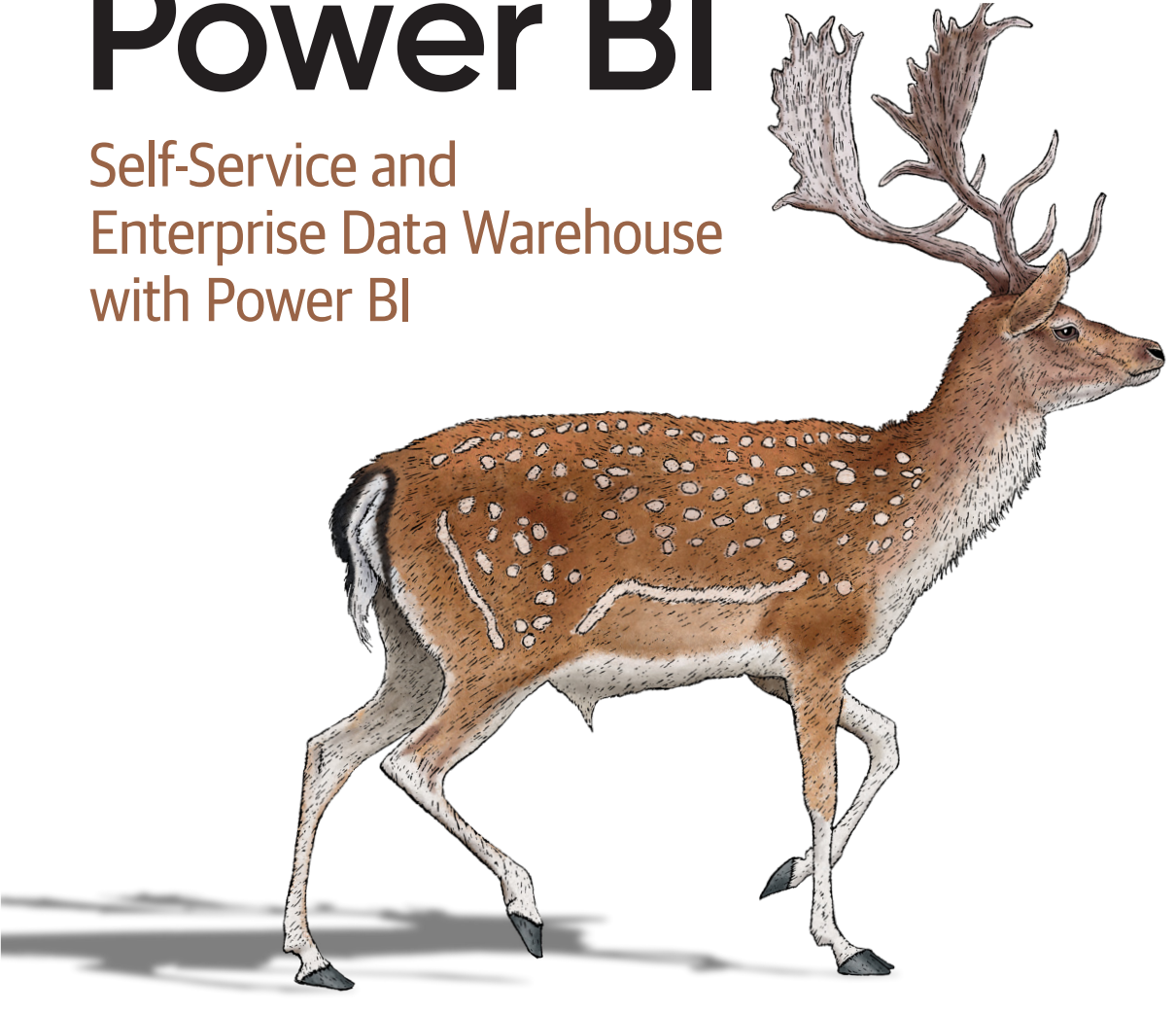
O'REILLY®

Data Modeling with Microsoft Power BI

Self-Service and Enterprise Data Warehouse with Power BI



Markus Ehrenmueller-Jensen

O'REILLY®

Data Modeling with Microsoft Power BI

Data modeling is the single most overlooked feature in Power BI Desktop, yet it's what sets Power BI apart from other tools on the market. This practical book serves as your fast-forward button for data modeling with Power BI, Analysis Services tabular, and SQL databases. It serves as a starting point for data modeling, as well as a handy refresher.

Author Markus Ehrenmueller-Jensen, founder of Savory Data, shows you the basic concepts of Power BI's semantic model with hands-on examples in DAX, Power Query, and T-SQL.

You'll learn how to:

- **Normalize and denormalize**
- **Apply best practices for calculations, flags, and indicators, time and date, role-playing dimensions, and slowly changing dimensions**
- **Solve challenges such as binning, budget, localized models, composite models, and key value tables**
- **Discover and tackle performance issues via the data model**
- **Work with tables, relations, set operations, normal forms, dimensional modeling, and ETL**

"This book is a comprehensive tutorial that covers the subject in language that is easy to understand yet thorough, concise, and accurate. Markus's mastery of the art and science of data modeling provides value for any data professional working with Power BI."

—**Paul Turley**
Microsoft Data Platform MVP

Markus Ehrenmueller-Jensen, founder of Savory Data, has worked as a project leader, trainer, and consultant for data engineering, business intelligence, and data science since 1994. He's a software engineer and professor at HTL Leonding (technical college), teaching databases and project engineering. He has several Microsoft certifications and is a Microsoft Data Platform MVP.

DATA

US $69.99 CAN $87.99
ISBN: 978-1-098-14855-3

linkedin.com/company/oreilly-media
youtube.com/oreillymedia

Praise for *Data Modeling with Microsoft Power BI*

This excellent book will tell you why to "star schema all the things." It explains in great depth why data modeling is important and provides ample examples. Reading this book will make your life as a Power BI developer easier.

—Koen Verbeeck, Senior BI Architect, Star Schema Aficionado, Microsoft Data Platform MVP

Markus's book *Data Modeling with Microsoft Power BI* provides a very good introduction to data modeling principles for an effective data model in Power BI as well as in Excel's data model, Power Pivot.

The book is written in an explanatory way, using clear language that can be read by both novices and experts alike. It is very accessible and a must-have for those who want to learn more about data modeling.

I especially like Markus's division of the book into a kind of matrix, where each of the five main sections is divided into four chapters dealing with the same four subtopics—understanding the data model, building a data model, examples from the real world, and performance optimization—which become more and more complex throughout the book so you gradually get more and more insight into the many facets of data modeling.

—Jørgen Koch, Innovate, Microsoft Power BI and Office Enthusiast (SME), author and Microsoft Certified Trainer

Creating a fancy report and tinkering with DAX or M-Code in the times of AI is not hard—creating a high-performing model that will work is. Markus is a "Model Wizard" and has fixed more of my work than I would like to admit. This hands-on guide will give you a shot at mastering the model-building challenges ahead of you.

Although Power BI has evolved in leaps and bounds, the challenge of building a solid and performing model has not. Markus has seen endless environments and setups. I am a witness to his magic; he helped me fix broken models just by taking a quick look. With his new book he shares his knowledge with all of us and with his hands-on approach, he will guide you to becoming a Model Wizard yourself.

—Joel Ruh, Digital Transformation Lead at SkyFrame

Data Modeling with Microsoft Power BI is a must-read for anyone looking to master this powerful tool. Markus Ehrenmueller-Jensen has created an easy-to-read and fun guide that will help you unlock the full potential of Power BI.

—Carola Seyr, Business Intelligence Team Lead at Porsche Holding Salzburg

Data Modeling with Microsoft Power BI

Self-Service and Enterprise Data Warehouses with Power BI

Markus Ehrenmueller-Jensen

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Data Modeling with Microsoft Power BI
by Markus Ehrenmueller-Jensen

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Acquisitions Editor: Michelle Smith
Development Editor: Shira Evans
Production Editor: Katherine Tozer
Copyeditor: Liz Wheeler
Proofreader: M & R Consultants Corporation
Indexer: WordCo Indexing Services, Inc.
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Kate Dullea

June 2024: First Edition

Revision History for the First Edition
2024-06-11: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781098148553* for release details.

978-1-098-14855-3

[LSI]

I want to dedicate this book to the best thing that ever happened to me: my lovely children Clara (Alex) and Victor.

Table of Contents

Part II. Data Modeling in Power BI

Part IV. Data Modeling for Power BI with the Help of Power Query

Foreword

No topic in the data industry is more debated than data modeling. It is the source of memes, T-shirts, and endless debates at conferences, and its demise has been predicted for years.

Yet here we are with a new book about data modeling. And it is sorely needed; data modeling is a foundational skill with many applications. It makes tough problems easier to solve, data easier to work with, and Data Analysis Expressions (DAX) easier to write. It improves performance and eventually saves costs. However, you must be willing to put in the work; it's necessary to start thinking about data modeling early in the process, whether you're designing a data warehouse, lakehouse, or semantic model in Power BI.

The data model is the cornerstone of your project. I have learned this from working with customers on all variations of analysis services over the years (from Power Pivot to SSAS and Power BI). With proper data modeling, you won't have to resort to as many DAX gymnastics. A good data model simplifies your calculations.

I've known Markus for many years and always enjoy his sessions at conferences. He explains tough topics in a simple manner, and this book is no exception.

In *Data Modeling with Microsoft Power BI*, Markus explores the many facets and long history of data modeling (who doesn't have the Kimball data warehousing book on the shelf?): how we need to think about data, and how we can translate requirements into entities and attributes. Markus does a great job applying these theoretical practices to real life.

How do these data modeling practices help you with everyday Power BI and SQL challenges? Markus explains the basics of data modeling in Power BI by looking at tables, relationships, and analysis of data granularity. He then shows how to translate common requirements like role-playing dimensions, slowly changing dimensions, binning, and translations into SQL, DAX, M, or in the model itself so you can use them at any step of your project.

The lessons in this book are very valuable, helping you simplify your day-to-day work as a data engineer, and it all starts with the model.

— Kasper de Jonge
Principal Program Manager at Microsoft Fabric

Preface

Welcome to this journey into data modeling concepts and practical examples for Power BI, including DAX, Power Query and T-SQL. This book is your companion on your journey to gain a comprehensive understanding about the steps needed to make building reports in Power BI Desktop and Power BI Report Builder, and creating measures in DAX, easier.

Power BI supports a wide variety of data sources (covering databases from different vendors, like Microsoft, Oracle or Teradata; flat files, like CSV, text, or Excel; web services like an https link to a web page, etc.). The only way to get data into Power BI is through Power Query. It's best practice to add calculations as *(explicit) measures* in DAX (as opposed to *calculated columns* in DAX or as columns in Power Query or in the data source). Creating *calculated tables* in DAX should be an exception; depending on your skills and preferences, you will implement transformations to shape the data model either in Power Query (in the user interface or by writing code in the M language) or in the data source. For example, in the case of a relational data warehouse implemented in Microsoft's relational database engines, you might use T-SQL in the data warehouse, as laid out in Figure P-1.

The first part of this book, which is written in an agnostic way, introduces the necessary concepts in a general way: you can apply this to any analytical system. The second part of the book explains the properties of a data model in Power BI. The rest of this book addresses DAX, Power Query, and SQL.

The book is designed for you, the reader, to have an individual experience based on your knowledge. You may not know DAX, Power Query, *and* SQL, but you may have familiarity with one or two of them; you can pick and choose to fill in gaps in your knowledge. Maybe you need a refresher on the composition of a data model. Part I has you covered. Maybe you struggle with dealing with a bunch of Excel files from which you need to create reports? The part on Power Query will be your starting point. Maybe your task is to build a data warehouse to which other people connect

with Power BI Desktop? Then the part about SQL will present you with solutions to typical problems.

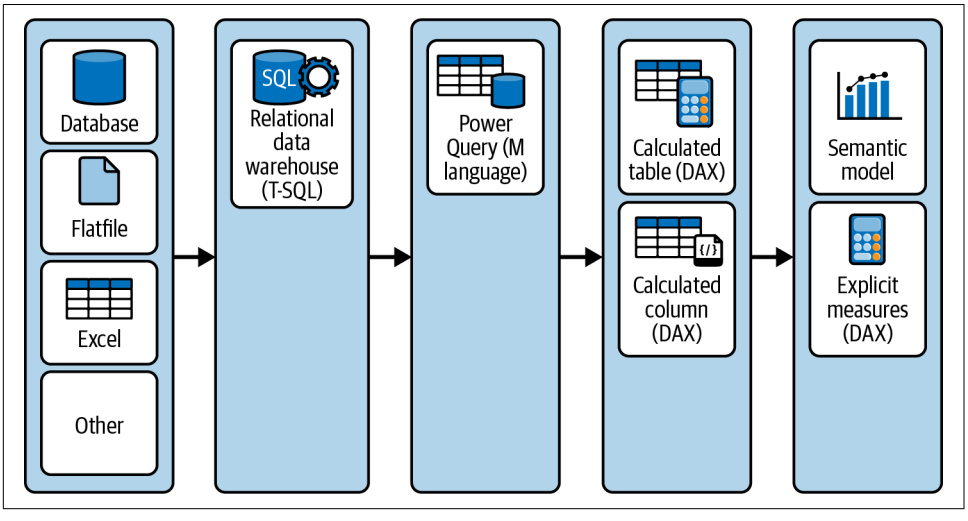


Figure P-1. Power BI data-shaping architecture

Data modeling is definitely the single most underestimated task when working with Power BI Desktop. It is a crucial part in your steps, from raw data to business intelligence and analytics. Decisions made during data modeling will influence how much detail your reports can show, how user-friendly the database or semantic model is for creating reports and analysis, and how easy it is to add more data and implement calculations on existing data. Wrong decisions at the start are very expensive to fix later, as changes to the data model will break existing reports. I speak from experience: I had to learn this the hard way—and I see also other people struggling with the repercussions almost every day in my work as a trainer and consultant. This book is your guide to getting data modeling right from the beginning.

You will learn that it is less important how complex the steps done in the "back end" (DAX, Power Query, or T-SQL) are, as long as the result is an easier-to-understand and easier-to-use data model for the user who creates reports and does analytics based on this data model: report authors, business analysts, data scientists, etc. These steps can be as simple as changing technical names (e.g., `CSTNM4711`) into user-friendly names (e.g., `Customer Name`) or as complicated as combining or splitting tables into a whole new structure. You will learn how to add calculations to the data model and enrich plain data with metadata (hierarchies, translations, etc.). This book is full of practical examples from challenges I faced over more than 25 years in the field. Keep in mind that the essential goal is to remove the burden from the report creator.

A common analogy for data transformation is a restaurant. As the restaurant's customer, you expect the dishes served attractively arranged on plates so you can enjoy the meal with only common tools (spoon, knife, fork, chopsticks, or maybe your fingers). To make this happen, the restaurant not only reserves significant space and resources in the kitchen but also owns expensive tools (convection oven, broiler, sous-vide cooker, blender, etc.) and employs formally trained and skilled people (cooks) to control those devices to transform the raw ingredients into the dishes. Think of the people using a data model to create reports and do analytics as restaurant guests: they will prefer to have all information presented in an easy-to-digest (pun intended) form, which they can consume with common tools (Power BI, Excel, etc.). Set yourself into the role of a cook who puts all her experience together to create a data model that invokes the appetite to consume it.

Are you a data cook? Read on!

Who Is This Book For?

Are you accessing data in Power BI Desktop and interacting with it via visuals? Did you gather the most important data into one Power BI Desktop file so others can build reports on it? Are you in charge of a data warehouse and want to make sure that the data model is optimized for usage in Power BI Desktop? If you answer "Yes" to any of these questions, then this book is for you.

The primary audience is the enthusiastic Power BI report creator who wants to apply best practices in the data model for performant reports and easy DAX calculations. The secondary audience is the IT Pro who wants to support report creators with a connect-and-go data source (a data model created in Power BI Desktop). You should be comfortable with creating reports in Power BI Desktop and have a basic understanding of at least DAX, Power Query/M, or SQL, so you can follow the code examples provided in this book.

Throughout the book, you'll not only learn about the data modeling options in Power BI but also about modeling options in other tools, so you can create a data model that is optimal for Power BI. This is covered in Part V.

Why is it worth it to read (and write!) a whole book about data modeling? The next section delivers the answer.

What Is Data Modeling?

The challenges of storing data in different logical formats are as old as data itself. Before electronic computers were invented, data was put into different physical files and structured in physical folders, filling the shelves of big cabinets or even whole rooms or basements. Optional indexes (alphabetically ordered small cards with important terms and a reference on which shelf in which folder and file the information can be found) allowed you to scan the data not only by its physical order but by different tags. This terminology remains, but data is now stored in files and folders on hard disks, with additional indexes speeding up reading access.

Different approaches were invented and discussed over history to make for easy storage (e.g., avoiding redundancy and standardization of physical data storage) and easy read-access (e.g., indexing and reintroducing of some redundancy to speed up access to data). The concept of relational databases (e.g., SQL Server and Azure SQL Database) goes back to the year 1970. Dimensional modeling is even older—but nevertheless still very useful today.

To master Power BI, you need to master data modeling, because Power BI is a model-oriented analytics tool (as opposed to some other tools on the market). The next section gives you an overview about what parts Power BI is composed of and where you define the data model.

What Is Power BI?

Power BI is not a single tool but a whole suite of tools that became widely available in 2015. In case you are new to Power BI, I will give you a brief overview of the different tools:

Power BI Desktop
: Power BI Desktop is the full client with which you can achieve a lot of tasks (see Figure P-2). You connect to data sources, clean and transform data (with Power Query), develop a data model (in the *Model view*), and create reports (in the *Report view*). This is a tool you will spend a lot of time with when re-creating the examples in this book. All examples and screenshots in this book covering Power BI data modeling, DAX, and Power Query are based on Power BI Desktop. Files created with Power BI Desktop have the extension `.pbix` or `.pbip`. Power BI Desktop is free of charge—you need no sort of license, and you can skip the sign-in step when the tool prompts you.

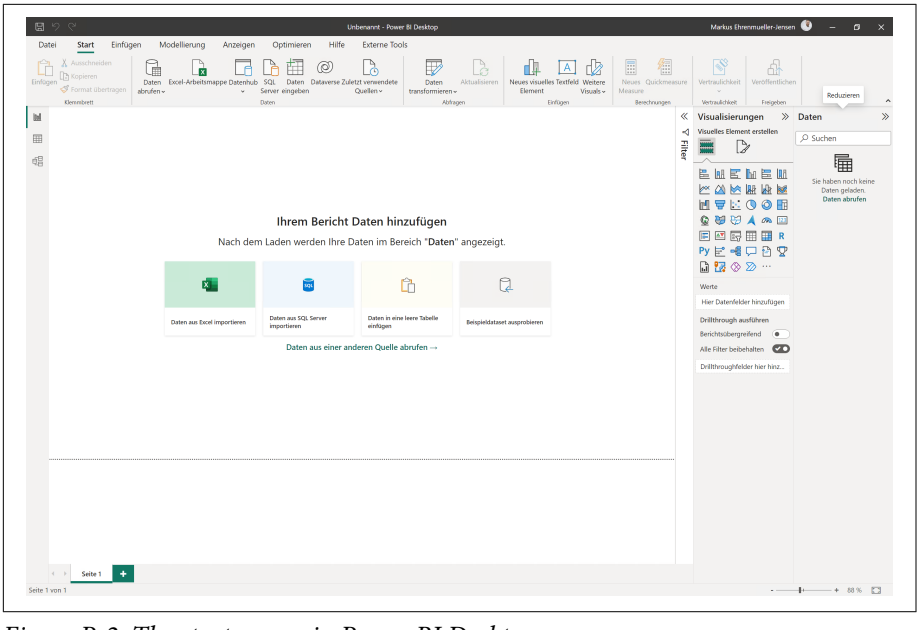


Figure P-2. The start screen in Power BI Desktop

The Power BI service

The Power BI service is where you host files you have created with Power BI Desktop so that others can consume a report or create a new report based on your data model. The Power BI service offers additional features (like Metric, Dashboard, Power BI App, Analyze in Excel, Export to Excel, etc.) not available in Power BI Desktop. The Power BI service is hosted in Microsoft's cloud data centers and is available with an internet browser of your choice via the Power BI service (see Figure P-3). You can also edit reports directly in the Power BI service, and Microsoft is working hard to enable editing of data models available in the Power BI service. At the time of writing, this lacks important features (like version control and collaborative editing)—so I'd currently recommend relying on Power BI Desktop instead. The Power BI service comes with different licenses.

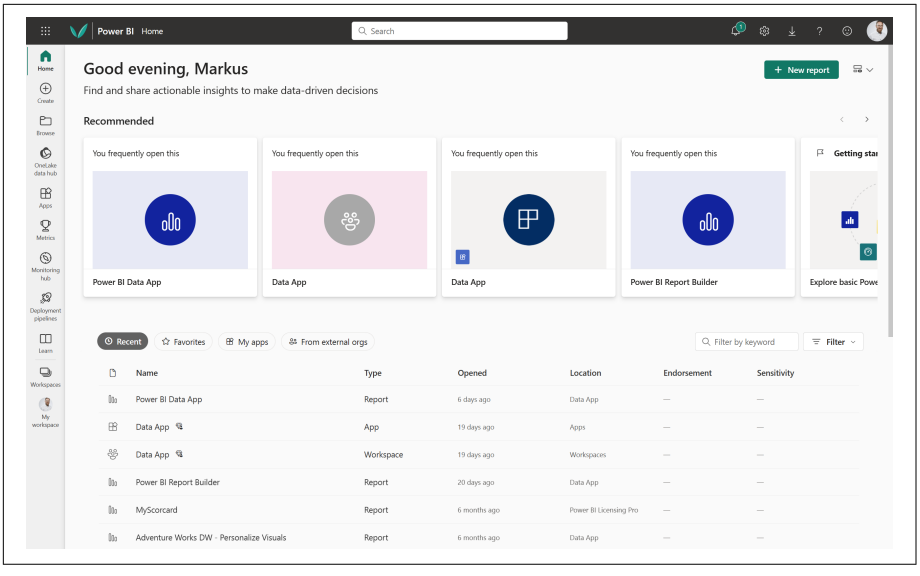


Figure P-3. The Power BI service says hello

Power BI Report Server

Power BI Report Server (see Figure P-4) is an alternative to the Power BI service, which you can install on your own premises. Power BI Report Server comes with a limited feature set, and new versions are released (only) three times a year. You need to use a matching version of Power BI Desktop (called *Power BI Desktop for Report Server*, which shows the month and year in the title of the application) when you intend to publish a Power BI Desktop report on a Power BI Report Server, as the monthly released version of Power BI Desktop might contain artifacts not compatible (yet) with the Power BI Server you are using.

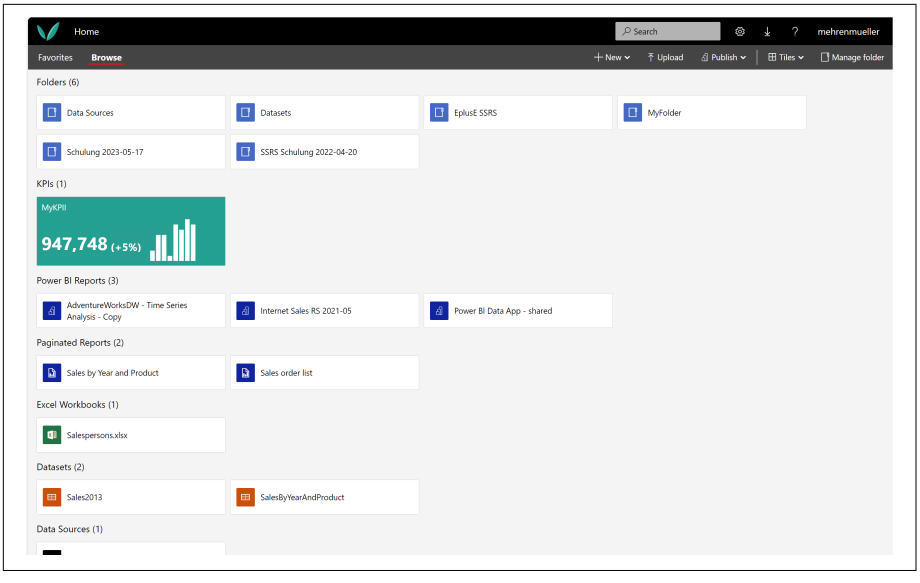


Figure P-4. The Power BI Server says hello

Power BI Report Builder

When you need to create pixel-perfect reports, with lists of data covering several pages, or you need to export in file formats not available for reports created with Power BI Desktop, then Power BI Report Builder is the tool for you (see Figure P-5). Power BI Report Builder is free of charge.

If you intend to publish paginated reports on a Power BI Report Server (as opposed to the Power BI service) you need to use *SQL Server Report Builder* instead of *Power BI Report Builder*.

You cannot create data models in (any version of) Report Builder.

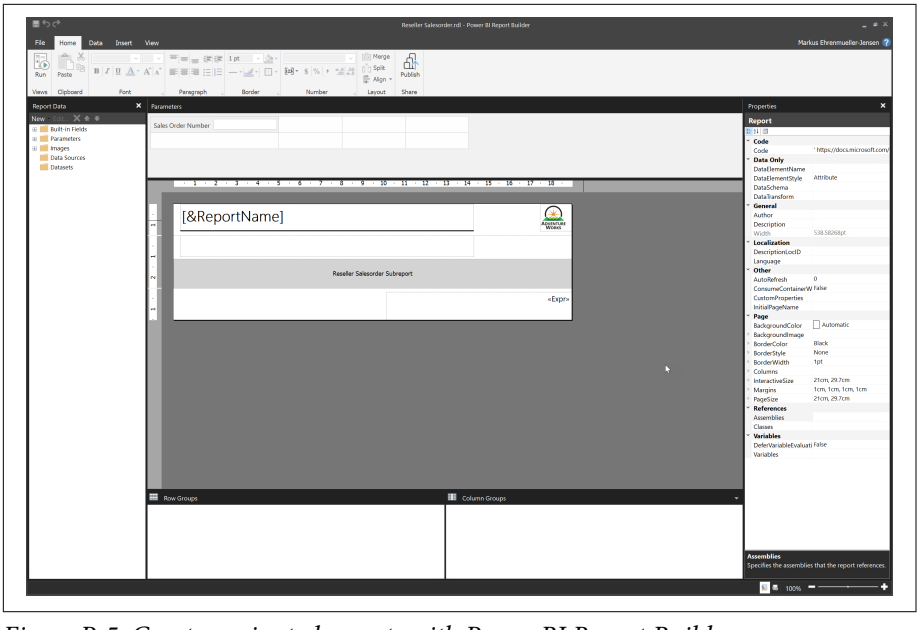


Figure P-5. Create paginated reports with Power BI Report Builder

The Analysis Services tabular model

You can think of Analysis Services tabular (which is available as Azure Analysis Services and as SQL Server Analysis Services tabular) as Power BI Desktop stripped from the report creation feature and reduced to the table view, the Model view, and Power Query. It shares the same storage engine (*VertiPaq*) and modeling capabilities as Power BI Desktop, and you can connect any reporting tool (including Power BI Desktop and Power BI Report Builder) to Analysis Services to create reports and analytics. You develop and deploy such a database with Visual Studio (see Figure P-6). Some of my customers use Azure Analysis Services instead of the Power BI service to host data because they can scale the costs for data storage at a more granular level (compared to the Power BI licensing costs); others use SQL Server Analysis Services because they cannot or do not want to host their data in the cloud.

Microsoft's vision is to make Power BI a super-set of Analysis Services tabular; with the announcement of Microsoft Fabric at the Build conference in May 2023, Microsoft made a big step toward it. Fabric will also allow scaling costs at a more granular level, compared to Power BI's licensing.

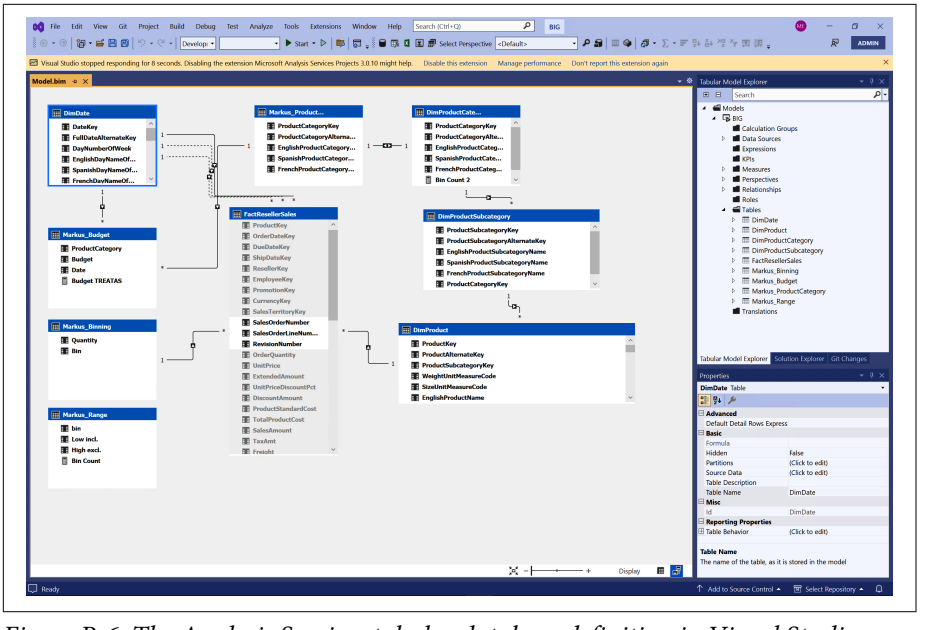


Figure P-6. The Analysis Services tabular database definition in Visual Studio

Let's talk about why you should care so much about the data model you build in Power BI Desktop.

What Is So Special About a Power BI Data Model?

Power BI is very versatile when it comes to the shapes of data models you can use (how you spread information between the tables or if you combine everything into a single table). But don't step into a trap here: the storage engine behind Power BI (called VertiPaq) and the language to define formulas for measure (DAX) are optimized for a certain shape called a *star schema*. This book is your guide to understand what a star schema is, why it is so important to take the time to transform the table(s) of your data source(s) into this shape and, most importantly, how to actually implement these steps.

In a nutshell, *star schema* is a term used for a data model where you have a fact table in the middle (forming the center of the star) surrounded by dimension tables (Figure P-7). I know, you need a lot of imagination to see such a star—but still, this concept is important and useful when building analytical systems in general, and especially when it comes to Power BI.

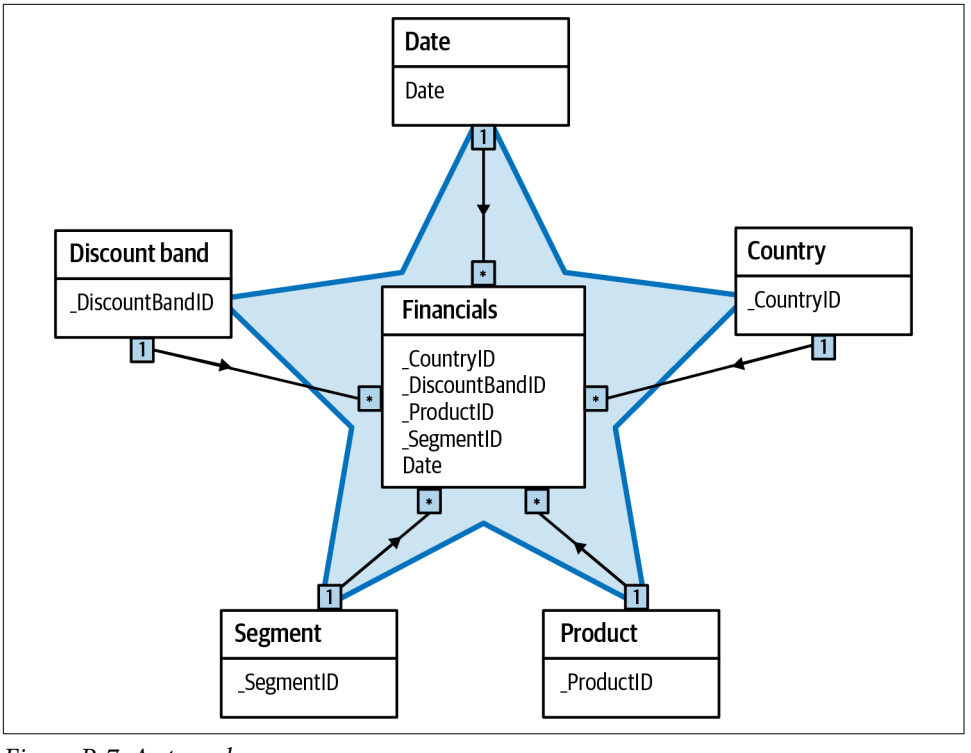


Figure P-7. A star schema

In a perfect world, the tables in the data source would already be in the shape of a star schema. If this is true for all your data sources, I consider you really lucky—and you can stop reading this book and ask for a refund. The mere mortals who are not as lucky have several options, which I discuss in this book. You can build a data warehouse layer (e.g., in form of a relational database or a data lakehouse). The SQL language will be your tool of choice. Or you can use Power BI's Power Query to reshape the tables. The third option is DAX, which is discussed in the next section.

What Is DAX?

Data Analysis Expressions (DAX) is a formula expression language used to create calculated columns, measures, calculated tables, and row-level security and to write queries. As you will learn, it's important to master DAX, as certain types of calculations can only be done in DAX (and not Power BI's data source or Power Query).

To reshape a data model into a star schema, you must move information from one table into another via calculated columns or create new tables as calculated tables. You'll also learn how to use DAX when you hit limits of Power BI's data modeling capabilities.

Overall, DAX isn't my first choice as a data-shaping tool if I can solve a problem with Power Query. Whenever possible, I push transformations into Power Query or, if available, into the data warehouse layer because it's best practice to push transformations as far "upstream" in the data processing pipeline as possible, to increase reusability of a transformation. A transformation in Power Query can be pushed into a Power BI dataflow to re-use it in different data models. Transformations in DAX are tightly applied to the data model they reside in. Another reason is that in Power Query and SQL, I can shape the data *before* I actually load it into Power BI, while in DAX I can only add calculated columns and tables on top of a model; any suboptimal part still occupies resources in my data model.

If you don't feel ready for Power Query (or the M language) or if you don't have a data warehouse at hand, then modeling the data in DAX is way better than not modeling your data at all. And remember, some problems can only be solved with a DAX measure.

What Is Power Query?

As the name suggests, Power Query is a tool to create queries you send against your data source(s). At the time of writing, Power Query has connectors to over 120 data sources, from flat files like CSV, XLS, or JSON to relational (SQL Server, Oracle, DB2, Teradata, etc.) and analytical databases (Analysis Services). Via a graphical user interface (GUI), you can clean and transform the original tables: renaming tables and columns to give them a more user-friendly name, removing unnecessary columns, and adding new tables and columns based on the existing information to shape a better data model.

All steps you apply in the GUI are added as lines of code to a Power Query script (called M for short), similar to the macro-recorder in Excel. Through the course of this book, you will learn how to use the GUI and when to edit the resulting query. Any time you refresh the content of a table, this script is executed and, therefore, all transformation steps are applied to the new data as well. That's why I ask people to hand me over "raw" CSV or Excel files (in cases when files are the preferred data source). There's no need to put effort into shaping the content manually every time they send me the new data if I can implement transformations once and re-apply them during the refresh of my data model.

Power Query (and therefore scripts in its mashup language, M) is not only available in Power BI Desktop but as Power BI dataflows and mashup tasks in Azure Data Factory as well. But maybe you want to push transformation further up the data stream (and if you ask me, you should). That's why there's so much content in this book about Azure SQL DB and T-SQL.

What Is SQL?

SQL is an acronym for *Structured Query Language*. In the context of Microsoft, "SQL" can refer to Microsoft's relational database engines offerings as well. Azure SQL DB is Microsoft's cloud offering for relational databases, which is available for on-premises' usage under the name Microsoft SQL Server Database Engine.

In this book, Azure SQL DB is used as an example of how to model your data outside of Power BI. I still consider this best practice: model your data in a database, and model your data as early as possible, or in other words, in a data warehouse layer. It is less important how you implement this layer: as a physically hosted relational database, as a data lakehouse, or in any other data store, like somewhere in Microsoft's Fabric.

Pushing the transformations as far as possible toward the data source makes it easier for people down the data stream to re-use it. If you wait until the last moment (that is, to determine the tool you use to show your data, e.g., Power BI Desktop, Power BI Report Builder, Excel or any other tool your end users might use to access the data and do their analytics) then you have accumulated a technical debt. The end users are then responsible for cleaning the data and bringing it into a useful shape. Due to lack of education, they might fail—leading to overcomplicated reports and possibly wrong numbers (you will learn about this problem in Chapter 5). On top of that, all the work is then redone or copied over into the next file—adding the problem of duplicated versions of these overcomplicated reports. Soon, you could end up in so-called Excel Hell, where nobody has a complete picture of the different versions of logic applied to the data. On the other hand, a data warehouse layer guarantees a single version of the truth.

Part V concentrates on solutions built with `SELECT` statements and SQL's procedural extension, T-SQL. I use the SQL dialect available in Azure SQL DB and SQL Server Database Engine. There is some chance that simple `SELECT` statements in this book will also run on other relational databases, or even *NO-SQL* databases. There is almost no chance that the procedural extensions (loops, functions, procedures, etc.) will work without any change on other database management systems. You'll have to find a way to migrate the code to your destination system if you aren't using Microsoft's SQL-based databases.

T-SQL as a language is quite stable in terms of how rarely new extensions are made or what parts are deprecated. Power BI is a different beast—new versions are released every month.

A New Release Every Few Weeks

The team behind Power BI and its tools and services at Microsoft is very busy delivering new versions of Power BI Desktop every month and rolling out changes in the cloud-based services weekly. This is a challenge for everybody: the UI changes, icons are redesigned, and buttons are moved to different places. This is also a challenge for every book project: some of the screenshots may be outdated when you read this book. Therefore, I include only portions of the screen, when sufficient. In many places, I also link to Microsoft's official documentation, which Microsoft and the community keeps up-to-date; you can double-check it if your Power BI Desktop looks different.

The general concepts on how to create an optimal data model for Power BI haven't changed much in the past, and therefore, there is hope that this knowledge is here to stay and will help you in the future as well. Read on to learn how I divide all the necessary knowledge and skills into digestible portions.

How to Read This Book

The book is organized into five parts—five books for the price of one:

Data Modeling 101 (Part I)
: The first part gently introduces all the theory and concepts and teaches you why data modeling matters. It covers all the content in a practical but tool-agnostic way. Look at this part as a "reader's digest" version of Ralph Kimball and Margy Ross's *The Data Warehouse Toolkit*, Third Edition (Wiley, 2013) and Christopher Adamson's *Star Schema: The Complete Reference* (McGraw Hill, 2010).

Data Modeling in Power BI (Part II)
: The second part covers data modeling features of Power BI Desktop, where I show you how to apply all the theory (from the first part) in concrete examples. I guide you around the menu and ribbon and into the properties of a data model.

Data Modeling for Power BI with the Help of DAX (Part III)
: This part shows how you can bring information from data source(s) into the necessary shape (discussed in Part II) with the help of DAX.

  The advantage of using DAX is that you need to learn it anyway when you want to master Power BI (there's no way around writing calculated measures in DAX for anything but very simple data models). Another is that changes in the formula will be calculated immediately after you press OK, without accessing the data source. The disadvantage of using DAX to reshape a data model is that the result of calculated columns and calculated tables will occupy disk space when you save the file and memory when you open it. Another disadvantage is that

your data model will contain both the original shape and the reshaped version of the data model (therefore occupying an unnecessary amount of space).

DAX's main purpose is to define measures and not to shape data, but some of the examples still provide additional insight about what you can achieve with DAX. Learning the full capabilities of the DAX language will also help you use complex DAX measures when you for some reason are not able to solve the problem in the model itself.

Data Modeling for Power BI with the Help of Power Query (Part IV)
: This part shows how to bring information from data source(s) into the necessary shape with the help of Power Query's UI and the M language. You'll see that you can come a long way with the UI alone before handling code in M.

  Unfortunately, M is very different from DAX. For one thing, in M you apply transformations based on the the result of a previous step; in DAX you add content on top of existing content of the data model. M is also case sensitive for keywords and the content of columns, and DAX is case insensitive. M is similar to F#, whereas DAX is similar to Excel formulas. The two can hardly be compared to each other. As you will see in this part, M as a language is much better suited to the task of data shaping than DAX.

  When the data I'm loading isn't already in the desired shape, Power Query and M are my tools of choice. Only the final result of transformations made in Power Query are loaded into the Power BI data model, avoiding unnecessary tables and columns. While working with Power Query, you need active access to the data source.

Data Modeling for Power BI with the Help of SQL (Part V)
: This part shows how you can bring the information from data source(s) into the necessary shape with the help of SQL and T-SQL. Despite trends to gather data in lakes, I still strongly believe that an enterprise organization needs a central place where someone takes care of the data; a place where natural (and dirty) data is cleaned and brought into the right shape. Whether this place is a physical database or "just" views on some data store is not so important. But it is important that you have such a (data warehouse) layer instead of putting the burden onto the end user and the tools they use.

  If you're eager to learn about relational databases or to make the life of Power BI users in your organization easier, then this part of the book is especially for you. In Microsoft Fabric you can access your data residing in a data lake.

Each part has four chapters, which cover the following topics:

Understanding a data model
: These chapters are about the basic terms and concepts. If you're new to data modeling, ensure you read and understand these chapters. If you already have some background in data modeling, you might quickly check these chapters to refresh your memory. Look out for the key takeaways at the end of each chapter.

Building a data model
: These are all about the meat-and-potatoes of data modeling. These chapters discuss problems and solutions to common problems. I'm convinced that you will face at least some of those on your journey of refining your data into usable information.

Real-world examples
: For these chapters, I address advanced challenges. For simple data models, the problems discussed here might not be an issue. You might enjoy a break on your first reading here and return when you find yourself facing one of the issues. These chapters dive deeper into data modeling, DAX, Power Query, and SQL and cover not-so-common features you might need to solve a certain problem. These chapters are aimed for the advanced data cook.

Performance tuning
: Every part closes with a chapter on performance tuning, which is most often the last step in the whole journey of data modeling. If you start by learning to follow all best practices pointed out in this book, you will have plenty to do before report performance will be an issue. For really large implementations (I'm talking about billions of rows of data), though, even following these best practices won't be enough. Then it's time to dive into the technical layer of data modeling and learn how to tune the available storage modes (Import, DirectQuery, live connection, and Direct Lake) to your advantage.

The whole book, and therefore also the chapters about performance tuning, concentrates on data modeling and not so much on how to write performant code in DAX, Power Query or SQL.

All the parts and topics are brought together in 20 chapters (see Figure P-8). They progress in complexity and difficulty, allowing you to read the book cover-to-cover. Or you could jump over DAX or ignore SQL, for example, if you intend to use Power Query to solve your challenges. Imagine you jump to Chapter 10 and discover that you're not sure why you should build a date table for your Power BI data model. You could jump to Chapter 6 to find out. You might also review the general concept of a date table (and why it's useful in general and for analytical systems built with other tools) in Chapter 2. Or maybe you decide against building the date table in DAX and

explore other solutions. You could find the same implementations done in Power Query in Chapter 14 and in SQL in Chapter 18. Figure P-9 illustrates this journey.

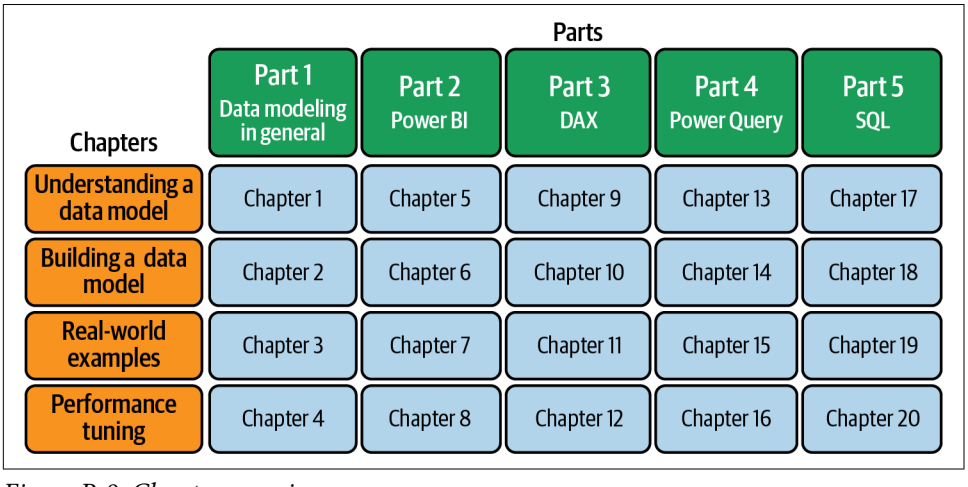


Figure P-8. Chapter overview

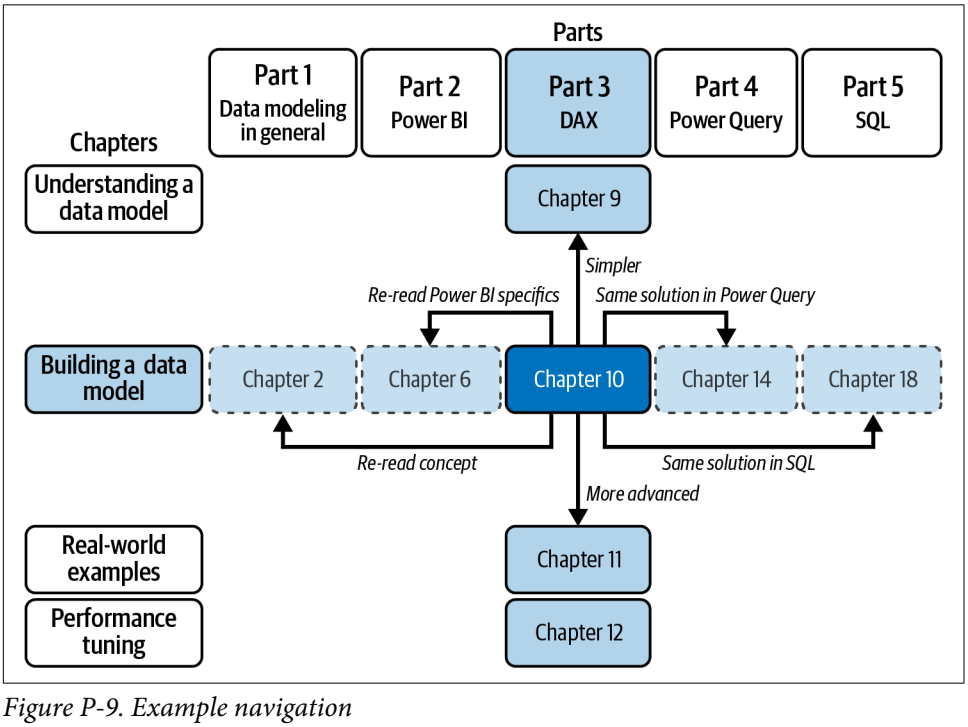


Figure P-9. Example navigation

Ready to begin? Ensure that you have the necessary software installed.

Installing Necessary Software

You need to install the following software to open the demo files I've provided and replicate the exercises described in this book:

Power BI Desktop
: To open the *.pbix* files, you need to either install the Windows Store version from the Power BI Desktop Store or download the installation files from Power BI Desktop installer. Make sure to download and install the *.msi* file regularly, or install Power BI Desktop from the Microsoft Store, which will update automatically.

SQL Client
: For the exercises in SQL, I use SQL Server Management Studio (SSMS), which you can download and install from Download SQL Server Management Studio (SSMS). Alternatively, you can use Azure Data Studio, which you get from Download and install Azure Data Studio, or you use your preferred SQL client.

SQL Server or Azure SQL DB
: For the exercises in SQL, you need access to one of the relational databases available from Microsoft: either SQL Server (installed on your premises) or Azure SQL DB (software-as-a-service in Microsoft's cloud platform). You can download SQL Server from Microsoft. Use either the Express or Developer edition. Both are free of charge and sufficient for the exercises.

  Alternatively, you can sign up for an Azure SQL DB. For the exercises, a free trial access will be sufficient.

Write access to database "AdventureWorksDW"
: Most of the examples are based on a data warehouse schema of the fictitious company I call "Adventure Works," a sport retailer that earns the majority of its revenue through selling bikes on three continents either via resellers or directly over its web shop. To create the necessary database objects (tables, views, procedures, functions, schemas), you need write access to the database "AdventureWorksDW" (the one with the "DW" in the suffix of the name, not: "AdventureWorks" or "AdventureWorksLT" or "Adventure Works OLTP"). You can find the backup file and a description of how to install it on SQL Server here: AdventureWorks sample databases.

  To install database "AdventureWorksDW" as an Azure SQL DB, you can use the "Data-tier Application" (also known as a BACPAC file). Install this BACPAC file via SQL Server Management Studio (SSMS). Right-click Databases and choose "Import Data-tier Application," as shown in Figure P-10.

  Provide the folder and file names of where you downloaded it (Figure P-11).

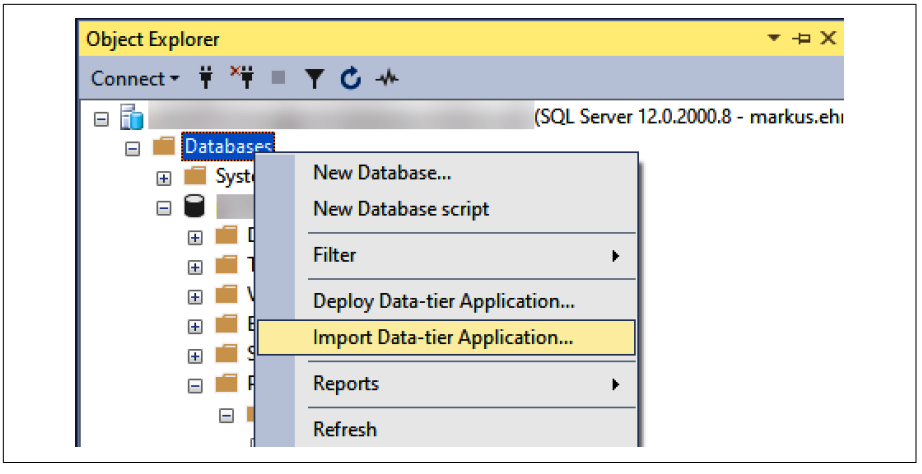


Figure P-10. Import Data-tier Application in SQL Server Management Studio

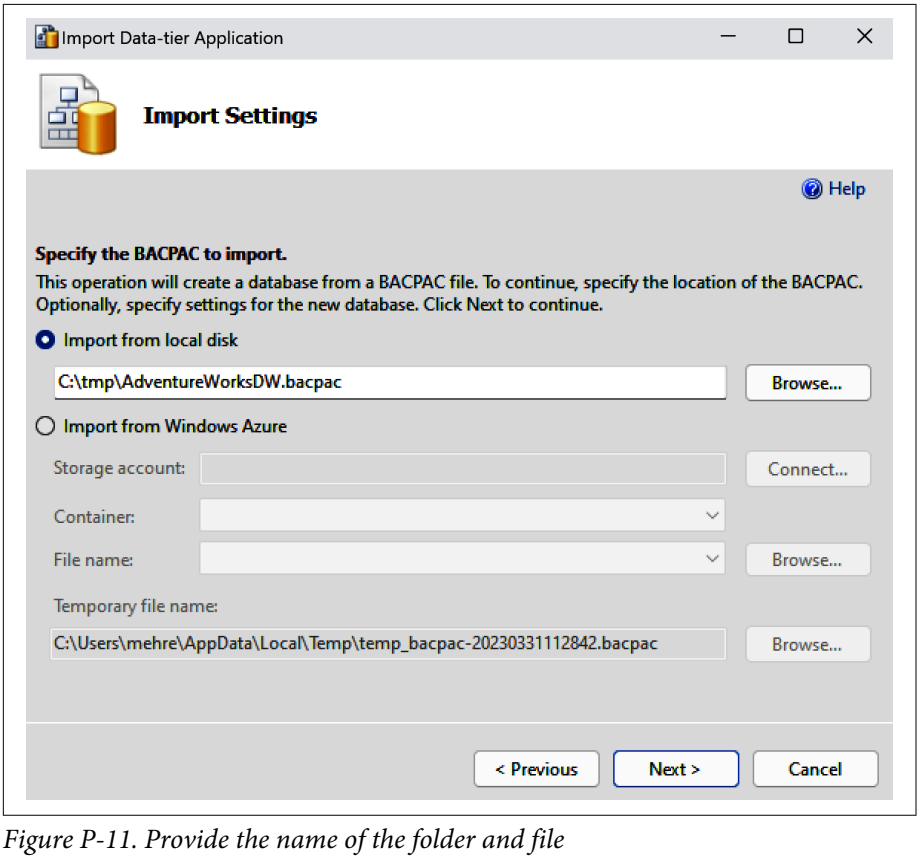


Figure P-11. Provide the name of the folder and file

The smallest/cheapest edition will be sufficient (Figure P-12).

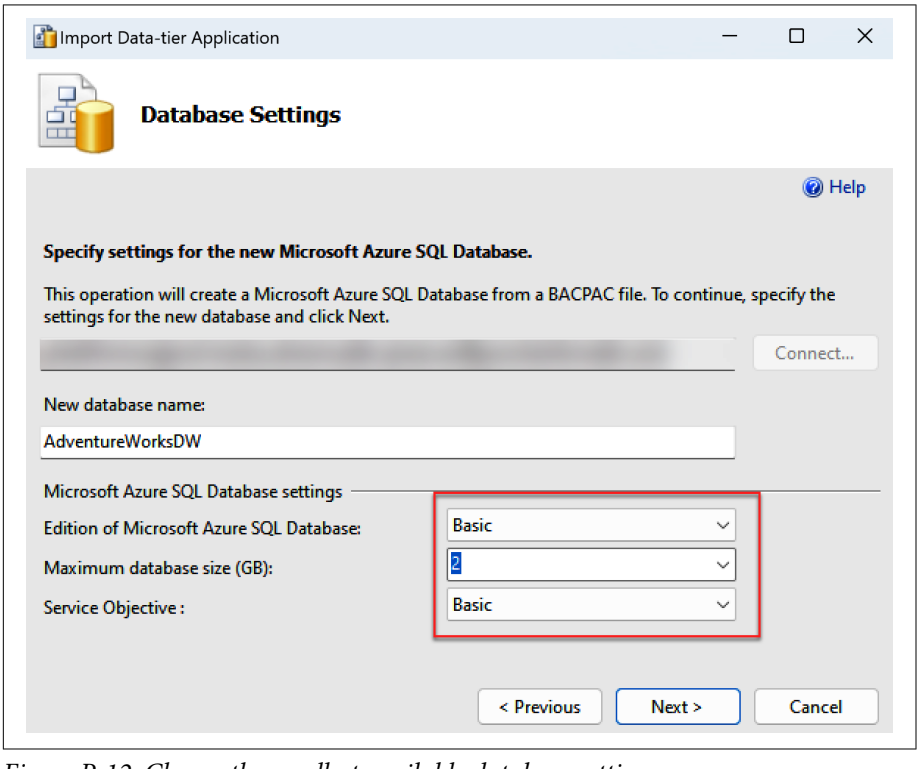


Figure P-12. Choose the smallest available database setting

If you choose to run the database *serverless*, you can further reduce costs, as the database (and all costs) will hibernate if you don't use it for one hour. Next time you access the database, it will wake up for you, which can lead to a timeout. Just reconnect again, and the connection will be established successfully.

Finally, run the script `001 Preparation.sql` when connected to database *AdventureWorksDW* to create all artifacts for the demo environment.

Additional Tools

The data community around Power BI is great. I really admire the smart people who combine a deep knowledge of the technology behind Power BI with understanding the needs of professional Power BI developers, and who also have the skill set to develop useful tools. Some of these people even develop their tools as open source and share them with the community. In this book, I refer to two such tools:

- Tabular Editor V2, the open source version of Tabular Editor V3, by Daniel Otykier (CTO at Tabular Editor ApS)
- DAX Studio, an open source tool by Darren Gosbell (senior program manager at Microsoft)

Demo Files

For every problem, I share one single Power BI Desktop file (*.pbix*) that contains the whole data model and all the solutions in DAX and in Power Query/M, as well as a connection to the tables in Azure SQL. This allows you to easily compare the different technical solutions with on another.

The majority of the examples have an educational purpose: for instance, in the file *Data multiple.pbix* you'll find the table `Order Date` three times with the exact same content. `Order Date (DAX)` is a version completely created in DAX, `Order Date (PQ)` is a version completely created in Power Query, and `Order Date (SQL)`, as you might guess, is a version completely created in SQL and just loaded as is into Power BI. To avoid any misconceptions: in a practical scenario, it neither makes sense to create a table with the same content several times in your data model nor does it make sense to do some transformations in DAX while doing others in Power Query and/or SQL. Stick to one tool to make it easier to find any transformation.

You can find all files in the GitHub repository for this book.

Now you're set for the first chapter, which will give you a basic understanding of what a data model is and what it consists of.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Shows for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

For all DAX code (mostly in Part III), the following conventions also apply:

```
[Measure Name] :=
    <definition of a measure>

'Table Name'[Column Name] =
    <definition of a calculated column>

[Table Name] = /* calculated table */
    <definition of a calculated table>
```

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *http://github.com/MEhrenmueller/DataModeling*.

If you have a technical question or a problem using the code examples, please email *bookquestions@oreilly.com*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but generally do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Data Modeling with Microsoft Power BI* by Markus Ehrenmueller-Jensen (O'Reilly). Copyright 2024 Savory Data Gmbh, 978-1-098-14855-3."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

O'Reilly Online Learning

For more than 40 years, *O'Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, visit *https://oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-889-8969 (in the United States or Canada)
707-827-7019 (international or local)
707-829-0104 (fax)
support@oreilly.com
https://www.oreilly.com/about/contact.html

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *https://oreil.ly/DataModelingwithMicrosoftPowerBI*.

For news and information about our books and courses, visit *https://oreilly.com*.

Find us on LinkedIn: *https://linkedin.com/company/oreilly-media*.

Watch us on YouTube: *https://youtube.com/oreillymedia*.

Acknowledgments

Writing a book looked like a lonely work in the beginning: writing down a concept and draft, going through my own material and doing research, developing and improving the examples, creating the screenshots and diagrams, transforming the thoughts and ideas into words, etc. But I soon found out that it is a collective endeavor; I was lucky enough to work with very engaged and motivated people who I want to bring in front of the curtain here.

Michelle Smith was the acquisition editor who challenged my book proposal and helped it become a useful draft to start from, and Shira Evans was the excellent development editor. I had countless hours of fruitful discussions with Shira in which she challenged the structure of the book, made sure I aligned with the publisher's guidelines, asked the right questions to make the book more readable, and kept me motivated during the whole process all the way through publication with her knowledge, experience, and happy mood. Thanks also to Katherine Tozer, Kate Dullea, Liz Wheeler, Alexis Browsh, and all the folks at WordCo Indexing Services for their work on the final pieces in production.

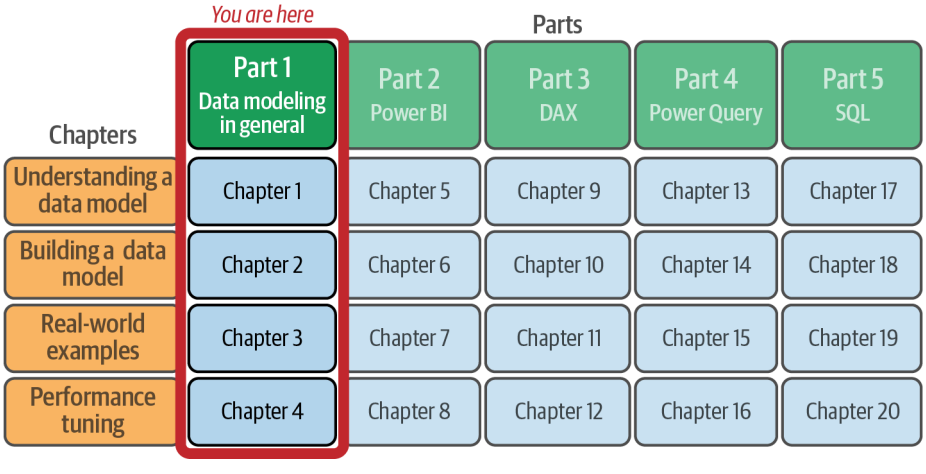
I am also very pleased to have convinced renowned top experts Shabnam Watson, Matt Allington, and Nikola Ilic to join in as technical reviewers for this project. I

appreciate the time they dedicated to this book. Their comments and findings made the book so much better.

Furthermore, I want to thank all my clients who keep me busy in projects and attendees of my seminars, workshops, and webinars through the past years. Solving their challenging problems, preparing materials for presentations, and answering their questions led finally to the idea of bringing everything together for this book.

PART I

Data Modeling 101



The goal of this first part of the book is to bring everybody onto the same page when it comes to the topic of data modeling. The chapters in this part are agnostic: the content, problems, and solutions are not specific to a particular type of database management system. You will be able to apply the knowledge you gain from this part to any relational or analytical database. This includes, of course, Power BI, Analysis Services tabular and Azure SQL DB, but is not limited to them, and you can apply all statements, information, and conclusions to database management systems from vendors other than Microsoft—to classical cubes, to data lakehouses, and so on.

These concepts have existed for decades and are so mature that I bet they'll be around for decades to come. Make sure to learn of all them so you'll understand why I insist on applying one transformation or another when it comes to data modeling in Power BI in the later parts of this book.

Chapter 1 introduces the following basic terms and concepts:

- Entities and tables
- Relations and their cardinality
- Primary and foreign keys

You'll learn how to combine information spread out into different tables with the help of *Set operators* and *join operators*, including problems you might face that, for example, could result in missing data or duplication of data. I discuss the three core possibilities of modeling data (combining everything into one single table, splitting the information in a way that avoids duplicates under all circumstances, and a compromise in the form of a so-called *dimensional model*). You'll learn to decide which to use under which conditions.

Based on the information in Chapter 1, Chapter 2 teaches you how to transform the data model of your data source into a data model of the intended shape. This can be done via transformation steps, including the following:

- Normalizing and denormalizing tables
- Adding calculations
- Transforming flags and indicators into meaningful text
- Adding a dedicated table to contain all dates and/or timestamps
- Modeling dimensions when they can play different roles (e.g., a person can be in the role of an employee or in the role of a customer)
- Modeling dimensions when we need to track attribute changes over time
- Implementing hierarchies

In my experience, the challenges in Chapter 2 are common in most data models. Chapter 3 talks about rarer challenges, which you won't see in every data model. All of them are real-world problems I've had to tackle for my customers. To solve them, you'll need to combine different techniques, which make them more advanced.

Finally, in this part's performance tuning section, Chapter 4, I introduce what role a data model plays when it comes to guaranteeing a fast performance of the reports and queries based on the data model. Basically you need to decide whether you want to persist data in the shape you need it in for your analytics or only store the code for a query, which instead transforms the original data into the necessary shape on the fly.

CHAPTER 1
What Is a Data Model?

This chapter covers the basics of data modeling, starting with basic terms, which will help establish the reasoning behind taking so much care of the data model. A data model that is optimized for creating reports and doing analysis is much easier to work with than one that is optimized for other purposes (e.g., to store data for an application or data collected in a spreadsheet). When it comes to analytical databases and data warehouses, you have more than one option.

The goal throughout this book is to create the data model as a star schema. By the end of this chapter, you'll know which characteristics of a star schema differentiate it from other modeling approaches. Each and every chapter reinforces why a star schema is so important when it comes to analytical databases in general and Power BI and Analysis Services tabular in particular. You will learn how to transform any data model into a star schema.

Transforming information of your data source(s) into a star schema is usually *not* an easy task. Quite the opposite; it can be difficult. It might take several iterations. It might take discussions with the people who you build the data model for—and the people using the reports, as well as those who are responsible for the data sources. You might face doubts (from others and yourself) about whether it's really worth all the effort instead of avoiding the struggle and pulling the data in as it is. At such a point, it's important to take a deep breath and evaluate whether a transformation would make the report creator's life easier. If so, then it's worth the effort. Repeat the data modeler's mantra with me: *make the report creator's life easier*.

Before I talk about transformations, I'll introduce some basic terms and concepts:

- What is a data model?
- What is an entity? What does an entity have to do with a table?

- Why should you care about relationships?
- Why do you need to identify different keys (primary, foreign, and surrogate) and understand the general meaning of cardinality?
- How can you combine tables (set operators and joins)?
- What are the data modeling options?

The first stop on our journey toward the star schema is learning what a data model is in general terms.

Data Model

A model is something that *represents* the real world. It does not replicate it. Think of a map. A map replicating the real world 1:1 would be impractical: it would cover the whole planet. Instead, maps scale down distances. Maps are created for special purposes. A hiking map contains (and omits) different types of information than a road map does, and a nautical chart looks completely different still. All of these are maps but with different purposes.

The same applies to a data model, which represents a certain business logic. As with a map, a data model will look different for different use cases. Therefore, models for different industries will not be the same. And even organizations within the same industry will need different data models (even for basically identical business processes), as they will concentrate on different requirements. The challenges and solutions in this book will help you overcome obstacles when you build a data model for your organization.

Now for the bad news: there isn't one data model that rules them all. Also, it's impossible to create a useful data model with technical knowledge and no domain knowledge. But there is good news: this book will guide you through the technical knowledge necessary to successfully build data models for Power BI and/or Analysis Services tabular.

Don't forget to collect or record all requirements from the business before creating the data model. Here are some examples of requirements in natural language:

- We sell goods to customers and need to know the day and the SKU of the product we sold
- We need to analyze the 12-month rolling average of the sold quantity for each product
- Up to 10 employees form a project team and we need to report the working hours per project task

These requirements will help you determine what information you need to store in which combinations and at which level of detail. There might be more than one option for the design of a data model for a certain use case.

And, very importantly, you need the create the correct data model right from the beginning. As soon as the first reports are created on a data model, every change in the model bears the risk of breaking those reports. The later you discover inconsistences and mistakes in your data model, the more expensive it will be to correct them. This cost hits everybody who created those reports—you yourself, but also other users who built reports based upon your data model.

The data model's design has a huge impact on the performance of your reports, which query data from the data model to feed the visualizations as well. A well-designed data model lessens the need for query tuning later. And a well-designed data model can be used intuitively by the report creators, saving them time and effort (and saving your organization money). From a different point of view, problems with the performance of a report or report creators who are unsure of which tables and columns to use to gain certain insights are a sure sign of a data model that can be improved by a better choice of design.

In "Entity Relationship Diagrams" on page 25, I describe graphical ways to document a data model's shape. But first, let's discuss the entities of a data model.

Basic Components

You need to understand a few key components of a data model before you dive in. In this section, I explain the basic parts of a data model. In "Combining Tables" on page 11, I will walk you through different ways of combining tables with the help of set operators and joins and which kind of problems you can face and how to solve them.

Remember that this part of the book isn't specific to Power BI. Some concepts may apply only when you prepare data before you connect Power Query to it (e.g., when writing a SQL statement in a relational database).

Entity

An *entity* is someone or something that can be individually identified. In natural language, entities are nouns. Think of a real person (your favorite teacher, for example), a product you bought recently (ice cream, anybody?), or a term (e.g., *entity*).

Entities can be both real and fictitious. And most have attributes: a name, value, category, point in time of creation, etc. These attributes are the information we are after when it comes to data. Attributes are displayed in reports to help the reader provide context, gain insights, and make decisions. They are used to filter displayed information to narrow down an analysis, too.

How do such entities make it into a data model? They're stored in tables.

Tables

Tables are the base of a data model. They have been part of data models since at least 1970, when Edgar F. Codd developed the relational data model for his employer, IBM. But collecting information as lists or tables was done way before the invention of computers, as you can see when looking at old books.

Tables host entities: every entity is represented by a row in a table. Every attribute of an entity is represented by a column in a table. A column in a table has a name (e.g., birthday) and a data type (e.g., date). All rows of a single column must conform to this data type (it isn't possible to store the place of birth in the column "birthday" for any row, for example). This is an important difference between a (database's) table and a spreadsheet's worksheet (e.g., in Excel). A single column in a single table contains content. You can see an example in Table 1-1.

Table 1-1. A table containing the name of doctors

Doctor's name	Hire date
Smith	1993-06-03
Grey	2005-03-27
Young	2004-12-01
Stevens	2000-06-07
Karev	1998-09-19
O'Malley	2003-02-14

Entities do not exist just on their own but are related to each other.

Relationships

In most cases, relationships connect only two entities. In natural language, the relationship is represented by a verb (e.g., bought). Between the same two entities, more than one single relationship might exist. For example, a customer might have first ordered a certain product, which we later shipped. It's the same customer and the same product but different relationships (ordered versus shipped).

Some relationships can be self-referencing. That means that there can be a relationship between one entity (one row in this table) and another entity of the same type (a different row in the same table). Organizational hierarchies a are a typical example. Every employee (except maybe the CEO) needs to report to a supervisor. The reference to the boss is therefore an attribute. One column contains the identifier of the employee (e.g., `[Employee ID]`) and another contains the identifier of who this

employee reports to (e.g., `[Manager ID]`). The `[Manager ID]` in one row can be found as the `[Employee ID]` of another row in the same table.

Here are a few examples of relationships expressed in natural language:

- Dr. Smith *treats* Mr. Jones.
- Michael *attended* a data modeling course.
- Mr. Gates *owns* Microsoft.
- Mr. Nadella *is* the CEO.

When you start collecting the requirements for a report (and therefore for your data model) it makes sense to write them down as sentences in natural language, as in the preceding list. This is the first step. In "Entity Relationship Diagrams" on page 25, you will learn to draw tables and their relationships as entity relationship diagrams.

Sometimes the existence of a relationships alone is enough information to collect and satisfy analysis. But some relationships might have attributes, which we collect for more in-depth analysis:

- Dr. Smith treats Mr. Jones *for the flu*.
- Michael completed a data modeling course *with an A grade*.
- Mr. Gates owns *50% of* Microsoft.
- Mr. Nadella has been CEO *since February 4, 2014*.

You learned that entities are represented as rows in tables. The question that remains is how you can then connect rows with each other to represent relationships. The first step is to find a (combination of) columns that uniquely identify a row. Such a unique identifier is called a *primary key*.

Primary Keys

Both in the real world and in a data model, it's important that you can uniquely identify a certain entity (a row in a table). People, for example, are identified via their names in the real world. When you know two people with the same (first) name, you might add something to their name (e.g., their last name) or invent a nickname (which is usually shorter than the first name and last name combined), so that you can make clear who you are referring to (without spending too much time). If somebody isn't paying attention, they might end up referring to one person while another is referring to a different person ("Ah, you're talking about the other John!").