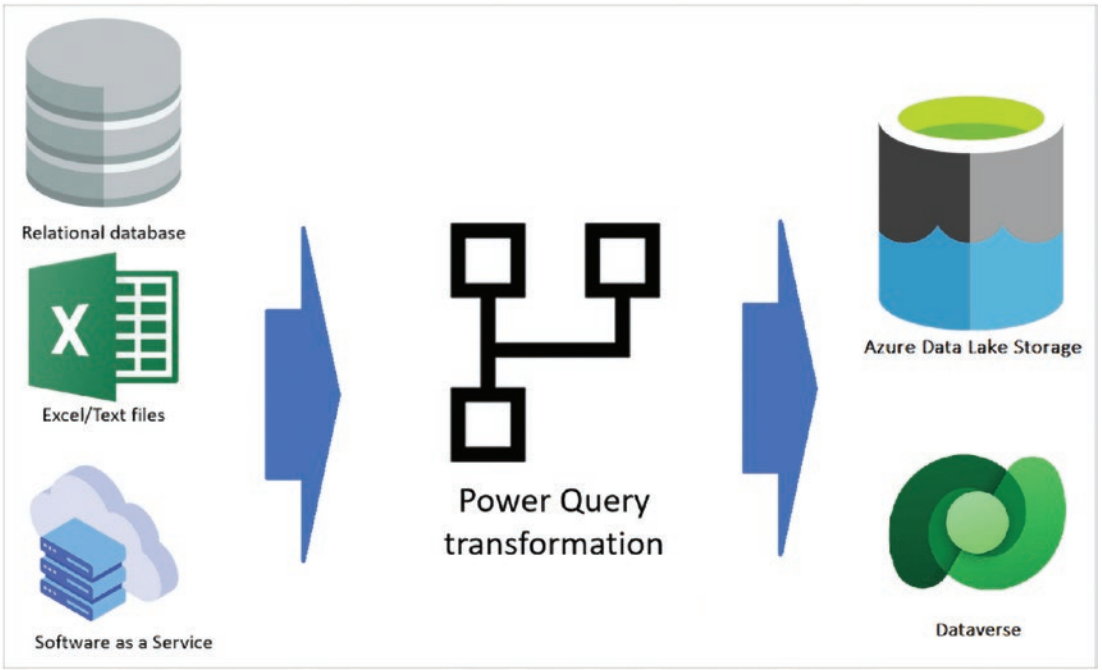


Figure 12-2. *Power BI dataflows*

Figure 12-3 features a diagram of using dataflows with multiple Power BI files.

Figure 12-3. *Shared tables can be processed in a dataflow and then reused in multiple Power BI files*

Shared Datasets for the Data Modeling Layer

Although dataflows create shared data sources for Power BI models, if you want to create multiple Power BI models, you need to set field-level formatting inside the Power BI file. You need to create DAX calculations inside each file and create hierarchies and any other modeling requirements inside the Power BI model. These settings cannot be done in the dataflow. These are parts of the Power BI dataset.

Instead of doing all the modeling and calculations in each Power BI file separately, you can leverage the shared dataset concept, shown in Figure 12-4. The shared dataset concept enables you to create a Power BI file with a model only and no visualization pages.

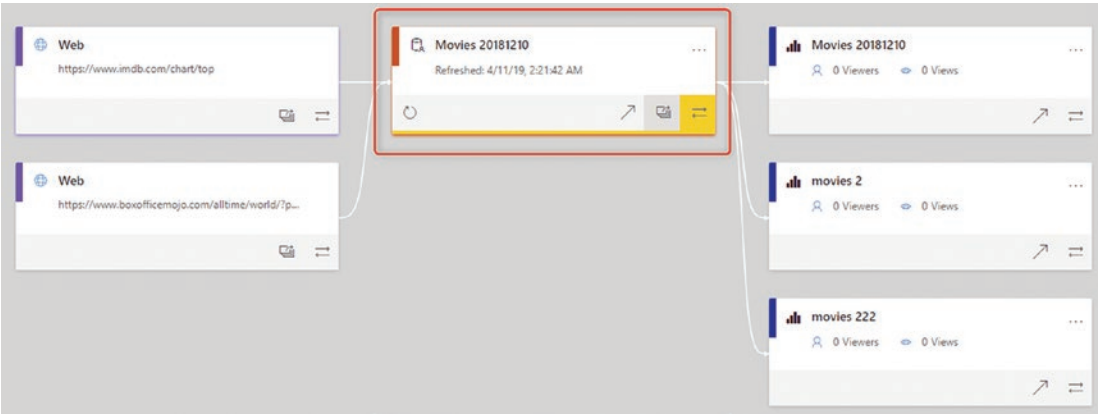


Figure 12-4. *A Power BI shared dataset*

A shared dataset then can be used in multiple Power BI reports as a centralized model. You don't need to duplicate your DAX calculations, hierarchies, and field-level formatting using the shared dataset. The shared dataset will act like a modeling layer of your Power BI solution.

Thin Power BI Reports, Paginated Reports, and the Analyze in Excel Option

Now that you have the Power BI data model in a shared dataset, you can create Power BI reports that get data from that shared dataset. This process is shown in Figure 12-5. These reports will create a live connection to that dataset. These are called thin reports.

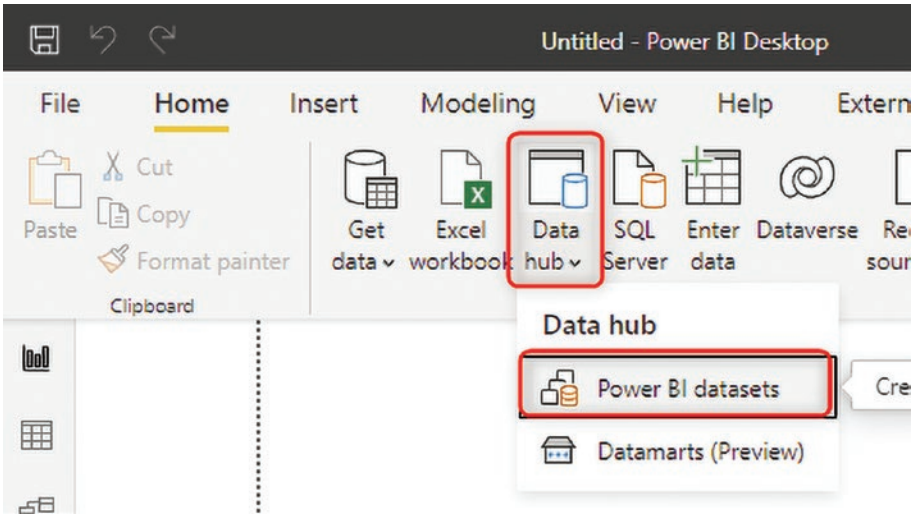


Figure 12-5. *Creating thin reports with a live connection to Power BI shared datasets*

You can create multiple thin reports by getting data from the shared dataset. This enables multiple report visualizers to build visualizations at the same time.

Not only can you create Power BI reports on top of the shared dataset, but you can also create paginated reports. As you can see in Figure 12-6, you connect to the same Power BI dataset in Excel.

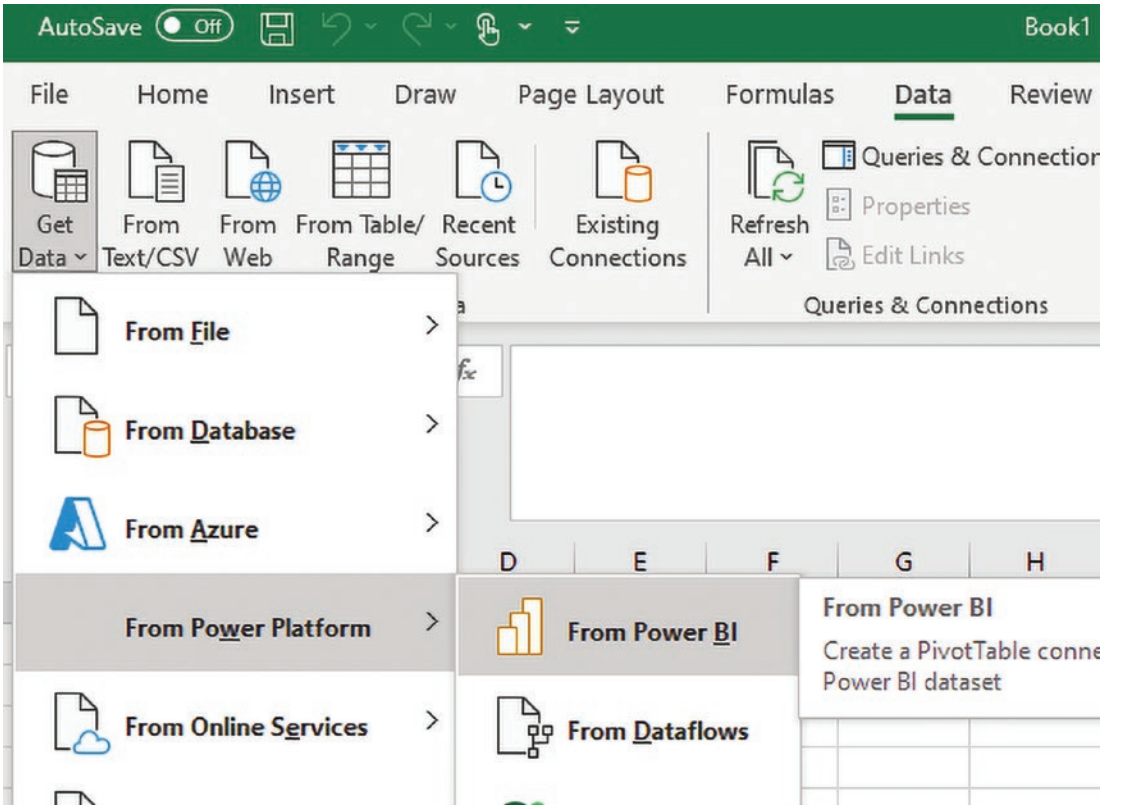


Figure 12-6. *Connecting to the Power BI shared dataset from Excel*

Power BI Architecture with Dataflows, Shared Datasets, and Thin Reports

As a result, I recommend the multi-layered Power BI architecture shown in Figure 12-7. This architecture doesn't include datamarts (which are explained later in this chapter) because the licensing for datamarts might require a different setup.

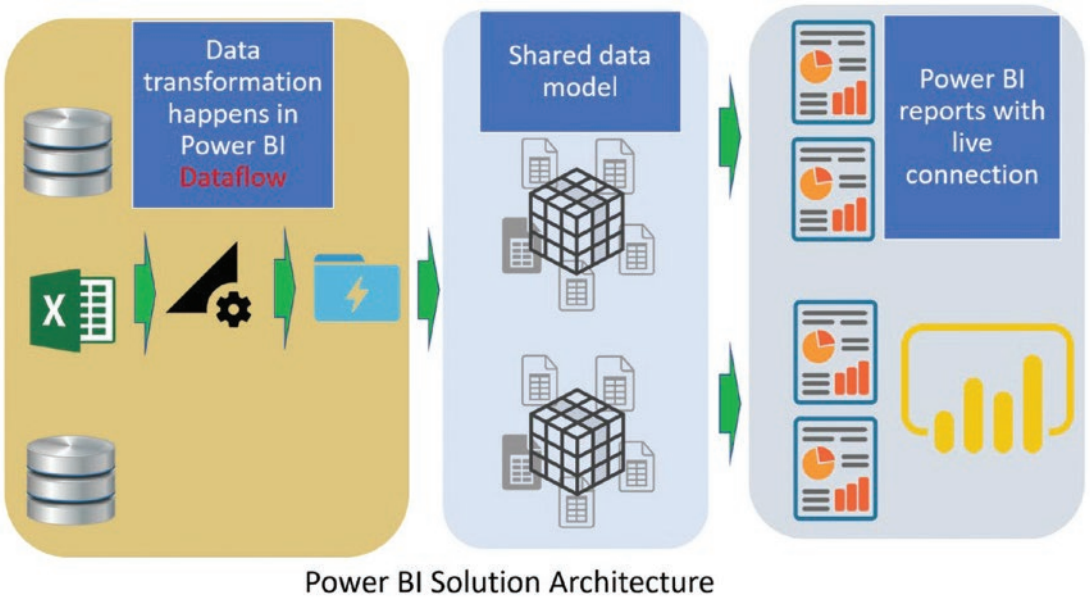


Figure 12-7. *The Power BI Solution Architecture with dataflows and a shared dataset*

This architecture uses layers as follows:

- Dataflow for the ETL or data preparation layer
- Shared dataset for the data modeling layer
- Get Data from Power BI dataset for the visualization layer (thin reports), paginated reports, or the Analyze option in Excel

Benefits of a Multi-Layered Architecture

Every architecture plan has its pros and cons. The following sections covers some of the benefits of this architecture.

Decoupling the Data Preparation Layer Using Dataflows

The development time you put using Power Query will be preserved and can be used in other Power BI models. The dataflow stores the data in an Azure Data Lake. Decoupling the data preparation from the modeling means that you can apply changes to the data transformation with minimum effect on the other layers (modeling).[1]

[1] Moving shared data tables to a Power Query dataflow is a recommended approach I have previously written about. Learn more at `radacad.com/move-your-shared-tables-to-dataflow-build-a-consistent-table-in-power-bi`.

A Multi-Layered Architecture Without the Need for Other Developer Tools

You can use SSIS, Azure Data Factory, or other ETL tools to take care of the data preparation process. However, the power of Power BI resides in its simplicity. Everyone with fair knowledge of Power Query can use dataflow. You don't need to be a C# or SQL Server developer to work with this tool. When you build a shared dataset, you are still using Power BI. All the skills you need are in the Power BI skillset.

Multi-Developer Environments

The proposed architecture supports multiple developers simultaneously on one Power BI solution. You can have multiple ETL developers (or data engineers) working on dataflows, data modelers working on the shared dataset, and multiple report designers (or data visualizers) building reports. They don't need to wait for each other to finish their work and then continue. They can work independently.

In this environment, everyone can do the job they are skilled to do. The dataflow developer requires Power Query and M skills but not DAX. The data modelers need to understand the relationships and DAX, and the report visualizer needs to understand the art of visualization and tips and tricks on how to build informative dashboards.

Reusing Calculations and Modeling with Shared Datasets

Having a centralized model will reduce the need for writing DAX calculations repeatedly. You do the modeling once and reuse it in multiple reports. On the other hand, you are sure that all reports are getting their data from a reconciled and fully tested model, or in other words, it is a gold data model.

Minimum Redundancy, Maximum Consistency, and Low Maintenance

A solution implementation that reduces the need for code duplication (using dataflows and a shared dataset) and reusing the existing code (using Get Data from the dataflows and datasets) will be highly consistent. Maintenance of such an implementation is possible with minimal effort.

Enhancing the Architecture Using Power BI Datamarts

Datamarts take the architecture of Power BI implementation to the next level by bringing one unified UI to build the dataflow, the data warehouse using an Azure SQL database, and the shared dataset. Because datamarts require premium (PPU or Premium capacity) licensing, I do not explain them in the original version of the architecture.

If you have the required licensing, I recommend using datamarts, which will introduce the data warehouse as the fourth layer of the Power BI architecture, as shown in Figure 12-8.

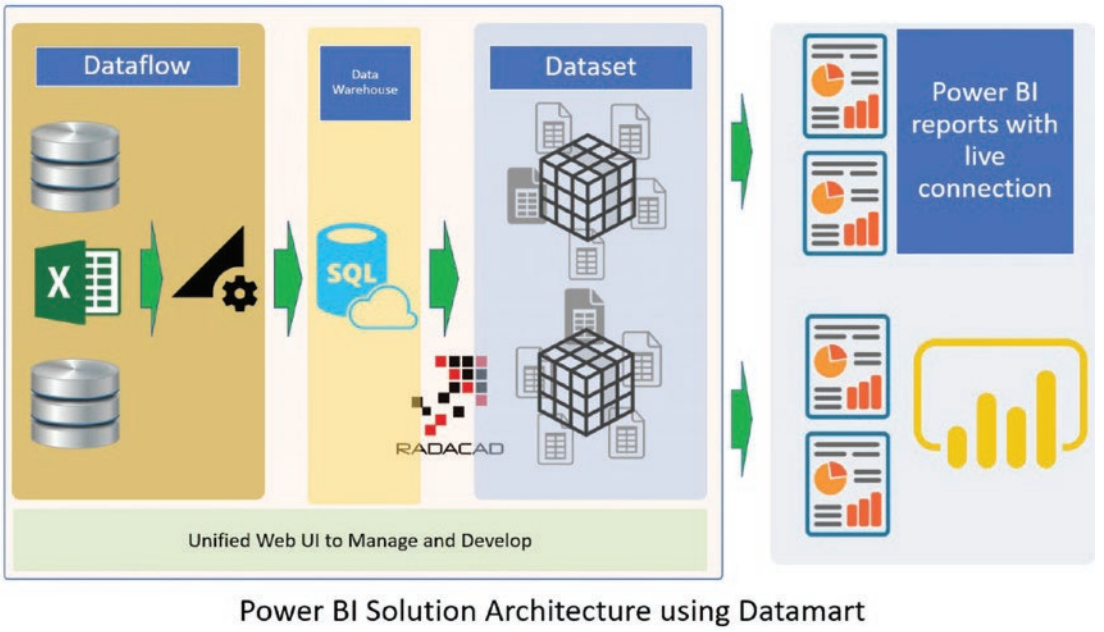


Figure 12-8. *Power BI solution architecture using datamarts*

Datamarts are explained in the previous chapter, which I recommend you study.

Layered Architecture for Dataflow

This architecture includes three layers (or four if you use datamarts). However, this can be expanded into many more layers in a real-world situation. The Data Preparation layer, which is done by dataflow, can split into multiple layers.[2]

Staging Dataflows

One of the key points in any data integration system is to reduce the number of reads from the source operational system. In a traditional data integration architecture, this reduction is made by creating a new database, called a *staging database*. The purpose of the staging database is to load data as is from the data source into the staging database on a regular schedule. The rest of the data integration will then use the staging database as the source for further transformation and convert it to the dimensional model structure.

I recommended that you follow the same approach using dataflows. Create a set of dataflows that are responsible for loading data as is from the source system (and only for the tables you need). The result is then stored in the storage structure of the dataflow (either Azure Data Lake Storage or a dataverse). This change ensures that the read operations from the source system are minimal.

[2] I explain this concept in my article at `docs.microsoft.com/en-us/power-query/dataflows/best-practices-for-dimensional-model-using-dataflows*`

Next, you can create other dataflows that source their data from staging dataflows. The benefits of this approach include:

- Reducing the number of read operations from the source system and reducing the load on the source system as a result.
- Reducing the load on data gateways if an on-premises data source is used.
- Having an intermediate copy of the data for reconciliation purposes in case the source system data changes.
- Making the transformation dataflows source-independent.

Figure 12-9 shows how staging dataflows works.

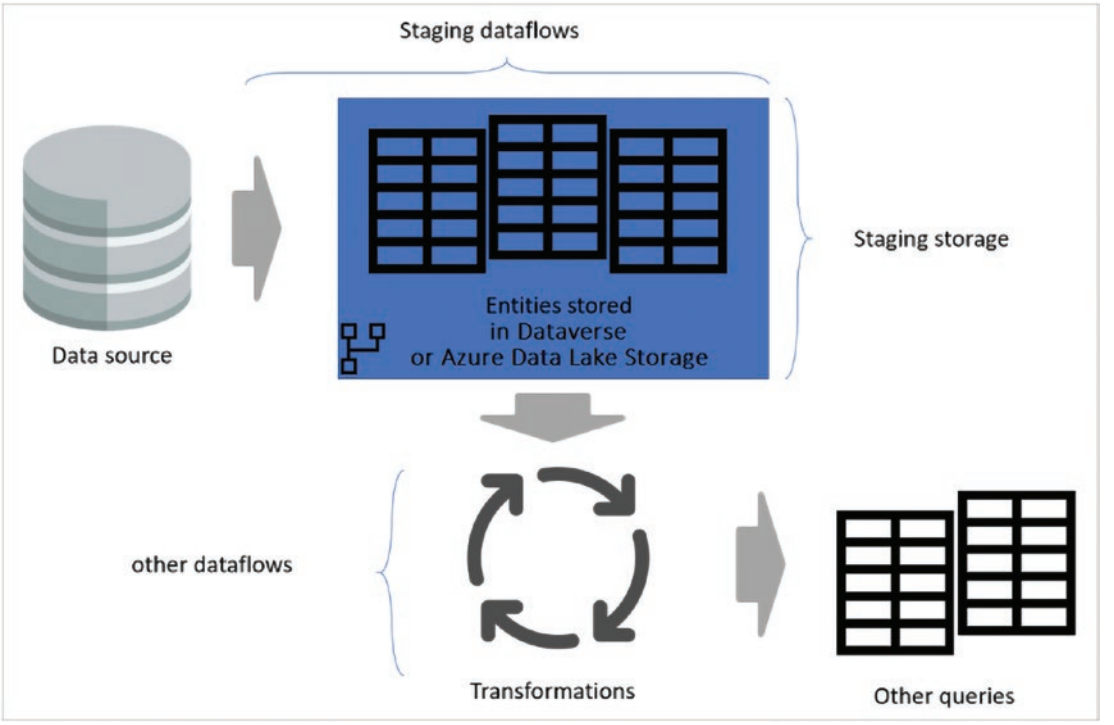


Figure 12-9. *Staging dataflows*

Figure 12-9 emphasizes staging dataflows and staging storage, and it shows the data being accessed from the data source by the staging dataflow. Entities are being stored in either Cadavers or Azure Data Lake Storage. These entities are then transformed along with other dataflows, which are sent out as queries.

Transforming Dataflows

When you've separated your transformation dataflows from the staging dataflows, the transformation will be independent of the source. This separation helps if you're migrating the source system to a new system. All you need to do in that case is change the staging dataflows. The transformation dataflows are likely to work without any problems because they're sourced only from the staging dataflows.

This separation also helps when the source system connection is slow. The transformation dataflow, shown in Figure 12-10, won't need to wait to get records coming through a slow connection from the source system. The staging dataflow has already done that part, and the data will be ready for the transformation layer.

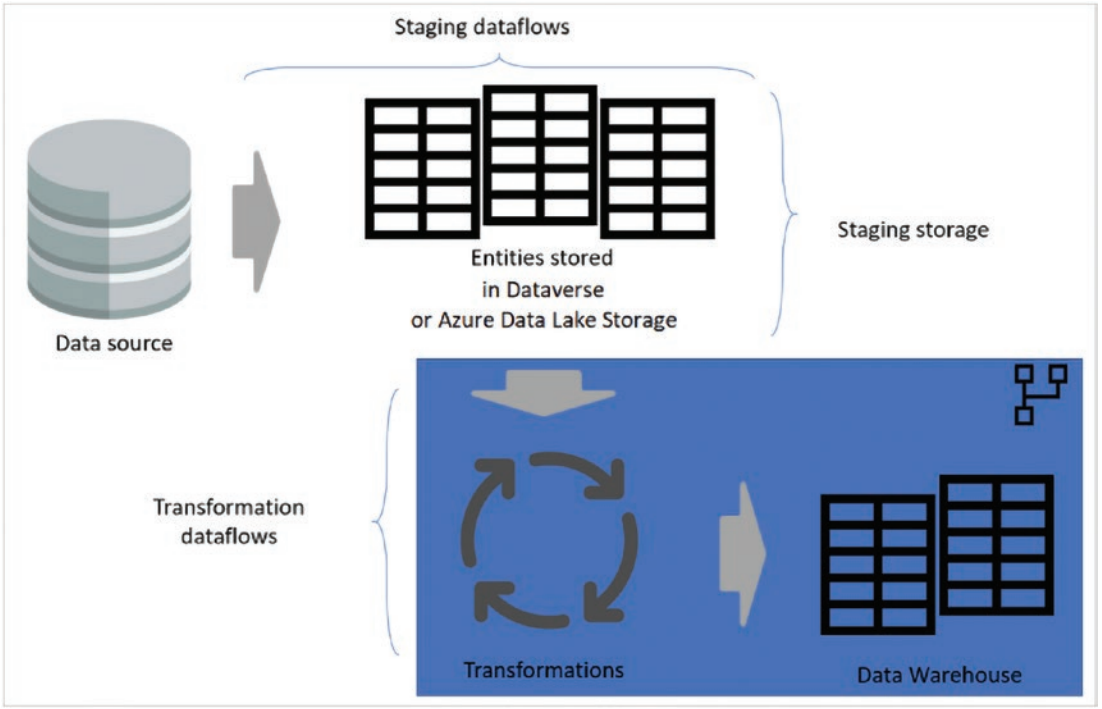


Figure 12-10. *Transforming dataflows*

Layered Dataflow Architecture

A layered architecture is an architecture in which you perform actions in separate layers. The staging and transformation dataflows can be two layers of a multi-layered dataflow architecture. Trying to do actions in layers ensures minimum maintenance is required. When you want to change something, you just need to change it in the layer in which it's located. The other layers should all continue to work fine.

Figure 12-11 shows a multi-layered architecture for dataflows in which their entities are then used in Power BI datasets.

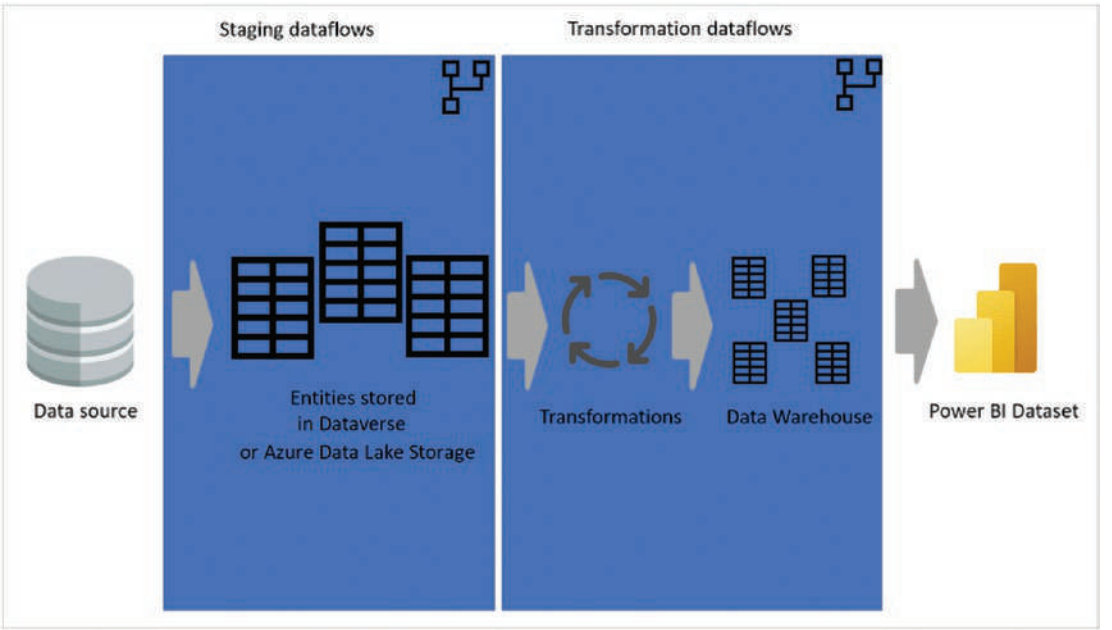


Figure 12-11. *Layered architecture of a dataflow*

The dataflow layered architecture is not only helpful when you are using the three-layered architecture, but also when you use the four-layered architecture with a datamart. A datamart can get data from dataflows that are ingesting data from some other transformation dataflow, which are then getting data from the data source using staging dataflows.[3]

Chained Datasets

Power BI datasets can also be implemented in multiple layers, by way of chained datasets. Chained datasets are datasets that use the DirectQuery to Power BI Dataset scenario. These chained datasets can be further modeled or can include data from other sources. This is particularly helpful in scenarios in which data analysts use a centralized model built by the BI team and extend it to a smaller chained model for their use cases. See Figure 12-12.

[3] Information about layered dataflow architecture borrows from my previous work, found at `docs.microsoft.com/en-us/power-query/dataflows/best-practices-for-dimensional-model-using-dataflows*`

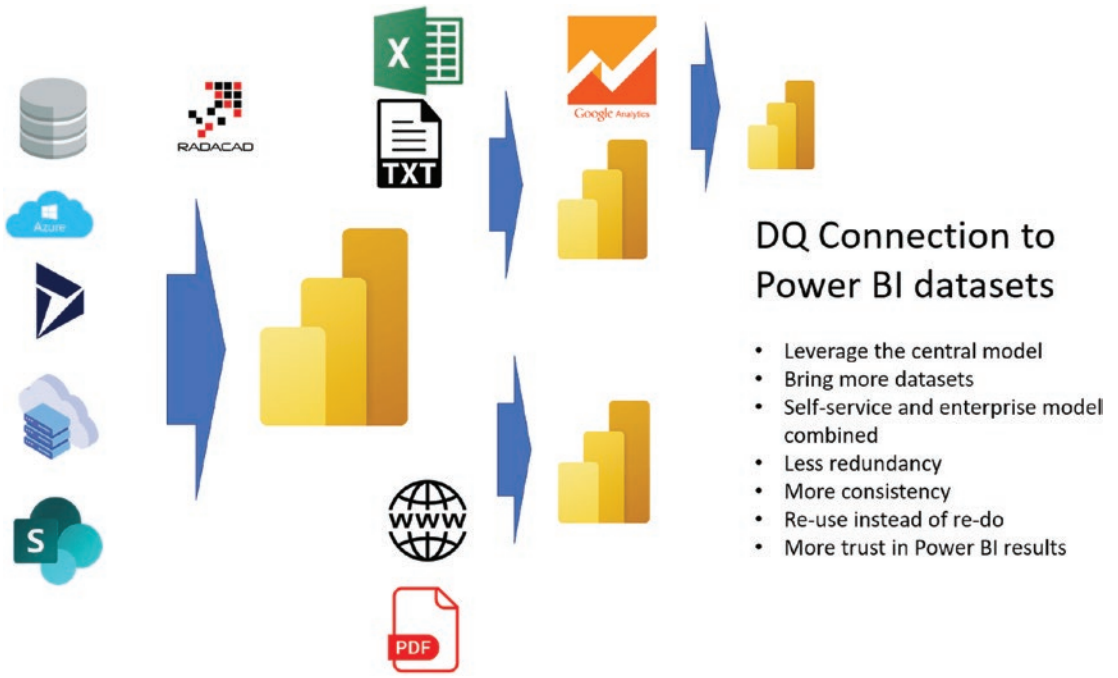


Figure 12-12. *Power BI chained datasets*

As you see in Figure 12-12, the dataflow and the dataset can become multiple layers. The number of layers in the architecture is not important. Designing layers in a way that leads to less maintenance and more reusable objects and components is the critical thing to consider when you design the architecture of a Power BI solution.

What About Enterprise Architecture?

You might already use Power BI in an enterprise solution using Azure Data Factory or SSIS as the ETL tool, SSAS as the modeling tool, and Power BI Live Connection as the visualization tool (see Figure 12-13). Such an architecture is similar to what Figure 12-13 shows. If you already have that architecture in place, I recommend you proceed.

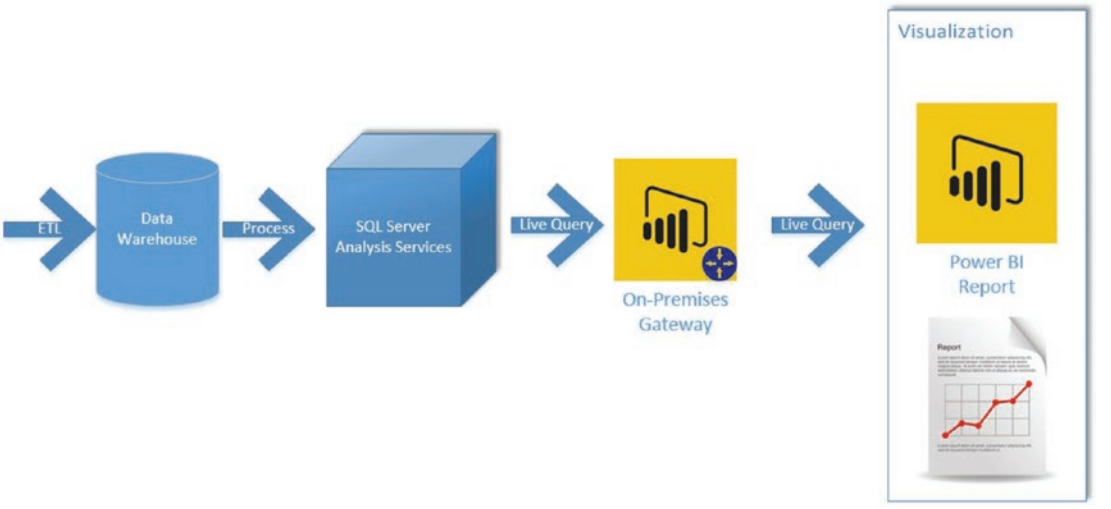


Figure 12-13. *Enterprise architecture for a Power BI implementation using other tools and services for ETL and data modeling*

Summary

Implementing a Power BI solution is simple. However, maintaining it is not! You must have a proper architecture if you want low maintenance, highly scalable, consistent, and robust Power BI implementation. This chapter explained the benefits of a proposed architecture that leverages dataflows for the data preparation layer and shared datasets for the data modeling layer. The architecture can also be enhanced using Power BI datamarts. The proposed architecture will require minimum maintenance efforts—it is highly scalable. I strongly recommend you consider using this architecture in your implementations.

CHAPTER 13

Datamarts vs. Dataflows vs. Datasets

Datamarts, dataflows, and datasets are all Power BI components that deal with data. I have presented about these a lot, and one of the questions I get is what is the difference between them? In this chapter, you learn the differences between these three components, when and where you use each, and how they work together with the other components of Power BI.

What Is a Dataflow?

A Power BI dataflow is the data transformation component in Power BI. It is a Power Query process that runs in the cloud, independent of the Power BI report and dataset, and it stores the data into Azure Data Lake storage (or a dataverse). Dataflows are not limited to Power BI; they can be created and used in other services such as Power Platform (Power Apps). Dataflows give you the transformation engine of Power Query plus the storage option. Dataflows give you a reusable ETL (Extract-Transform-Load) component. Figure 13-1 shows an overview of dataflows.

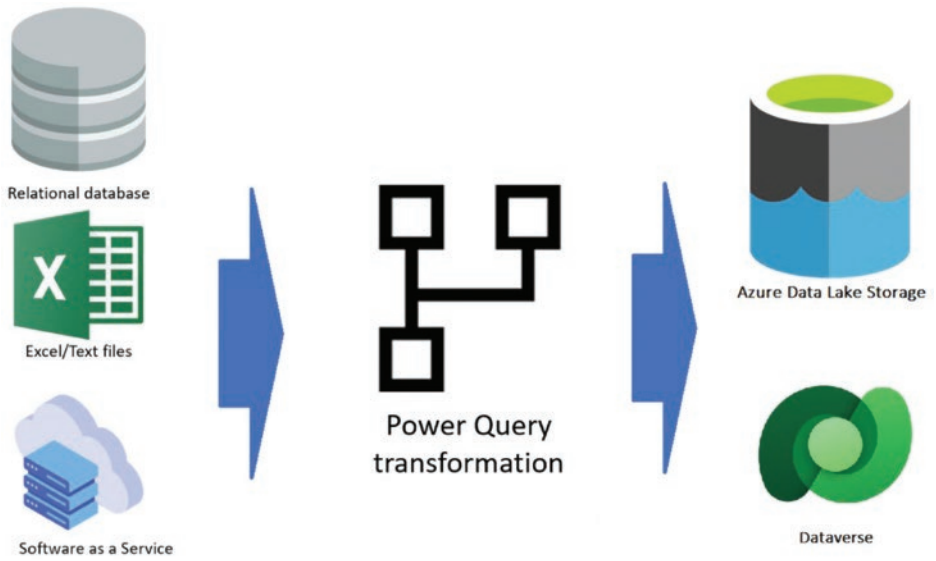


Figure 13-1. *A Power BI dataflow*

To learn more about dataflows, I recommend the earlier chapter in this book.

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_13

What Is a Dataset?

A Power BI dataset is the object that contains the connection to the data source, data tables, the data itself, the relationship between tables, and DAX calculations. Usually, a Power BI dataset is hidden from the Power BI Desktop view but can easily be seen in the Power BI Service. A Power BI dataset is used commonly when sharing a model between multiple visualization objects (such as multiple Power BI reports, paginated reports, and Excel reports). The ability to use a shared dataset will give you a reusable modeling layer in Power BI. Figure 13-2 shows how datasets appear in your workspace.

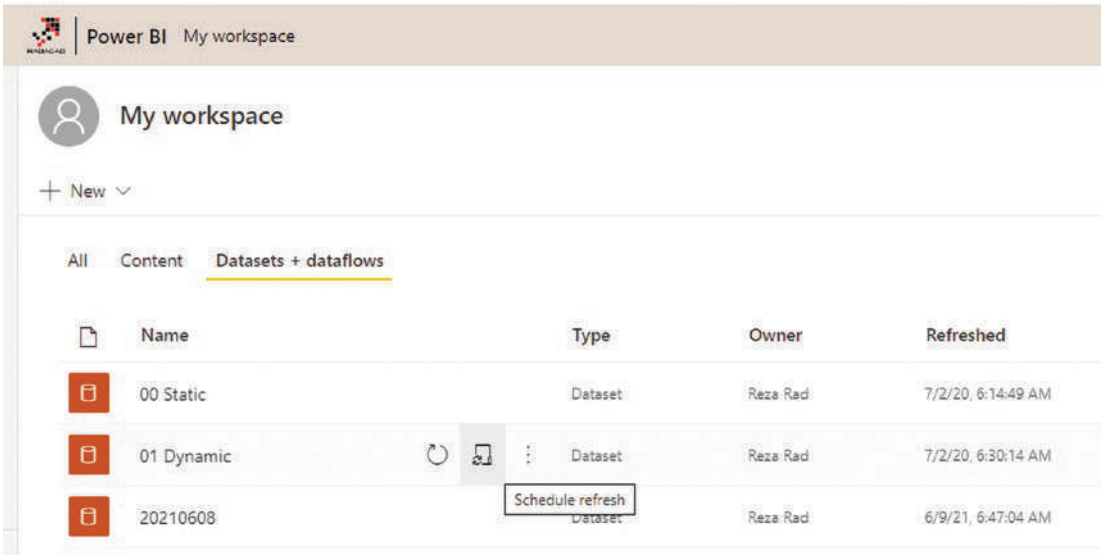


Figure 13-2. *A Power BI dataset*

To learn more about datasets and how they are used in Power BI, read the chapter about them earlier in this book.

What Is a Datamart?

A Power BI datamart is a recently added component to the Power BI ecosystem. Power BI datamarts are a combination of dataflows, an Azure SQL database (acting like a data warehouse), and a dataset. Power BI datamarts also come with a unified editor in the Power BI Service. Power BI datamarts are more like containers around other components of Power BI (dataflows, datasets, and an Azure SQL database), as illustrated in Figure 13-3.

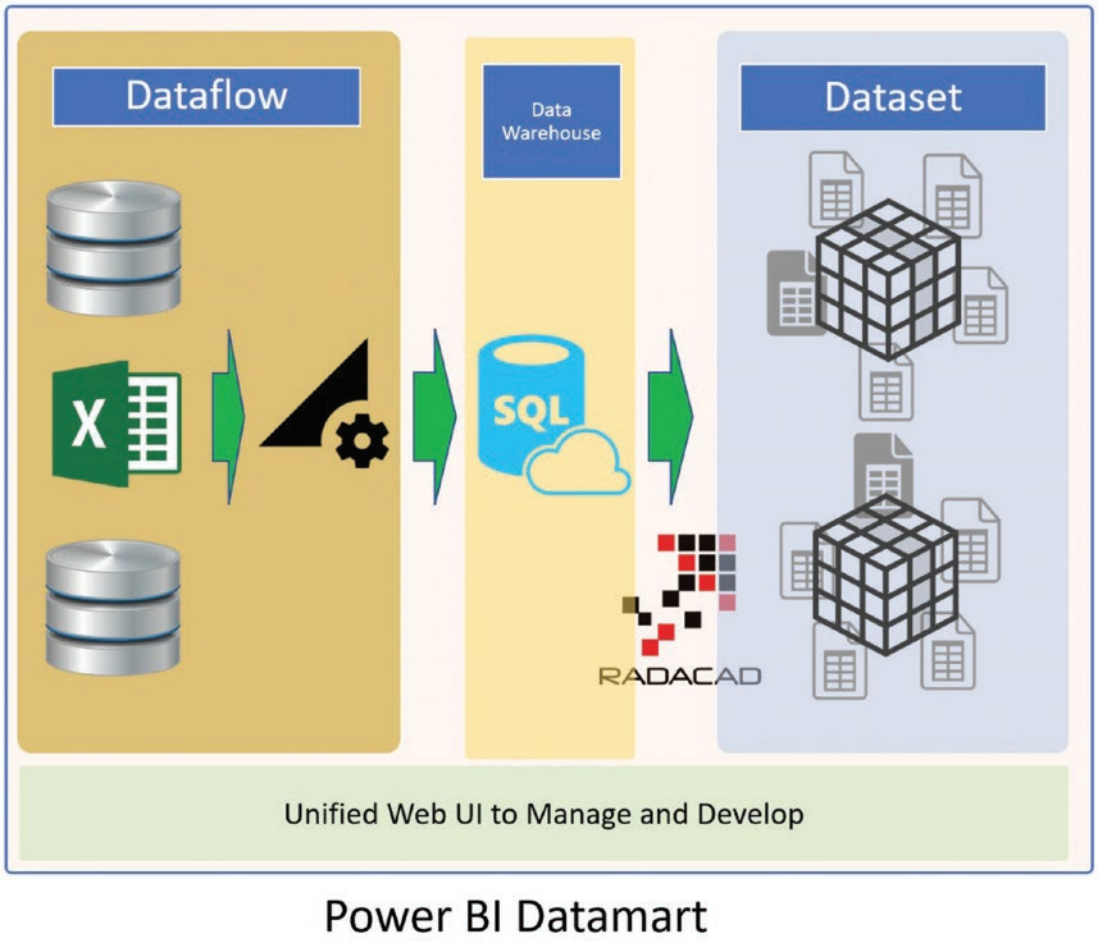


Figure 13-3. *A Power BI datamart*

To learn more about datamarts, read the earlier chapter in this book.

Differences Between Dataflows and Datasets

Datamarts, dataflows, and datasets are Power BI components that store and work with data. Now that you know what they are, let's look at the differences between these three components.

A dataflow is the Power Query component.

Dataflows decouple the Power Query logic and code from the Power BI file so that it can be used in multiple files. You can get data from many different data sources, do the transformations using Power Query online, get the data in the shape you want, set a scheduled refresh for it, and load it into storage options (such as Azure Data Lake storage or a dataverse). See Figure 13-4.

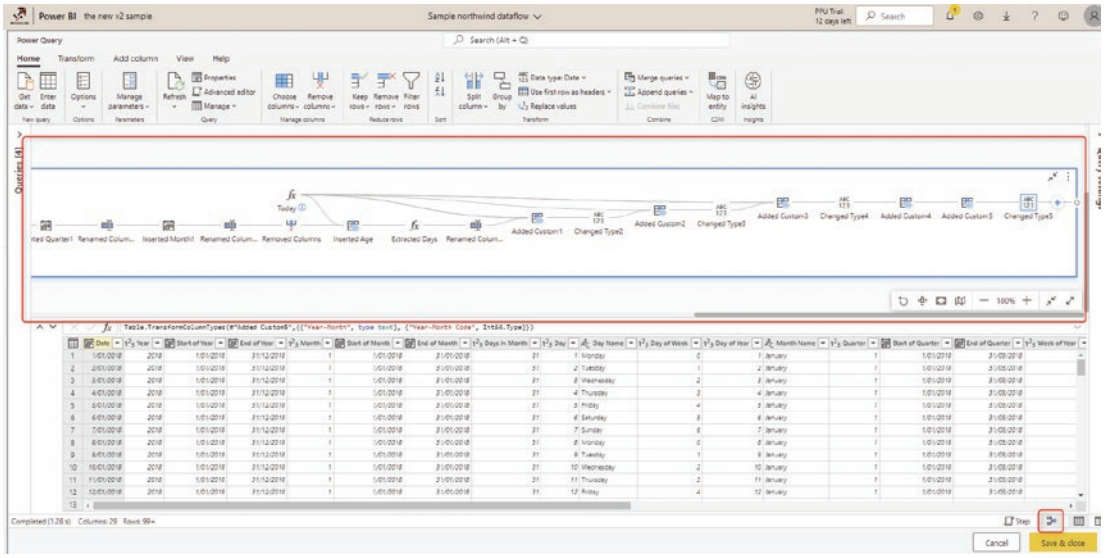


Figure 13-4. *Dataflows are the Power Query component*

Dataflows are not just for Power BI.

The dataflow is the only component of the three that can also be created outside of Power BI. You do not need a business intelligence or data analysis requirement to use dataflows. Sometimes you may need a dataflow for just data integration. For example, you may want to get data from some source systems, transform it, and store it in data storage. This might be for other applications to use.

Dataflows in Power BI can be used for data analysis purposes, but you can also create dataflows in the Power Platform under the Power Apps portal, as shown in Figure 13-5.

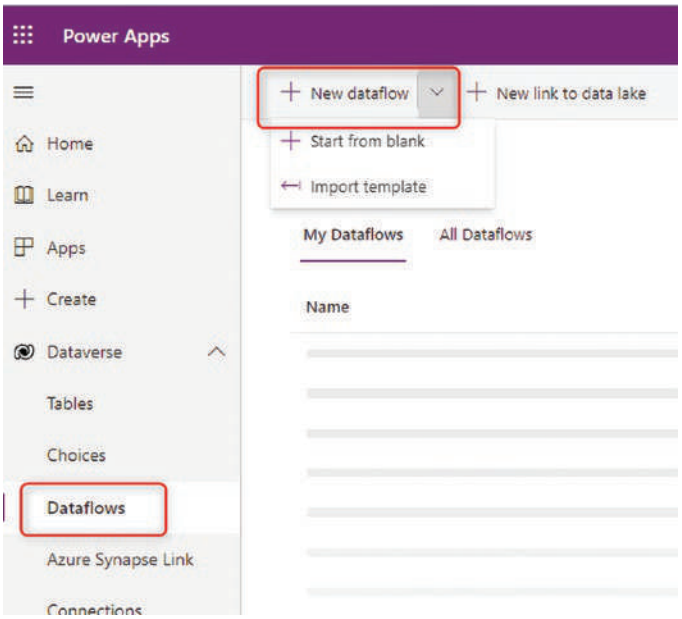


Figure 13-5. *Dataflows in the Power Apps portal*

Datasets replace DAX calculations and relationships.

Using a shared dataset, you can reuse the DAX calculations and relationships you created for one model in other Power BI files. If you want to reuse the DAX measures, the calculated columns and tables, the hierarchies, the field-level formatting, and the relationships defined in your model for multiple files, shared datasets do that for you. You can connect multiple reports to a shared dataset and reuse the data model, as shown in Figure 13-6.

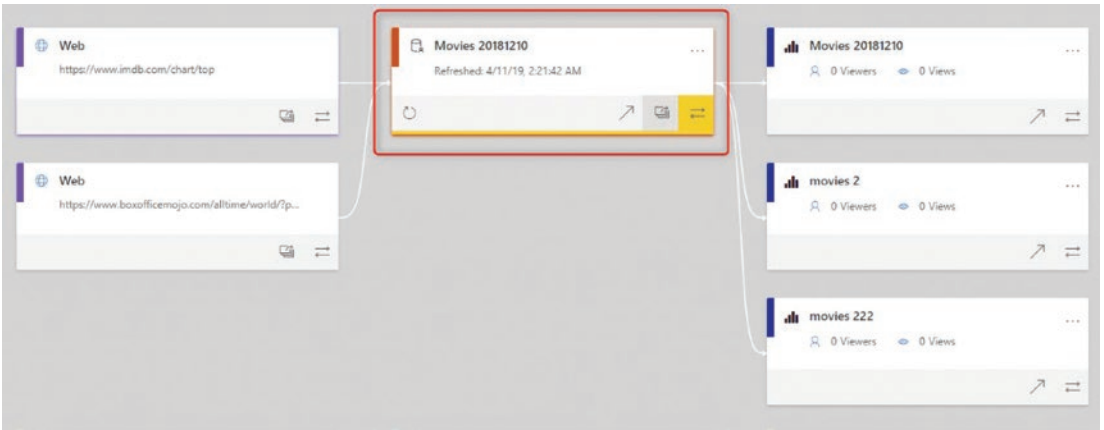


Figure 13-6. *Shared datasets replace your DAX and any calculations*

Why not use DirectQuery from the source instead of a dataset?

The question that normally comes to mind is, what if you already have a data warehouse? Or even if there is no data warehouse, what if you consider the storage of your dataflow a data warehouse? Can you connect DirectQuery to that? Isn't a Power BI dataset an unnecessary layer on top of that? Why would you need Power BI datasets in those cases?

Power BI datasets use in-memory engine storage for the data. The in-memory storage for the data ensures the best performance in the report and visualization, as the interaction in the report would be the fastest. It also includes a powerful calculation language called DAX that helps with some analytical requirements and calculations. So even if you already have a data warehouse, I still highly recommend using a dataset on top of that.

Dataflows represent the ETL layer.

Dataflows represent the Data Transformation layer in your Power BI implementation, highlighted in Figure 13-7. The terminology for this layer is ETL (Extract, Transform, Load). It will extract data from the data sources, transform the data, and load it into the CDM.

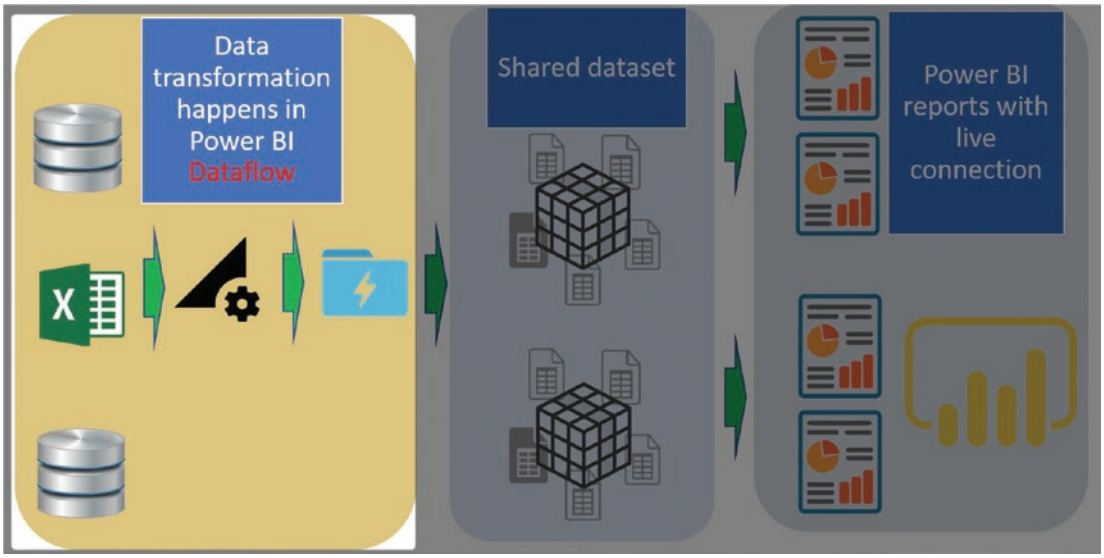


Figure 13-7. *Dataflows are the ETL layer*

Datasets represent the modeling layer.

Datasets represent the layer of all the calculations and modeling, highlighted in Figure 13-8. They get data from the dataflow (or other sources) and build an in-memory data model using the Power BI (Analysis Services) Engine.

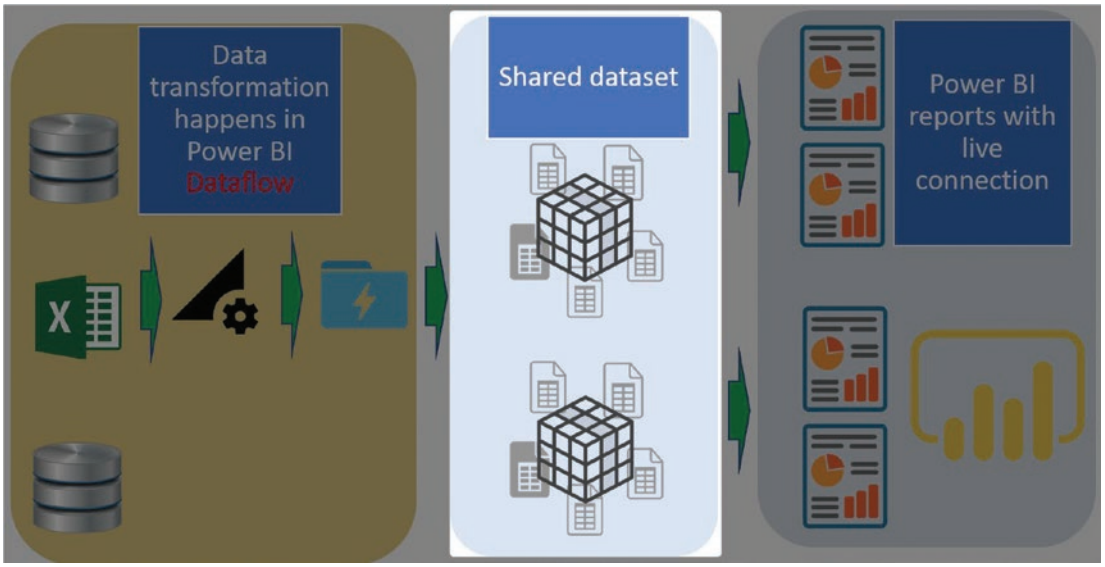


Figure 13-8. *Datasets make up the modeling layer*

Dataflows feed data into the dataset.

The result of the dataflow will be fed into a dataset for further modeling; a dataflow by itself is not a visualization-ready component.

Datasets feed data into visualizations.

Because the dataset is an in-memory model built and is ready for visualization, its result is used to build a visualization.

Dataflows access the data source directly.

Unless you use a linked or computed entity, a dataflow usually gets data directly from the data source.

Datasets can access data from the dataflow.

Although a dataset can directly get data from a data source, it is a best practice that a shared dataset gets the data from dataflows. This supports a multi-developer implementation of Power BI.

A dataflow developer needs Power Query skills.

One of the reasons to use dataflows and shared datasets is to decouple the layers, so you have multiple developers building the Power BI solution at the same time. In such an environment, the skillset needed for a dataflow developer include Power Query and how to build Star-Schemas, and so on. No DAX or visualization skills are required for dataflow developers.

A dataset developer needs DAX and modeling skills.

On the other hand, the dataset developer needs to know everything about the relationships in Power BI and the calculations in Power BI using DAX. Although the dataset developer can know Power Query and visualization, it does not need to be their primary skill.

Dataflow users are called data modelers.

The dataflow's result can be used for data modelers. It is not a great approach to give the output of dataflow to report visualizers. The dataflow has to be loaded into a model with proper relationships and calculations added to it.

Dataset users are called report visualizers.

The result of a dataset is ready for report visualizers. They can have a live connection to the shared dataset and build their visualizations from it.

Dataflows solve the problem of having multiple versions of the same table in different PBIX files.

Using a dataflow, you reduce the need to copy and paste your Power Query scripts into other files. You can reuse a table in multiple files.

Datasets solve the problem of having multiple versions of the same DAX code in different PBIX files.

Using a shared dataset, you can have multiple reports using the same calculations and data model without duplicating the code.

Dataflows vs. Datasets

Dataflow and datasets are not the same thing. As Table 13-1 demonstrates, there are considerable differences between them.

Table 13-1. *Dataflows vs. Datasets*

Dataflow Vs. Dataset

Dataflow	Dataset
Replacement of your Power Query layer	Replacement of your modeling, relationship, DAX expressions
ETL Layer	Modeling Layer
Feeds Data to the Dataset	Feeds the Visualization Layer
Access the data source directly (usually)	Access the data from dataflow (best practice)
Dataflow developer needs to have Power Query skills	Dataset developer needs DAX and modelling skills
Users of dataflow are data modelers	Users of dataset are report visualizers
Dataflow solves the problem of having multiple version of the same table in different PBIX files	Dataset solves the problem of having multiple version of the same DAX code in different PBIX files

They are two separate, essential components of Power BI and both have their places in the Power BI architecture for a multi-developer scenario. Figure 13-9 shows a representation of their essential-but-separate roles. Dataflows and datasets are not replacements for each other; they are the complements of the other.

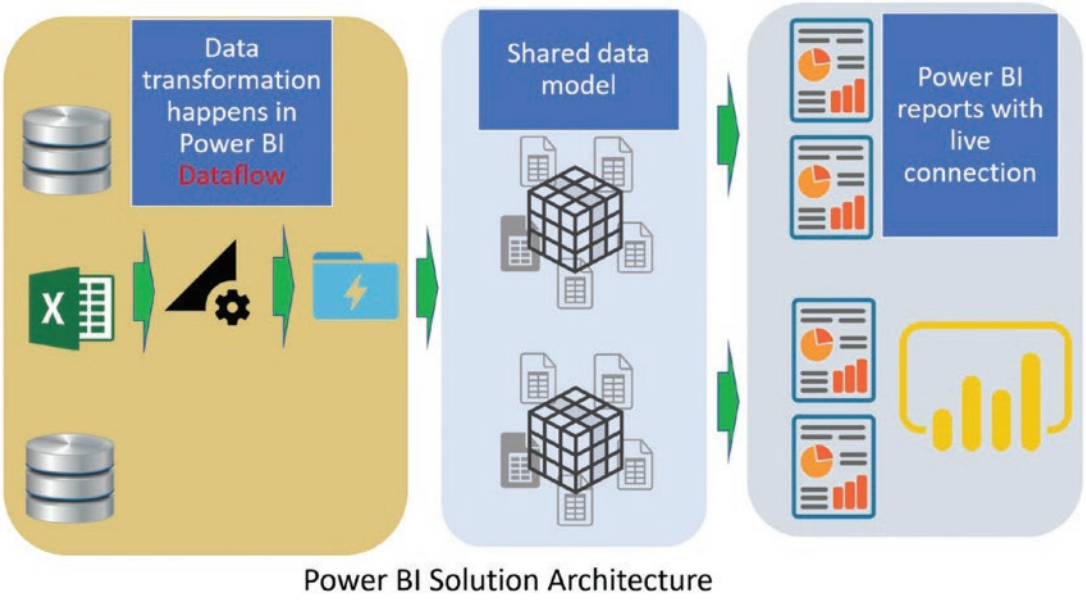


Figure 13-9. *The Power BI Solution architecture with dataflows and datasets*

What About Datamarts?

You learned a lot about the differences between dataflows and datasets. But what about datamarts? Shouldn't they be in the comparison list?

Power BI datamarts are more like containers of components rather than single objects (see Figure 13-3). When you create a Power BI datamart, you are creating a dataflow, an Azure SQL database, and a dataset. This means that the datamart benefits from the features mentioned for dataflows and datasets. Datamarts also have an extra component: they store data in the Azure SQL database. After processing by the dataflow, the data is loaded into the Azure SQL database. Some call this a data warehouse, and some may even call this a datamart. But Power BI datamarts include all these components: dataflows, an Azure SQL database, and a dataset.

Power BI Datamart Components

Power BI datamarts also provide a unified Web UI to build and manage all these components. With the addition of the datamart, the multi-developer architecture of Power BI looks more like Figure 13-10.

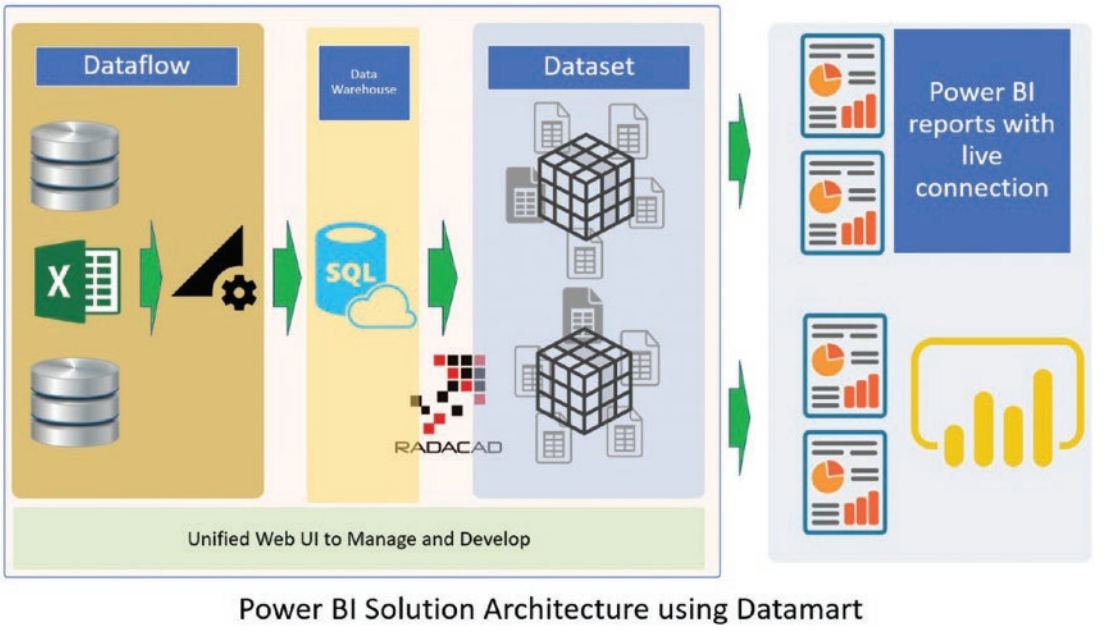


Figure 13-10. *Power BI solution architecture using datamarts*

You might wonder whether datamarts will replace datasets and dataflows. The next section considers this question.

Will Dataflows Be Replaced by Datamarts?

No. Certainly not. A dataflow is a component by itself. As mentioned, you can build and use a dataflow without needing a BI solution. Datamarts are normally useful when you are building a BI solution. You may just want to integrate some tables into storage and reuse them in other applications; in that case, you can use a dataflow by itself.

Another use case of a dataflow by itself is that even if you have a datamart, you may still want to implement multiple layers of dataflows for staging and transformation. Having multiple layers of dataflows (see Figure 13-11) provides an effective technique in data source isolation and reuse of dataflow components.

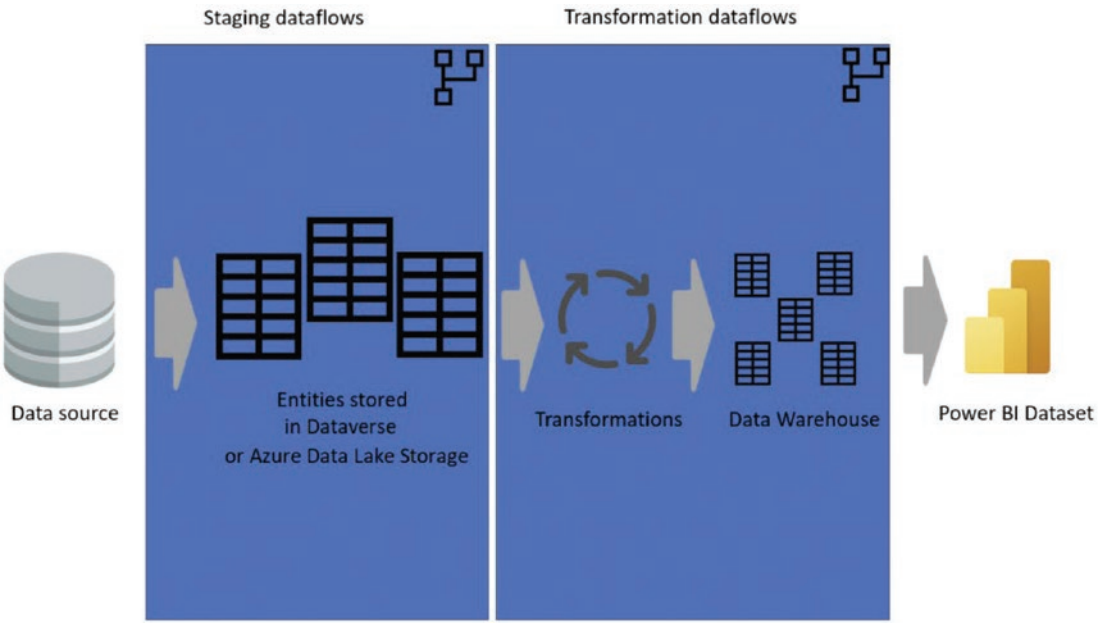


Figure 13-11. *Multi-layered dataflow architecture*

Are Datasets Being Replaced by Datamarts?

The answer to this question is also no. Although it is now easier to create a dataset from the unified UI of the datamart, that doesn't reduce the need for the dataset to be a separate component of its own. There are plenty of use cases for a dataset as a component on its own. Imagine you are implementing an architecture in which the data transformation is done using another service (such as Azure Data Factory), and the data warehouse is in Azure Synapse. You can still use Power BI datasets to build the data model and calculations on top of that without building a full datamart.

Another use case is that even if you use a Power BI datamart, you may still create chained datasets on top of your existing dataset. These chained datasets are DirectQuery to Power BI datasets (which in this case is part of a datamart), but they can include other data sources. Chained datasets (see Figure 13-12) are a very useful way to work with self-service data analysts in an enterprise organization.

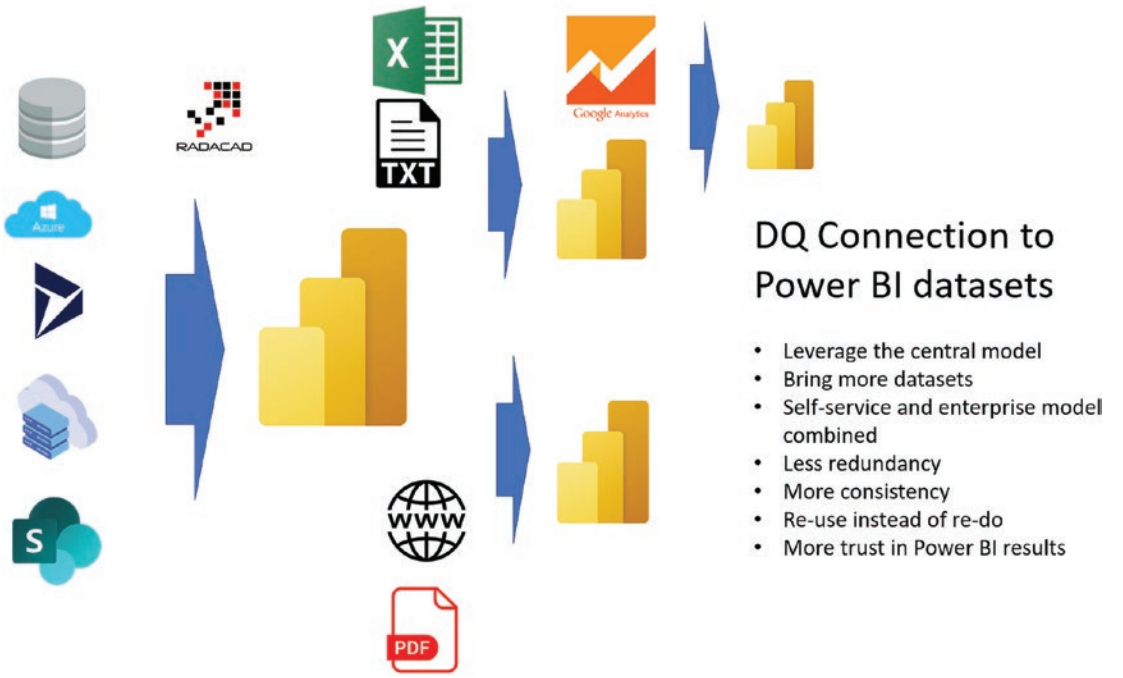


Figure 13-12. *Power BI chained datasets*

Summary

Now is time for the million-dollar question; which one of these components should you use? Each of these comes with some benefits, as you've learned. Let's answer this question using a scenario.

James is a BI developer who is building a new BI solution. His implementation includes stages such as getting data from the source, doing transformations, loading data into a data warehouse, writing calculations in the model, and visualization. Power BI datamarts enable him to build most of that in one unified structure but still use a multi-layered approach that can be easily maintained in the future.

After some time, James realizes that the data transformation side of his implementation is heavy. He wants the data transformation to be isolated from the data source so that if the source changes, his solution will still work with minimal changes. So, he uses staging and transformation dataflows in his architecture.

James has some colleagues who are business analysts in other departments. They want to use the dataset that James provided as the main source, but add data sources and calculations to it. They create chained datasets on top of that.

This scenario uses all three components in an architecture. Choosing which components you need is mainly based on what reusable components you have and where you want to use them.

CHAPTER 14

Paginated Reports

If you want to print a Power BI report and have a tabular visual in your report (such as a table or matrix), your options are limited. Fortunately, the Power BI Paginated Report feature can help with that. This chapter explains the paginated report and the differences between it and a normal Power BI report, plus situations that you need to use each.

What Is a Paginated Report in Power BI?

If you look at Figure 14-1, you can see two objects in the Power BI service. They both are report type, but they have different icons.

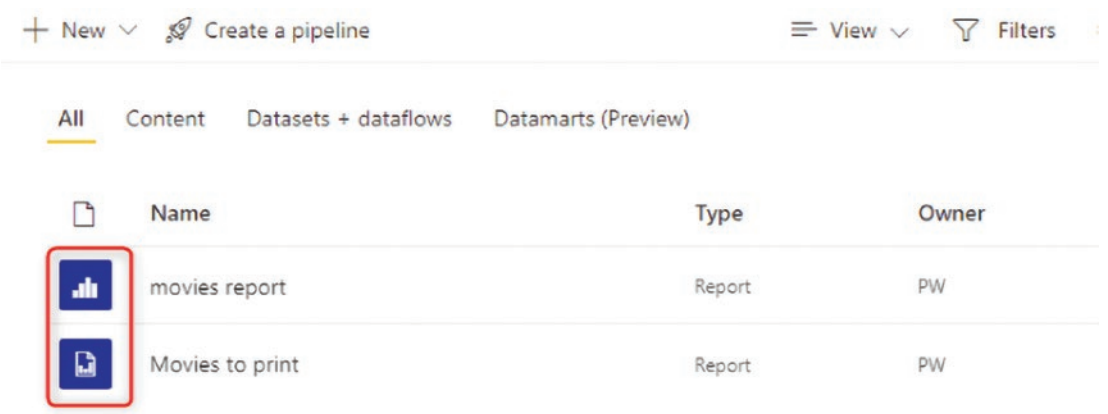


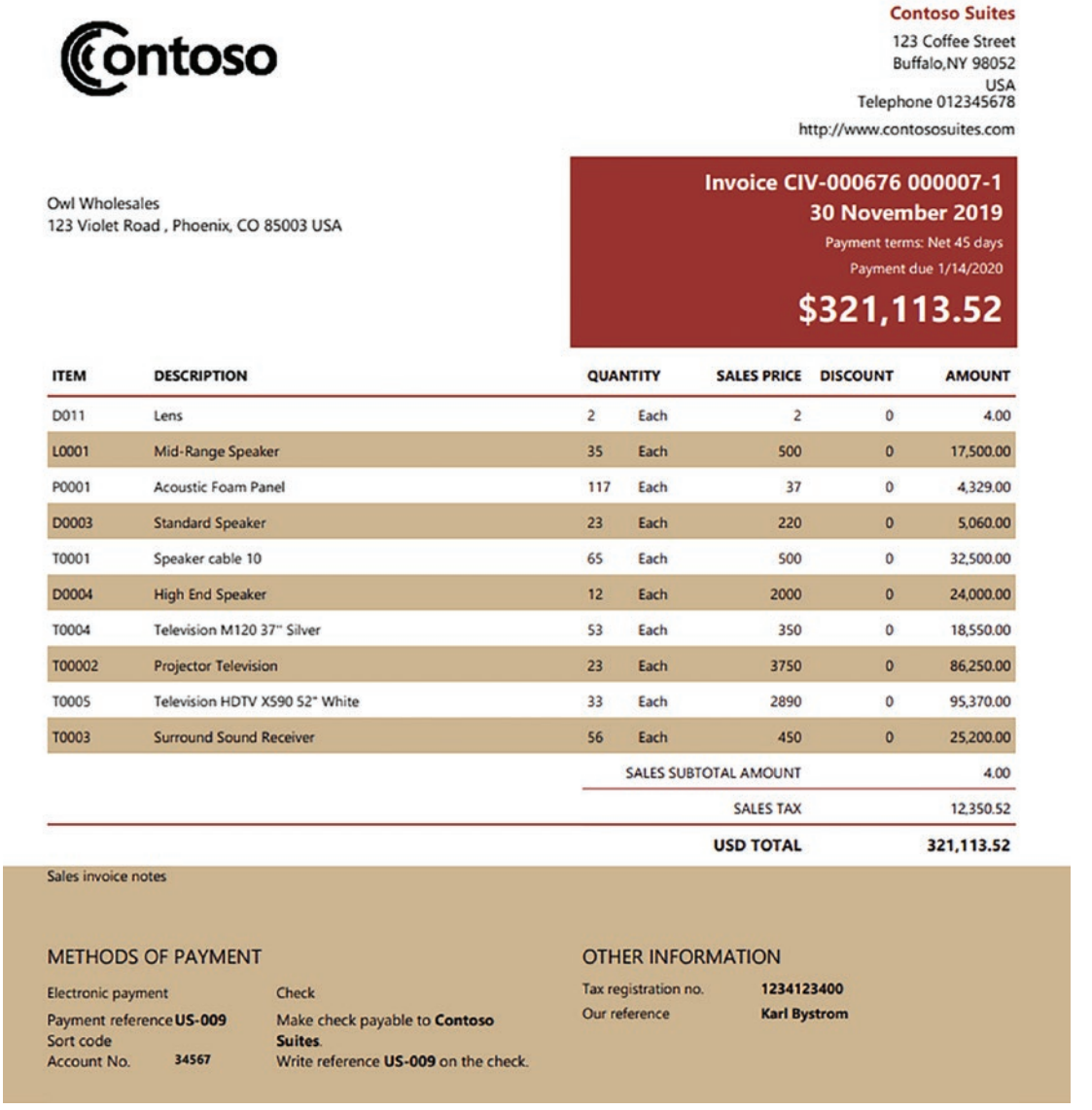
Figure 14-1. *Two types of reports in the Power BI Service*

The Movies to Print report is a paginated report. Paginated reports, although they look like normal reports, are a different type of report in Power BI.

Paginated reports are created using another tool (not using the Power BI Desktop) and for a different use case. Paginated reports are designed for pixel-perfect printing. They are ideal for exporting to PDF or printing, especially when the data presented in the report is multiple pages (such as a table with thousands of rows). That is why they are called paginated. You can design the report over multiple pages. You can set page headers and footers and categorize them so that certain information goes into certain report pages.

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_14

Figure 14-2 shows an example paginated report.



Contoso

Contoso Suites
123 Coffee Street
Buffalo,NY 98052
USA
Telephone 012345678
http://www.contososuites.com

Owl Wholesales
123 Violet Road , Phoenix, CO 85003 USA

Invoice CIV-000676 000007-1
30 November 2019
Payment terms: Net 45 days
Payment due 1/14/2020
$321,113.52

ITEM	DESCRIPTION	QUANTITY		SALES PRICE	DISCOUNT	AMOUNT
D011	Lens	2	Each	2	0	4.00
L0001	Mid-Range Speaker	35	Each	500	0	17,500.00
P0001	Acoustic Foam Panel	117	Each	37	0	4,329.00
D0003	Standard Speaker	23	Each	220	0	5,060.00
T0001	Speaker cable 10	65	Each	500	0	32,500.00
D0004	High End Speaker	12	Each	2000	0	24,000.00
T0004	Television M120 37" Silver	53	Each	350	0	18,550.00
T00002	Projector Television	23	Each	3750	0	86,250.00
T0005	Television HDTV X590 52" White	33	Each	2890	0	95,370.00
T0003	Surround Sound Receiver	56	Each	450	0	25,200.00
				SALES SUBTOTAL AMOUNT		4.00
				SALES TAX		12,350.52
				USD TOTAL		**321,113.52**

Sales invoice notes

METHODS OF PAYMENT

Electronic payment
Payment reference **US-009**
Sort code
Account No. **34567**

Check
Make check payable to **Contoso Suites**.
Write reference **US-009** on the check.

OTHER INFORMATION

Tax registration no. **1234123400**
Our reference **Karl Bystrom**

Figure 14-2. *Power BI paginated report sample*

Paginated reports are created using the Power BI Report Builder. This is a different editor than the Power BI Desktop and has to be downloaded and installed separately.

Why Paginated Reports?

As you already know, you can use the Power BI Desktop to create any report you want. So you might wonder why you would need to create paginated reports. To answer this, you must understand how printing works with a normal Power BI report.

Say you have a normal Power BI report (a report that is created using the Power BI Desktop), and your report looks like Figure 14-3.

Figure 14-3. *Power BI report with table visuals*

Figure 14-3 shows a table visual in the Power BI report. Only a few rows in that table can be presented on the screen, so it includes a scroll bar on the right. In the context of an interactive report, you can easily scroll down to get to the records you want, or you can filter the report using slicers or other visuals.

However, the printed version of the report would look like Figure 14-4.

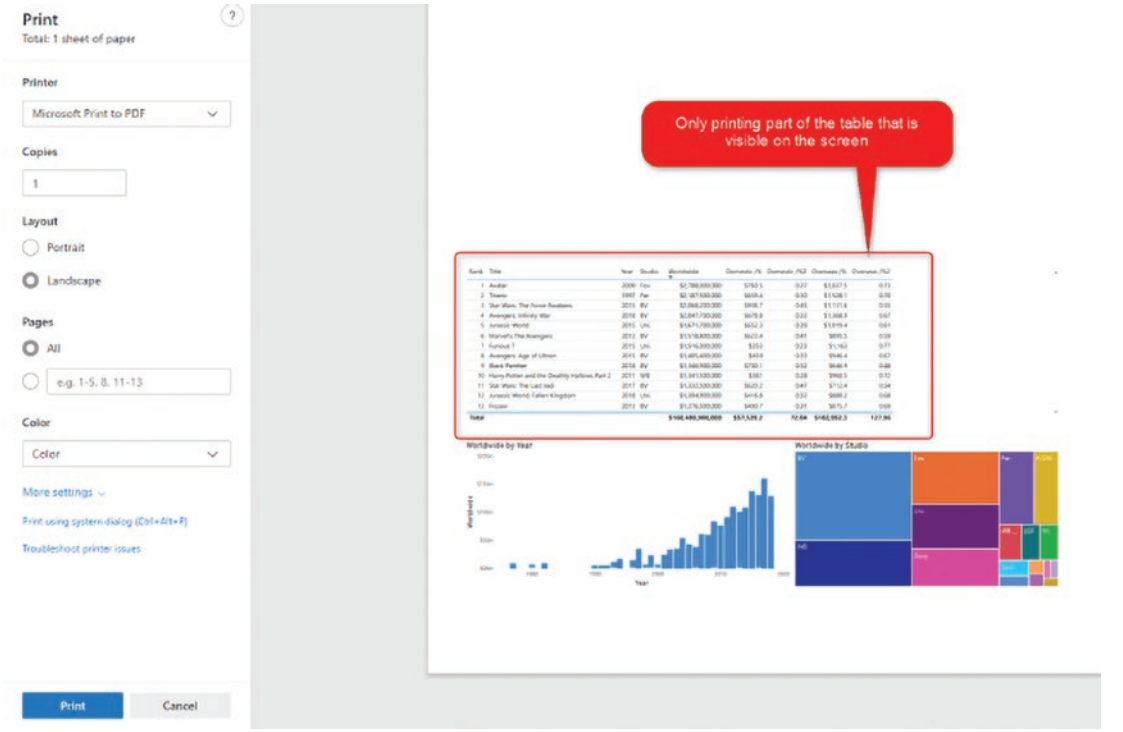


Figure 14-4. *Printed version of the Power BI report includes only the part of table rows visible on the screen*

The printed version only shows the few rows visible on the screen. All other rows, which you must scroll down to get to, are not included in the output. Printing the content of a table or matrix is not simple; you need pagination. You may need page headers and footers and some other settings. This is why there are paginated reports in Power BI. A paginated report can be designed exactly as you want to print it in the output. See Figure 14-5.

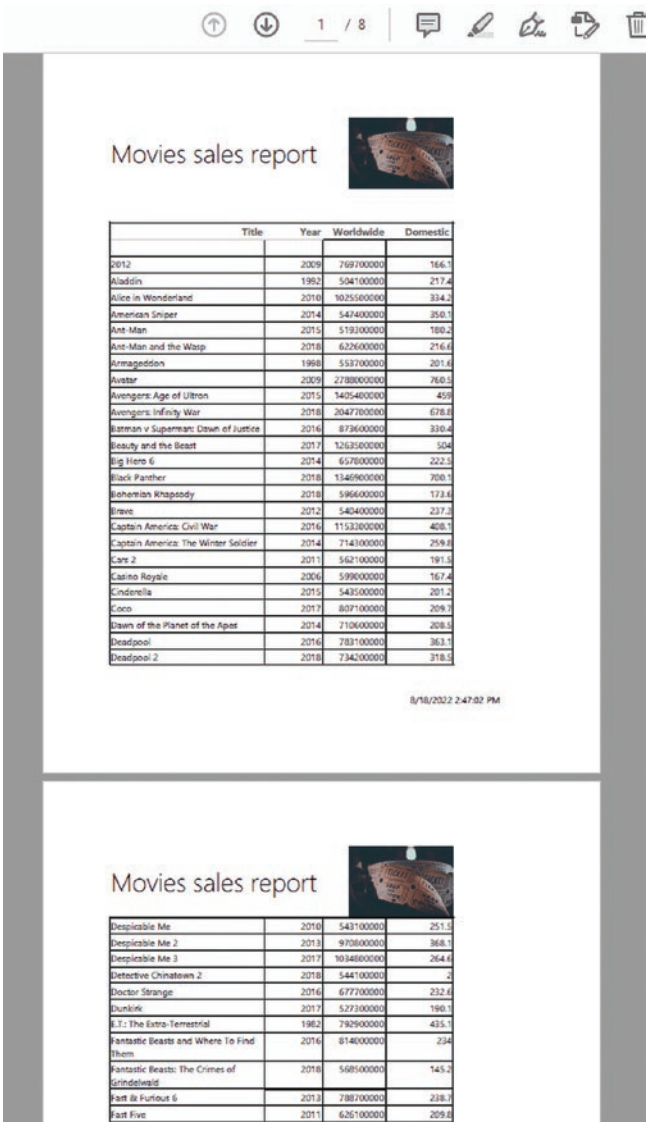


Figure 14-5. *A printed output of a Power BI paginated report*

Differences Between Power BI Reports and Paginated Reports

Although these two are reports hosted in the Power BI service, they are generated for different purposes and have different capabilities. Table 14-1 summarizes a few key differences.

Table 14-1. *Power BI Reports vs. Paginated Reports*

	Power BI Report	Paginated Report
It can be created Using	Power BI DesktopPower BI Service	Power BI Report BuilderVisual Studio
License needed	Power BI Free, Pro, Premium, PPU	Premium, PPU
Ideal for	Interactive analytical reports	Pixel-perfect print-ready reports
Design Process	Powerful and user-friendly GUI of the Power BI Desktop	Powerful but not user-friendly (more developer-focused) UI of Report Builder or Visual Studio
File extension	.PBIX	.RDL

Do You Need a Paginated Report?

If you need to print your reports on occasion, you will likely need to use paginated reports. One of the main reasons that users print reports is to carry them to the meetings, discuss them, and show them to others. When using Power BI mobile applications, especially if you design mobile-friendly reports, you can certainly have normal Power BI reports that can be carried to meetings (by phone), be discussed (with annotation on them), and shown to others. See the example in Figure 14-6. My experience shows that mobile-friendly Power BI reports can replace many (but not all) paginated reports.

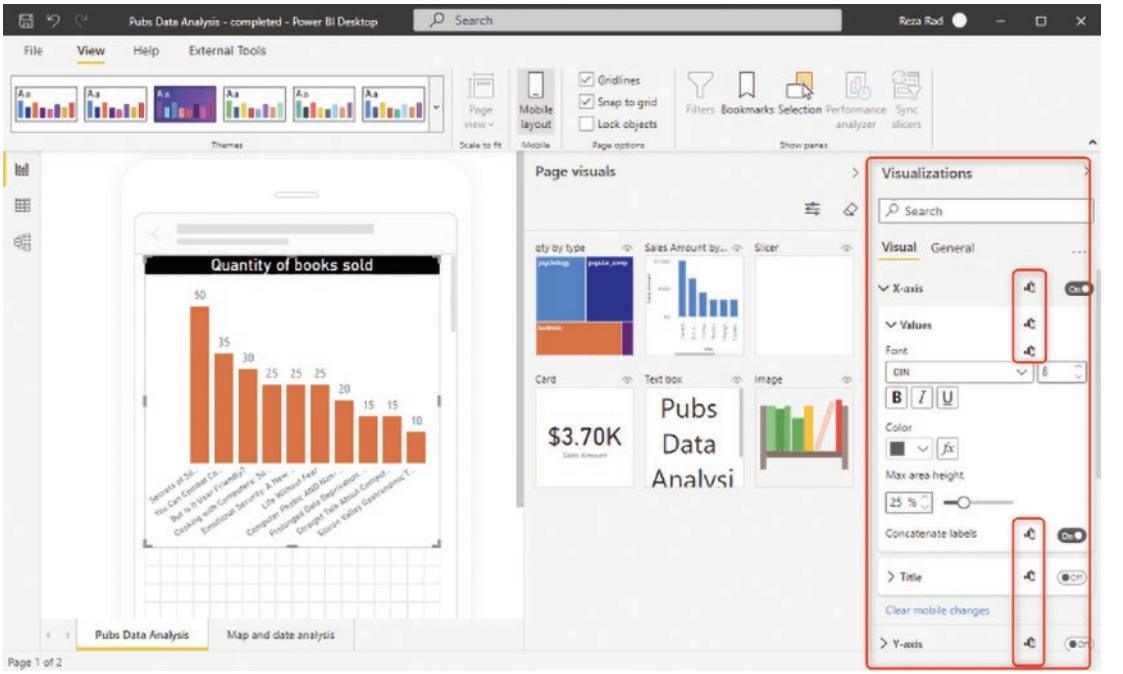


Figure 14-6. *Designing mobile-friendly reports in the Power BI Desktop*

If, after designing and adopting mobile-friendly Power BI reports, you still feel you need printed reports, it's time to create paginated reports.[1]

Architecture Best Practice: One Dataset to Rule Them All

If you create paginated reports, I urge you to consider the best architecture practices. The best architecture for development is the one that requires less maintenance and reuses components as much as possible. I explained how a dataset could be shared in multiple Power BI objects in the multi-layered architecture chapter and the shared dataset chapter.

Paginated reports can get data from a Power BI dataset. I strongly advise you to do that because your normal Power BI and paginated reports are sourced from the same dataset. Any changes to that dataset will then be available to both reports, as Figure 14-7 makes clear.

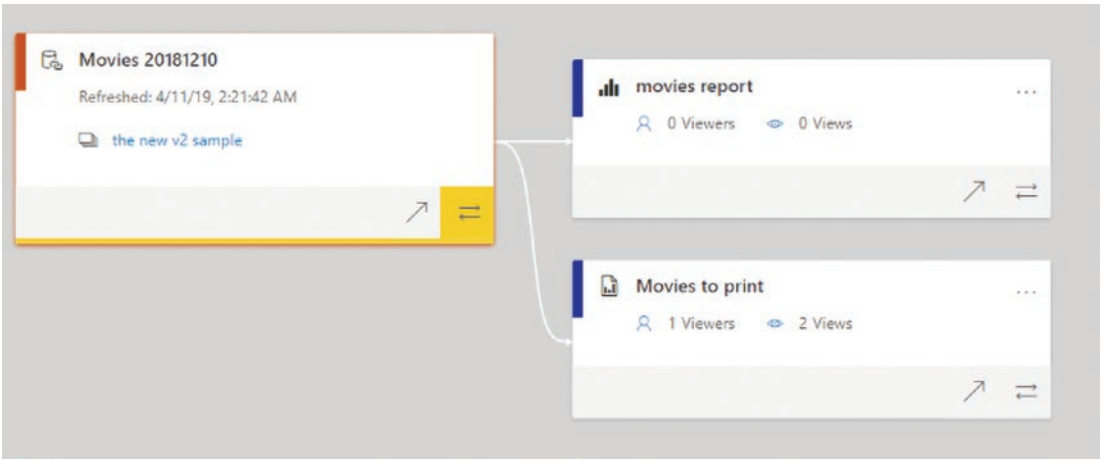


Figure 14-7. *Paginated reports and Power BI reports are sourced from a single shared dataset*

If you use this design, you can build a powerful reporting system. The Power BI report can be the interactive landing page report for the users. If they want more paginated details, you can build a master-detail scenario, passing details using parameters to the paginated report and navigating between these two. This is a great way to combine the power of both types of reports into one analytical system.

The Power BI Report Builder

Your tool to create a paginated report is called the Power BI Report Builder. This tool (before the age of Power BI) was called the Reporting Services Report Builder, or SSRS Report Builder. It is a lightweight tool for creating print-ready reports. However, compared to the Power BI Desktop, the Report Builder is not a user-friendly tool. You can download and install the Power BI Report Builder from `aka.ms/pbireportbuilder`.

As Figure 14-8 shows, this tool has a wizard that you use to start building a report. Usually, this is a good starting point if you are a beginner.

[1] To learn more about designing and adopting mobile-friendly Power BI reports, visit `radacad.com/power-bi-design-tip-design-for-mobile-device`.

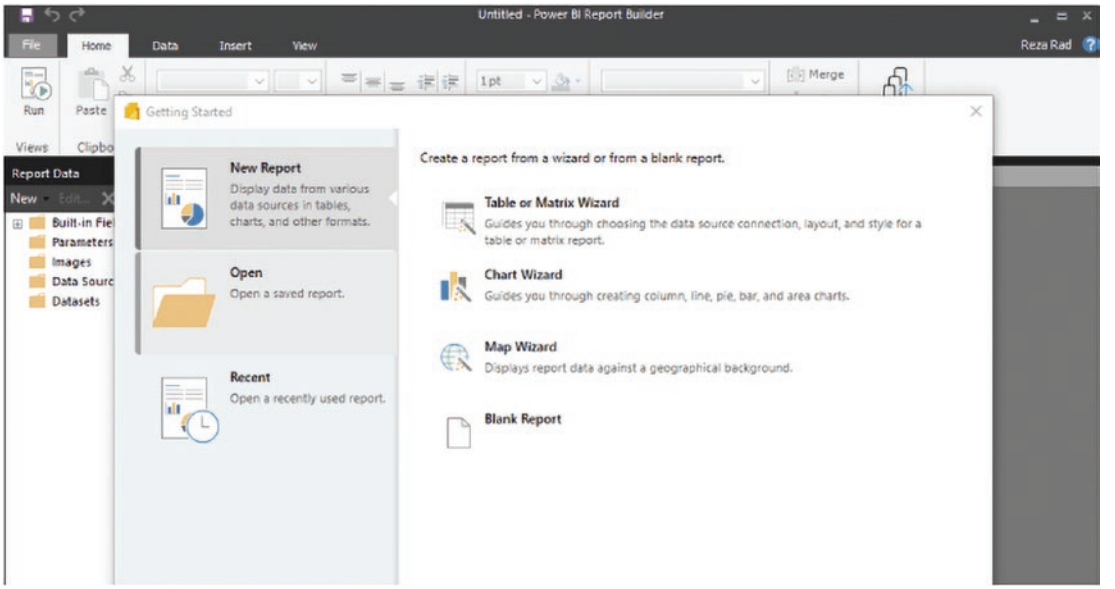


Figure 14-8. *The Power BI Report Builder*

The Report Builder has settings and configurations that are used to create a print-ready report. Figure 14-9 offers a sample view of the designer and its capabilities.

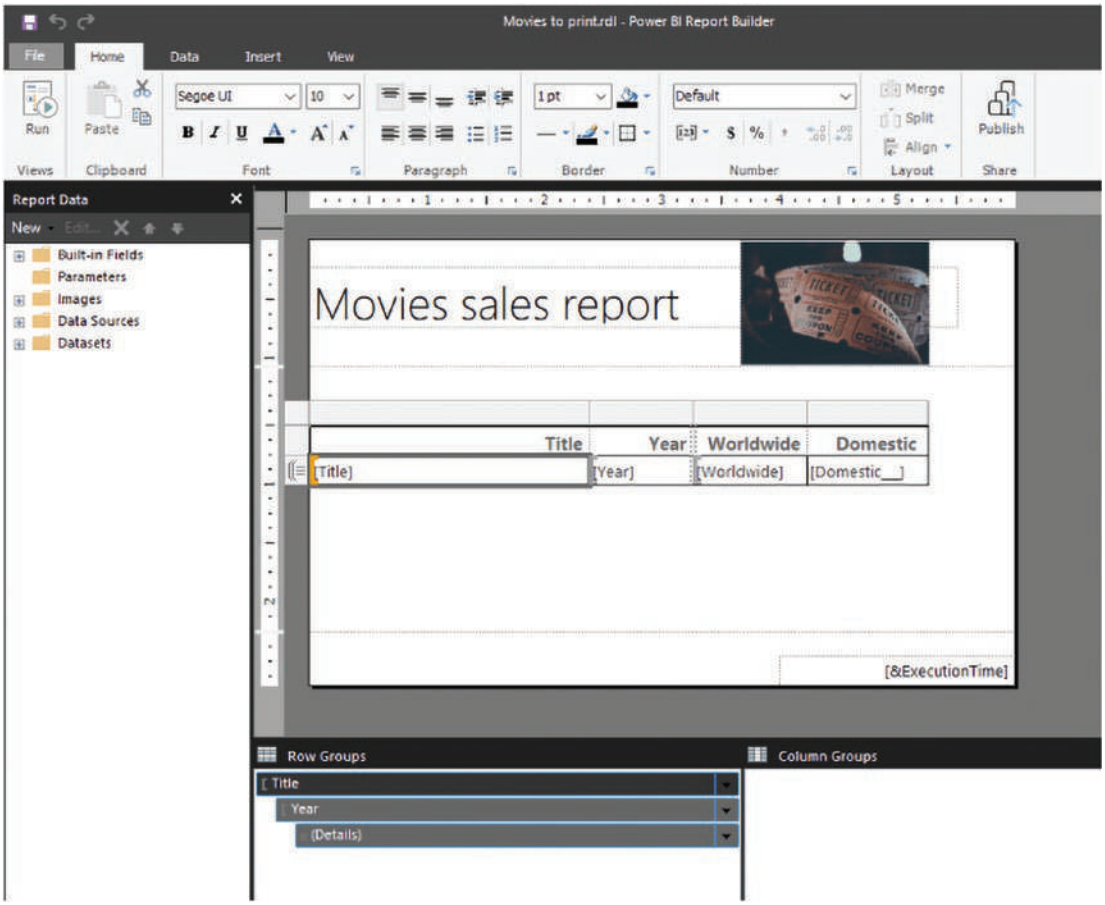


Figure 14-9. *An example of the designer for the Report Builder*

This differs from the Power BI environment, where things are more drag-and-drop, interaction-friendly. Remember that the Report Builder was created during the age of SQL Server Reporting Services, and the UI is older.

In Report Builder, you can get data from a Power BI dataset, which is what I recommend you do; see Figure 14-10.

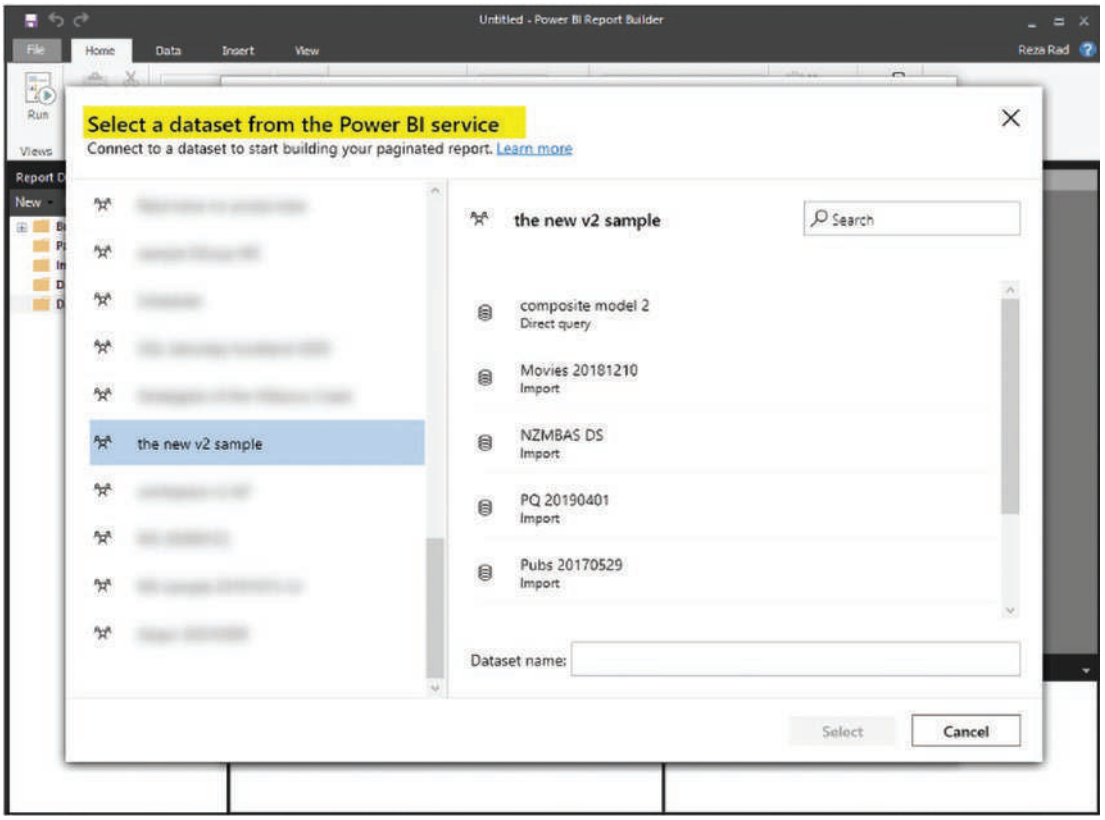


Figure 14-10. *Select a dataset from the Power BI service*

After building your report, you can publish it to the service, as Figure 14-11 shows.

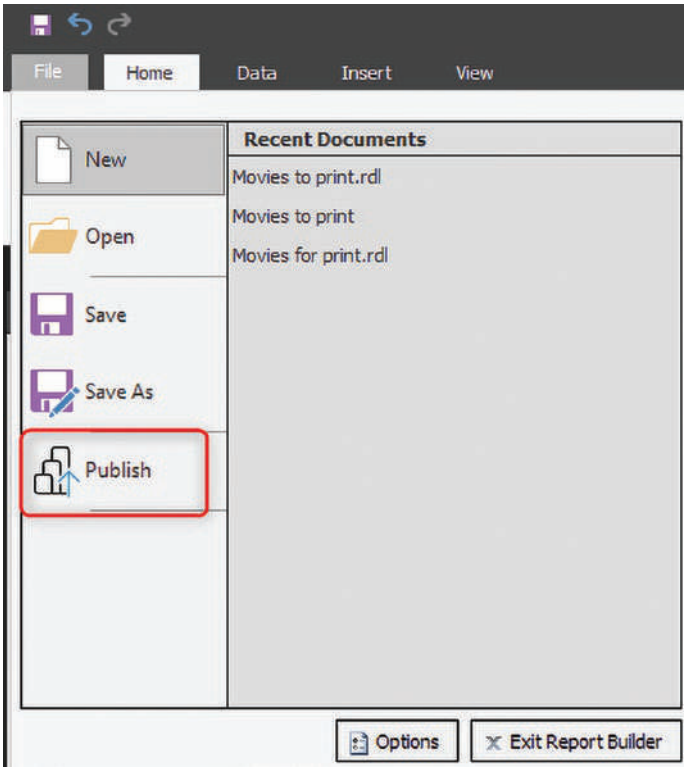
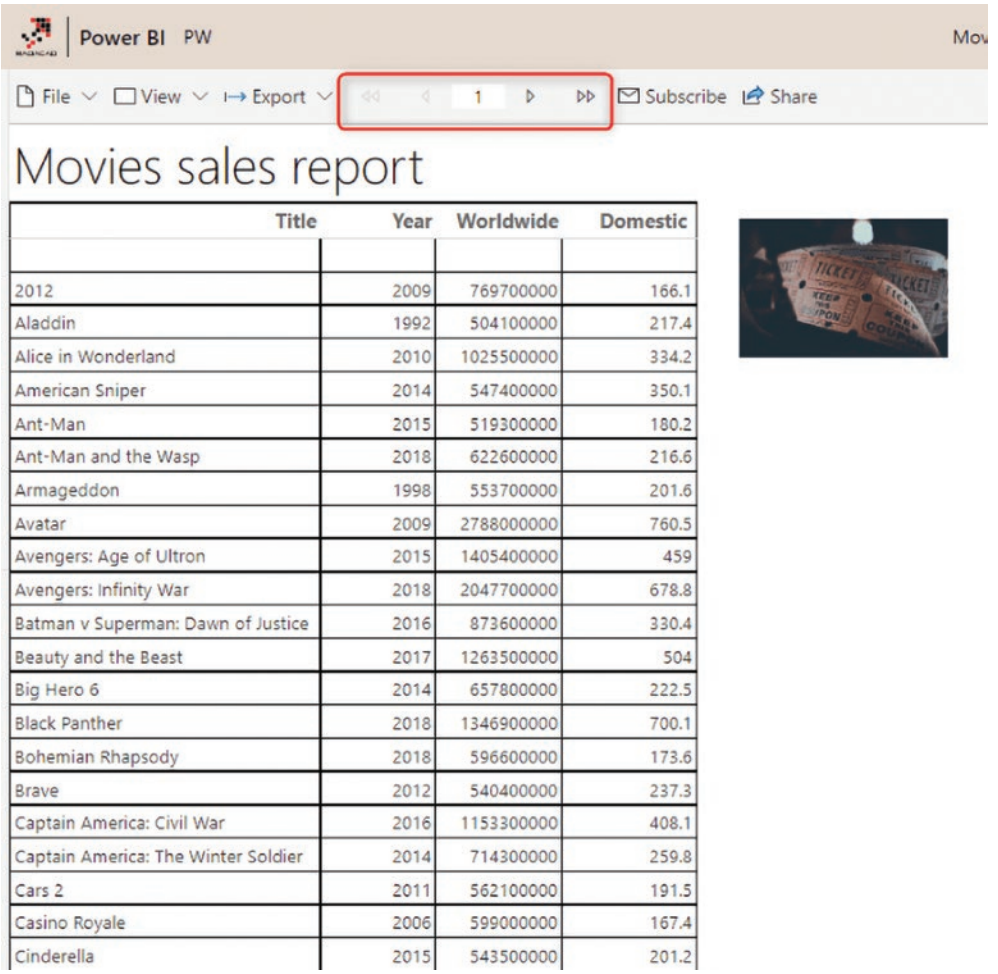


Figure 14-11. *Publishing a paginated report to the Power BI Service*

Publishing is only possible in Premium or PPU workspaces. Figure 14-12 shows a view of a paginated report.



Title	Year	Worldwide	Domestic
2012	2009	769700000	166.1
Aladdin	1992	504100000	217.4
Alice in Wonderland	2010	1025500000	334.2
American Sniper	2014	547400000	350.1
Ant-Man	2015	519300000	180.2
Ant-Man and the Wasp	2018	622600000	216.6
Armageddon	1998	553700000	201.6
Avatar	2009	2788000000	760.5
Avengers: Age of Ultron	2015	1405400000	459
Avengers: Infinity War	2018	2047700000	678.8
Batman v Superman: Dawn of Justice	2016	873600000	330.4
Beauty and the Beast	2017	1263500000	504
Big Hero 6	2014	657800000	222.5
Black Panther	2018	1346900000	700.1
Bohemian Rhapsody	2018	596600000	173.6
Brave	2012	540400000	237.3
Captain America: Civil War	2016	1153300000	408.1
Captain America: The Winter Soldier	2014	714300000	259.8
Cars 2	2011	562100000	191.5
Casino Royale	2006	599000000	167.4
Cinderella	2015	543500000	201.2

Figure 14-12. *Power BI paginated report in the Power BI Service*

The learning curve for paginated reports is longer and steeper than for the Power BI Desktop. Some of the changes you can do only in Visual Studio. Many of the configurations are not as user-friendly as the Power BI Desktop. However, paginated reports are SSRS reports by nature, and SSRS is a very powerful visualization engine. There are books of over a thousand pages covering that technology.

Summary

Paginated reports are pixel-perfect, print-friendly reports. These reports are designed using Power BI Report Builder (and they can also be extended using Visual Studio). They are not usually as interactive as the normal Power BI reports, but they are perfect for printing. Before creating these reports, always think twice whether you actually need print-ready reports. Maybe accessible reports (such as mobile-friendly reports) would serve you better. If you are creating paginated reports, use shared datasets to access a good, layered architecture for your Power BI solution.

CHAPTER 15

Excel and Power BI Integration

Power BI and Excel are longtime friends of each other, not only because Power BI components come from add-ins introduced in Excel, but also because of how these two tools interact from the Power BI Service. This chapter is not about using Power Query or Power Pivot components in Excel. This chapter discusses the interaction between Excel and Power BI through the service.

Power BI and Excel integration through the service allows users to use Excel as their slicing and dicing tool while connected to a live Power BI dataset. On the other hand, you can pin a range of cells from an Excel document into the Power BI dashboard. Excel files also can be uploaded to the Workbook tab of the Power BI service. In this chapter, you learn the following ways that Excel and Power BI interact with each other through the Power BI Service:

- Analyze in Excel feature
- Publish to Power BI from Excel
- Import Excel into the Power BI Desktop

The Analyze in Excel Feature

Every company has users with good Excel experience. Excel users can use Excel to connect to the Power BI dataset and then use the Excel features such as PivotTable and PivotChart to slice and dice the data. The connection to the Power BI dataset is a live one, which means whenever users refresh the Excel file, they get the most up-to-date data from the Power BI service. Let's look at how the Analyze in Excel feature works in action.

The Analyze in Excel Feature from the Power BI Service

You can initiate the Analyze in Excel feature from a Power BI Service report or a dataset. Log in to the Power BI Service and click the More options of a dataset (or a report). Choose the Analyze in Excel option, as shown in Figure 15-1.

R. Rad, *Pro Power BI Architecture*, https://doi.org/10.1007/978-1-4842-9538-0_15

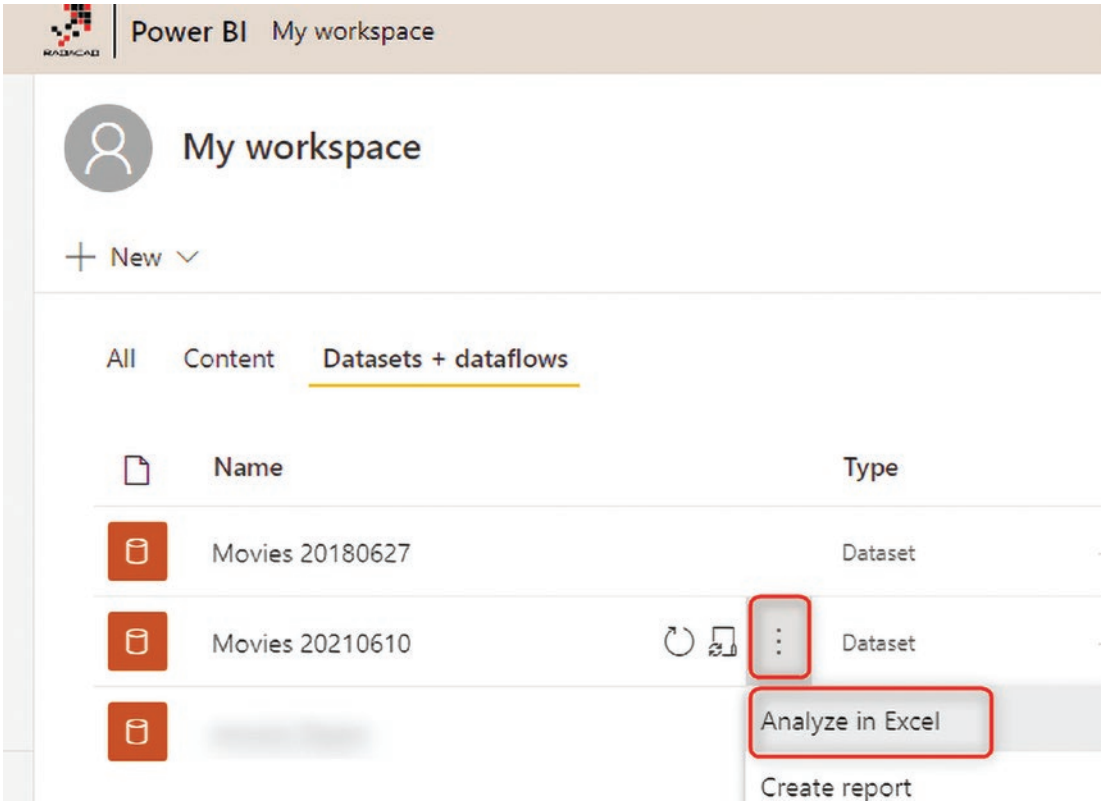


Figure 15-1. *Analyze in Excel from Power BI report or dataset*

Power BI will generate an Excel file in OneDrive and create a connection from that Excel file to the Power BI dataset. As shown Figure 15-2, you get a notification when the Excel file is ready. (If you don't have OneDrive for Business in your tenant, then clicking Analyze in Excel will download the workbook to your local computer.)

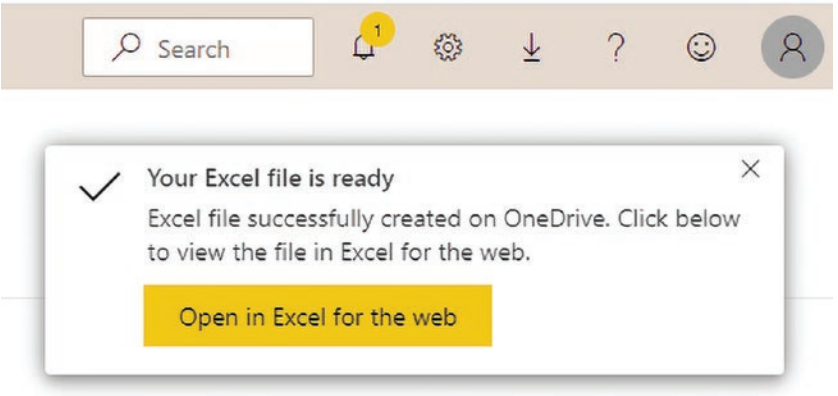


Figure 15-2. *Opening Analyze in Excel*

This file, by default, is opened in Excel online (the web version). However, you can download the file and store it locally. When you open the file, you get a warning about the data coming from outside of this workbook (Power BI Dataset), as shown in Figure 15-3.

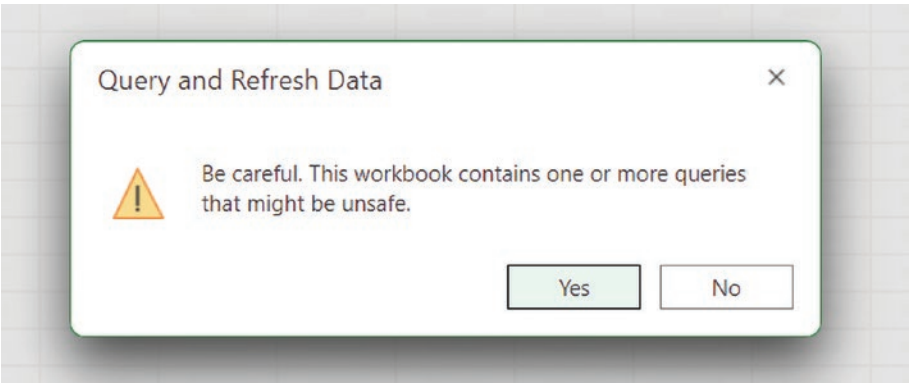


Figure 15-3. *Data is coming from outside of the Excel workbook*

Once you click Yes, you will see an Excel workbook with a PivotTable, which is the data sourced from the Power BI dataset in the service (see Figure 15-4). Analyze in Excel will use the same Power BI account username and password that you used to access the report (because the Power BI account is an Office 365 account).

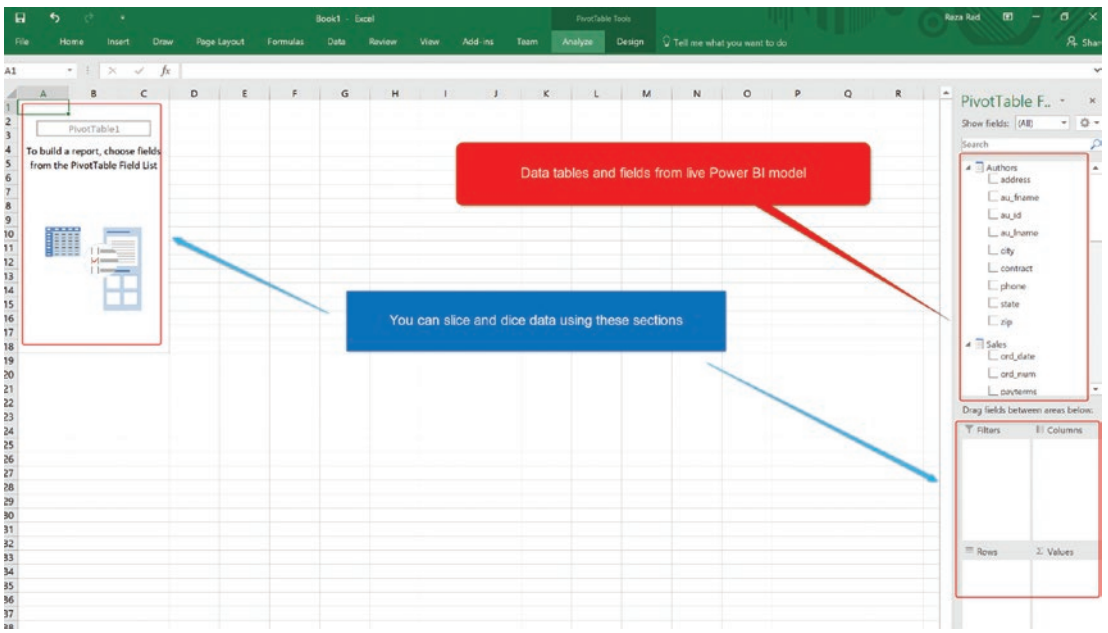


Figure 15-4. *PivotTable in Excel presenting Power BI dataset's data*

Drag the data fields into the slicing and dicing area (under the fields pane), and you will see a result in PivotTable. This result is fetched live from the Power BI dataset in the Power BI service. See Figure 15-5.

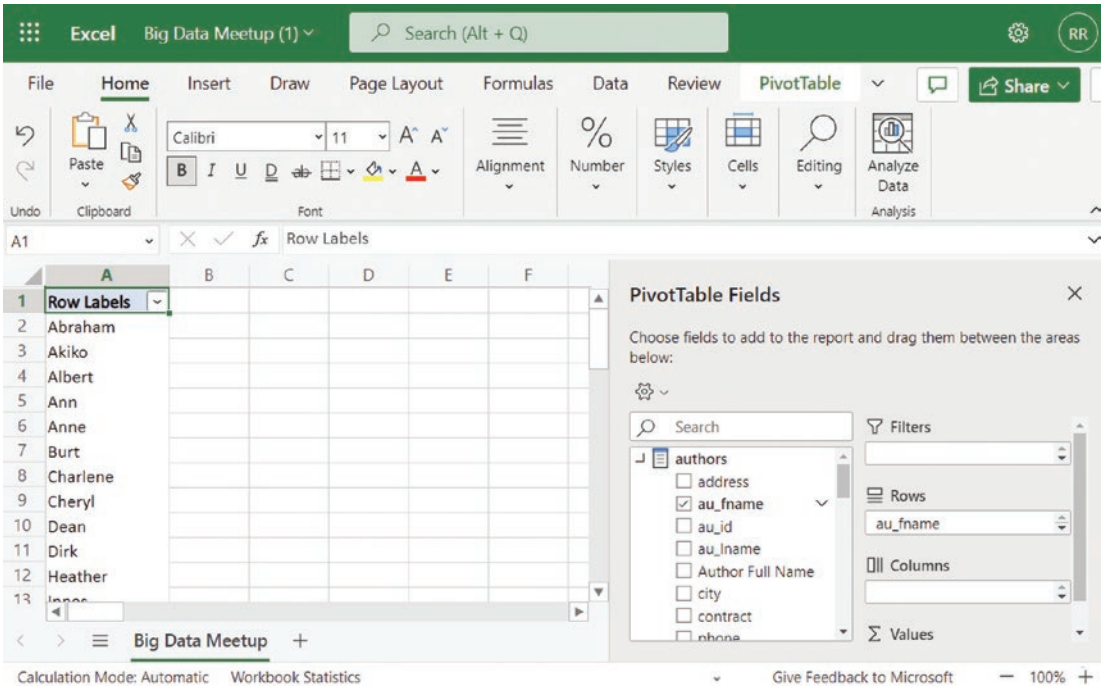


Figure 15-5. *Power BI data in tabular format in Excel*

Implicit Measures Don't Work in Excel

Implicit measures are a measure that Power BI creates automatically. Power BI automatically applies auto summarization on numeric fields (that haven't been part of a relationship). Behind the scenes, Power BI creates a measure for those fields; these measures are called implicit measures. These measures are marked with a small Sum or Sigma icon beside their name in the Power BI Desktop, as shown in Figure 15-6.

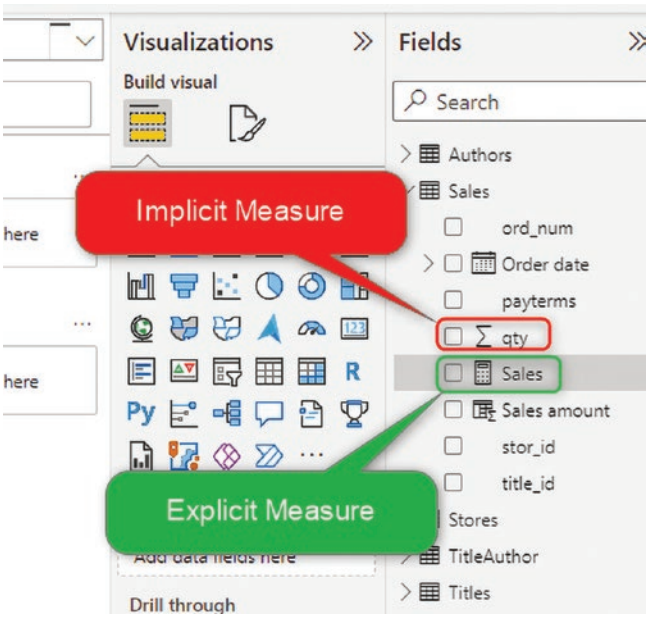


Figure 15-6. *Implicit vs. explicit measures in Power BI*

The implicit measure cannot be used in the Analyze in Excel feature; if you try dragging them to a PivotTable, you will see individual values instead of the aggregation. This limitation is illustrated in Figure 15-7.

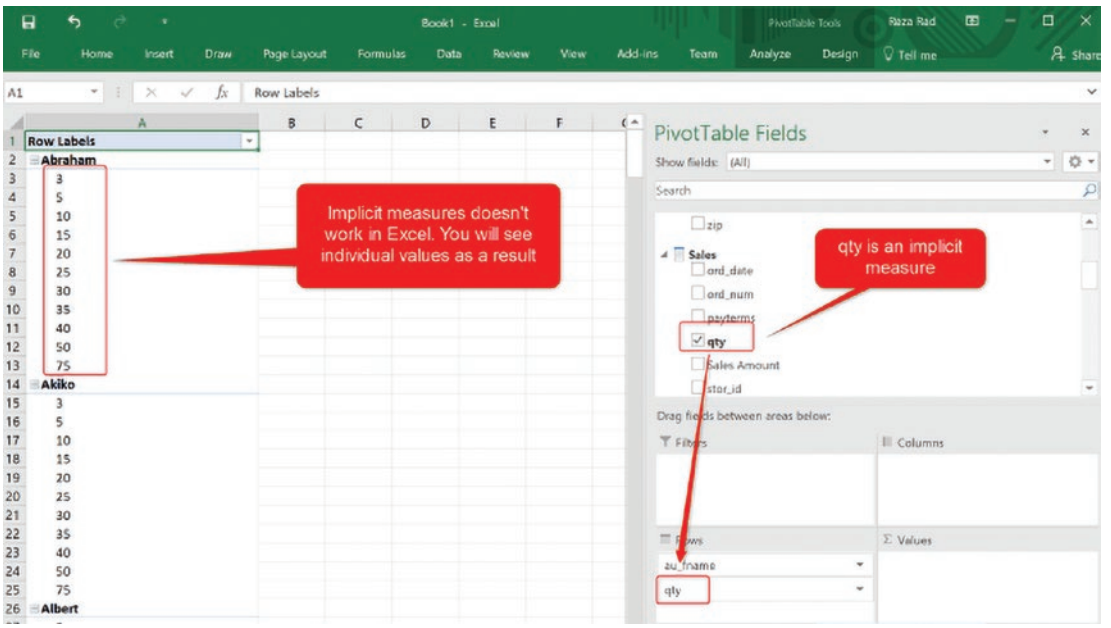


Figure 15-7. *Implicit measures aren't allowed in the Value section of the PivotTable*

However, if you create explicit measures (which are DAX measures created by you), similar to Figure 15-8, you can then use them in PivotTables as a normal measure and see the correct result, such as in Figure 15-9.

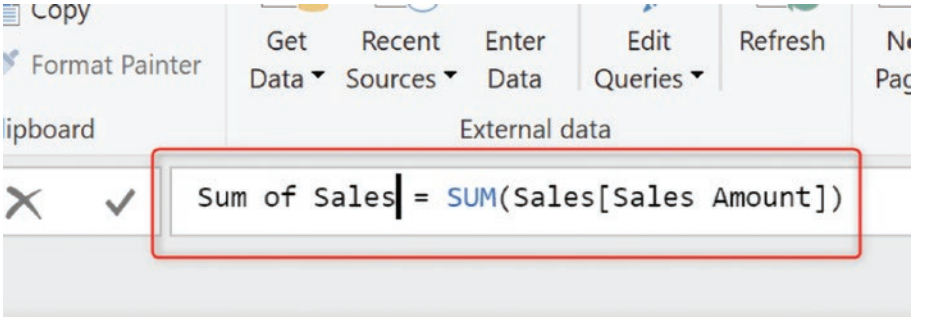


Figure 15-8. *An explicit measure*

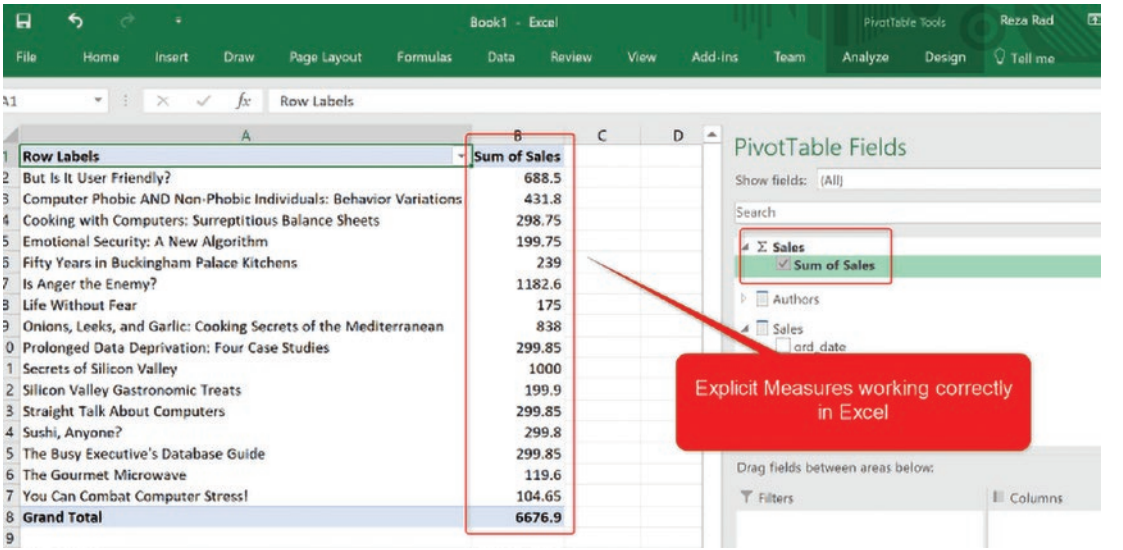


Figure 15-9. *Explicit measures work correctly in PivotTables*

If you consider that some users use Excel as their front-end tool to connect to Power BI models, you have to consider creating the Explicit measure.

Excel Is Connected Live to the Power BI Model in the Service

The wonderful thing about the Excel connection to the Power BI Service is that the connection is live. Using a live connection means Excel fetches the data directly from the dataset in the Power BI service. Any time you refresh the Excel file, you get the most up-to-date data from the service. This feature is completely different from the Export to Excel option. The Export to Excel option you see on visuals in the Power BI service only downloads data offline. However, Analyze in Excel is an online and live connection to the dataset.

You can check the connection properties in the Data tab, under the Connections, Properties section, as shown in Figure 15-10.

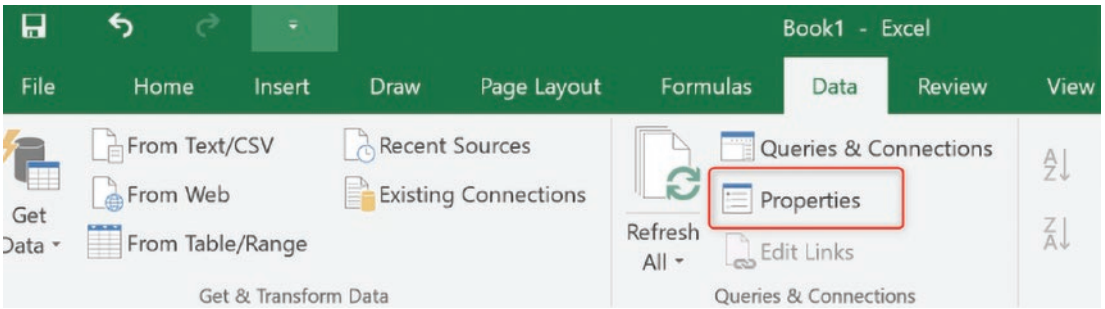


Figure 15-10. *Getting properties of the connection from Excel*

The Connection properties will have an Azure address with the ID of the dataset in the Power BI service. You can even use this connection in any other Excel file to connect to the same dataset. See Figure 15-11.

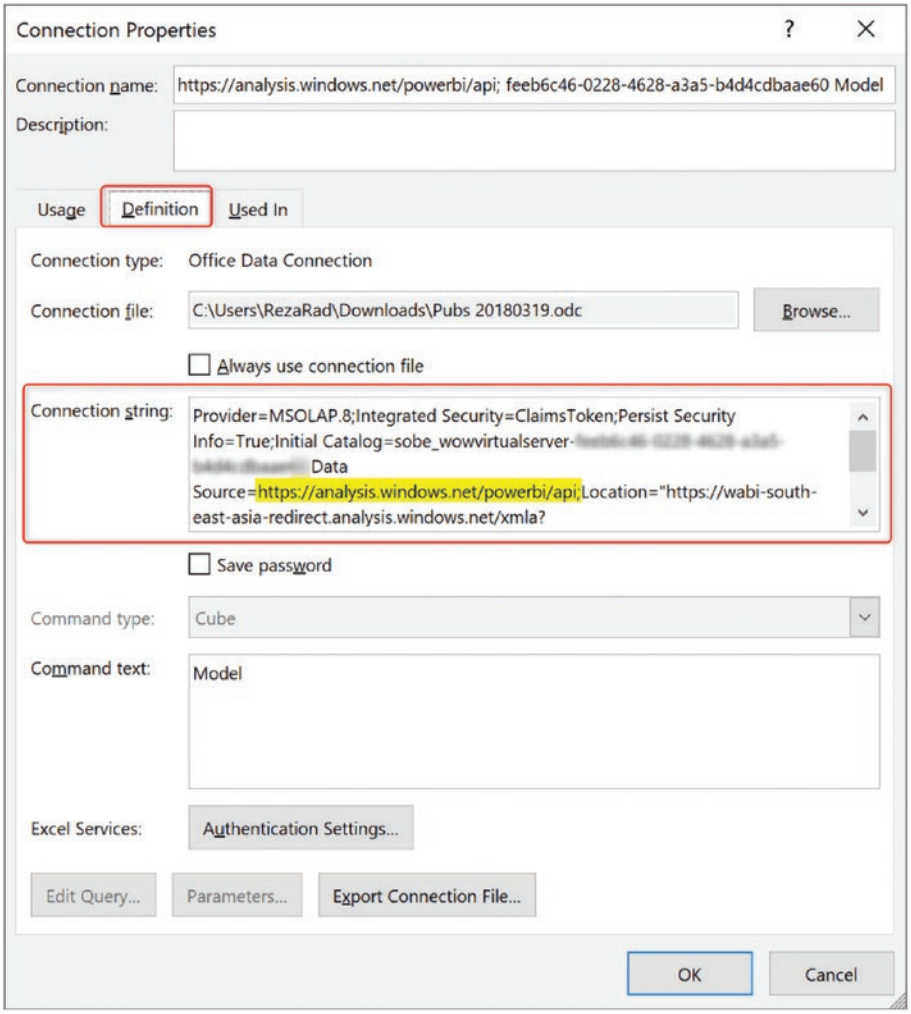


Figure 15-11. *Excel is connected to Azure Analysis Services (where the Power BI dataset is hosted)*

Getting Data from Power BI Dataset

Another very similar approach to Analyze in Excel is getting data from the Power BI dataset in Excel. If you have an Excel file and you want to do some data analysis on the data of a Power BI dataset, you can go to the Data tab and choose Get Data. Then choose From Power Platform, Dataset, as shown in Figure 15-12.

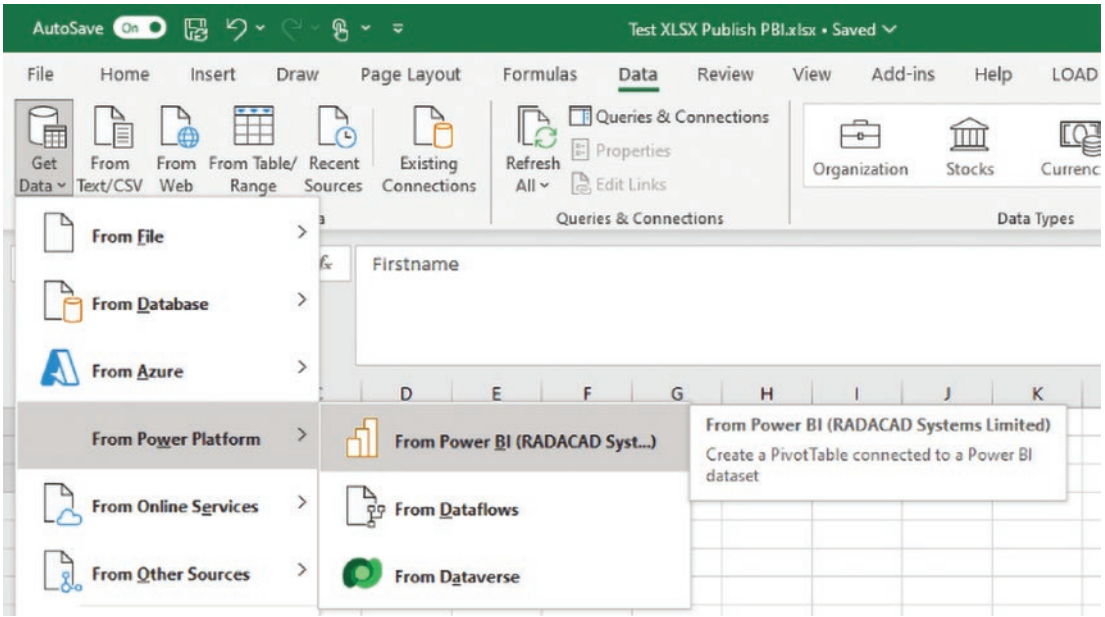


Figure 15-12. *Excel getting data from the Power BI dataset*

This will show you the list of Power BI datasets that you have access to, as shown in Figure 15-13) and you can select any of those.

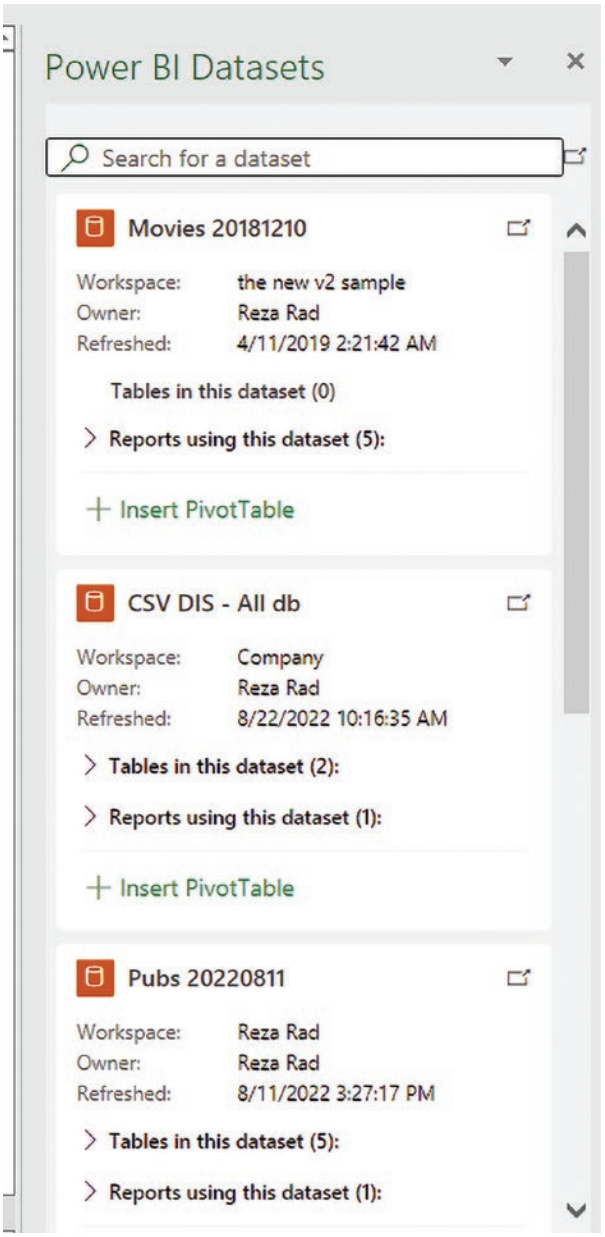


Figure 15-13. *List of Power BI datasets loaded in Excel*

Doing this will create a live connection to the Power BI dataset (similar to a thin report).

Getting Data from a Power BI Dataflow

Using Excel, you can also connect to a Power Platform dataflow and get data from it, as Figure 15-14 demonstrates.

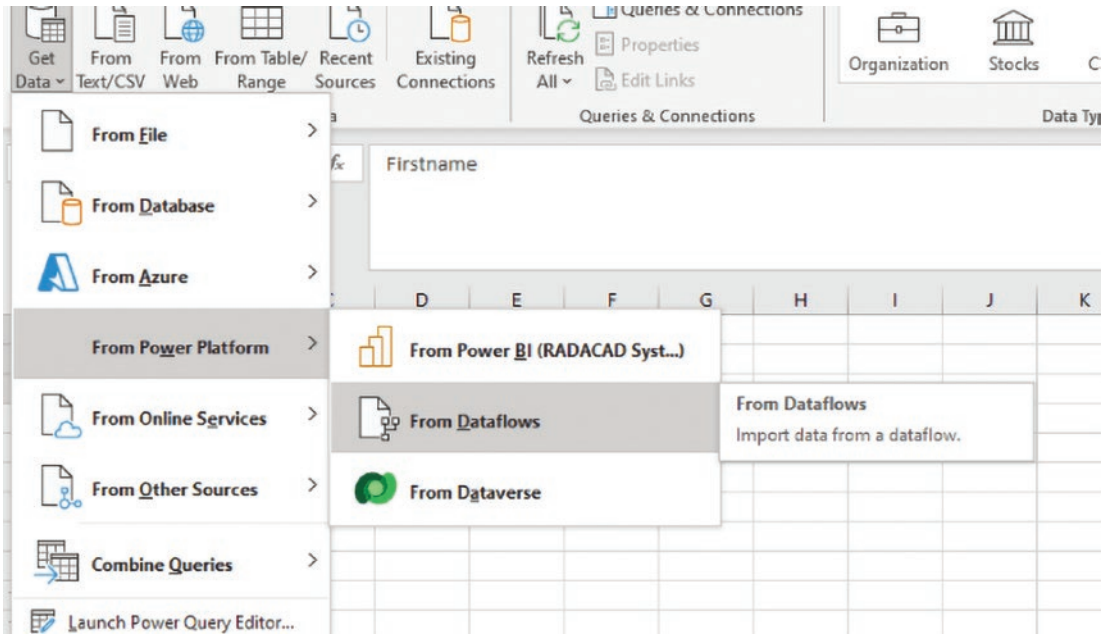


Figure 15-14. *Excel can get data from the Power Platform and Power BI dataflows*

The Navigator window will show you the Power Platform dataflows (under Environments) and Power BI dataflow (under workspaces), as shown in Figure 15-15.

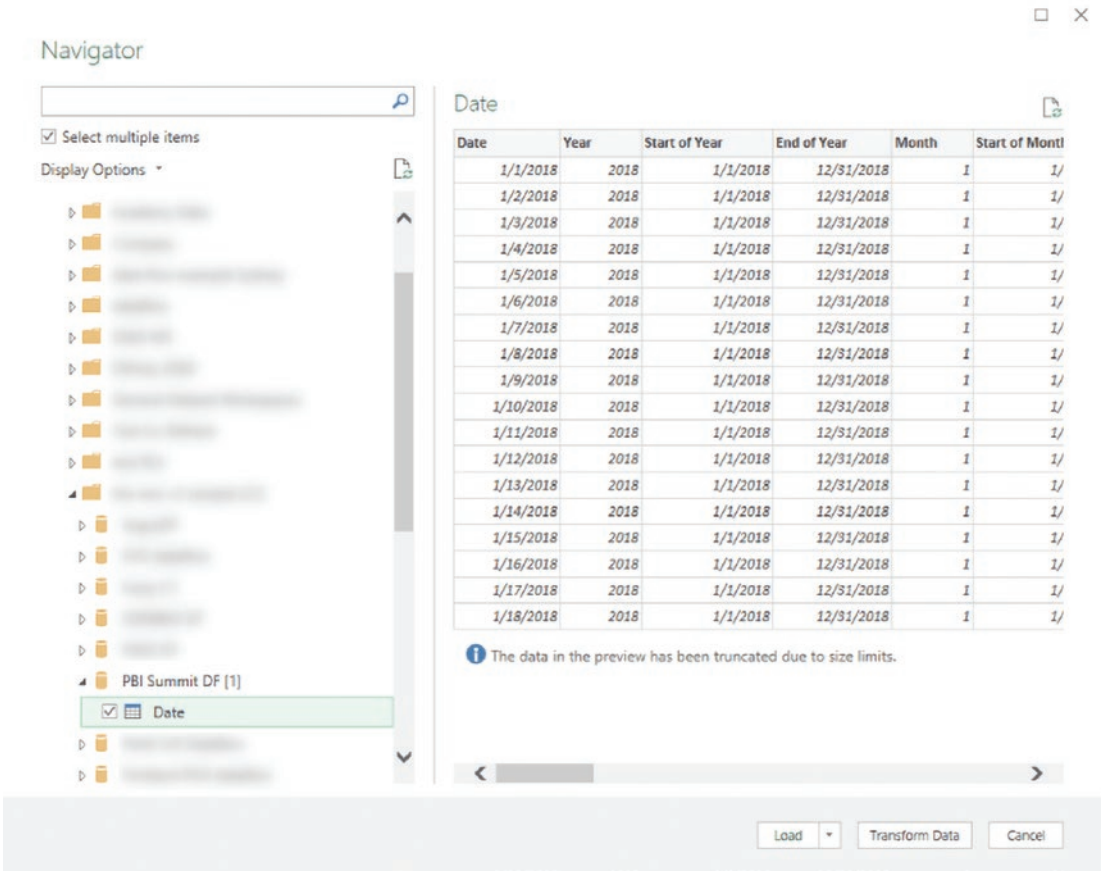


Figure 15-15. *Power Platform dataflows listed in the Excel connector for dataflows*

Why is Analyze in Excel Better than Using Export Data?

There are many reasons that Analyze in Excel is a better option than exporting data. The following sections explore some of those reasons.

Analyze in Excel is Live

As soon as you export data from a Power BI report or dataset, you create a snapshot of the data. If the report refreshes the next day, your exported data is no longer up-to-date. It is offline and outdated.

However, when you use Analyze in Excel, the data is fetched live from the Power BI dataset in the service. You can see that in the connection properties of Excel. You can check the connection properties in the Data tab, under the Connections, Properties section. The Connection properties will have an Azure address with the ID of the dataset in the Power BI service. You can even use this connection in any other Excel file to connect to the same dataset.

Export Data Is Limited to 150,000 Rows

Export data has a limitation on the number of rows. Analyze in Excel doesn't. You are connected to the model live. It looks like a live connection to the SSAS Tabular; you can do whatever you want with the data.

The export is limited to 150K rows if you export to Excel and 30K rows for CSV files.

Export Data Is Limited to Specific Visuals and Fields

When you use Export Data, you only do it from a specific visual. The fields in that visual (or related fields) are exported. When you use Analyze in Excel, you have access to all the tables, columns, and calculated fields and measures in Excel.

Analyze in Excel is the entire model: all the tables and calculations.

Export Data Is Not Secure

You might export data using an account that has access to everything and then share the exported data file with someone who should not have access to the data. Because the security and the data are decoupled when you use the export data, there is no security around it. You have to be careful to secure the exported data yourself. That said, Power BI has the system to apply sensitivity labels using other Azure services, which can help. Analyze in Excel uses Power BI account credentials.

However, when you use Analyze in Excel, you have to log in with your Power BI account. You can see the data only if you have access to it.

Analyze in Excel Supports Row-Level Security

Because users log in from Excel, when they use the Analyze in Excel feature, they will only see the part of the data they are allowed to see. That means if row-level security is enabled on the data, users will only see their part of the data, and nothing more (as long as they don't have Edit access on the dataset).

Analyze in Excel Uses Shared Datasets

When you use the Analyze in Excel feature, you use the shared Power BI dataset from the Excel frontend. Whenever the dataset refreshes, you get the new data. You get the updates if the dataset gets a new table or field.

As Figure 15-16 shows, Analyze in Excel works well with multi-layered architecture for Power BI development. The visualization layer can be Power BI reports, paginated reports, or even Excel.

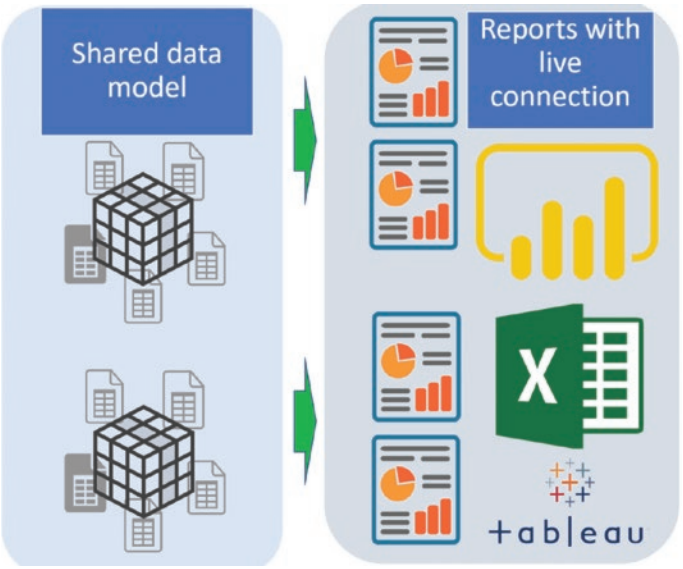


Figure 15-16. *Analyze in Excel works well with the Multi-Layered architecture for Power BI*

Publish to Power BI from Excel

Not only can Excel read data from the Power BI Service, but also it can publish data into Power BI. This is possible in the Excel 2016 editions or Office 365 editions of Excel. Once you click the Publish button from the File menu, you will see an option to publish into Power BI, as shown in Figure 15-17.

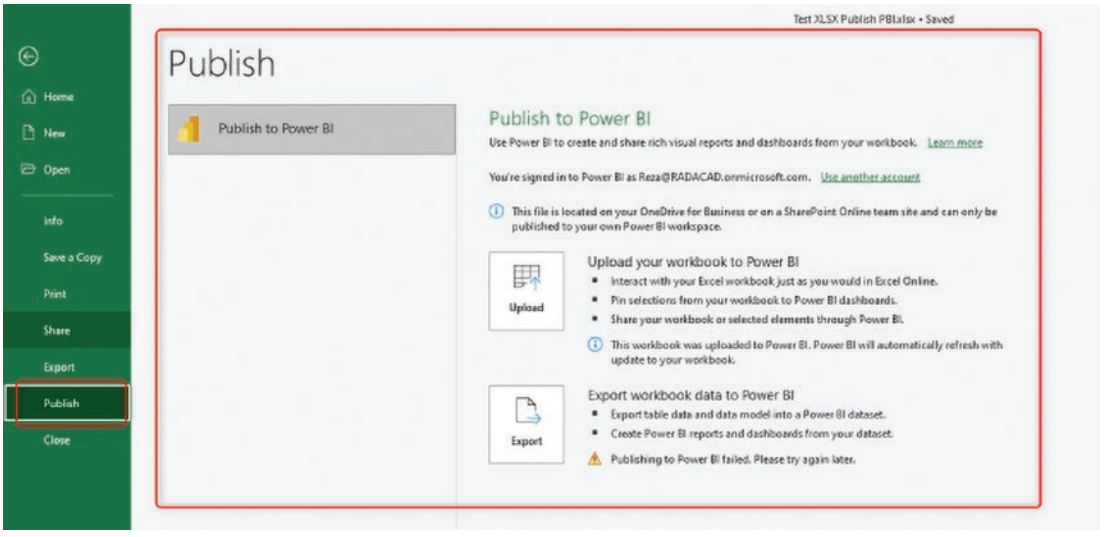


Figure 15-17. *The Publish to Power BI option from Excel*

Upload Your Workbook to Power BI

If you have an Excel workbook file, which you want to keep beside your Power BI objects (reports, dashboards, and datasets), this option is for you. You can upload the workbook into the Power BI workspace using this option. (See Figure 15-17 as an example.) Once the process is complete, the Excel workbook will be accessible from the Power BI workspace and can be opened and edited in Excel online. See Figure 15-18.

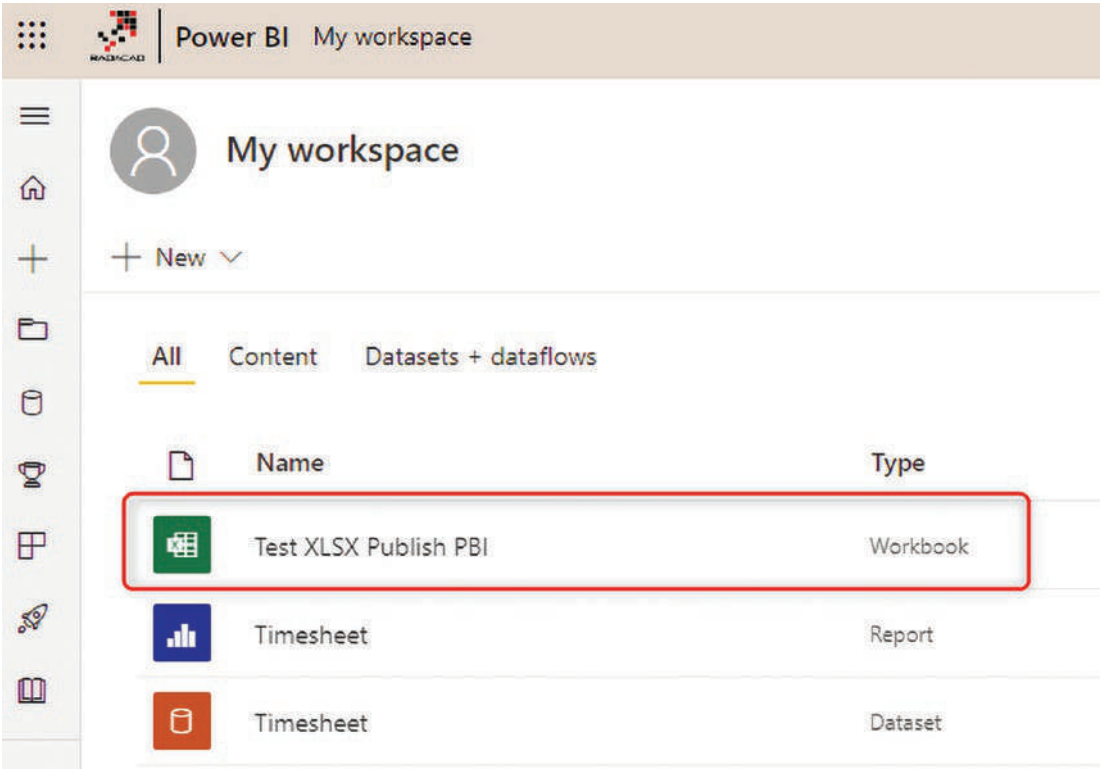


Figure 15-18. *Uploading an Excel workbook into the Power BI workspace*

The Excel workbook, as mentioned, can be opened and edited or viewed online, as shown in Figure 15-19.

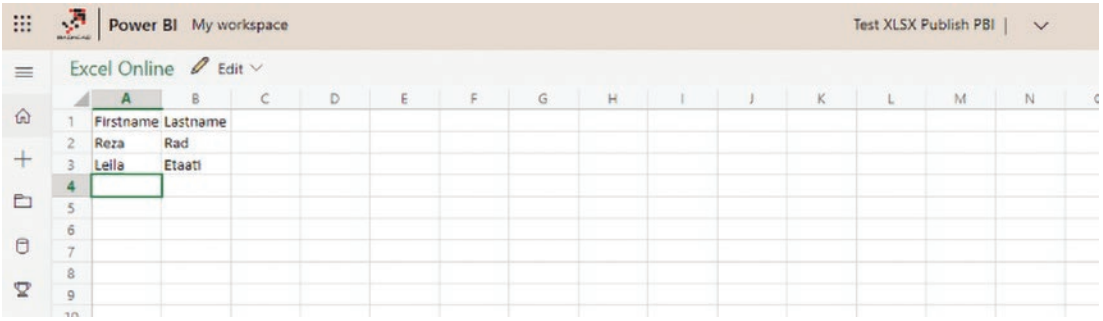


Figure 15-19. *Opening an Excel workbook from the Power BI workspace*

Export Workbook Data to Power BI

Here comes the exciting feature; not only can you upload the Excel file as a workbook, but you can also export it as a Power BI dataset. To do this, you must have data in the Excel file as tables. The table data will then be exported into a Power BI dataset from Excel (and it can be updated by updating the Excel file). See Figure 15-20.

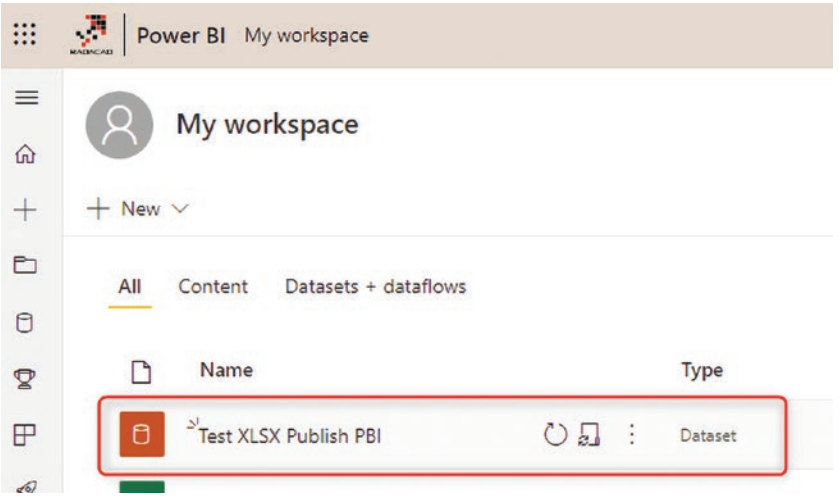


Figure 15-20. *Exporting workbook data into Power BI*

After these publish options, you will see a short message showing the publish process to the Power BI Service, as shown in Figure 15-21.

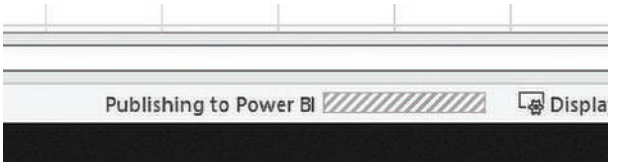


Figure 15-21. *The progress of publishing to Power BI from Excel*

These two options might not be very commonly used, but they certainly bring an interesting aspect to the integration between Excel and Power BI.

Import Excel into the Power BI Desktop

I cannot discuss the Excel and Power BI integration options without mentioning the most important developer option. You can import an Excel model, which includes a PowerPivot model, into Power BI. The Import into Power BI doesn't just import the data from Excel, it imports the entire model, including all the tables, relationships, and calculations, into a Power BI report. This scenario is a very handy option when you have an existing PowerPivot model in Excel, as shown in Figure 15-22.

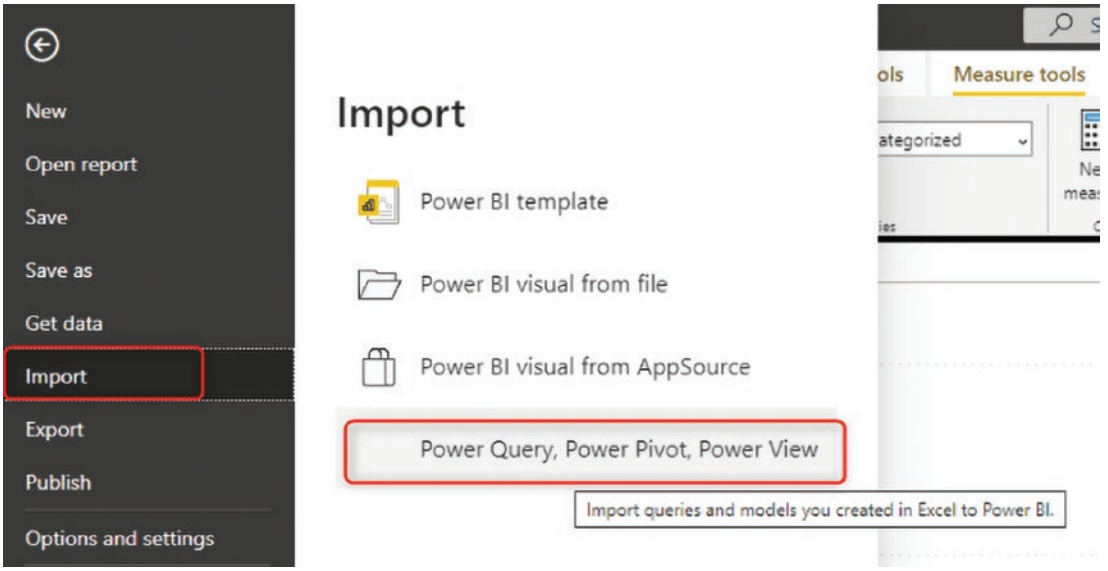


Figure 15-22. *Import Power Query, Power Pivot, and Power View into the Power BI Desktop*

Importing an Excel workbook is a very easy and straightforward process. The only important note in this process is that your Excel file should not be password-protected.

Summary

In this chapter, you learned that Excel and Power BI can work together. Each of the modes of integration will give you some features. Analyze in Excel allows you to slice and dice the data model of the Power BI Service easily from Excel through PivotTables and PivotCharts. Excel can publish the entire workbook or the data part as a dataset into Power BI. Both of the features will be well received by any business users who are good with Excel as a frontend tool. Last but not least, you can import an entire Excel PowerPivot model with all tables, relationships, and calculations into a Power BI report instead of having to recreate it.