

# Extend Microsoft Access Applications to the Cloud



 Professional

Andrew Couch

# Extend Microsoft Access Applications to the Cloud

Andrew Couch

PUBLISHED BY  
Microsoft Press  
A division of Microsoft Corporation  
One Microsoft Way  
Redmond, Washington 98052-6399

Copyright © 2015 by Andrew Couch. All rights reserved.

No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Library of Congress Control Number: 2014946864  
ISBN: 978-0-7356-6768-6

Printed and bound in the United States of America.

First Printing

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Book Support at [mspinput@microsoft.com](mailto:mspinput@microsoft.com). Please tell us what you think of this book at <http://aka.ms/tellpress>.

This book is provided “as-is” and expresses the author’s views and opinions. The views, opinions and information expressed in this book, including URL and other Internet website references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

Microsoft and the trademarks listed at <http://www.microsoft.com> on the “Trademarks” webpage are trademarks of the Microsoft group of companies. All other marks are property of their respective owners.

**Acquisitions and Developmental Editor:** Devon Musgrave  
**Project Editor:** Rosemary Caperton  
**Editorial Production:** Waypoint Press ([www.waypointpress.com](http://www.waypointpress.com))  
**Technical Reviewer:** Jeff Conrad  
**Copyeditor:** Roger LeBlanc  
**Indexer:** Cristina Yeager  
**Cover:** Twist Creative • Seattle and Joel Panchot

*For Charlotte and Michael. Thanks for all the fun and the way you have enriched our lives.*

—ANDREW COUCH

*This page intentionally left blank*

# Contents at a glance

CHAPTER 1	Finding your way around Office 365	1
CHAPTER 2	Finding your way around Access 2013	57
CHAPTER 3	Converting a desktop database to a web app	99
CHAPTER 4	Creating a blank web app and using templates	133
CHAPTER 5	Displaying data in views	149
CHAPTER 6	Creating data sources by using queries	195
CHAPTER 7	Programming a web app by using macros	215
CHAPTER 8	Managing security and a public-facing web app	289
CHAPTER 9	Looking under the hood at Microsoft Azure SQL Database	301
CHAPTER 10	Other techniques for reporting	339
CHAPTER 11	Using apps for Office with Access	375

*This page intentionally left blank*

# Contents

<i>Introduction</i> .....	<i>i</i>
<b>Chapter 1 Finding your way around Office 365</b> .....	<b>1</b>
Office 365 subscriptions .....	2
Office products in a browser .....	5
Getting started with Office 365 .....	5
Team site, personal site, and public site .....	7
Installing Office Professional .....	10
OneDrive and OneDrive For Business 2013 .....	13
Creating a web app using a template .....	19
Opening a web app with Access .....	23
Saving a web app as a package .....	24
Uploading a web app package .....	26
Editing a web app package .....	30
Displaying a web app in a browser .....	32
Sharing a web app with external users .....	32
Creating sites and subsites .....	36
Creating site collections (Enterprise subscription) .....	39
Applying a theme .....	45
Document storage and libraries .....	47
Microsoft Azure SQL Database, Office 365, and Access .....	51
App Catalog .....	53
Summary .....	55

---

## What do you think of this book? We want to hear from you!

Microsoft is interested in hearing your feedback so we can continually improve our books and learning resources for you. To participate in a brief online survey, please visit:

[microsoft.com/learning/booksurvey](https://microsoft.com/learning/booksurvey)



<b>Chapter 2</b>	<b>Finding your way around Access 2013</b>	<b>57</b>
	A new way of building applications . . . . .	57
	Create a custom web app . . . . .	61
	Importing data from Access . . . . .	62
	Using the navigation pane . . . . .	64
	Working with tables, lookups, and relationships . . . . .	65
	Displaying data in a browser . . . . .	68
	Working with different views . . . . .	71
	Controls available in a view . . . . .	73
	Autocomplete control . . . . .	74
	Related items control . . . . .	75
	Subview control . . . . .	76
	Action bar . . . . .	77
	List control . . . . .	77
	Hyperlink control . . . . .	78
	Multiline Text Box control . . . . .	79
	Web Browser control . . . . .	79
	Overview of macro programming and data macros . . . . .	80
	Upgrading and deploying a web app . . . . .	82
	Side-loading a web app . . . . .	96
	Designing and commissioning with existing data . . . . .	96
	Summary . . . . .	98
<b>Chapter 3</b>	<b>Converting a desktop database to a web app</b>	<b>99</b>
	Tables and primary key fields . . . . .	99
	Relationships and lookups . . . . .	102
	Table and field properties . . . . .	108
	Boolean data . . . . .	112
	Value-based lookups . . . . .	112
	Multi-value data . . . . .	113

Importing different data types.....	113
Long Text (memo data).....	115
Calculated fields.....	115
Image data type.....	115
Attachments and OLE objects.....	115
Reporting.....	116
References.....	119
Quick conversion to external writer.....	120
Attachment and OLE data.....	121
Uploading multiple files.....	122
Extracting attachment files.....	123
Uploading image files.....	125
Summary.....	132
<b>Chapter 4 Creating a blank web app and using templates</b>	<b>133</b>
Creating a blank web app.....	133
Adding template tables.....	134
Creating a new table.....	136
Editing the design of a table.....	137
Creating lookups and relationships.....	139
Adding indexing.....	142
Adding validation rules.....	143
Adding calculated fields.....	145
Summary.....	146
<b>Chapter 5 Displaying data in views</b>	<b>149</b>
Customizing the table selector.....	150
Customizing the view selector.....	151
Creating a pop-up window interface.....	152
Datasheet view.....	156
Views and record sources.....	164

	Duplicating views .....	167
	List Details view .....	169
	Summary view .....	173
	Standalone view .....	177
	Related items control .....	180
	Subview control .....	182
	View controls .....	186
	Combo box control .....	187
	Combo box synchronization .....	188
	Autocomplete control .....	190
	Web browser control .....	191
	Adding color to controls .....	192
	Summary .....	193
<b>Chapter 6</b>	<b>Creating data sources by using queries</b>	<b>195</b>
	Creating basic queries .....	196
	Adding criteria to queries .....	198
	Adding calculations to queries .....	200
	Adding parameters to queries .....	202
	Totals and queries .....	208
	Top value queries .....	209
	Unique values in queries .....	212
	Summary .....	214
<b>Chapter 7</b>	<b>Programming a web app by using macros</b>	<b>215</b>
	Macro-programming capabilities .....	216
	Macro editor and standalone user-interface macros .....	219
	User-interface events for views and controls .....	222
	User-interface macros .....	227
	Named data macros and stored procedures .....	228

Data macros and triggers .....	239
Data macro tracing .....	247
On Start macro .....	249
Transactions and recursion .....	250
Indirect recursion .....	250
Direct recursion .....	251
Presenting a view for printing .....	253
Creating a cross-tabulation of data .....	262
Side-loading a web app .....	267
On Deploy Macro .....	274
Using an alias with a macro action .....	287
Summary .....	288
<b>Chapter 8 Managing security and a public-facing web app</b>	<b>289</b>
Creating a public website .....	289
Creating a public-facing web app in Office 365 .....	291
Managing security in a public web app .....	292
Summary .....	300
<b>Chapter 9 Looking under the hood at Microsoft Azure SQL Database</b>	<b>301</b>
SQL Server Management Studio .....	302
ODBC drivers .....	303
Connecting to Microsoft Azure SQL Database .....	303
Schemas .....	308
Security in SSMS .....	309
Linked Microsoft SharePoint lists .....	309
Manually creating an ODBC DSN connection .....	310
Linking from the desktop to use an ODBC connection .....	317
Creating DSN-less connections with program code .....	319

Extracting information on relationships . . . . .	321
Displaying structural information using data access objects . . . . .	323
Displaying structural information using ADOX . . . . .	327
Displaying structural information with a query . . . . .	330
Validation rules . . . . .	330
Data macros under the hood . . . . .	331
Views and table-valued functions . . . . .	336
Summary . . . . .	338
<b>Chapter 10 Other techniques for reporting</b>	<b>339</b>
Excel and data connections . . . . .	339
Excel PowerPivot . . . . .	344
Creating a report for SQL Server Reporting Services . . . . .	351
Publishing a report to SSRS . . . . .	360
Linking a web app to an SSRS report . . . . .	363
Using Visual Studio 2013 with a web app . . . . .	364
Summary . . . . .	374
<b>Chapter 11 Using apps for Office with Access</b>	<b>375</b>
Apps for Access concepts . . . . .	375
Consuming apps for Access . . . . .	377
Developing apps for Access . . . . .	381
Summary . . . . .	386
<b>Index</b>	<b>387</b>

---

**What do you think of this book? We want to hear from you!**

Microsoft is interested in hearing your feedback so we can continually improve our books and learning resources for you. To participate in a brief online survey, please visit:

[microsoft.com/learning/booksurvey](http://microsoft.com/learning/booksurvey)

# Introduction

If you are already experienced in working with Microsoft Access desktop databases, now is the time to suspend all your existing knowledge and prepare for something completely new. Even though you will find a familiar development interface, it looks a bit different. And even though the user interface is similar to past versions of Access, you will need to become familiar with a lot of new concepts.

If you have never used Access, you should realize that the product can be used to construct both desktop database solutions (which is not covered in this book) and the new web app database solutions. You can use these new web app solutions to quickly and productively deliver solutions in a browser window.

The new Access web apps fit into a strategic initiative to move the world of Microsoft Office into the cloud. Although each Office product offers its own isolated web experience, they all seem to be moving in a similar direction, and they have been consolidated onto the single common platform, Microsoft Office 365. You also can host these technologies with your own on-premise Microsoft SharePoint 2013 Server with Access Services—in which case, you should read the references to Office 365 within this book as meaning “your own SharePoint 2013 Server instance.” This book is written from the perspective of Office 365, so you might find that some of the terminology varies from what you see when using your own on-premise solution.

Historically, there was a time when communicating between Access, Excel, and Word on the desktop was quite difficult. Then, in Office 95 and later when a number of issues were resolved in Office 97, we had seamless integration between the Office products. The process of integration required the adoption of a common programming language called Visual Basic for Applications (VBA). In the new web browser–based interface, VBA is not supported. We hope that Microsoft will improve integration through web services or other technologies, and the addition of new “Apps for Office” features is a step toward offering better integration between the Office products and other technologies.

Microsoft has tried twice in the past to leverage Access into a web browser; both times the technologies did not have great longevity. The first attempt, called *Data Access Pages*, delivered individual web pages and never got a great deal of support from the Access community. The second attempt, called *Access 2010 Web Databases*, took the cloud ideas further and had a lot of great features, but it suffered from too much reliance on storing data within SharePoint’s internal structures and a publication

model that was lengthy and prone to problems; both these issues have been addressed in the new web app technology.

Access 2013 demonstrates how the Access product is being transformed to deliver exceptional productivity in creating a web browser–based experience.

## Product updates and Office 365

---

Office 365 gives Microsoft a platform on which to offer a more frequent mechanism for delivering updates and enhancements to the Office products. These updates, which are scheduled on a monthly basis, are automatically applied. In general, updates to the desktop products will continue to be available on a basis similar to the existing service packs.

However, because Access web apps have such tight integration with the cloud, the Access development team can deploy more significant updates on a shorter, more regular basis as service updates. Some of these updates are visible through the desktop experience of designing a web app.

As an example of the benefits of this approach, in July 2013 new updates were made to Access Services within Office 365 that provide a great new synchronization feature between combo box controls and include more diligent checking for inconsistent objects when creating packages.

In the first quarter of 2014, Microsoft released the first service pack for Office 2013 on Office 365. This service pack includes three significant enhancements to web-app development. This book reflects these changes:

- The first change was to provide a mechanism for developers to deploy changes to web apps without needing to apply those changes directly into the live system. Changes now can be made in a test environment and then subsequently deployed to a production environment.
- The second change is also related to deploying changes—it introduces the idea of locking an application to protect a developer’s intellectual property rights and limit permission to modify the web app. This feature is aimed at software houses that plan to make applications available in the Office Store and need to provide an upgrade path while locking down the source code for the application.

- The third change is of great significance. For some time, Office products such as Excel have benefited from the ability to augment an application with an App for Office—for example, enabling map data to be seamlessly displayed in a spreadsheet. This feature now has been extended to support Access. This opens up opportunities to download a component from the Office Store, as well as to create your own components that, for example, can read and write data through to external company data sources. A long time ago, Access developers benefited from the introduction of OLE Automation, which allowed for interoperability between the Office products. You can view this new integration feature as offering a similar capability in the new development environment.

If you have an on-premise SharePoint 2013 environment, you might find that some newer features are available only at the point of applying service packs. The main use of service packs is to remedy problems, not to provide additional features. Major releases of the product for the desktop environment are still planned to take place. When you consider these updates together with the service update schedule, it is evident that Microsoft intends to compress the overall release cycle.

Because Office 365 can be updated on a regular basis, some of the screen shots might not match what you see on your screen as you follow along with the examples in this book. I also made the decision to focus on Small Business Premium and Enterprise subscriptions; for other subscriptions, you might also see differences in the screen shots, although the basic techniques and concepts will be similar to those I describe.

## Choosing Access to develop in the cloud

---

In Visual Studio a number of years ago, Microsoft created the Entity Data Model as a scaffolding process where, after defining data structures, the tool then created web-browser forms to navigate between related information areas. This approach can be seen as a precursor to some ideas in Access 2013 web apps. Access 2013 uses the relationships in the same way Visual Studio did to create a scaffolding interface where most of the layout work is automatically constructed. This enhances productivity for you as the app developer.

If you are looking for .NET programming–style features, you will find this supported only when developing with the Apps for Office features, because the approach taken when developing with Access will not give you that “dig down deep” opportunity to endlessly extend your own code and functionality. The product is just not designed with that mindset. Currently, most new features in Access are aimed at the power user, with an emphasis on simplicity of use.



If you build sophisticated Access desktop applications and products and regard yourself as a serious Access developer, this book was written for you. But you need to bear in mind that the technology is still evolving, and the chances of reconstructing all your existing Access desktop applications into browser-based apps are slim. However, you will enjoy both the exposition of how to embrace the new technology and starting to see how you can use this technology on new projects and to extend existing projects. I also strive to demonstrate in the book how you can go beyond the built-in features and extend your web app.

If you have a new project where you or a potential client of yours might have increased geographical reach, and you want to save money by not funding infrastructure and get ahead of the competition, this technology offers a uniquely productive experience.

Before you decide to use Access, you need to consider the alternatives. Because you picked up this book, I assume you want a solution in the cloud. So, what are the alternative technologies that can deliver a pure browser-based experience?

The most obvious choice within the Microsoft product portfolio is Visual Studio. The advantage of using Visual Studio is it can give you everything you could ever want on the web. The disadvantage is that you will need to make a serious commitment to understanding how to write complex code. In our experience, people who develop Access applications can find this challenge to be a bit too difficult a step to take, and this is where the new web app technology will help you to take a big leap forward in the browser.

Here are some key benefits to consider in choosing Access 2013 web apps with Office 365:

- Your data is held in Azure SQL, so you have a scalable and efficient platform in which to store your data.
- Unlike many of the competing tools, you have a true graphical, easy-to-use GUI for positioning controls in the browser window, writing program code, and customizing the user experience.
- Although you cannot indirectly (using a tool like SQL Server Management Studio) change the database design in Azure SQL, you can dig down deep into the design to ensure that processing is performed correctly. You also can use the standard Microsoft SQL Server Management Studio (SSMS) to bulk modify data and investigate the database structure.
- You can link existing desktop Access database applications and other Office applications, such as Excel spreadsheets, to the data sets held in Azure SQL.

- You can use PowerPivot to display drill-down charts and summaries of your data in a browser interface.
- SQL Server Reporting Services (SSRS) can be used to supplement a web app delivering up reports in a browser.
- The automatic creation of views and drill-down capabilities will create a rich interface for slicing and dicing views of your data.
- When you change the design of tables, the views (unless customized) are automatically updated to reflect the design changes made in the dependent tables.
- The web app shares and builds on other technologies, such as SharePoint and Azure SQL.
- It offers a low-cost model for creating a functionally rich web experience for your users and provides an easy and intuitive path to get started.
- You can use Visual Studio 2013 to create completely standalone, browser-based Web Form applications linked to your Access data held in the Azure SQL Database.
- You can use Apps for Office targeted for Access to add new features to a web app you developed yourself or to one you obtained from the Office Store.

The preceding list contains just some of the great benefits of choosing Access 2013 to deploy a solution in a browser window. Here are the restrictions:

- The Azure SQL Database does not provide any permission on the master database. This means that other products such as Visio and older versions of Visual Studio will not work with the Azure SQL Database. Some useful standard features in SSMS are not supported, such as setting identity insert on tables (which would enable repeated, indirect uploading of data, replacing primary key values in the data). Another unsupported feature is scripting out the database design (although I have a workaround for this in Chapter 9, “Looking under the hood at Microsoft Azure SQL Database,” in the section “Extracting information on relationships”).
- Although reporting can be achieved in a browser using SSRS or PowerPivot, there is no native reporting features in a web app. This also means there is no ability to acceptably print out a screen of data. (I demonstrate a workaround you can use for small layouts in Chapter 7, “Programming a web app by using macros,” in the section “Presenting a view for printing.”)

## Web app software life cycle

---

You'll find it worthwhile to consider how you will make changes to your application after the application is in use. An Access web app consists of both data and data-related design objects (such as queries and data macros) that are held in an Azure SQL Database as objects and execute in Azure SQL. The user interface design objects (such as UI macros and web app views) are also held in Azure SQL in system tables, but they execute in the browser interface.

Prior to service pack 1, there was no feature allowing you to upload a new set of user interface components without also replacing the Azure SQL Database data, but now there are new features enabling developers to evolve and test changes to a design deploying updates to the application without replacing live data.

In Chapter 1, "Finding your way around Office 365," I describe a simple approach to downloading your design and creating a backup using a package suitable for use either when you start developing an application or when you are the only user of the application. Following this, in Chapter 2, "Finding your way around Access 2013," I provide a more detailed description of the deployment techniques required when maintaining an application for which software updates are required. There is a further related topic, called the *On Deploy Macro*, which is discussed in Chapter 7.

Access 2013 web apps present a unique opportunity to deploy an extremely productive user experience. The scaffolding functionality is fantastic, and the integration into an Azure SQL back end provides a solid scalable platform.

A critique of the initial release of Access 2013 would point toward a lack of support for both upgrading and extending web apps and integrating with external data. Both these issues were addressed in the first service pack, and the remedies for them point toward Access continuing to be an exciting and rich development tool.

## Reference material

---

If you find that you would like to gain a deeper knowledge of the VBA code in Chapter 3, I recommend reading *Microsoft Access 2010 VBA Programming Inside Out* from Microsoft Press (2011).

For a quick reference to techniques for rapidly achieving productivity increases with more general Access functionality, rather than reading an in-depth discussion of the topics, I recommend you read *Microsoft Access 2013 Plain & Simple* from Microsoft Press (2013).

For an in-depth look at Access 2013 in both the context of desktop and web apps, I recommend *Microsoft Access 2013 Inside Out* from Microsoft Press. This Microsoft Access book has a great amount of content that will supplement this book in providing an ideal reference text for an Access developer.

## Who should read this book

---

This book is written for two types of readers. It's written for experienced Access developers who want to understand the detailed design choices they are making when creating a web app, and it's also written for readers who are starting to develop web apps and want to better understand how to develop an Access web app using Office 365.

## Organization of this book

---

This book is divided into three parts. In Part I, "Working with Office 365 and Access web apps," I introduce Office 365 and the Access web app development environment. I also look at issues related to taking an existing desktop database and converting the data it contains into a web app. After reading the first part of the book, you should have the necessary overall understanding to plan in detail how you will move existing data into a web app or start creating a web app with new data in the cloud.

In Part II, "Designing Access web apps," I dive into the details of how to construct a web app. This coverage includes designing tables, queries, and views. I also introduce data macro programming, which is essential to fitting together the different components of the web app to construct a completed application.

In Part III, "Extending Access web apps," I start with a case study illustrating how to create a public-facing web app and manage the public-facing, web app security. Then I look behind the scenes at Azure SQL to get a better understanding of how our web app works with data, queries, and data-macro programming. I also look at linking other technologies—such as PowerPivot, Visual Studio 2013, and SQL Server Reporting Services (SSRS)—to the back-end data in Azure SQL. In a final section, I introduce developing and working with the new Apps for Office technology.

Following is a brief description of each chapter.

In Chapter 1, "Finding your way around Office 365," I describe how to work with Office 365. I take you through concepts such as sites, OneDrive, security, themes, and all the other basic features you need to understand before creating a web app.

In Chapter 2, “Finding your way around Access 2013,” I help you get comfortable with the new GUI interface you will use to design Access 2013 apps. I provide an overview of how all the components in a web app fit together.

In Chapter 3, “Converting a desktop database to a web app,” I discuss taking an existing desktop database and converting the database to a web app. I show you all the issues to avoid and how to best resolve problems. Toward the end of this chapter, I include examples showing VBA code, and I also make the assumption that you are familiar with the Access desktop database experience. This chapter is written for experienced Access developers who need to tackle more demanding problems with converting existing applications.

In Chapter 4, “Creating a blank web app and using templates,” I go back to the basic ideas of a web app and show how to construct a blank web app, add tables, and create relationships between tables.

In Chapter 5, “Displaying data in views,” I give you a look at how to create and manage the basic components for interacting with data, including the list details, datasheet, blank and summary views. Then I provide a systematic examination of the available controls in each view.

In Chapter 6, “Creating data sources by using queries,” I show you how to take advantage of using queries in your web app, including working with calculations and parameters.

In Chapter 7, “Programming a web app by using macros,” I look in detail at writing both UI and data macros. I also delve deeper into SQL Azure to see how the data macros get implemented as stored procedures and triggers.

In Chapter 8, “Managing security and a public-facing web app,” I look at a number of great features for making your web app available to anonymous users and configuring security for external users.

In Chapter 9, “Looking under the hood at Microsoft Azure SQL Database,” I provide a more detailed look at working with Azure SQL and guiding you through the process of using the SQL Server Management Studio (SSMS) to connect to and manage your web app.

In Chapter 10, “Other techniques for reporting,” I look at how to extend support for your web app using PowerPivot, Visual Studio 2013, and SSRS.

In Chapter 11, “Using Apps for Office with Access,” I introduce the new features for integrating Apps for Office in your web apps.

## Conventions and features in this book

---

This book presents information using the following conventions, which are designed to make the information readable and easy to follow:

- Boxed elements with labels such as “Note” provide additional information or alternative methods for completing a step successfully.
- Text that you type (apart from code blocks) appears in bold.
- A plus sign (+) between two key names means that you must press those keys at the same time. For example, “Press Alt+Tab” means that you hold down the Alt key while you press the Tab key.
- A vertical bar between two or more menu items (for example, File | Close) means that you should select the first menu or menu item, then the next, and so on.

## System requirements

---

You will need the following hardware and software to complete the practice exercises in this book:

- Windows 7 or Windows 8.
- Office 365 subscription that includes Office 365 ProPlus. (Access 2013 is required for all chapters, and in Chapter 10 Excel 2013 is required.)
- Visual Studio 2013 standard edition for some examples in Chapters 10 and 11.
- SQL Server 2012 Express Edition (or newer edition), with SQL Server Management Studio 2012 Express or newer for Chapter 9.
- SQL Server 2012 with Advanced Services (developer, standard, or newer edition) for SSRS examples in Chapter 10.
- SQL Server Business Development Studio (obtained with SQL Server 2008 or 2008R2 express or higher edition) for SSRS examples in Chapter 10.
- A computer or device that has a 1.6-GHz or faster processor (2 GHz recommended).
- 1 GB (32 bit) or 2 GB (64 bit) RAM. (Add 512 MB if running in a virtual machine or SQL Server Express Editions, and add more for advanced SQL Server editions.)

- 5400-RPM hard disk drive.
- Internet connection to download software or chapter examples

Depending on your Windows configuration, you might require Local Administrator rights to install or configure Visual Studio 2013 and SQL Server 2012 products.

## Downloads: Companion content

---

Most of the chapters in this book include companion content that let you interactively try out new material learned in the main text. All sample web apps, can be downloaded from the following page:

*<http://aka.ms/AccessApps/files>*

Follow the instructions to download the `AccessApps_667686_CompanionContent.zip` file.

### Installing the companion content

Follow these steps to install the companion content on your computer or device so that you can use them with the exercises in this book:

- Unzip the `AccessApps_667686_CompanionContent.zip` file that you downloaded from the book's website (the contents of which should be extracted to your desktop or another appropriate location).

### Using the companion content

The folder created by the `ExtendAccess.zip` file contains the following subfolders:

- **Chapter2** This folder contains the desktop database `NorthwindRestructuredData.accdb` which is used as a source of data when you create your `NorthwindData` web app.
- **Chapter3** This folder contains the desktop databases `Chapter3_ReKeyedTables.accdb`, `Chapter3_TablesToConvert.accdb`, `ConvertAttachments.accdb`, and `NorthwindReportsChangeToExternalWriter.accdb`.

- **Chapter6** This folder contains the completed web app for Chapters 2, 3, 4, 5, and 6. The web app NorthwindData\_Completed.app can be uploaded to Office 365 as described in Chapter 1 in the section “Uploading a web app package.”
- **Chapter7** This folder contains the desktop database NorthwindRestructuredMacros.accdb, which is used as a source of data when you create your NorthwindRestructuredMacros web app. The web apps NorthwindRestructuredMacrosCompleted.app and Projects with Printable Sales Orders - MultiUser.app contain the completed examples and can be uploaded to Office 365 as described in Chapter 1 in the section “Uploading a web app package.”
- **Chapter8** This folder contains the sample web app UKAUG\_PublicWebSite. app and can be uploaded to Office 365 as described in Chapter 1 in the section “Uploading a web app package.”
- **Chapter9** This folder contains the desktop databases DSNManagement. accdb, NorthwindADOX.accdb, and NorthwindDAO.accdb, and the script files ListRelationships.sql and ScriptOutDesign.sql.
- **Chapter10** This folder contains the desktop database NorthwindReportsForSSRS.accdb and a folder named SSRSWebAppReports that contains the SSRSWebAppReports.sln Visual Studio project with the converted SSRS report Invoice.rdl.



**Note** To follow along with the examples in Chapter 9, you will need a copy of Microsoft SQL Server 2012. (The Express edition can be used.) For parts of Chapter 10 and Chapter 11, you will need a copy of Microsoft Visual Studio 2013 (the minimum Standard edition with any current service packs).

For the section in Chapter 10 on SSRS, you will need a Developer or Standard edition of SQL Server 2012 with Advanced Services. (The Express edition cannot be used.) You also will need the SQL Server Business Development Studio (available as part of a SQL Server 2008 or 2008R2 installation). The Express edition can be used to obtain this feature).



## Acknowledgments

---

I'd like to thank Jeff Conrad at Microsoft for performing the technical review of the book and for his invaluable guidance in helping to clarify explanations in the text. My thanks also to Kevin Bell for his assistance through many discussions on newer product features. Thanks also to other members of the Access Team at Microsoft for taking the time to help on several technical issues.

My thanks to the staff at Microsoft Press, including Devon Musgrave and Rosemary Caperton for editorial guidance. Thanks also to Roger LeBlanc for his diligent copy-editing and Steve Sagman for editorial and production management. Thanks also to Kenyon Brown for instigating the work that has resulted in this text.

My final thanks is to my wife, Pamela, for her understanding and patience with me as I undertook the writing.

## Errata, updates, & book support

---

We've made every effort to ensure the accuracy of this book and its companion content. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

*<http://aka.ms/AccessApps/errata>*

If you discover an error that is not already listed, please submit it to us at the same page.

If you need additional support, email Microsoft Press Book Support at [mspinput@microsoft.com](mailto:mspinput@microsoft.com).

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to <http://support.microsoft.com>.

## Free ebooks from Microsoft Press

---

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at:

*<http://aka.ms/mspressfree>*

Check back often to see what is new!

## We want to hear from you

---

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

*<http://aka.ms/tellpress>*

We know you're busy, so we've kept it short with just a few questions. Your answers go directly to the editors at Microsoft Press. (No personal information will be requested.) Thanks in advance for your input!

## Stay in touch

---

Let's keep the conversation going! We're on Twitter: *<http://twitter.com/MicrosoftPress>*

*This page intentionally left blank*

# Creating a blank web app and using templates

**In this chapter:**

Creating a blank web app . . . . .	133
Adding template tables . . . . .	134
Creating a new table . . . . .	136
Editing the design of a table . . . . .	137
Creating lookups and relationships . . . . .	139
Adding indexing . . . . .	142
Adding validation rules . . . . .	143
Adding calculated fields . . . . .	145
Summary . . . . .	146

In this chapter, you'll see how you can create a new blank web app using design tools without needing to import existing data. You'll also look at basic design activities such as creating lookups and relationships, adding validation rules, indexing, and using calculated fields.

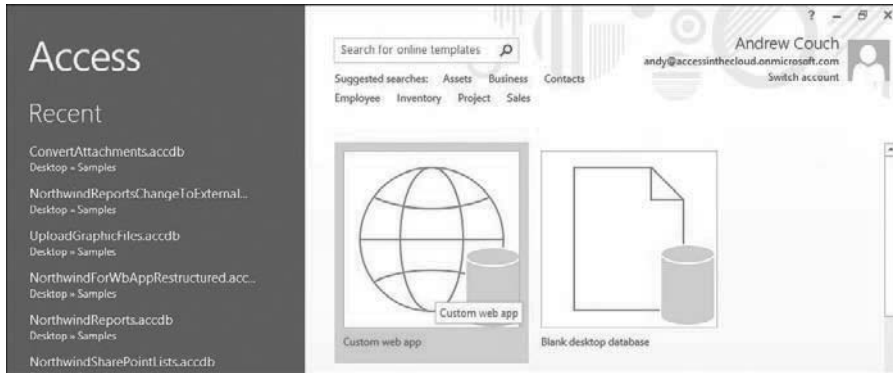
The web app you create in this chapter is not used in subsequent chapters.

## Creating a blank web app

---

When using Access, you can create either a desktop database or a web app. The list of choices for each option also includes built-in templates for creating popular database structures, both for the desktop and for a web app. The templates for the desktop databases begin with the word "Desktop" in their name, as listed on the Office Start screen. In a web app the idea of a template database is of more limited importance, which you'll see when working with the new feature for table templates in the next section.

After you start Access, select Custom Web App, as shown in Figure 4-1.



**FIGURE 4-1** Creating a blank web app.

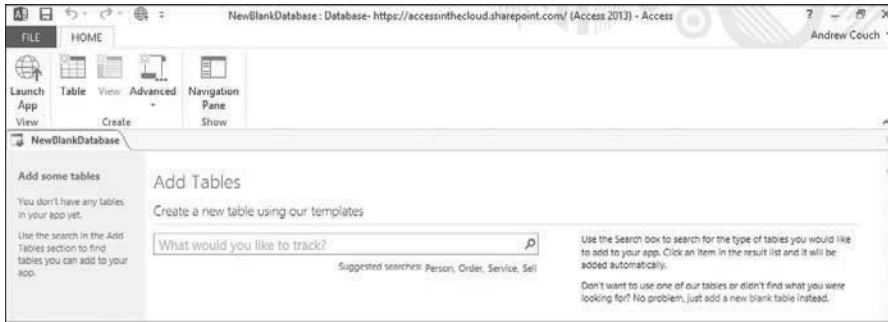
After you make this selection, you will be prompted to select a name and location for the application. As shown in Figure 4-2, name your web app **NewBlankDatabase**.



**FIGURE 4-2** Entering the App name and selecting a location.

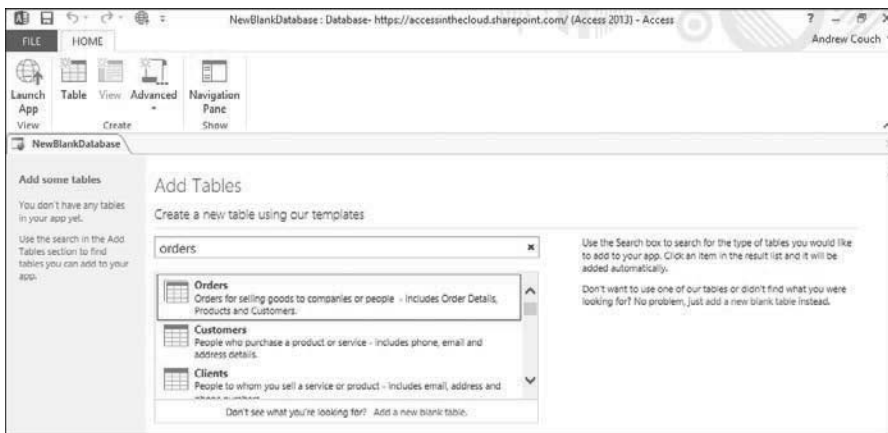
## Adding template tables

After creating your blank web app, you will be presented with the screen shown in Figure 4-3. Notice that below the search box are hyperlinks for suggested searches on Person, Order, Service, and Sell.



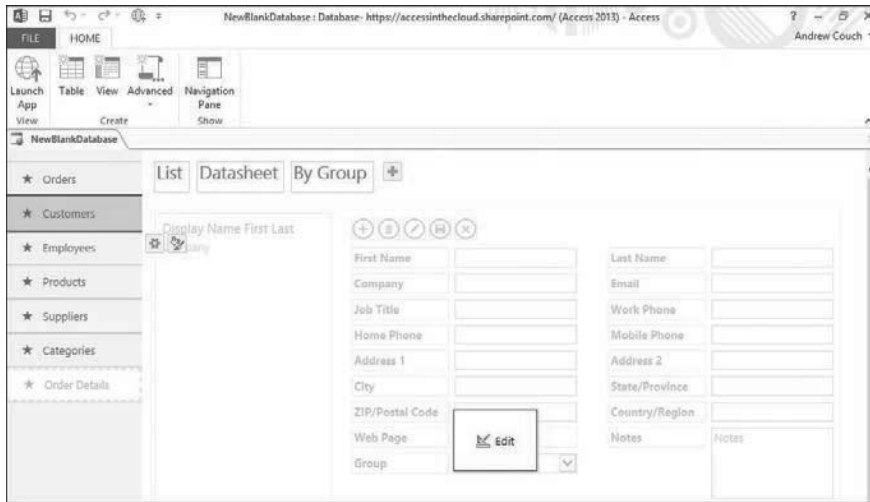
**FIGURE 4-3** The Add Tables page.

If you type **Orders** in the search box, Access will try to match your request with the set of options most appropriate to your needs. You will notice that the results can show either a box with a single outline, such as Customers (which means you get one table), or a box named Orders with a shadowed outline (which means you get more than one table). Select the Orders table option shown in Figure 4-4 to follow the example in this chapter.



**FIGURE 4-4** Selecting a template table.

The result of making this selection is shown in Figure 4-5, where not only have new tables been added, but on the Table Selector on the left the Order Details view is hidden from the user.



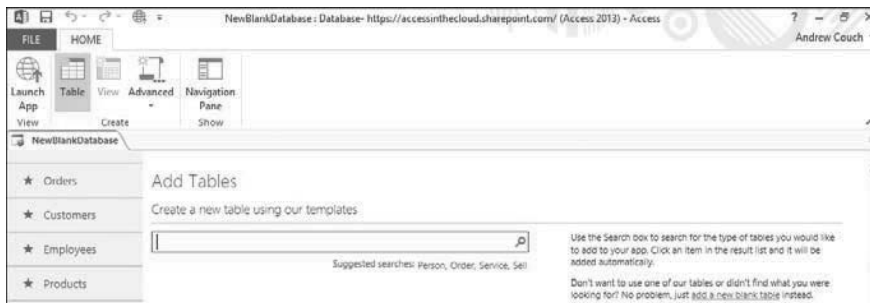
**FIGURE 4-5** Multiple tables displayed in the Table Selector.



**Tip** Access 2013 tries to get you started with a productive business solution by helping to structure your data and then, with the data structures created, adding views to display the data. The underpinning concept to this scaffolding is the relationships between the tables. With one or two clicks, you have a set of tables and views for the data. This is the true power of this new design tool, and later you will see that this approach of automatically creating views also applies when you construct your own blank or imported tables.

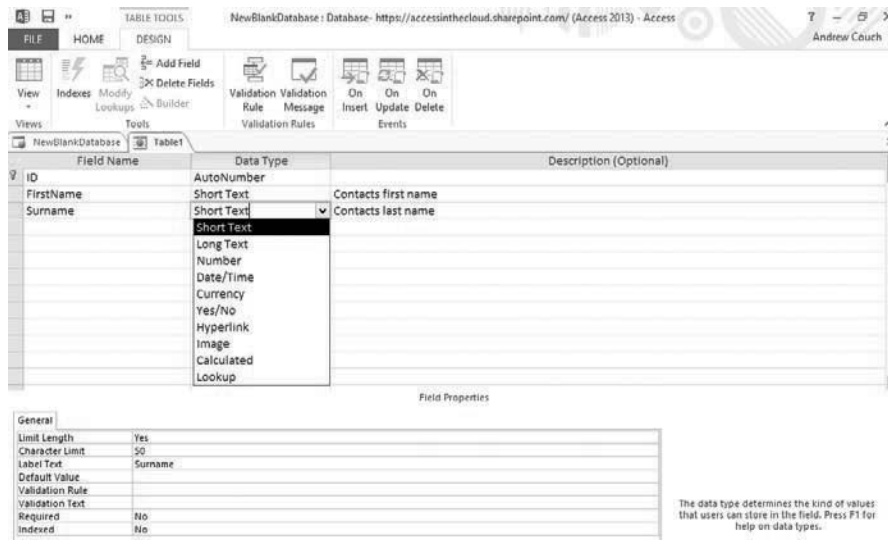
## Creating a new table

To create new tables, click the Table icon on the home ribbon or, at the bottom of the Table Selector, click the Add New Table button. Either method will display the Add Tables screen shown in Figure 4-6. On the right side of the screen at the end of the second paragraph of text, click the Add A New Blank Table hyperlink.



**FIGURE 4-6** Add Tables page.

After you click the link, Access displays the design interface. Here you need to add two new fields to the table design (FirstName and Surname, which are both Short Text data types), as shown in Figure 4-7. In Chapter 3, “Converting a desktop database to a web app,” in the section “Importing different data types.” I listed the available data types for use in a web app.



**FIGURE 4-7** A new table viewed in the design interface.

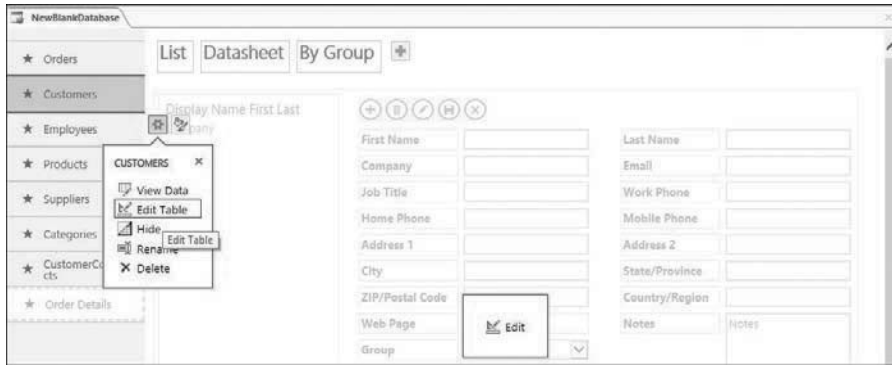
Each table will automatically be assigned a primary key AutoNumber called *ID*. This field can be renamed but not deleted from the table design. You can change the name of the key to a more meaningful name—for example, ContactID, which is used for the new table design in this chapter.

Save the table with the name CustomerContacts. Before getting deeper into the design, you will change the names of all the primary and foreign-key fields to make them easier to understand, rather than accepting the default names for the fields.

## Editing the design of a table

When viewing the Table Selector, you can choose from one of two methods to quickly switch a table into design view to change the table structure. You can double-click on the table caption in the Table Selector. Or you can click once on the table caption, either right-click on the table or directly click on the Settings/Actions charm, and then choose Edit Table, as shown in Figure 4-8. You could also display the navigation pane and double-click or right-click on a table.





**FIGURE 4-8** Opening a table in Design view.



**Warning** Don't click on the large edit button in the center of the screen because that is for editing the view that is displayed in the design preview on the screen, and in this situation we want to change the design of the table and not alter a view layout.

Make the following name changes to the primary and foreign keys in the tables, and save your changes.

**TABLE 4-1** Changes for the primary and foreign keys

Table Name	Old Field Name	New Field Name
Orders	ID	OrderID
	Customer (Lookup)	CustomerID
	Employee (Lookup)	EmployeeID
Customers	ID	CustomerID
Employees	ID	EmployeeID
Products	ID	ProductID
	Supplier (Lookup)	SupplierID
	Category (Lookup)	CategoryID
Suppliers	ID	SupplierID
Categories	ID	CategoryID
Order Details	ID	OrderDetailsID
	Product (Lookup)	ProductID
	Order (Lookup)	OrderID

As you are going through this, you can also change the captions on the foreign keys and primary keys—for example, change *ProductID* to *Product*. That has not been done for the example used in this chapter.

If you look at Figure 4-9, you will see how the web app has altered all views to reflect the changes you made to the field names—in this case, *OrderID* and *ProductID* in the Order Details table.



**FIGURE 4-9** Automatic updates to the design of the quickly created views.

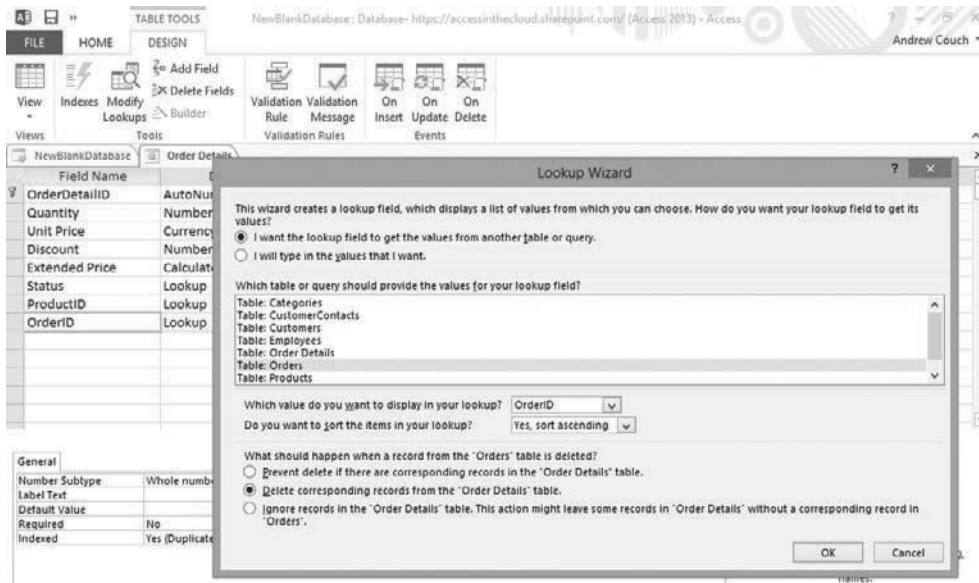


**Tip** As long as you have not edited any of the quickly created views, any changes to elements in the table will be automatically reflected in the views (which have not been edited). This means that making any significant structural changes before starting to work on view design will save you design effort.

## Creating lookups and relationships

In Chapter 3, I described the rules that are applied when importing lookups and relationships, and I described how lookups and relationships are combined into one feature in the user interface even though behind the scenes real relationships can be created in the back-end Microsoft Azure SQL Database.

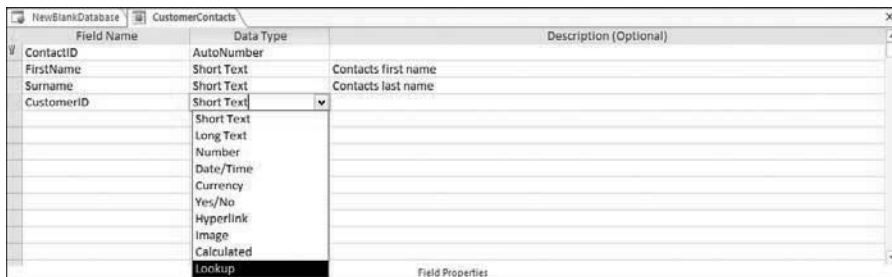
Before looking at how to create a new lookup, you'll find it useful to see how to display existing lookups. View a table in design view, select a field that is a lookup, and then click the Modify Lookups button on the Design contextual ribbon tab. Access opens the Lookup Wizard, as shown in Figure 4-10.



**FIGURE 4-10** The Modify Lookups selection, displaying the Lookup Wizard.

You will notice that the Lookup Wizard supports both a lookup to data in a table and a list of values you typed in. In the Order Details table, if you choose to modify the lookup for the Status field, you will see an example of a value-based lookup.

Return to the new CustomerContacts table. Using the design view, type in the foreign-key name **CustomerID** on a new line and change the data type from Short Text to Lookup, as shown in Figure 4-11.



**FIGURE 4-11** Adding a new lookup field.

In the Lookup Wizard, select the first option, I Want The Lookup Field To Get The Values From Another Table Or Query. Select the Customers table from the list of tables, and then select Company from the drop-down list to be the field displayed. You can also change the sorting order if required, as shown in Figure 4-12.

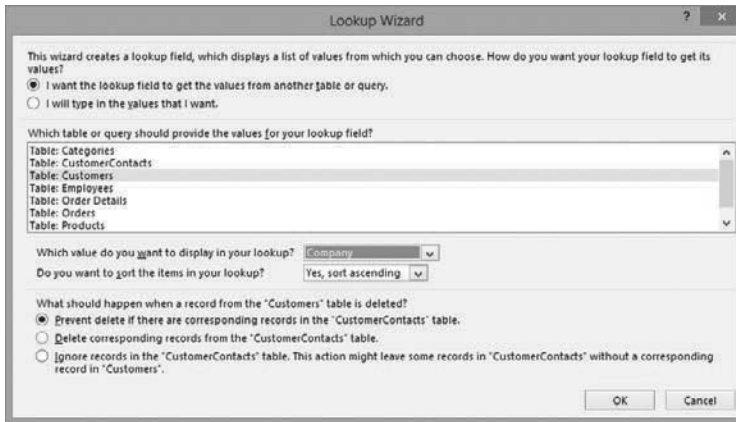


FIGURE 4-12 Completing selections in the Lookup Wizard.

In the lower part of Figure 4-12 are three choices that control how the relationship will operate:

- **Prevent Delete If There Is Are Corresponding Records In The “CustomerContacts” Table** This option creates a relationship with referential integrity between the two tables.
- **Delete Corresponding Records From The “CustomerContacts” Table** This option acts similar to the preceding one, but it adds a cascade delete to the relationship.
- **Ignore Records In The “CustomerContacts” Table** This action might leave some records in “CustomerContacts” without a corresponding record in “Customers.” No relationship will be created.

The web app does not have any point at which you can view all the relationships to see a diagram of how the tables fit together. (There is no equivalent to a desktop database diagram.)



**Tip** The information on lookups that enforce referential integrity is stored inside the Microsoft Azure SQL Database. How to extract this information is described in Chapter 9, “Looking under the hood at Microsoft Azure SQL Database.”

After you create this new lookup between Customers and CustomerContacts choosing the first option to enforce referential integrity, if you display the Customers table caption and select the list view, at the bottom of the view you will see that the web app has automatically added a new tab to the related item control (RELIC). It now displays the related CustomerContacts tab (in addition to the existing Orders tab), as shown in Figure 4-13.



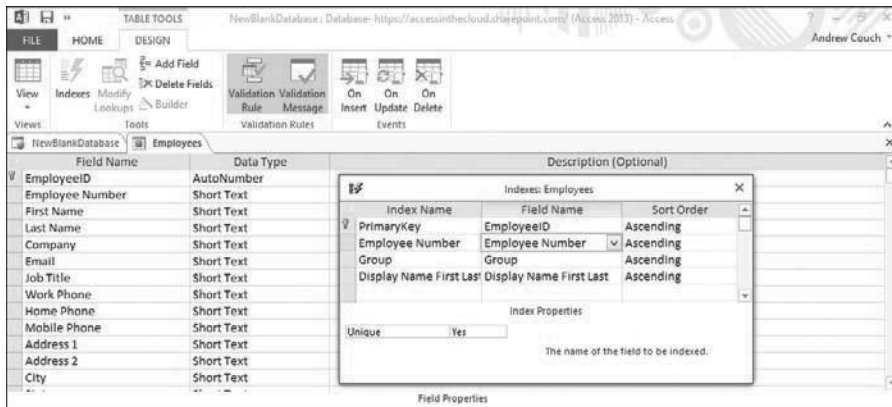
**FIGURE 4-13** Automatic changes to related information for the Customers list view.

## Adding indexing

If you create a new table, the primary key for the table will be indexed, and if you add a lookup, the foreign key for the lookup will also be indexed. The indexing on foreign keys is created to help improve performance when joining together two tables of data to return with queries.

Adding additional indexes is useful when you have a commonly searched field and you believe an index could help to retrieve the data. An index can also be created that is unique, and then you can use the index to enforce a business rule ensuring unique values.

Take a look at the design of the Employees template table. If you click the Indexes button on the Design contextual ribbon tab, Access opens the Indexes dialog box and displays all indexes defined in the table, as shown in Figure 4-14.



**FIGURE 4-14** Displaying indexing on the Employees table.

In Figure 4-14, although the PrimaryKey index is unique, the Employee Number index is also unique. This means that the second index is enforcing a business rule stating that each employee has a unique employee number.



**Tip** A value of NULL (no value) is treated as also being unique, so this index seems to allow one employee, and only one, to have no value for the employee number. But because the field is a required field (NOT NULL), there is no inconsistency. If you created a field that was not required (NULLABLE) and created a unique index, you could get some strange side effects.

If you have an activity that is slow to execute, adding an index on, for example, a field used for filtering the data can improve performance. The only way to test this is to add the index and then see if the performance has improved. Indexes can also be created using multiple fields. One key question to ask before adding an index is, “Will the field values have sufficiently varied values to make the index selective?” For example, indexing a field with only four or five distinct values is unlikely to be of great benefit, while indexing a field for a postal code is likely to be beneficial if you are constantly filtering or searching on this field.

You also incur a cost with having indexes: the database will need to update the indexes when data changes. This means that adding many indexes to a table can result in operations to change the data slowing down (because the indexes will need to be updated). It is a good idea not to create too many indexes. Exactly what is meant by too many is difficult to quantify. However, if you kept the number of indexes to 5 or 6, that would not be an unreasonable number of indexes.



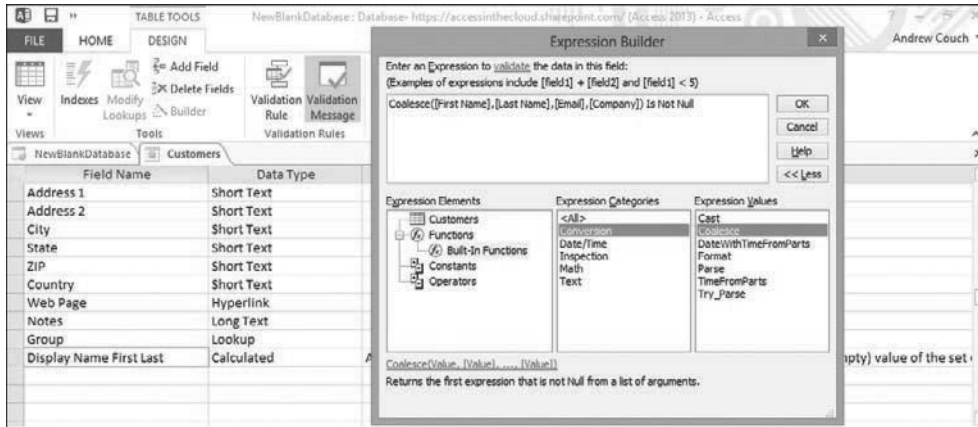
**Tip** Although the Microsoft Azure SQL Database has tools for displaying how indexing is being used (specifically, showing the execution plan for your SQL), unfortunately at the time of this writing, you will not have sufficient permissions in Microsoft Azure SQL Database to use these features.

## Adding validation rules

---

A web app has validation features similar to a desktop database, but it uses a different set of functions to support validation. Each table is allowed a validation rule and message. The table-level validation rule is used to cross-validate data and compare the values in several fields. Depending on the data type, each field can have a separate validation rule, which normally refers only to the field but can make references to the values in other fields. (This is a feature that is not available in desktop database field validation rules.)

In the sample web app in this chapter, look at the template table Customers and click the Validation Rule button on the Design contextual ribbon tab. You will see the rule displayed in the Expression Builder dialog box, as shown in Figure 4-15.



**FIGURE 4-15** Expression Builder dialog displaying a validation rule.

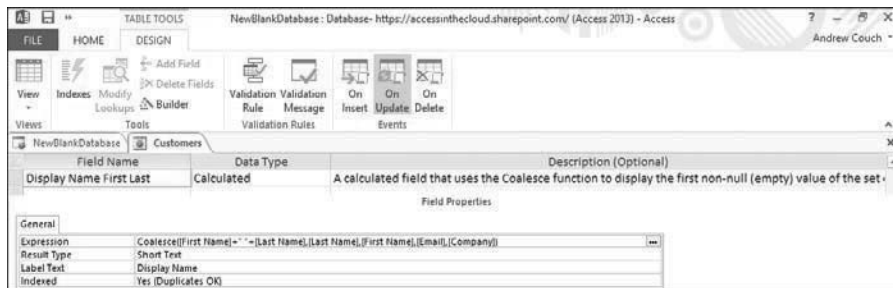


**Tip** In addition to the help information provided within Access, the validation rule functions (such as COALESCE, which returns the first NOT NULL value from a list of choices) are SQL Server functions.) You can find detailed examples and further information when looking in the SQL Server help; this information is most easily found either by searching online or by using the help features in the SQL Server Management Studio (SSMS), which is described in Chapter 9.

The preceding example of a table-level validation rule ensures that one of the four specified fields has to contain a value:

- Coalesce([First Name],[Last Name],[Email],[Company]) Is Not Null

Individual fields can also have a single validation rule. For example, in the Products table, if you want to ensure that each product code starts with the string "01-", you could add the validation rule shown in Figure 4-16.



**FIGURE 4-16** Field validation rule and text.

In this example, you find the rule `Left([Product Code], 3) = "01-"`, which checks that each product code starts with "01-". When entering the rule, you can choose to use single quotes (such as '01-'), which would also be accepted.



**Tip** To delete a field or table validation rule or message, display the text and remove the text or rule using the backspace key.

## Adding calculated fields

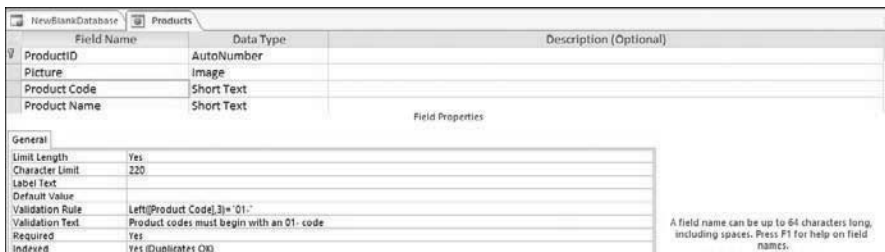
Tables support the use of calculated fields, where the calculation can be based on a combination of other fields in the table. There are two advantages to using a calculated field rather than an expression in a query. The first advantage is that rather than possibly repeating the calculation in several places, the calculation is defined in one place in the design of the table.

The second benefit is that it might give you better performance, because the web app holds the results of the calculation with the data in the table. This means that it needs to recalculate the result only if any data in the dependent fields change, which it will do automatically. The storage type used for the calculated field will vary to suit the needs of the calculation, as defined in the Result Type field property for the calculation (which is a read-only property).



**Tip** Because the calculation results are saved, you are not allowed to use expressions in a calculation that could vary over time. For example, you can calculate the number of days between two date fields in the table but not the number of days between a date field and today's date (because that calculation varies depending on the current date). In SQL Server terminology, you are allowed to use only *deterministic* expressions, not *nondeterministic* expressions.

Figure 4-17 shows an example of a calculated expression in the Customers table.



**FIGURE 4-17** Calculated field expression.

In this example, the calculation used is the following:

```
Coalesce([First Name]+' '+[Last Name],[Last Name],[First Name],[Email],[Company])
```



The *Coalesce* function returns the first non-null value in the list of expressions. Notice that the plus sign (+) symbol is used for string concatenation; this is because that is what SQL Server uses for concatenating strings. An Access desktop database can use either + or & for string concatenation. The advantage of using & is that if one part of the expression is NULL, it will still give a result. Here is an example:

```
"Andrew" + NULL = NULL, but "Andrew" & NULL = "Andrew"
```

To obtain a similar expression to the Access desktop database expression `Trim$( [First Name] & " " & [Last Name] )`, you would need to protect the string concatenation against a NULL—because you cannot use the & symbol but must use the + symbol.

You could use the following expression:

```
LTRIM(RTRIM(COALESCE([First Name], "") + " " + COALESCE([Last Name], "")))
```

Here you need to use the combination of left and right trim operations because the TRIM function has no equivalent function in SQL Server.



**Tip** If you examined this calculation in Microsoft Azure SQL Database, you would see the following expression:

```
(ltrim(rtrim((coalesce([First Name],N'')+N'')+coalesce([Last Name],N'')))),
```

Here you can see that the double quotes have been switched to single quotes, and the use of the *N* symbol next to the string expressions indicates that they should be read as Unicode strings.

## Summary

---

In Chapter 3, I focused on dealing with converting an existing desktop data. In this chapter, I described the alternative scenario, where you need to create new table structures or alter the design of imported table structures.

You started by creating a blank database and then adding tables using the built-in table templates. You used this great new feature to add individual tables or sets of tables to either a new or existing web app.

This chapter also demonstrated the essential skills you need to do the following:

- Create a blank new table.
- Edit the design of an existing table.
- Create lookups and relationships.

- Add indexing to improve performance.
- Add validation rules.
- Define calculated fields.

Now that you understand both how to manage the importing of data from existing databases and the creation of new database structures, you can move forward in Chapter 5, “Displaying data with views.” In that chapter, you develop an understanding of how to customize the basic UI interface displayed when new tables are created or imported into a web app.



*This page intentionally left blank*

# Creating data sources by using queries

## In this chapter:

Creating basic queries . . . . .	196
Adding criteria to queries. . . . .	198
Adding calculations to queries . . . . .	200
Adding parameters to queries. . . . .	202
Totals and queries. . . . .	208
TOP value queries. . . . .	209
Unique values in queries. . . . .	212
Summary. . . . .	214

In this chapter, we will look at how to create queries in a web app. These queries can then be used either with views to display information or with data macros to process information. If you are familiar with this subject, you will notice that in the contents there are no topics on creating queries to modify data. Queries that modify data are not supported in a web app; instead, you use data macros to perform equivalent operations.

In Chapter 5, “Displaying data in views,” I demonstrated creating a simple query for creating a jump list to summarize product data in our application. In this chapter, I provide a more detailed explanation of how queries are created and how they can then be used in your application. As you did in Chapter 5, you’ll use some simple user-interface macros to link your queries to views. In Chapter 7, “Programming a web app by using macros,” I will provide a more detailed discussion of macro design.

If you want to follow along with the design steps in this chapter, continue to use the NorthwindData web app used in Chapter 5.

# Creating basic queries

Queries are created using the Query option on the Advanced menu, as shown in Figure 6-1.

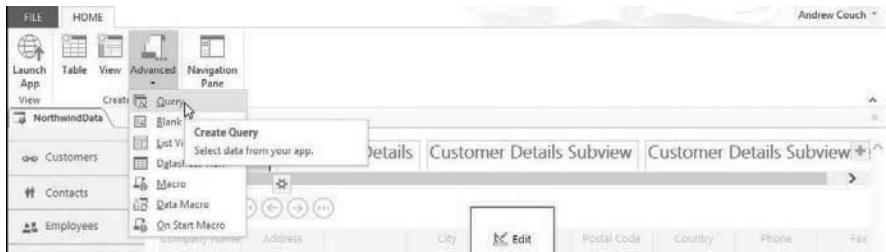


FIGURE 6-1 Creating a query.

When the query designer opens, the Show Table popup is displayed. It has three tabs you can click on to display tables, queries, or both. If you double-click a table or query in the Show Table popup (or click the Add button when the table or query is highlighted), it will be added to the query, as shown in Figure 6-2. If you close the Show Table popup, you can return later to add other tables or queries by clicking the Show Table icon on the ribbon. If tables are related, lines will be added between the linking fields as the tables are added to the query.

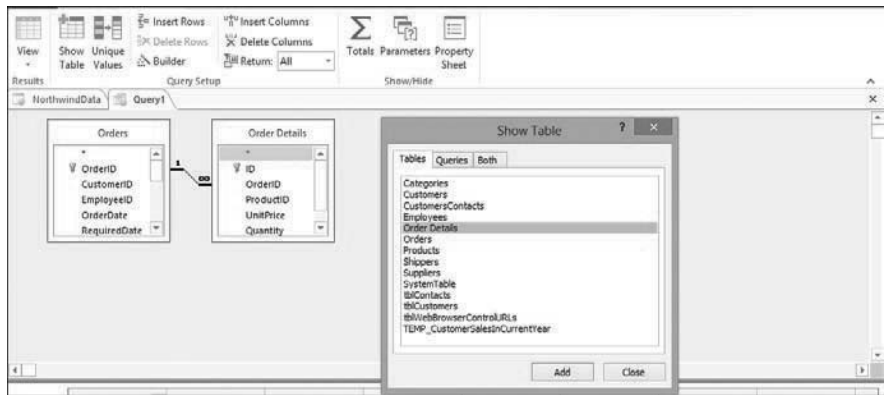


FIGURE 6-2 Adding tables or other queries to a query.

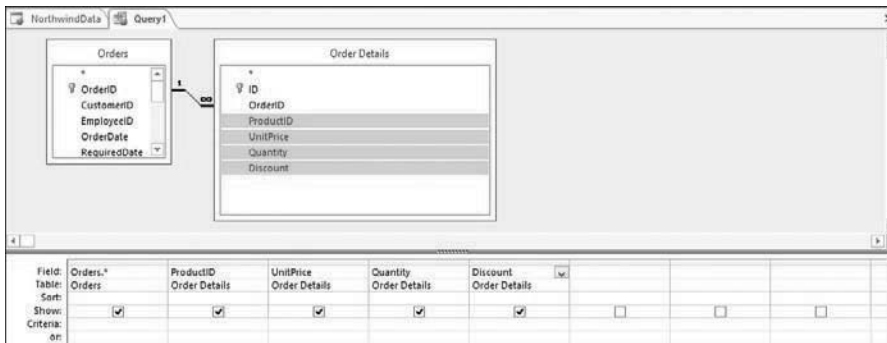


**Tip** Joining-relationship lines can be removed by clicking on the line and pressing delete. New lines can be added by clicking to select a field in one table and dragging the mouse over the related table. If tables are not joined, this produces a display multiplying all rows in one table by all the rows in another table. This is called a *Cartesian product*. Normally, you will not want to see this, but in certain situations it can be useful. In those situations, you are likely to either have only one record in one of the tables or need to add additional filtering criteria to the query grid to reduce the number of returned records. (You will see an example of this in Chapter 7.)

After you select the tables and queries to be used in creating a new query, the next step is to select the fields to work with. Fields can be added using several different techniques:

- Double-click or drag and drop the \* at the top of the list of fields to add all fields to the query grid. This would be shown, for example, as Orders.\*
- Double-click or drag and drop a single field onto the query grid.
- Press the Shift key to select multiple fields, and then drag and drop the multiple selections onto the query grid.
- From the query grid, in a blank column, use the Field drop-down list to select a field.

Select all the fields from the Orders table by dragging and dropping the \* at the top of the field list for the Orders table onto the query grid, as shown in Figure 6-3. Then select multiple fields—such as ProductID, UnitPrice, Quantity, and Discount—holding down the shift key to select the fields from the Order Details table, which are then dragged onto the query grid.

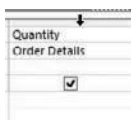


**FIGURE 6-3** Adding fields to the query grid.



**Tip** The advantages of using the \* to select all fields are that it's easy and does not clutter up the query grid. The disadvantages are that you need to again select again a field in the set if you want to sort or filter by the field. Also, if you ask for more fields than needed, displaying all the data will take more time than if you select only the fields you need.

Fields on the query grid can be selected by clicking just above the field name, as shown in Figure 6-4. After a field is selected, the column will appear darkened to indicate it is selected. After the fields are selected, they can be re-ordered using drag and drop, or deleted by pressing the Delete key; the Shift key can be used to select multiple fields.



**FIGURE 6-4** Hovering the cursor to select a field.



**Tip** Avoid selecting to show the same field name more than once. If you use the \* and then need to add a field for filtering or sorting, clear the Show box below the added field. This means the field can be used for filtering or sorting but is not displayed. Having a field selected more than once is allowed, but that can lead to confusion when you are linking views to the query or working with data macros.

A field name can be changed by prefixing the field using an alternative name ending with a colon, as shown in Figure 6-5. I displayed the Zoom box by pressing Shift+F2 and adjusted the display font to display the expression.



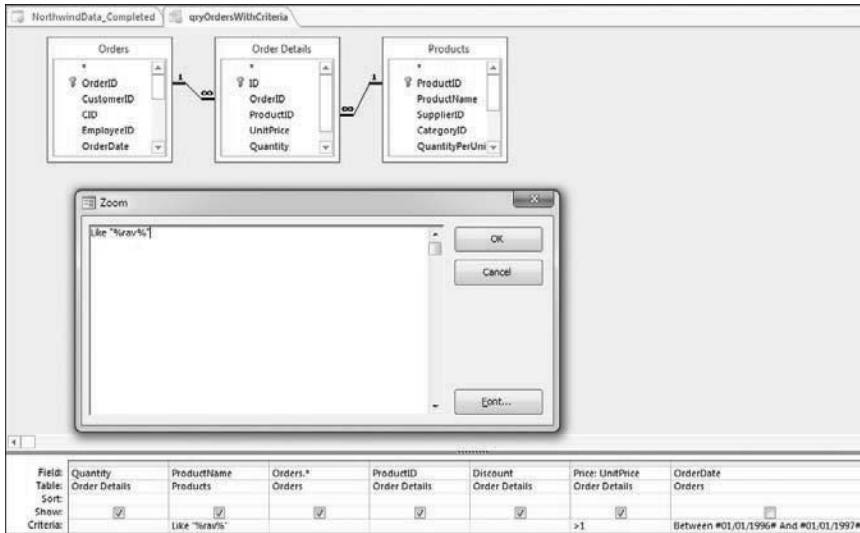
**FIGURE 6-5** Changing the name of a field.

You can display the results of your query by clicking on the View icon, which is located at the top left on the Design ribbon. You will be prompted to save the query before viewing the results. Save the query with the name *qryOrdersWithCriteria*.

## Adding criteria to queries

Figure 6-6 shows an example of criteria being added below some of the fields. I started with the previously created query, *qryOrdersWithCriteria*, and used the Show Table icon on the design ribbon to add the Products table to the query. Note that I cleared the Show box below [Order Date] because I already included this field using *Orders.\**. Also, note that the date range is displayed using the standard # symbols around the date, but the *like* criteria uses the % symbol as a wildcard and not the \* (which is not supported). You can surround the text string by either single or double quotation marks. I chose double quotes in this example.





**FIGURE 6-6** Sample query with criteria against numeric, date, and text fields.

If you looked at this saved query (which is saved as a view) in Azure SQL Database, you would notice that the criteria looks like this:

WHERE

```
[Order Details].[UnitPrice] > 1.0 AND [Orders].[OrderDate]
BETWEEN DATEFROMPARTS(1996, 1, 1) AND DATEFROMPARTS(1997, 1, 1)
AND [Products].[ProductName] LIKE N'%rav%'
```

From this code, you can see how the web app displays the information in a format that is more familiar to people working with Access for the date criteria, but it actually saves the query in a true Azure SQL Database format.



**Tip** For a wildcard search against text criteria, to save typing you can enter **\*rav\***. This will then be displayed as *Like '\*rav%'*. You can then change the \* to %. Although the resulting view will use single quotes, the information you enter is preserved for display in either double or single quotes, depending on what you typed in. Getting used to using single quotes will help when switching between the web app and the Azure SQL Database. When searching on a Yes/No field, you cannot use True/False; you can use Yes/No or 1/0 or <>0/0. (In the Azure SQL Database, 0 is false and 1 is true.)

The query grid has multiple lines, each starting with the keyword OR. This allows you to construct several alternative sets of criteria for filtering the data. When you have multiple criteria against multiple columns on a single line of criteria, they act together to filter the data, such as criteria1 AND criteria2 AND criteria3. Each line of criteria acts as a set of alternatives. If we had criteria1 on the first line and criteria2 on the second line this would be criteria1 OR criteria2.

## Adding calculations to queries

If you are familiar with creating calculations using a desktop database, you will find that the calculations in a web app are significantly different. The reason for this is that the queries are stored in Azure SQL Database as views and must conform to the SQL language syntax used in the Azure SQL Database, the built-in Azure SQL Database functions, and additional Access support functions in the Azure SQL Database—not the syntax and functions used in a desktop database.



**Tip** The query calculation syntax does not allow the use of the & symbol when concatenating strings. In a desktop database, developers often use the & when concatenating strings because it protects against a NULL value causing the resulting string to be NULL. In the web app, you must use the plus sign (+) when concatenating strings and use a built-in function to avoid problems with NULL values, such as COALESCE([FieldName], ''). The COALESCE function takes multiple arguments and returns the first non-null value.

There are 52 built-in functions you can use in web app queries, together with 25 constants. Rather than simply listing all of these (which you can see when using the builder in the query design tool), I will provide a few guidelines about using popular functions to get you started constructing expressions.

The Azure SQL Database has more strict rules than the desktop Access environment when it comes to adding together different data types and performing arithmetic operations—for example, on dates.

For example, if you try to construct the following expression, you will get an error:

```
NameAndID: [Products].[ProductID]+' '+[ProductName]
```

In this case, you need to convert *ProductID* to a string, and you do this by using the CAST function to change one data type to another data type. (Note that the data types used here—for example, ShortText—are not true Azure SQL Database data types; they are the data type names used when creating fields in a web app.)

```
NameAndID: Cast([Products].[ProductID],ShortText)+' '+[ProductName]
```

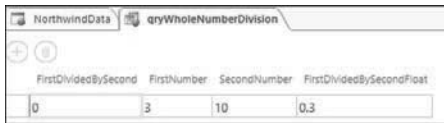
If you want to remove spaces from the beginning and end of a string, you need to use the following syntax because the *Trim* function is not supported:

```
NoSpaceField: LTRIM(RTRIM([FieldName]))
```

To replace a substring inside a larger string, use the REPLACE function. To extract part of the text in a string, use the SUBSTR function. There are also STUFF and REPLICATE string functions.

Another useful feature in a query is if you divide, for example, an integer by 10, you will find that in the view this is replaced by division by 10.0. This is because in the Azure SQL Database  $3/10 = 0$  and  $3/10.0 = 0.3$ . However, you need to be a bit careful with this. Figure 6-7 shows the result of dividing

two whole-number columns and an alternative calculation that forces the divisor to be a floating point number. Note that this happens only because we are using whole-number columns for the FirstNumber and SecondNumber fields.



FirstDividedBySecond	FirstNumber	SecondNumber	FirstDividedBySecondFloat
0	3	10	0.3

**FIGURE 6-7** Dividing 3/10 is zero when using integer arithmetic.

The two expressions are as follows:

FirstDividedBySecond: [FirstNumber]/[SecondNumber]

FirstDividedBySecondFloat: [FirstNumber]/Cast([SecondNumber],Float)

Calculations can also be constructed using the Builder icon (shown in Figure 6-8) on the design ribbon, or by right-clicking in a column header or criteria and selecting Build.



**FIGURE 6-8** Expression Builder.

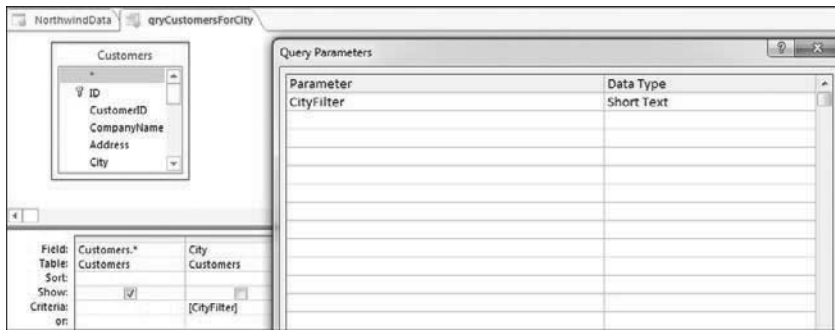


**Tip** You might think that the integer division issue is a bug or problem in the web app, but this is not the case. This is simply a contrast between how an Access desktop database performs a calculation and how the Azure SQL Database performs calculations. You will also find that when you add a Number field to a table, the default Number Subtype is Fixed-Point Number (6 decimal places), and because this is not an integer, the division issue does not arise.

## Adding parameters to queries

Parameters enable you to create a query that has a placeholder that can be populated at runtime to filter the data. Unlike a desktop database, where a user can directly open a query (or an object using the query) and enter a value for the parameter, in a web app you need to provide the values for any parameters before opening a view that has been constructed to use the parameterized query.

You start by creating a query from the advanced menu. For this example, select all fields from the Customers table, select the City field, and clear the Show check box for this field. Then click on the Parameters icon on the design ribbon, select a name for your parameter (CityFilter), and specify an appropriate data type (Short Text), as shown in Figure 6-9. Click OK to dismiss the Query Parameters dialog when you finish defining the new parameter.

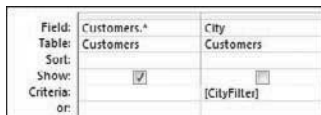


**FIGURE 6-9** Creating a query and adding a parameter.



**Tip** Ensure that you do not enter any spaces in your parameter names, because you will be prevented from saving a query when the parameter names contain spaces. If you use parameters in a query, ensure that you use the Query Parameters window to define the parameters.

As you start to type the parameter name in the criteria below the City field, IntelliSense will assist in selecting the parameter. Once you are done, the parameter name is shown in square brackets, as you can see in Figure 6-10.



**FIGURE 6-10** Adding the parameter as a filter against an existing field.

If you look at this using SQL Server Management Studio (SSMS) in the Azure SQL Database, you would find that your query is saved using a special type of function called a *table-valued function*. This is shown in Figure 6-11. (I deleted some fields in the select statement to make this clearer.)

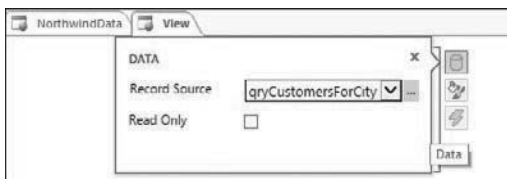


**FIGURE 6-11** Displaying the resulting table-valued function in SSMS.



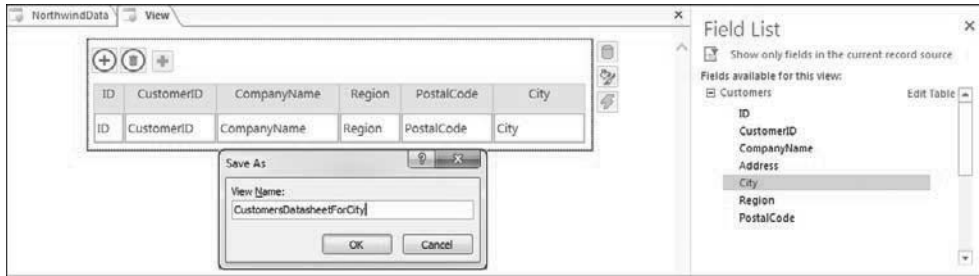
**Tip** The advantage of Access using a table-valued function in the Azure SQL Database rather than a stored procedure is that the table-valued function can return results that might be updateable, whereas the results from a stored procedure are always read-only.

After creating the parameterized query, you then construct a View that uses the parameterized query as a record source, as shown in Figure 6-12. However, you will not be able to directly open the view from the table selector, because we must use the OpenPopup macro command to supply a value for the parameter. This means you should create this as a standalone view using the Advanced Menu. Choose to create a datasheet view.



**FIGURE 6-12** Selecting the parameterized query as a datasource for a datasheet view.

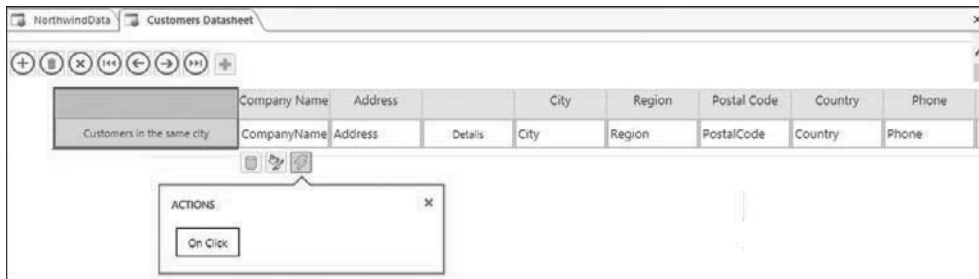
Figure 6-13 shows the Field List on the right being used to select the fields ID, CustomerID, CompanyName, Region, and PostalCode to display on the datasheet. The view has been saved with the name CustomersDatasheetForCity.



**FIGURE 6-13** Adding fields to the datasheet view, and saving the view.

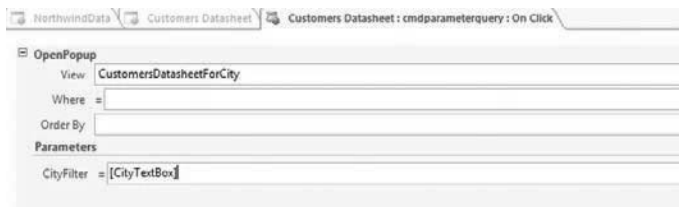
You have now created a parameterized query and associated view. Next, you need to provide a point from which to display the data. Normally, a blank form with selection controls is a good choice of starting point, but here you will add a button to the Customers datasheet. Clicking the button displays your view, which shows customers in the same city for the currently displayed customer.

Figure 6-14 shows a button on the Customers datasheet that will be used to open the new parameterized view.



**FIGURE 6-14** Adding a button to open a parameterized view.

By clicking the On Click event for the button, you can then add the OnPopup macro action and select your CustomersDatasheetForCity view. As shown in Figure 6-15, a great feature of a parameterized view when combined with the OpenPopup action is that it automatically identifies the parameters and displays a list of the parameters on the macro design surface, which can then be set to reference a control or field on the view. In our example, I refer to the control called [CityTextBox].

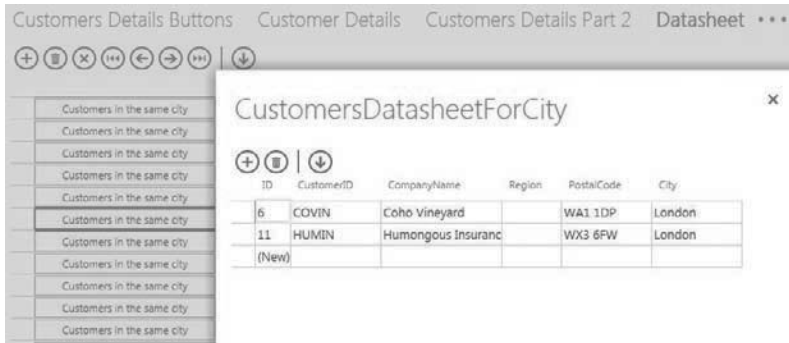


**FIGURE 6-15** Adding the OpenPopup macro action and providing the Parameters.



**Tip** In Figure 6-15, the City field value is referred to by using the control name [CityTextBox], because that is the option offered with the IntelliSense. However, you could have entered the equivalent field name [City].

In Figure 6-16, you can see the related records displayed in the browser window.

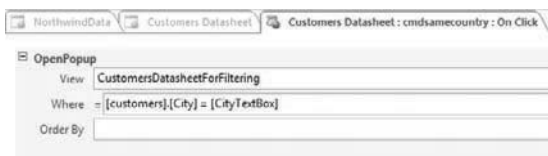


**FIGURE 6-16** The view will display another popup window based on a parameterized query.



**Tip** If you edit data in the popup, when you close the popup, the underlying data will not reflect any changes. It is not possible to automatically refresh the data, but you could add an action button that uses the RequeryRecords macro action to refresh the data.

Looking back at Figure 6-15, you'll notice that the OpenPopup macro action also supports a Where argument for filtering. This can be used either as an alternative to using a parameterized query or to provide additional filtering. Figure 6-17 shows the equivalent macro argument expression used to filter by city without using a parameterized query.



**FIGURE 6-17** OpenPopup macro action with an expression entered in the Where argument.

It is up to you whether you want to use parameterized queries or *where* clauses. By choosing the *where* clause approach, you can use the target view in several parts of an application, by using different *where* clauses. The parameterized query is fixed in terms of the parameters. But the parameterized query can seem simpler to work with, because it automatically detects the parameters and provides separate options for each parameter.



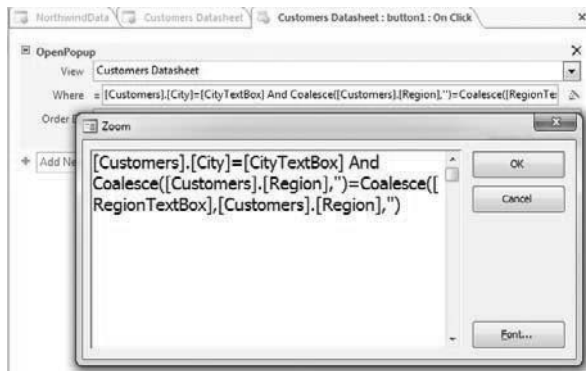
**Tip** In a parameterized query, you can create columns that use the parameter values for calculations and other expressions. You cannot do this using the *where*-clause technique.

In the Customers table, you assume that all customers have a city, but sometimes a customer also has a region. The question is how you can make your parameters conditional so that there is additional filtering by region when the customer has a value for the Region field. You want to construct optional parameter filtering.

The choice of whether to use a parameterized query to solve this or a *where* clause depends on how confident you are at writing *where* clauses, and how much you prefer to have popup boxes and graphical assistance. You should also consider, if you start to add more conditional logic, which method would be simpler for you to remember and understand.

The *where* clause approach with the OpenPopup macro action is shown in Figure 6-18 and uses the *where* clause:

```
[Customers].[City]=[CityTextBox] And Coalesce([Customers].[Region], '')=
Coalesce([RegionTextBox],[Customers].[Region], '')
```



**FIGURE 6-18** Conditional logic in a *where* clause for optional filtering.

Remember that the COALESCE function returns the first non-null value. So if a for a customer record you have a value in [RegionTextBox], this is compared against Coalesce([Customers].[Region], ''). [Region], ''), which will either match the region value or return an empty string that will not match the region value.

If the current record has no value or NULL in [RegionTextBox], the expression Coalesce([RegionTextBox], [Customers].[Region], '') returns either the region value [Customers].[Region] or an empty string. This is equivalent to comparing [Customers].[Region]= [Customers].[Region] or "".

You might be wondering why you create the comparison '' = ''. The reason is that you cannot allow the comparison NULL = NULL, because this expression is never true in a database or web app and would not match any records.

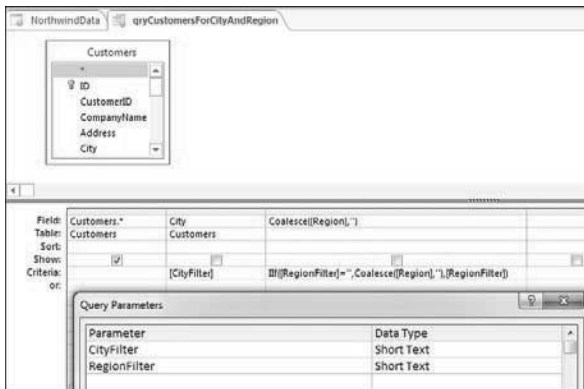




**Tip** You could, as an alternative to our logic, use IIF conditional logic in these expressions. You will see an example of that in the following description of using a parameterized query.

Before I show how to create a parameterized query approach to this problem, you need to understand that you are not allowed to have a parameter that can have a value of NULL, because Variant type parameters are not supported. This means that in a later step you will force the values provided to your parameterized query to have a safe value, such as an empty string, when the value is NULL.

Figure 6-19 shows a parameter query with two parameters, [CityFilter] and [RegionFilter], which are both Short Text data types. The [CityFilter] has already been explained earlier in this section. For the region filter, you create a column expression `Coalesce([Region], '')`. (Remember you can't compare NULL values, and you stated that NULL values will be converted to empty strings.) Then you compare this in the where clause against the expression `IIF([RegionFilter]='', Coalesce([Region], ''), [RegionFilter])`. The explanation of this logic is the same as previously described for the *where* clause example.



**FIGURE 6-19** Creating a parameterized query with optional parameters.

If you look in the Azure SQL Database, you would see the following table-valued function. (I removed the list of fields to focus on the important part of the SQL—the parameter data types and the *where* clause.)

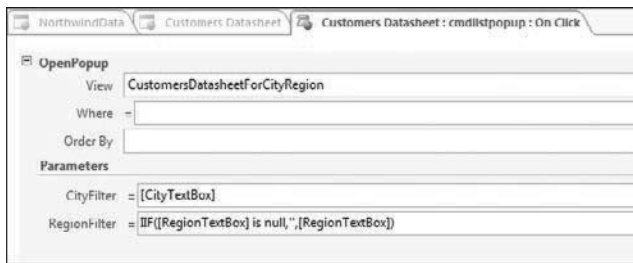
```
CREATE FUNCTION [Access].[qryCustomersForCityAndRegion]
(
    @CityFilter NVarChar(4000),
    @RegionFilter NVarChar(4000)
)
RETURNS TABLE
AS
RETURN
(
```

```

SELECT
    ...
FROM
    [Access].[Customers]
WHERE
    [Customers].[City] = @CityFilter
    AND COALESCE([Region], N'') =
        IIF(@RegionFilter = N'', COALESCE([Region], N''), @RegionFilter)

```

Next construct a new view based on our parameterized query called *CustomersDatasheetForCityRegion*; this method for constructing a standalone datasheet was described earlier in this section. Then add a button on your customers datasheet view to open the parameterized view as shown in Figure 6-20. Note that the RegionFilter parameter uses the expression `IIF([RegionTextBox] Is Null, '', [RegionTextBox])`; remember you indicated earlier that you cannot pass NULL as a parameter, so you convert that value to an empty string.



**FIGURE 6-20** Protecting a parameter from passing a NULL value.



**Tip** Another benefit of using parameterized queries rather than a *where* clause is improved performance. Because the parameterized query is saved as an object in the Azure SQL Database, it can offer slightly better performance.

## Totals and queries

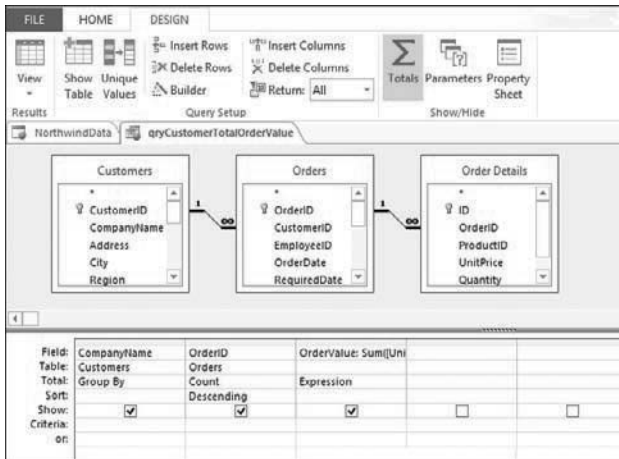
In this section, we look at how to use the Totals feature to produce summarized data; these queries are read only.

Figure 6-21 shows the new query with the Totals icon clicked on the design ribbon. This results in a new Total row on the query grid. The default choice below each field is GroupBy in the Total row, where the query groups records with similar values. Other options include the aggregate functions SUM, Avg, Min, Max, Count, the statistical function StDev, Var, Expression for calculations, and Where for filtering data.



**Tip** When working with a calculation and entering an aggregate such as SUM in the Total row, after saving and re-opening the query, you will see Expression in the Total row and the SUM operation moved into the column title. This is because the web app standardizes the way it saves the SQL. So the look of your layout can change, although the resulting data that is displayed will be unchanged.

In our example we have the expression `OrderValue:SUM([UnitPrice]*[Quantity])`.



**FIGURE 6-21** A totals query allowing data to display summarized calculations.

The datasheet preview for this Totals query is shown in Figure 6-22. In the results of this query, you can see that grouping for each company name displays the count of orders (`CountOfOrderID`) and sum of order values (`OrderValue`) as totals.

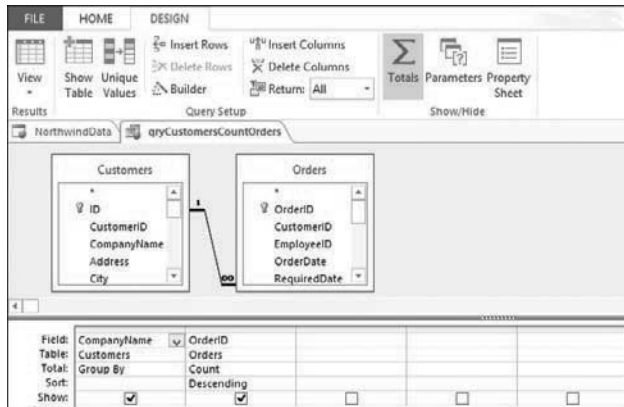
Company Name	CountOfOrderID	OrderValue
Litware, Inc.	102	113236.68
Proseware, Inc.	48	28722.71
Coho Vineyard	30	13806.5
Contoso, Ltd	27	19230

**FIGURE 6-22** Summarized results of a Totals query in the datasheet preview.

## Top value queries

Another feature of queries is to limit the number of records displayed, either specifying this as a number of records or as a percentage of the total records that could be displayed.

Figure 6-23 shows a summary query that counts the number of orders for each customer.



**FIGURE 6-23** Customers sorted by the greatest number of orders.



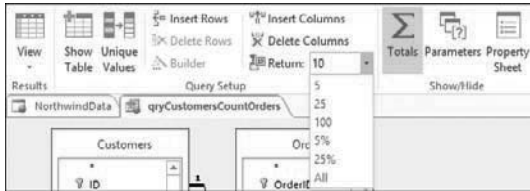
**Tip** In a web app, you cannot create a copy of an existing query. This means if you need several similar queries, you must create each one from a new query. (You can use the clipboard to copy and paste complex query expressions from one query to the next.)

Figure 6-24 shows the query results in the datasheet preview. Note that in the 13 records displayed, the last three records in the screen shot show the same count of 7. This is an important point to note for the following discussion of how the Top feature works in a web app.

Company Name	CountOfOrderID
Litware, Inc.	30
Proseware, Inc.	15
Coho Vineyard	13
Contoso, Ltd.	12
Wingtip Toys	11
Tailspin Toys	10
The Phone Com	9
Graphic Design	8
Humongous Int	8
Margie's Travel	7
City Power & Li	7
Coho Vineyard	7

**FIGURE 6-24** Resulting datasheet preview showing customers sorted by the greatest number of orders.

Returning to the design view, you can use the Returns icon on the design ribbon to select to display only the top 10 records, as shown in Figure 6-25. (In our example, we typed in the value 10, which is not shown in the drop-down list of choices.)



**FIGURE 6-25** In the design view, a screen that shows the use of the Returns drop-down list to return the top 10 records.



**Tip** If you look at the query properties, you will see that although the ribbon displays the word *Returns*, the property is called *Top Values*.

Figure 6-26 shows the resulting top records.

Company Name	CountOfOrderID
Litware, Inc.	30
Proseware, Inc.	15
Coho Vineyard	13
Contoso, Ltd.	12
Wingtip Toys	11
Tailspin Toys	10
The Phone Com	9
Humongous Int	8
Graphic Design	8
City Power & Li	7

**FIGURE 6-26** The Top Values property returns the top matched records.

If you are familiar with using TOP in an Access desktop database, you might know that it would have returned more than 10 rows in the previous example. This is because if the last row ties with other rows, all matched rows that tie in value are displayed. This is called TOP WITH TIES. Because the web app relies on the Azure SQL Database, which by default does not show tied values, you see only the first of the records that tie for last place.



**Tip** The Azure SQL Database does offer a special TOP WITH TIES feature, but this is not available for you to use in a web app.

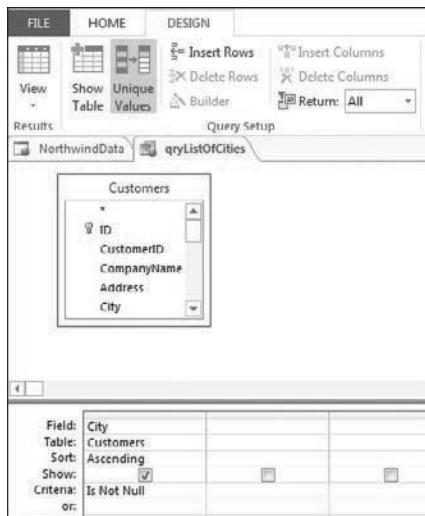
# Unique values in queries

When you create a query, one row will be displayed for each underlying record, regardless of which fields you choose and whether the output shows several records with the same value. The Unique values property enables you to eliminate any duplicate values and makes the output of each record unique across the combination of selected fields. These queries are useful when summarizing data, making lists of unique choices, and when re-organizing data to identify new lists of unique combinations with which to construct new tables. These queries are read only.



**Tip** Although you cannot create a “make table” query or “append” query in a web app, you can create an empty table and then use SSMS to execute an “append” query to populate the data.

As an example of this, Figure 6-27 shows a query to select the City field from the Customers table and display a list of unique values when clicking on the Unique Values icon on the design ribbon. We also add a criteria to exclude any null values and sort the results.

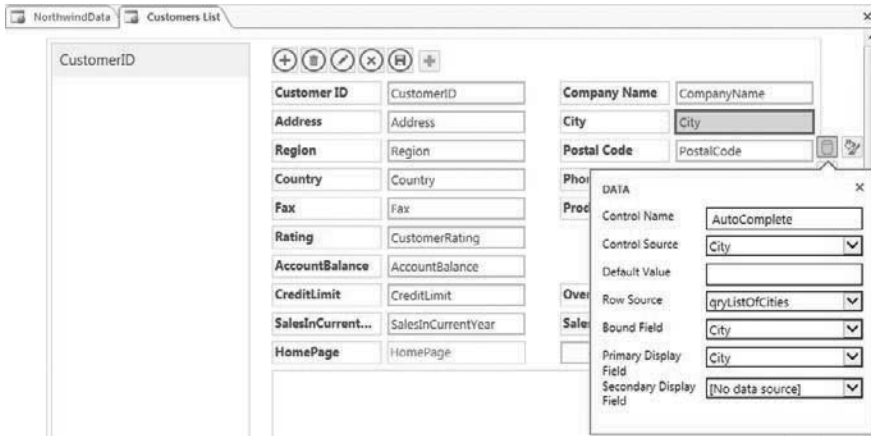


**FIGURE 6-27** Specifying the Unique Values setting for a query.



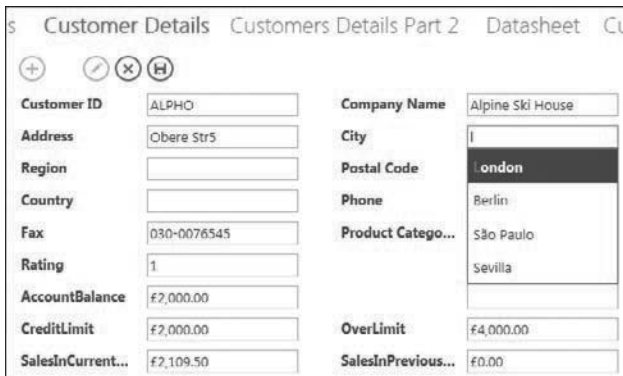
**Tip** When you specify unique values in the Azure SQL Database, the `SELECT DISTINCT` syntax will be added to the query. An equivalent technique to using the unique values property is to use a Totals query and the Group By operation.

You can then edit your customers view and replace the text box for the City field with an Autocomplete control, which uses the query `qryListOfCities` as the row source for the control, as shown in Figure 6-28.



**FIGURE 6-28** Using a Unique Values query in an autocomplete control.

In Figure 6-29, you can see the result of using the autocomplete control to display a list of values for the City selection based on the unique values used in the existing customer records.



**FIGURE 6-29** Completed view with autocomplete control.

In Figure 6-29, the autocomplete control sorts the results based on matching characters. The word *London* is displayed before *Berlin*, because the letter *l* is first matched as the first letter in the word *London*. This means that the autocomplete control will override any sorting you specified earlier when constructing the underlying query `qryListOfCountries` (where you had an ascending sort on the City name).



**Tip** This method, although it makes queries very quick to construct and avoids the need to create a new table, does not allow you to enter a new value for City, which is not already in the list of values. In this case, you need an alternative view, such as a datasheet view, to enter a new value.

## Summary

---

In this chapter, you saw how to create queries that can then be used to construct views that display your data. You can use queries to bring together data from several tables or other queries and construct calculated expressions. A query can contain criteria to filter the data, provide totals to summarize the data, use unique values to remove duplicate data, and limit the results using the Top property.

The most complex feature of working with queries is in using parameters to supply values to a query at runtime. This technique requires both the construction of a parameterized query and the use of a macro to supply the runtime parameters.

In Chapter 7, we will look at further examples of working with macros and queries to enhance your applications.



# Index

## Symbols

- & (ampersand), 200
- \* (asterisk), selecting with, 197–198

## A

- About Me, 8
- Access
  - application types, 19
  - backstage view, 24
  - in browsers, 5
  - design tool, 136
  - downloading web apps to, 23–24
  - DSNs, creating, 311
  - interrelation with Office 365 and Azure SQL Database, 51–53
  - Office-related files, 15–16
  - opening web apps with, 23–24
  - Service Pack 1 updates, 82–96, 23–24
  - starting, 15
  - Team Site, browsing from, 18
  - view preview, 157
- Access App button, 27
- Access design environment
  - building apps in, 57–61
  - controls in views, 73–80
  - custom web apps, 61–62
  - On Deploy Macro feature, 84–85
  - displaying data in browser, 68–71
  - importing data from, 62–64
  - locking applications, 83–84
  - locking tables, 83
  - lookups, 65–68
  - macro programming, 80–82. *See also* macro programming; macros
  - navigation pane, 64–65
  - packaging options, 85–86. *See also* packages
  - pop-up capability, 61
  - relationships, 65–68. *See also* relationships
  - tables, 65–68
  - upgrading packages, 93–96. *See also* upgrade process
  - uploading packages, 86–93
  - views, 71–73. *See also* views
- Access.Documents view, 309–310
- AccessRuntime schema, 308
- Access schema, 308
- AccessSystem schema, 308
- action bar, 51, 77, 222
  - built-in buttons, 172
  - custom buttons, 222–225
  - hiding commands, 297
  - navigation buttons, adding, 160–161
  - Undo button, adding, 159–160
- action buttons, 160
- Action Catalog, 220
- Action charm, 160
- Active Data Objects (ADO), 119–120
  - connections with, 328
- Active Data Objects Extension (ADOX) Library, displaying structural information, 327–329
- Add Button* macro action, 223
- Add Else command, 224
- Add Else IF command, 224
- Add Tables screen, 62, 101, 135–136
- Administrator users, 292
  - identifying, 295
  - restricting view to, 296
- ADODB Library, adding references to, 126
- Advanced menu
  - Query option, 173–174, 196
  - view options, 177–178
- AFTER DELETE triggers, 332, 334
- AFTER INSERT triggers, 332, 334

## first first-level entry

- After Triggers, 241, 332, 334-217
- After Update data macro, 331
- After Update event, 226
- AFTER UPDATE triggers, 332, 334
- AJAX. *See* Asynchronous JavaScript and XML (AJAX)
- aliases with macro actions, 287–288
- ampersand (&), 200
- anonymous users, managing, 294–295
- Anonymous Users group, 290
- API Tutorial For Office page, 385–386
- App Catalog, 53–55
  - Apps For SharePoint page, 87–88
  - displaying, 53–54
  - upgrades with, 95
  - uploading apps to, 54–55
  - uploading deployment packages to, 87–89, 92–93, 285
  - uploading snapshots to, 86–88, 278–279
- appdb.dacpac file, 93
- append queries, 324
- .app file extension, 30
- App Home View page, 249
- applications, locking, 83–84
- app manifest, 30, 376
  - for apps for Office, 383
  - GUID attribute, 85–86
  - version number attribute, 85–86, 91
- apps for Access, 375–377
  - adding to List or Blank view, 377–379
  - app manifests, 376
  - consuming, 377–380
  - data source, 380
  - developing, 381–386
  - displaying available apps, 378
  - interaction with web apps, 377
  - linking to data in web app, 379–380
  - Select Data button, 378–379
  - webpage component, 376
- apps for Office, 375–386
  - app manifest, 383
  - Basic App, 382
  - components, 375–376
  - creating, 381–386
  - data source, 380
  - Document Visualization App, 382
  - forms, 376
  - object map, 385
  - presentation features, 380
  - running, 384
  - visualization content, 383
- Apps For Office control, 380
- Apps In Testing folder, 22, 272
  - removing web apps from, 277
- ASP.NET Ajax Control Toolkit, 367
- ASP.NET applications
  - connections, adding, 368–369
  - displaying data in, 364–374
  - launching in browser, 372–373
  - select statement, configuring and testing, 369–370
  - Web Forms, 366–374
- asterisk (\*), selecting with, 197–198
- Asynchronous JavaScript and XML (AJAX), 217
- Attachment data type conversion, 121–132
- attachments
  - converting, 115
  - extracting, 123–125
  - hyperlinking to records, 125
- autocomplete control, 67, 74–75, 190–191
  - as data source, 246–247
  - Unique values property, 212–213
- AutoNumber data type as primary key, 106–107
- AutoNumber field not primary key, 101
- Azure SQL Database, 301–338
  - check constraints, 330–331
  - connecting to, 303–307
  - connecting with Excel, 339–344
  - connecting with SQL Server, 365
  - database connection options, 304–305
  - data macro execution in, 216
  - data macros, 331–335
  - direct connections, constructing, 115–121
  - DSN-less ODBC connections, 310–316, 319–321
  - firewall rules, 302, 304–305
  - indexing, displaying, 143
  - instances, 52
  - integer division, 200–201
  - interrelation with Access and Office 365, 51–53
  - ODBC drivers for connecting, 303
  - queries, saving as views or table-valued functions, 336–337
  - referential integrity, 141
  - relationships, information on, 321–330
  - schemas, 308
  - security of design, 302
  - table names, listing, 320
  - table-valued functions, 203, 336–337
  - validation rules, 330–331
  - VARBINARY(MAX) field, 130
  - views, 336–337

**B**

- backstage view, 24
- backups with snapshots, 24–25
- Bell, Kevin, 129
- bit fields, 112
- blank desktop databases, 19
- blank view, 177–180
  - creating, 152–156
- Boolean data, importing, 112
- bound controls, 187, 191
- browsers
  - displaying data in, 68–71, 372–373
  - displaying Excel spreadsheets in, 344–350
  - displaying web apps in, 22, 32, 59–60
  - Keep Me Signed In option, 18
  - macro execution in, 216
  - Office products in, 5
  - page layout, 253
  - On Start macros, 61
- Builder icon, 201
- built-in actions, controlling, 222–225
- business rules, enforcing with indexes, 142–143
- buttons
  - adding to views, 152–153
  - built-in, 172
  - captions, formatting, 162
  - On Click events, 162
  - control properties, 173
  - displaying, 171–172
  - enabling and disabling by context, 171
  - hiding, 171–172, 297

**C**

- calculated fields, 115, 145–146, 174
- calculations
  - adding to queries, 200–201
  - query syntax, 200
- captions, browser title bar, 172
- captions, table
  - deleting, 151
  - hiding, 150, 155
  - renaming, 150, 182
  - re-ordering, 150–151
  - views, duplicating to, 167–169
  - views associated with, 151–152
- captions, view, 182
- Cartesian product, 196
- cascading deletes, 102–103, 141
- cascading updates, 103
- CAST function, 200
- Change The Look option, 45
- ChangeView* macro action, 161, 185–186, 266, 296
  - where* clause, 163
- charms, 64–65
- check constraints, 330–331
- cloud services and apps, 2
- Coalesce* function, 144–146, 200, 206
- code execution location, 216
- code indentation, 235–237, 264
- color, adding to view controls, 192
- columns
  - data properties, 158–159
  - renaming, 85
- combo box control, 187–189
- commissioning a system, 96–97
- communications, 2
- concatenation, string, 200
- Concat* function, 238
- conditional logic, 206
- connections
  - adding to website, 368–369
  - with ADO, 328
  - in ASP.NET applications, 368–369
  - in Azure SQL Database, 115–121, 303–307, 339–344, 365
  - to desktop databases, 317–319, 339–344
  - displaying, 303–304
  - DSN-less, 310–316, 319–321
  - DSN-less ODBC, 310–316, 319–321
  - with Excel, 339–344
  - ODBC, 310–319
  - in PowerPivot, 346
  - read-write, 120–121, 125–126
  - in Web Forms, 368–369
- connection strings
  - displaying, 117–118
  - managing, 118–119
  - passwords, saving in, 320–321
  - tabledef*, 310
- constants, 200
- Contribute permissions, 35
- controls
  - adding to action bar, 159–160
  - adding to Web Forms, 371
  - Apps For Office, 380
  - ASP.NET Ajax Control Toolkit, 367
  - autocomplete, 67, 74–75, 190–191, 212–213, 246–247

## first first-level entry

- controls (*continued*)
    - bound and unbound, 163, 187, 191
    - color, adding, 192
    - combo box, 187–189
    - for data display, 371
    - DetailsView, 371
    - displaying actions for, 153
    - enabling and disabling by context, 171
    - EntityDataSource, 367
    - events, 226
    - Filter Box, 72
    - GridView, 371
    - Hyperlink, 78–79
    - ImageControl, 131–132
    - LinqDataSource, 367
    - List, 72, 77–78, 169–170
    - ListView, 371–372
    - Multiline Text Box, 79, 189
    - positioning, 161
    - RELIC, 66–67, 72, 75–76, 141–142, 180–182
    - Report Viewer, 351
    - SQLDataSource, 367
    - Subview, 76–77, 182–186
    - synchronization, 188–189
    - in views, 73–80. *See also* view controls
  - control source, autocomplete control as, 246–247
  - conversion
    - Attachment data type, 121–132
    - of attachments, 115
    - of calculated fields, 115
    - desktop database to web app, 99–132
    - ExternalReader to ExternalWriter, 120–121
    - of graphical images, 115
    - Image data type, 115
    - of lookups, 102–107
    - of NULL values, 206–208
    - of OLE data types, 121–132
    - primary key changes, 100–102
  - Corporate Catalog, 53
  - Create Reports icon, 116
  - creating a Cartesian product, 233–234
  - criteria, adding to queries, 198–199
  - cross-tab queries, 262
  - cross-tabulation of data, 262–266
  - cursors
    - opening, 243
    - in triggers, 333
  - Customize In Access option, 270
  - Custom Web App icon, 61
  - custom web apps, 19
    - creating, 61–62
- ## D
- DACPAC, 96
  - data
    - cross-tabulation of, 262–266
    - deleting existing, 263
    - displaying in browser, 68–71
    - duplicate values, eliminating, 212–213
    - editing, preventing, 297
    - filtering, 199
    - importing from Access, 62–64
    - in Microsoft Azure SQL Databases, 52
    - multi-value, 113
    - read-only, 83
    - referential integrity, 68
    - refreshing, 205
    - relationships, 65–68. *See also* relationships
    - remapping, 228–232
    - replacing, 96–97
    - saving in packages, 25
    - security, 53
    - summarizing, 208–209, 263–264
    - test data and live data, 96
    - unions, 266
    - validation rules, 143–145
    - views, 59. *See also* views
  - Data Access Objects (DAO), 120
    - structural information, displaying, 323–327
  - database diagrams, 322–323, 327
  - Data charm, 158–159
  - data connections. *See also* connections
    - adding to website, 368–369
    - displaying, 303–304
    - with Excel, 339–344
  - Data Definition Language (DDL) SQL, 326
  - data macros, 80–81, 216. *See also* macro programming; macros
    - in Azure SQL Database, 331–335
    - buttons to run, 231–232
    - debugging code, 296
    - deleting existing data, 263
    - embedded, 217
    - error handling, 334–335
    - executing, 265
    - IF statement, 334
    - named, 228–239

- standalone, 217
- starting, 229
- stored procedure code, 229–231
- testing with SSMS, 295–296
- tracing, 247–248, 332
- transactions, initiating and save points, 333
- triggers, 239–247, 332–335
  - for Windows Live login lookup, 293–294
- data migration, ODM for, 274
- data properties of columns, 158–159
- data retrieval and indexing, 142
- datasheet preview
  - displaying queries in, 70
  - opening tables in, 69–70
  - of query results, 175
- datasheet view, 59, 70, 156–164, 177–180
  - Action Bar, 77
  - captions, formatting, 162
  - components of, 72
  - filtering and sorting feature, 157
  - images, displaying, 158
  - undoing changes, 156
- Data Source Names (DSN)
  - DSN-less connections, 310–316, 319–321
- Data Source Names (DSNs), 310
  - 32-bit and 64-bit drivers, 312
  - creating, 311
- data sources
  - configuring, 372
  - linking to display controls, 371
- data types
  - changing, 200
  - editing, 140–141
  - importing, 113–115
- data validation, 298–299
- debugging macro code, 296
- default joins, 103
- default values for Boolean data, 112
- DELETED table, 331–332, 334
- DELETED view, 240, 243
- DELETE FROM TRACE command, 248
- DeleteRecord* macro action, 223
- deleting sites, 38–39
- deployment packages, 87
  - creating, 92, 285
  - options, 25, 86
  - test or reference data, 274
  - uploading to App Catalog, 87–89, 92–93, 285
- deployments
  - locked, 272
  - saving, 268–269
- design
  - changing, 5
  - from existing data, 96–97
  - protecting, 83–84, 267
  - saving in packages, 25
  - upgrading, 91–92
  - user changes to, 95
- design interface, 137
- design preview, 71
- design tool, 57–61, 136
- design view
  - editing in, 137–139
  - lookups, managing, 139–141
  - for tables, 100
- desktop databases
  - attachment files, extracting, 123–125
  - blank, 19
  - connecting to Azure SQL Database, 317–319
  - connecting with Excel, 339–344
  - converting to web app, 99–132
  - creating, 19
  - data types, 47
  - default joins, 103
  - images, displaying, 130
  - importing, 63
  - integer division, 200–201
  - lookups, 66–68
  - referential integrity, 103. *See also* referential integrity
  - relationships, 65–67. *See also* relationships
  - saving to OneDrive For Business 2013, 15–16
  - Shift key selection, 179
- desktop design environment, 21
- desktop design tool, 57–61
- desktop interface for web apps, 5
- desktop reporting databases, 52
  - connection strings, 117–119
  - creating, 116–117
  - editing or updating, 117–118
  - ExternalReader to ExternalWriter conversion, 120–121
  - references, 119–120
- desktop templates, 19
- DetailsView control, 371
- deterministic expressions, 145
- development copy, 90
- development process, macro programming, 80–82

## development sites

- development sites
  - creating, 90–91
  - uploading packages to, 91
- direct recursion, 251–252
- document management with SharePoint Online, 2
- document storage, 47–51
- document upload to Team Site, 122–123
- downloading web apps, 32
  - to Access, 23–24
- DSN-less connections
  - schema prefixes, removing, 319
  - user name and password saving, 319
- DSN-less ODBC connections
  - creating manually, 310–316
  - creating with code, 319–321
- duplicate values, eliminating, 212–213

## E

- editing
  - app packages, 30–32
    - preventing, 297
  - table design, 137–139
  - view layout, 138
  - views, 71
  - web app packages, 30–32
- EditRecord* macro action, 223, 235, 265
- email
  - addresses, validating, 298–299
  - Exchange Online, 2
- embedded macros, 81, 217
- embedded queries, 163–166
- enterprise solutions, 272
- Enterprise subscription
  - Office Professional, downloading and setting up, 10–13
  - promoting sites, 37–39
  - site collections, creating, 39–44
  - Site Contents window, 8, 22
- EntityDataSource control, 367
- errors in upgrades, 272–273
- ESC key, 159
- events
  - of controls on views, 226
  - macros, attaching to, 222
  - set, extending, 222–226
  - table events, 217, 239–247
  - on views, 222–226
- Excel
  - connecting spreadsheets to Azure SQL, 339–344
  - PowerPivot, 344–350
- Excel web app, 345
- Exchange Online, 2
- existing data, replacing, 96–97
- Expression Builder, 143–144, 201
- external databases, linking to, 317–319
- ExternalReader users, 119, 301, 309
  - conversion to ExternalWriter, 120–121
- external users, 6
  - sharing web apps with, 32–35
- ExternalWriter users, 119–121, 301, 309

## F

- field list, adding fields, 165
- field names, changing, 198
- field properties, 108–111
- fields
  - adding to queries, 197
  - adding to views, 158, 178–179
  - calculated, 145–146
  - indexing, 143
  - names, 100
  - NULL, NOT NULL, and NULLABLE, 143
  - properties for, 108–109
  - recursion, detecting, 252
  - renaming, 85
  - validation rules, 143–145
- File DSNs, 310–312
- file extensions, visibility, 31
- Filter Box control, 72
- filtering data, 199, 302
  - in datasheet view, 157
  - parameters, 206
  - with *where* argument, 205
- firewall rules in master database, 302
- folder options, locating, 31
- FOR DELETE triggers, 332
- For Each loops, 243–244
- For Each Record In* macro action, 234–235
  - alias, 287
- foreign keys
  - creating, 106
  - displaying, 129
  - indexing, 142
  - lookup fields, 228
  - names, changing, 138–139

- updating, 106–107
  - values, displaying, 246–247
  - values, updating, 103
- FOR INSERT triggers, 332
- Formatting charm, 162
  - for controls, 159
- formatting for controls, 192
- FOR UPDATE triggers, 332
- functions, 200

## G

- Get External Data wizard, 62–63
- Get It button, 93
- global data centers, 4
- globally unique identifiers (GUIDs), 85–86, 88, 271
- global variables, 218–219, 294
- GoToControl* macro action, 227
- GoToRecord* macro action, 161, 227
- graphical images
  - converting, 115
  - uploading, 121–122, 125–132
- GridView control, 371
- group permissions, 34
- GUIDs, 85–86, 88, 271

## H

- hiding table captions, 150
- Hyperlink control, 78–79
- hyperlinks
  - adding to web apps, 291–292
  - enabling, 259–260
  - viewer.aspx syntax, 253–254, 257

## I

- icons for web apps, 30
- ID* field, 101–102, 137
- IF code blocks, 224
- IIF conditional logic, 207
- ImageControl control, 131–132
- Image data type, 121
  - converting, 115
- images, 123
  - converting, 115
  - displaying, 129–132
  - displaying in datasheet view, 158
  - uploading, 121–122, 125–132

## importing

- Boolean data, 112
- data from Access, 62–64
- data types, 113–115
- properties, 108–109
- relationships, 105–106, 108
  - and special characters in field and table names, 100
- tables, 103
- unsupported data types, 63–64
- validation rules, 108–111
- in-code DSN-less connections, 311
- indexing, 142–143, 302
- indirect recursion, 250–251
- INSERTED table, 331–332, 334
- INSERTED view, 240, 243
- INSTEAD OF triggers, 332
- integer arithmetic, 200–201
- intellectual property, protecting, 83, 95–96
- IntelliSense, 173
- Invite People window, 34
- Is Edited flag, 95

## J

- “JavaScript API for Office,” 385
- JavaScript Object Notation (JSON), 217
- joining-relationship lines, 196
- jump lists, 176–177

## L

- language support, 30–31
- Launch App icon, 59–60, 70
- layouts for printing, 253–261
- legacy data types, 64
- libraries, 47–51
- licensed users, 6
- linked SharePoint lists, 309–310
- linked tables, 317–319
  - creating with code, 319–321
- linking
  - to attachments, 125
  - to data, 379–380
  - to data sources, 371
  - to display controls, 371
  - to documents, 47–51
  - to external databases, 317–319
  - ODBC connections to desktop, 317–319

## linking

- linking (*continued*)
  - to SharePoint lists, 47–52, 309–310
  - web apps to reports, 363–364
- LinqDataSource control, 367
- List control, 72, 77–78, 169–171
- List or List Details view, 59, 169–173, 177–180
  - action bar, 77, 171
  - components of, 71
- ListRelationships.sql script*, 321
- ListView control, 371
  - configuring, 371–372
  - linking to data source, 371
- live systems
  - About link, 93
  - customizing in Access, 90
  - re-creating web app in, 86–87
  - upgrade process, 93–94
  - uploading deployment package to App Catalog, 87–89
- local variables, 218
  - declaring and initializing, 229
- locked tables upgrade process, 282
- locking
  - applications, 83–84
  - tables, 82–83
  - web apps, 95–96
- Lock property, 276
- Long Text data type, 113–115
- Lookup A Record In* macro action, 234–235, 244–245
  - alias for, 287
  - filtering data, 264
- lookups
  - converting, 102–107
  - in desktop databases, 66–68
  - displaying, 139–140
  - gathering in table captions, 167
  - value-based, 112, 140
  - in web apps, 103–104, 139–142
- Lookup Wizard, 104–105, 139–141
- LostProperties table, 109–111
- Lync Online, 2

## M

- Machine Data Source, 131
- Machine DSNs, 310–311
- macro actions, 223–224. *See also under individual macro action names*
  - aliases, adding, 287
  - Database Objects set, 227
  - deleting, 156
  - embedded, 237–238
  - Where Condition character limit, 287
- macro design window, 153
- macro editor, 219–221
- macro programming, 80–82, 215–287
  - aliases with macro actions, 287–288
  - code placement, 217–218
  - cross-tabulation of data, 262–266
  - data macros, 228–239. *See also* data macros
  - On Deploy macro, 274–286
  - side-loading web apps, 267–274
  - SSMS on back end, 215
  - standalone macros, 217, 219–221
  - On Start macro, 154–155, 249–250
  - tracing, 82, 247–248, 332
  - triggers, 239–247
  - user-interface macros, 227. *See also* user-interface macros
  - variables, 218
  - web app capabilities, 216–219
- macros, 217
  - actions, 220
  - arguments, setting, 153–154
  - attaching to events, 222
  - embedded and standalone, 81
  - executing, 221
  - execution scope, 82
  - global variables, 82
  - parameters, 82
  - On Start macros, 154–155, 249–250
  - syntax, 82
  - testing, 220, 245–246
  - tracing tables, 82
- master database
  - firewall rules, 302
  - permissions on, 339
- Members users, 292
  - permissions of, 35
- memo fields, 113–115
- Microsoft Azure SQL Database. *See* Azure SQL Database
- Microsoft Office 15.0 Access Database Engine Object Library, 119–120
- Microsoft Office web apps, 375–386. *See also* apps for Access; apps for Office; web apps
- Microsoft SQL Server Management Studio (SSMS). *See* SQL Server Management Studio (SSMS)
- Microsoft Windows 8, 1–2



*modADOX\_ConnectToCatalog* routine, 328–329  
*modADOX\_GetTableNames* routine, 329  
*modDAO\_BuildTableSchema* routine, 324  
*modDAO\_CreateRelationships* routine, 326  
*modDAO\_CreateTables* routine, 325–327  
*modDAO\_CreateTargetDatabase* routine, 325  
*modDAO\_GetConnectionString* routine, 324  
*modDAO* routine, 325  
 model.sql file, 93  
 Move Down button, 236  
 Move Up button, 237  
 Multiline Text Box control, 79, 186  
 multi-value data, 113  
 my.sharepoint.com collection, 44

## N

named data macros, 217. *See also* data macros;  
 macro programming
 

- input parameters and return values, 218
- stored procedures and, 228–239

 named queries, 166. *See also* queries  
 naming conventions, 86  
 navigation, 59–60
 

- from popup window, 155–156
- between views, 184

 navigation buttons, adding to action bar, 160–161  
 navigation pane, 64–65
 

- displaying, 64–65
- duplicating views, 168–169
- macros, viewing, 82

 nested macros, 287. *See also* macro programming  
 nesting triggers, 251. *See also* triggers  
*NewRecord* macro action, 223  
*N* prefix, 114  
 NULL, NOT NULL, and NULLABLE expressions, 143, 146  
 NULL values, passing or converting, 206–208

## O

ODBC connections
 

- creating, 314
- linking from desktop to, 317–319
- manually creating, 310–316

 ODBC Data Source, 312  
 ODBC Data Source Administrator
 

- data source, creating, 314–315
- data source, testing, 316

 default database, 315–316  
 ODBC drivers, 313  
 SQL Server authentication, 315  
 System DSN tab, 314  
 ODBC Data Sources program, 312–313  
 ODBC drivers, 303, 314  
 Office 365
 

- account, 12–13
- Admin center, 10
- Apps In Testing folder, 22
- benefits of, 3
- downloading and setting up, 10–13
- downloading web apps from, 23–24
- getting started, 5–7
- global data centers, 4
- icons for web apps, 30
- image data, 123
- interrelation with Access and Microsoft Azure SQL Database, 51–53
- key features of, 4, 6–7
- OneDrive selection, 14
- public-facing web apps in, 291–292
- services of, 2–3
- signing in, 7
- sites, 7–9
- subscriptions, 2–5
- uploading packages to, 26–29

 Office 365 ProPlus, 2  
 Office API, 381  
 Office Developer Tools for Visual Studio, 381  
 Office products in browsers, 5  
 Office Professional
 

- Desktop products, 2
- installing, 10–13

 Office Store
 

- contributing apps to, 376
- locking applications for, 84, 95–96

 OLE data types
 

- conversion, 121–132
- uploading, 121–122

 OLE objects, 115
 

- in Image field, 130
- VARBINARY(MAX) field, 130

 On Click event, 226
 

- displaying, 162

 On Current event, 222
 

- add event handling, 226
- button for List Details view, 171–172

 On Delete event, 240

## On Delete (After Delete) macro action with [OLD]

- On Delete (After Delete) macro action with [OLD], 243–244
- On Deploy Data Macro (ODDM), 274
- On Deploy Macro (ODM), 84–85, 274–286
  - creating, 282–283
  - indentation levels, 284
  - testing, 284
- OneDrive, 6, 13–18
- OneDrive For Business 2013, 13–18
  - desktop databases, saving to, 15–16
  - folder options, 31
  - mapping to desktop drive, 15
  - synchronization options, 14
  - web apps, saving to, 15
- OneDrive Personal, 17
- On Insert event, 239–241
- On Insert (After Insert) macro action, 245
- On Load event, 222
  - button for List details view, 171–172
  - SetProperty*  macro action, 192
  - ViewState\_Open*  action, 226
- on-premises Microsoft SharePoint 2013 Server with Access Services, 1
- On Start macro, 61, 81, 218, 249–250
  - adding macro commands to, 154–155
  - creating and activating, 249–250
  - user logins, identifying, 293–295
- On Update event, 240–242
- On Update (After Update) event, 241
- Open Folder error message, 18
- OpenPopup*  macro action, 154–155, 162–163, 186
  - parameterized views and, 204
  - where*  argument for filtering, 205–206

## P

- packages
  - contents, displaying, 30
  - DACPAC, 96
  - defined, 24–25
  - design and data in, 25
  - editing, 30–32, 90–91
  - file extensions, changing, 90–91
  - location, 25
  - moving web apps with, 9
  - naming, 25
  - packaging options, 85–86
  - saving web apps as, 24–25
  - titles, changing, 273
- uploading, 26–29
  - uploading to App Catalog, 87–89, 92–93, 285
  - uploading to development sites, 91
  - version numbers, 85
- parameterized queries, 202
  - columns with parameter values, 206
  - optional parameters, 207
  - as record source, 203
  - as saved in Azure SQL, 337
  - views associated with, 203–205
- parameters
  - adding to queries, 202–208
  - creating, 358–359
  - data type, defining, 359
  - filtering, 206
  - listing, 204
  - names, 202
  - NULL values, 207
  - passing within a URL, 363
  - updating, 237, 239
  - views associated with, 203–204
- param\_UserName*  parameter, 263
- Parent Control property, 189
- pass-through to queries, 324
- passwords
  - saving as text files, 343
  - saving in connection strings, 320–321
- performance
  - calculated fields and, 145
  - indexing and, 143
  - parameterized queries and, 208
- permissions
  - identifying, 294
  - managing, 33–35
  - on master database, 339
  - for public websites, 290
  - read/write, 290–292
  - security and, 293
- Personal Site, 6–9
  - displaying, 8
  - Site Contents page, 9
  - uploading apps to, 29
  - URL, 40–41
- pop-up capability, 61, 74–75
- popup views, 177
  - captions, 182
  - editing data in, 205
  - read-only, 180
  - standalone views as, 179

- pop-up window interface, creating, 152–156
- PowerPivot
  - connection information, 346
  - data display, 348
  - enabling, 344
  - importing data options, 347–348
  - PivotChart data display, 348–349
  - refreshing data, 350
  - reporting with, 344–350
  - SQL Server data source, 345–346
  - trusting documents, 349–350
- primary keys
  - creating, 106
  - data types for, 101
  - fields, 99–102
  - ID field, 101–102, 137
  - indexing of, 142
  - names, changing, 138–139
- printing single-page layouts, 253–261
- Print Preview, 261
- Project Online, 3
- “Project with Printable Sales Orders” app, 255
- properties
  - desktop database vs. web app, 108–109
  - importing, 108–109
  - re-creating, 109–110
- Public Site, 6–9
- Public users, 292
- public web apps
  - in Office 365, 291–292
  - security, managing, 292–300
- public websites, 8
  - Anonymous Users group, 290
  - creating, 289–290
  - settings options, 290
  - site permissions, 290
  - uploading web apps to, 291
- publishing reports to SQL Server Reporting Services, 360–363

## Q

- queries
  - append queries, 324
  - calculated fields, adding, 174
  - calculations, adding, 200–201
  - creating, 196–198
  - creating from Advanced menu, 173–174

- criteria, adding, 198–199
- displaying in datasheet preview, 70
- fields, adding, 197
- parameters, adding, 202–208
- queries, adding, 196
- structural information, displaying, 330
- for summarizing data, 173–177, 209–210
- tables, adding, 196
- Top value, 209–211
- Totals, 208–209
- unique values in, 212–213
  - as views or table-valued functions, 336–337
- Query Builder, 164–165
- Query Designer, 358–359
- Query Parameters window, 202
- query results
  - datasheet preview of, 175
  - displaying, 198
- Quick Access toolbar, 153
- quotation marks, 198–199

## R

- RaiseError* data macro action, 240
- read-only users, 158
- read-write connections, 120–121, 125–126
- read/write permissions, 290–292
- Recent file list, 21–24
- records
  - attachments, hyperlinking to, 125
  - related information, 75–76
- Record Source build button, 165–166
- record sources
  - changing, 164–165
  - embedded queries, 163–166
  - multiple tables, 165–166
  - selecting, 178
  - for views, 164–166
- recursion, 250–252
- references
  - to ADODB Library, 126
  - displaying, 119–120
- referential integrity, 68
  - enforcing, 106–107, 141
  - lookup feature and, 105
  - rule-checking, 102–103
- Related Field property, 189
- related information, displaying, 66–68

## related items control (RELIC)

- related items control (RELIC), 66–67, 72, 75–76, 180–182
  - automatic changes in, 141–142
  - Data, Formatting, and Calculation charms, 181
  - data properties, 181
  - pop-up property, 72
- relationships, 102–107
  - with cascading delete, 141
  - creating, 107, 139–142
  - in desktop databases, 65–67
  - displaying, 104, 321–322
  - displaying with ADOX Library, 327–329
  - displaying with data access objects, 323–327
  - displaying with queries, 330
  - importing, 63–64, 105–106, 108
  - Lookup Wizard choices, 104–105
  - with referential integrity, 141
- REPLACE function, 200
- REPLICATE function, 200
- Report On My Data feature, 131–132
- reports and reporting, 116–121, 339–374. *See also* SQL Server Reporting Services (SSRS)
  - embedding credentials, 351–352
  - export formats, 362–363
  - filtering data, 358–360
  - hyperlinks for, 364
  - images, displaying, 131–132
  - linking web apps to, 363–364
  - paths, defining, 363–364
  - print layout, 253–261
  - Print Preview, 261
  - publishing to SQL Server Reporting Services, 360–363
  - security properties, 362
  - for SQL Server Reporting Services, 351–360
- Report Viewer control, 351
- RequeryRecords* macro action, 205, 223
- rip and replace upgrade technique, 83
- ROLLBACK command, 240
- rollbacks, 335
- rule-checking, 102–103
- RunDataMacro* macro action, 227, 232, 266
- RunMacro* macro action, 221, 225
- runtime experience, 5, 59–60
  - web apps in, 21

## S

- Save As New App option, 86
- Save As Snapshot option, 24, 86
- Save For Deployment option, 86
- SaveRecord* macro action, 223
- saving
  - views, 154
  - web apps, 32
- scaffolding, 68, 136
- scalability, site collections and, 40
- schemas, 308
- ScriptOutDesign.sql* script, 322
- searching with List control, 78
- security
  - of data, 53
  - logins, identifying, 293
  - member validation tables, 293
  - permissions control, 293
  - for public web apps, 292–300
  - restricting view to Administrator users, 296
  - in SQL Server Management Studio, 309
  - table-driven, 292–293
  - user email addresses, validating, 298–299
- SELECT DISTINCT syntax, 212
- Service Pack 1 (Access 2013)
  - On Deploy Macro, 84–85
  - locking applications, 82–83
  - locking tables, 82–83
  - new features, 82–96
  - packaging options, 85–86
- SetField* macro action, 264
- SetLocalVar* macro action, 236
- SetPrintableSalesOrder* macro action, 258–259
- SetProperty* macro action, 172–173, 192, 223, 227
- settings options for Team Site, 33
- SetVariable* macro action, 223, 266
- Shared With information, 33
- SharePoint lists, 47–51
  - linking to, 47–52, 309–310
  - saving files to, 121
- SharePoint Online, 2
- SharePoint site collections. *See* site collections
- SharePoint version control, 275
- sharing
  - with external users, 32–35
  - on Team Site, 7
  - web apps, 55
- Shift key selection, 179

- shortcut paths to web apps, 291
- Short Text data type, 113–114
- Show Table popup, 196
- side-loading, 26–29, 86, 91, 96, 267–274
  - version numbers in, 273
- site collections, 9, 36. *See also* Personal Site; Team Site page
  - creating, 39–44
  - default, 41–42
  - finding sites in, 43
  - functions of, 40
  - my.sharepoint.com, 44
  - naming, 41–42
  - personal sites of all users, 44
  - URLs of, 40–41
- Site Contents page, 7
  - adding apps from, 89, 291
  - for Personal Site, 9
  - removing web apps from, 277
  - for Team Site, 8
  - upgrading web apps from, 285–286
- site management with SharePoint Online, 2
- site permissions for public websites, 290
- sites, 9
  - adding apps to, 89
  - creating, 36–39
  - deleting, 38–39
  - displaying, 7, 43–44
  - home page, 37
  - promoting, 37–38, 43–44
  - settings, 38–39
  - themes, 45–47
  - URLs, 49
- Small Business Premium subscriptions
  - Office 365, downloading and setting up, 10–13
  - web apps, viewing, 8
- snapshots, 24–25, 86, 90, 267–268
  - creating and saving, 276
  - uploading, 269, 278
  - uploading to App Catalog, 86–88, 278–279
  - version numbers, 277–278
- sorting data in datasheet view, 157
- spaces, trimming, 200
- special characters, in field and table names, 100
- SQLDataSource control, 367
- SQL Server
  - connecting to Azure SQL database, 365
  - feature packs, 303
  - views and queries, 371
- SQL Server 2012 Express, 302–303
- SQL Server Business Intelligence Development Studio, 351
  - data source and data set, modifying, 354–356
  - displaying reports, 356–357
  - importing reports, 354
  - project properties, 356
  - Publishing reports, 361
  - Report Server Project template, 353
  - starting, 353
- SQL Server Connection Information window, 118
- SQL Server Data Tools (2012), 351
- SQL Server Management Studio (SSMS), 302–303
  - connecting to server, 306–307
  - Object Explorer, 307–309
  - security, 309
  - structural information, displaying, 321–330
  - testing data macro code with, 295–296
  - user security, 309
- SQL Server Object Explorer, 307
  - Schema folder, 308
  - viewing Access tables in, 365–366
- SQL Server Reporting Services (SSRS)
  - authentication, 351–352, 364
  - configuring, 360–361
  - displaying reports, 351
  - export formats, 362–363
  - linking web apps to reports, 363–364
  - memory requirements, 351
  - publishing reports to, 360–363
  - Remote and non-SQL data source support feature, 351
  - reporting for, 351–360
  - Reporting Services Configuration Manager, 360
  - Report Manager URL, 361
  - report server, starting, 360–361
  - uploading reports to, 361–362
  - Web Service URL, 361
- SrcID field, 129
- SSMS. *See* SQL Server Management Studio (SSMS)
- SSRS. *See* SQL Server Reporting Services (SSRS)
- standalone macros, 81, 217, 219–221
- standalone named data macros, 335
- standalone user-interface macros, 219–221
- standalone views, 177–180
  - creating, 152–156
  - other views, duplicating to, 168–169
- StopMacro* command, 294

## stored procedures

- stored procedures
  - data macros and, 228–239
  - standalone named data macros as, 335
- Stream objects, 126
- string concatenation, 146, 200
- structural information
  - displaying with ADOX Library, 327–329
  - displaying with data access objects, 323–327
  - displaying with queries, 330
- STUFF function, 200
- subscriptions
  - App Catalog, 53
  - to Office 365, 2–3
- subsites, 7
  - creating, 36–39
- SUBSTR function, 200
- Subview control, 76–77, 182–186
- subviews
  - arranging on-screen, 184
  - List details view as, 170
  - standalone views as, 179
- Suggested Searches, 134–135
- summarized data
  - Totals feature, 208–209
  - unique values, 212
- summary queries, 173–177, 209–210
- summary views, 59, 173–177
  - components of, 72–73
  - jump lists, 176–177
  - list area, 175–176
  - Popup View property, 176–177
  - as standalone view, 178
  - summary information area, 175–176
- synchronization of controls, 188–189
- System DSNs, 310–311
  - creating, 314–316

## T

- table captions, 58
- tabledef* connection string, 310
- table-driven security, 292–293
- table events. *See also* events
  - embedded data code in, 239–247
  - macros written on, 217
- tablename?Images* table, 123
- table properties, 108–111
- tables, 99–102. *See also* foreign keys; primary keys
  - calculated fields, 145–146

- captions, 150
- creating, 136–137
- deleting, 58
- design, editing, 137–139
- design view, 100
- foreign key lookup fields, 228
- hiding in selector, 58
- importing, 63–64, 103
- indexing, 142–143
- joining-relationship lines, 196
- large, 302
- locking, 58, 82–83, 276
- lookups. *See* lookups
- macros in, 81. *See also* macro programming
- names, 100
- navigating, 59–60
- opening in datasheet preview, 69–70
- order in selector, 58
- populating, 212
- queries, adding to, 196. *See also* queries
- related information, displaying, 66–68
- relationships. *See* relationships
- template tables, adding to web apps, 134–136
- validation rules, 143–145
- view controls, 73–80
- views, 59. *See also* views
- Table Selector, 57–61, 135–136
  - customizing, 150–151
- table-valued functions, 203, 336–337
- target view, referring to, 163
- Team Site page, 6–9
  - About Me selection, 8
  - Apps selection, 9
  - browsing from Access, 18
  - displaying web apps in, 21–22
  - permissions, 33–35
  - settings options, 33
  - Site Contents link, 8
  - Site Contents menu, 26–27
  - uploading documents, 122–123
  - URL, 40–41
  - Your Apps, 26–27
- templates, 19
  - creating web apps with, 19–22
- template tables, adding to web apps, 134–136
- testing macros, 220, 245–246
- themes, 6
- titles of web apps, 273
- Top Values property, 209–211
- TOP WITH TIES feature, 211

Totals queries, 208–209, 233–234, 262  
 tracing macro execution, 82, 247–248, 332  
 transactions  
   initiating and save points, 333  
   recursion and, 250–252  
   rollbacks, 335  
 Transaction Structured Query Language (TSQL),  
 216–217  
   in stored procedure code, 230  
 triggers  
   After Triggers, 240  
   code, 242–244, 332–335  
   cursors in, 333  
   data macros and, 239–247  
   displaying, 242  
   recursive, 251–252  
   saving, 247  
   transaction count, 335  
   values in, examining, 243  
 TRIM function, 146  
 trimming spaces, 200  
 TSQL. *See* Transaction Structured Query Language  
 (TSQL)

## U

unbound controls, 163, 187, 191  
 Undo button, adding to action bar, 159–160  
 undoing changes, 159  
*UndoRecord* macro action, 160, 223  
 Unicode, 114  
 unions of data, 266  
 Unique values property, 212–213  
 unsupported formats, removal and replacement of,  
 115  
 update operations, 229–231, 234–235  
 UpdateParameters link, 237, 239  
 upgrade process, 87, 93–94. *See also* side-loading  
   with App Catalog, 93–96  
   applying upgrades, 267–274  
   for app packages, 270  
   On Deploy Macro, creating, 282–283  
   with On Deploy Macro, 274–286  
   development copies, 279–280  
   errors, 272–273  
   for existing web apps, 90  
   locked tables, 282  
   locking upgrades, 272  
   rip and replace technique, 83  
   SharePoint version control, 275  
   side-loading, 96, 267–274  
   skipping version upgrades, 85, 274  
   upgrading web app from Site Contents, 285–286  
   uploading ODM, 284–285  
 uploading  
   to App Catalog, 54–55, 86–88, 92–93, 278–279,  
   285  
   apps to App Catalog, 54–55  
   apps to Personal Site, 29  
   apps to public websites, 291  
   documents to Team Site, 122–123  
   images, 121–122, 125–132  
   ODM, 284–285  
   OLE data types, 121–122  
   packages, 26–29, 86–93, 285  
   reports to SQL Server Reporting Services,  
   361–362  
   snapshots, 269, 278  
 URLs  
   parameters in, 363  
   of Personal Site, 40–41  
   of Report Manager, 361  
   of site collections, 40–41  
   of sites, 49  
   of Team Site, 40–41  
   of Web Service, 361  
 User Data Source Names (DSNs), 310  
*UserDisplayName()* function, 255, 257, 293–294  
*UserEmailAddress()* function, 293  
 user interface. *See also* Table Selector; view selector  
   developing, 149  
   pop-up window interface, 152–156  
 user-interface macros, 80–81, 216, 227. *See also*  
 macro programming; macros  
   action bar, hiding commands on, 297  
   actions for cross-tabulation of data, 266  
   debugging code, 296  
   embedded, 217  
   macro action sets, 227  
   standalone, 217, 219–221  
 user permissions, identifying, 294. *See also*  
 permissions  
 users  
   anonymous, 290, 294–295  
   email addresses, validating, 298–299  
   external, 6, 32–35  
   ExternalReader, 119–121, 301, 309  
   ExternalWriter, 119–121, 301, 309  
   Members, 35, 292

- users (*continued*)
  - personal sites, 44
  - Public, 292
  - restricting data editing, 297–298
  
- V**
- validation of user email addresses, 298–299
- validation rules, 143–145, 330–331
  - importing, 108–111
- value-based lookups, 112, 140
- VARBINARY(MAX) field, 130
- VARCHAR, 114
- variables
  - checking definitions, 230
  - defining, 249
  - global, 218–219, 294
  - local, 218, 229
  - in macros, 218
- Version 1 snapshot packages, 97
- Version 2 deployment packages, 96–97
- version control, 275
- version numbers, 85–86, 88
  - of deployment packages, 92
  - editing, 90–91
  - in side-loading, 273
- VersionNumber values, 282
- version upgrades, applying, 267–274
- view controls, 73–80, 186–187. *See also* controls
  - action bar, 77
  - autocomplete, 74–75, 190–191
  - color, adding, 192
  - combo box, 187–189
  - design controls, 73–74
  - Hyperlink control, 78–79
  - List control, 77–78
  - Multiline Text Box control, 79
  - Related Items control, 75–76
  - Subview control, 76–77
  - Web Browser, 79–80, 191
- viewer.aspx syntax, 253–254, 257
- view preview, 157
- views, 58
  - action buttons, 160
  - Apps For Office control, 380
  - in Azure SQL Database, 336–337
  - charms, 171
  - components of, 71–73
  - creating, 152–156
  - data source, 220–221

- design controls, 73–74
  - duplicating, 167–169
  - editing, 71, 138
  - embedding, 76–77
  - independent of table captions, 168
  - list of all, 168
  - macros in, 81, 217, 222
  - managing, 59
  - navigation between, 184
  - opening in browser, 70
  - parameterized queries in, 203
  - printing, 253–261
  - quickly created, 137, 139
  - record sources and, 164–166
  - saving, 154
  - set of events on, 222–226
  - Settings/Action charm, 70
  - settings and actions, 151–152
  - standalone, 152–156, 168–169, 177–180
  - subviews, 170
  - summary. *See* summary views
  - of tables, 59
  - testing macros in, 220
  - within views, 182–183
- view selector, 57–61
    - caption retitling, 182
    - customizing, 151–152
  - ViewState\_Add* macro action, 224
  - ViewState\_Cancel* macro action, 224
  - ViewState\_Delete* macro action, 224
  - ViewState\_Edit* macro action, 224
  - ViewState\_Open* macro action, 224
  - ViewState\_Save* macro action, 224
  - Visio Pro for Office 365, 3
  - Visitors permissions, 35
  - Visual Basic for Applications (VBA) functions
    - attachment files, extracting, 123–125
    - image files, uploading, 127–129
    - for scanning tables and fields, 110–111
  - Visual Basic for Applications (VBA) project references, 119–120
  - Visual Studio
    - apps for Access, creating, 381–386
    - New Website option, 364
    - Report Viewer control, 351
    - Web Forms, 366
    - websites, adding content, 366
    - websites, adding SQLDataSource, 367
  - Visual Studio 2013 with web apps, 364–374



## W

- web apps. *See also* apps for Access; apps for Office
  - Azure SQL Database details, 303–304
  - Azure SQL Database instances, 52
  - calculated fields, 145–146
  - converting desktop databases to, 99–132
  - creating, 19–21, 61–62, 133–134
  - creating with template, 19–22
  - design and data, saving, 25
  - development copies, 279–280
  - displaying in browser, 32
  - downloading, 32
  - hyperlinks, adding, 291–292
  - Image data type, 121
  - indexing, 142–143
  - Is Edited flag, 95
  - key components, 60
  - launching in browser, sites, or subsites, 21–22
  - linking to documents, 47–52
  - linking to SSRS reports, 363–364
  - location, 20, 134
  - locking, 267–269, 272
  - lookups, 68, 139–142
  - macro programming in, 81, 216–219
  - for Microsoft Office, 375–386
  - moving, 9
  - naming, 134
  - opening with Access, 23–24
  - recursion, 250–252
  - relationships, 139–142, 321–322. *See also* relationships
  - removing from Apps in Testing, 277
  - removing from Site Contents, 277
  - rule-checking, 103–104
  - runtime experience, 5
  - saving, 32
  - saving as package, 24–25. *See also* packages
  - saving for deployment, 268–269. *See also* deployments
  - SharePoint version control, 275
  - sharing, 32–35, 55
  - Shift key selection, 179
  - shortcut paths to, 291
  - side-loading, 267–274. *See also* side-loading
  - snapshots, 267–268
  - tables, 136–139. *See also* tables
  - template tables, adding, 134–136
  - themes, 45–47
  - titles, 273
  - upgrading, 270. *See also* upgrade process
  - uploading to App Catalog, 54–55
  - uploading to public sites, 291–292
  - validation rules, 143–145
  - VersionNumber values, 282
  - viewing on Team Site, 8
  - with Visual Studio 2013, 364–374
- Web Browser control, 79–80, 191
- Web Forms, 366
  - adding SQLDataSource, 367
  - connections, adding, 368–369
  - controls, adding, 371
  - data management, 367
  - launching in browser, 372–373
- web location, finding, 20
- webpage component of apps for Access, 376
- websites
  - adding connections, 368–369
  - adding content, 366
  - adding SQLDataSource, 367
  - New Website option, 364
  - permissions for, 290
  - public, 8, 289–290
  - uploading apps to, 291
- What's Your Style? tile, 45
- where clauses
  - character limit, 287
  - filtering with, 205–206
- wildcard search, 199
- Windows 8 Start screen
  - Office 365 tiles, 13
  - OneDrive tile, 16
- Windows Data Source Administration Tool, 311
- Windows Explorer, uploading files with, 122–123

## Y

- Your Apps, 26–27

*This page intentionally left blank*

# About the author




*Andrew Couch* has been working with Microsoft Access since 1992 as a developer, trainer and consultant. He is a joint founder of the UK Access User Group, which has been running for over 13 years. He has been a Microsoft Access MVP for the past 7 years.

Andrew has been involved with Access web apps from early on and has been working with members of the Microsoft team to communicate information on exploiting the power of this new technology.

Other publications with Microsoft Press include *Microsoft Access 2013 Plain & Simple* and *Microsoft Access 2010 VBA Programming Inside Out* and he is a co-author of *Microsoft Office Professional 2013 Step by Step*.

In addition to consulting and regularly speaking at community events, Andrew has developed the Migration Upsizing SQL Tool (MUST), which allows users to easily convert Access databases to SQL Server by using an Access-based application. Due to the success of MUST, which is used by over 200 companies, SQL Translation capabilities and WebForm code generators for .NET were added to the product range. He also provides free technical articles for Access at [www.upsizing.co.uk/TechLibrary.aspx](http://www.upsizing.co.uk/TechLibrary.aspx).



Now that  
you've  
read the  
book...

Tell us what you think!

Was it useful?

Did it teach you what you wanted to learn?

Was there room for improvement?

**Let us know at <http://aka.ms/tellpress>**

Your feedback goes directly to the staff at Microsoft Press,  
and we read every one of your responses. Thanks in advance!



**Microsoft**