Microsoft

Inside OUT

The ultimate, in-depth reference

Hundreds of timesaving solutions

Supremely organized, packed with expert advice

SQL Server 2017 Administration

William Assaf • Randolph West • Sven Aelterman • Mindy Curnutt

Foreword by Patrick LeBlanc, Microsoft

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Microsoft

SQL Server 2017 Administration Inside Out

William Assaf
Randolph West
Sven Aelterman
Mindy Curnutt

Published with the authorization of Microsoft Corporation by:
Pearson Education, Inc.

ISBN-13: 978-1-5093-0521-6
ISBN-10: 1-5093-0521-1

Library of Congress Control Number: 2017961300

Printed and bound in the United States of America.

1 18

Editor-in-Chief: Greg Wiegand
Acquisitions Editor: Trina MacDonald
Development Editor: Mark Renfrow
Technical Editor: Louis Davidson
Managing Editor: Sandra Schroeder
Senior Project Editor: Tracey Croom
Editorial Production: Octal Publishing, Inc.
Copy Editor: Octal Publishing, Inc.
Indexer: Octal Publishing, Inc.
Proofreader: Octal Publishing, Inc.
Cover Designer: Twist Creative, Seattle

To David III
for inspiring and enabling STEM careers for many, including my own.

—William Assaf

To Marinus and Trixie
for putting up with the lies of "almost done," and sharing my lap with a MacBook.

—Randolph West

To Ebony, Edward, and Sofia
in recognition of their sacrifices, support, and endless love.

—Sven Aelterman

To Chris
For believing in me more than I even believed in myself.

—Mindy Curnutt

This page intentionally left blank

Contents at a glance

This page intentionally left blank

Table of contents

Foreword

The world as we know it is being inundated with data. We live in a culture in which almost every individual has at least two devices, a smart phone, and a laptop or computer of some sort. Everything we do on these devices is constantly collecting, sharing, or producing data. This data is being used not only to help organizations make smarter decisions, but also to shape and transform how we as a society live, work, make decisions, and sometimes think.

This massive explosion can be attributed to the technological transformation that every business and nearly every industry is undergoing. Every click or purchase by an individual is now triggering some event that triggers another event that likely amounts to hundreds or possibly thousands of rows of data. Multiply this by every person in the world and now you have an unprecedented amount of stored data that no one could have ever imagined. Now, not only must organizations store this data, but also ensure that this data—this massive amount of data—is readily available for consumption at the click of a button or the swipe of a screen.

This is where the database comes into play. Databases are the backbone or back end to possibly every aspect of business today. Back when Ted Codd, the father of the relational database, came up with this seminal idea, he probably had no idea how widespread their use would be today. Initially, database usage was intended to store data and retrieve data. The primary purpose was to simply ensure the security, availability, and reliability of any information written by on-premises applications at varying scales.

Today, all of that has changed. Data must be available 24 hours per day, 7 days each week, primarily via the internet instead of just by way of on-premises applications. Microsoft SQL Server 2017 was designed with all of this in mind. It can support high-volume Online Transactional Processing (OLTP) databases and very large Online Analytical Processing (OLAP) systems out of the box. And, by taking advantage of Microsoft Azure, developers can grow and scale databases dynamically and transparently behind the scenes to accommodate planned and unplanned spikes in demand and resource utilization. In other words, the latest version of SQL Server was built to not only accommodate this new world of data, but to push the limits of what organizations are doing today and what they will be doing tomorrow and deeper into the future.

Close your eyes and imagine a world in which a DBA can configure a database system to automatically increase or decrease resource utilization based on end-user application usage. But that's not all. What if the relational database management system (RDBMS) could automatically tune performance based on usage patterns? All of this is now possible with SQL Server and Azure SQL Database. By using features such as the Query Store and Elastic Database Pools, DBAs can proactively design solutions that will scale and perform to meet any application Service-Level Agreement.

In addition to world-class performance, these databases also include security and high-availability features that are unparalleled to any other RDBMS. Organizations can build mission-critical secure applications by taking advantage of SQL Server out-of-the-box built-in features without purchasing additional software. These features are available both in the cloud and on-premises and can be managed using SQL Server Management Studio, SQL Server Data Tools, and SQL Operations Studio. All three tools are available to download for free, and will be familiar to DBAs and database developers.

Throughout this book, the authors highlight many of the capabilities that make it possible for organizations to successfully deploy and manage database solutions using a single platform. If you are a DBA or database developer looking to take advantage of the latest version of SQL Server, this book encompasses everything needed to understand how and when to take advantage of the robust set of features available within the product.

This book is based on the skills of a group of seasoned database professionals with several decades experience in designing, optimizing, and developing robust database solutions, all based on SQL Server technology. It is written for experienced DBAs and developers, aimed at teaching the advanced techniques of SQL Server.

SQL Server, Microsoft's core database platform, continues its maturity from supporting some of the smallest departmental tasks to supporting some the largest RDBMS deployments in the world. Each release not only includes capabilities that enhance its predecessor, but also boasts features that rival and exceed those of many competitors.

This trend continues with SQL Server 2017. This release, just like all past releases, continues to add capabilities to an already sophisticated and reliable toolkit. Features include a secure, elastic, and scalable cloud system; advanced in-memory technologies; faster and consolidated management and development experiences; and continued growth and enhancements in the area of high availability and disaster recovery. In addition, concerted efforts have been focused on making the number one secure RDBMS in the world even more secure, by adding capabilities such as row-level security, Always Encrypted, and dynamic data masking. Finally, and as always, performance is at the center of this release. With enhancements to the Query Store, DBAs can take a more proactive approach to monitoring and tuning performance.

All in all, this book is sort of like an "Inside Out" look of each of the core components of SQL Server 2017, with a few excursions into the depths of some very specific topics. Each chapter first provides and overview of the topic and then delves deeper into that topic and any corresponding related topics. Although it's impossible to cover every detail of every Transact-SQL statement, command, feature or capability, this book provides you with a comprehensive look into SQL Server 2017. After reading each page of this book, you will be able implement a cloud-based or on-premises scalable, performant, secure, and reliable database solution using SQL Server 2017.

Patrick LeBlanc, Microsoft

Introduction

The velocity of change for the Microsoft SQL Server DBA has increased this decade. The span between the releases of SQL Server 2016 and 2017 was only 16 months, the fastest new release ever. Gone are the days when DBAs had between three to five years to soak in and adjust to new features in the engine and surrounding technologies.

This book is written and edited by SQL Server experts with two goals in mind: to deliver a solid foundational skillset for all of the topics covered in SQL Server configuration and administration, and also to deliver awareness and functional, practical knowledge for the dramatic number of new features introduced in SQL Server 2016 and 2017. We haven't avoided new content—even content that stretched the boundaries of writing deadlines with late-breaking new releases. You will be presented with not only the "how" of new features, but also the "why" and the "when" for their use.

Who this book is for

SQL Server administration was never the narrow niche skillset that our employers might have suspected it was. Even now it continues to broaden, with new structures aside from the traditional rowstore, such as Columnstore and memory-optimized indexes, or new platforms such as Microsoft Azure SQL Database platform as a service (PaaS) and Azure infrastructure as a service (IaaS). This book is for the DBAs who are unafraid to add these new skillsets and features to their utility belt, and to give courage and confidence to those who are hesitant. SQL Server administrators should read this book to become more prepared and aware of features when talking to their colleagues in application development, business intelligence, and system administration.

Assumptions about you

We assume that you have some experience and basic vocabulary with administering a recent version of SQL Server. You might be curious, preparing, or accomplished with Microsoft Certifications for SQL Server. DBAs, architects, and developers can all benefit from the content provided in this book, especially those looking to take their databases to the cloud, to reach heights of performance, or to ensure the security of their data in an antagonistic, networked world.

This book mentions some of the advanced topics that you'll find covered in more detail elsewhere (such as custom development, business intelligence design, data integration, or data warehousing).

Book Features

These are the book's signature tips. In these tips, you'll get the straight scoop on what's going on with the software or service—inside information about why a feature works the way it does. You'll also find field-tested advice and guidance as well as details that give you the edge on deploying and managing like a pro.

How this book is organized

This book gives you a comprehensive look at the various features you will use. It is structured in a logical approach to all aspects of SQL Server 2017 Administration.

Chapter 1, "Getting started with SQL Server tools" gives you a tour of the tooling you need, from the installation media to the free downloads, not the least of which is the modern, rapidly evolving SQL Server Management Studio. We also cover SQL Server Data Tools, Configuration Manager, performance and reliability monitoring tools, provide an introduction to PowerShell, and more.

Chapter 2, "Introducing database server components," introduces the working vocabulary and concepts of database administration, starting with hardware-level topics such as memory, processors, storage, and networking. We then move into high availability basics (much more on those later), security, and hardware virtualization.

Chapter 3, "Designing and implementing a database infrastructure" introduces the architecture and configuration of SQL Server, including deep dives into transaction log virtual log files (VLFs), data files, in-memory Online Transaction Processing (OLTP), partitioning, and compression. We spend time with TempDB and its optimal configuration, and server-level configuration options. Here, we also cover running SQL Server in Azure virtual machines or Azure SQL databases as well as hybrid cloud architectures.

Chapter 4, "Provisioning databases" is a grand tour of SQL Server Setup, including all the included features and their initial installation and configuration. We review initial configurations, a post-installation checklist, and then the basics of creating SQL Server databases, including database-level configuration options for system and user databases.

Chapter 5, "Provisioning Azure SQL Database," introduces Microsoft's SQL Server database-as-a-service (DBaaS) offering. This Azure cloud service provides a database service with a very high degree of compatibility with SQL Server 2017. You will read about the concepts behind Azure SQL Database, learn how to create databases, and perform common management tasks for your databases.

Chapter 6, "Administering security and permissions" begins with the basics of authentication, the configuration, management, and troubleshooting of logins and users. Then, we dive into permissions, including how to grant and revoke server and database-level permissions and role membership, with a focus on moving security from server to server.

Chapter 7, "Securing the server and its data" takes the security responsibilities of the SQL Server DBA past the basics of authentication and permissions and discusses advanced topics including the various features and techniques for encryption, Always Encrypted, and row-level security. We discuss security measures to be taken for SQL Server instances and Azure SQL databases as well as the Enterprise-level SQL Server Audit feature.

Chapter 8, "Understanding and designing tables," is all about creating SQL Server tables, the object that holds data. In addition to covering the basics of table design, we cover special table types and data types in-depth. In this chapter, we also demonstrate techniques for discovering and tracking changes to data.

Chapter 9, "Performance tuning SQL Server" dives deep into isolation and concurrency options, including READ COMMITTED SNAPSHOT ISOLATION (RCSI), and why your developers shouldn't be using NOLOCK. We review execution plans, including what to look for, and the Query Store feature that was introduced in SQL Server 2016 and improved in SQL Server 2017.

Chapter 10, "Understanding and designing indexes" tackles performance from the angle of indexes, from their creation, monitoring, and tuning, and all the various forms of indexes at our disposal, past clustered and nonclustered indexes and into Columnstore, memory-optimized hash indexes, and more. We review indexes and index statistics in detail, though we cover their maintenance later on in Chapter 13.

Chapter 11, "Developing, deploying, and managing data recovery" covers the fundamentals of database backups in preparation for disaster recovery scenarios, including a backup and recovery strategy appropriate for your environment. Backups and restores in a hybrid environment, Azure SQL Database recovery, and geo-replication are important assets for the modern DBA, and we cover those, as well.

Chapter 12, "Implementing high availability and disaster recovery" goes beyond backups and into strategies for disaster recovery from the old (log shipping and replication) to the new (availability groups), including welcome new enhancements in SQL Server 2017 to support cross-platform and clusterless availability groups. We go deep into configuring clusters and availability groups on both Windows and Linux.

Chapter 13, "Managing and monitoring SQL Server" covers the care and feeding of SQL Server instances, including monitoring for database corruption, monitoring database activity, and index fragmentation. We dive into extended events, the superior alternative to traces, and also cover Resource Governor, used for insulating your critical workloads.

Chapter 14, "Automating SQL Server administration" includes an introduction to PowerShell, including features available in PowerShell 5.0. We also review the tools and features needed to automate tasks to your SQL Server, including database mail, SQL Server Agent jobs, Master/Target Agent jobs, proxies, and alerts. Finally, we review the vastly improved Maintenance Plans feature, including what to schedule and how.

About the companion content

We have included this companion content to enrich your learning experience. You can download this book's companion content from the following page:

https://aka.ms/SQLServ2017Admin/downloads

The companion content includes helpful Transact-SQL and PowerShell scripting, as mentioned in the book, for easy reference and adoption into your own toolbox of scripts.

Acknowledgments

From William Assaf:

I'd like to thank the influencers and mentors in my professional career who affected my trajectory, and to whom I remain grateful for technical and nontechnical lessons learned. In no particular order, I'd like to thank Connie Murla, David Alexander, Darren Schumaker, Ashagre Bishaw, Charles Sanders, Todd Howard, Chris Kimmel, Richard Caronna, and Mike Huguet. There's definitely a special love/hate relationship developed between an author and a tech editor, but I couldn't have asked for a better one than Louis Davidson. Finally, from user groups to SQLSaturdays to roadshow presentations to books, I am indebted to my friend Patrick Leblanc, who climbed the ladder and unfailingly turned to offer a hand and a hug.

From Randolph West:

In June 2017, I told my good friend Melody Zacharias that I'd like to finish at least one of the many books I've started before I die. She suggested that I might be interested in contributing to this one. Piece of cake, I thought.

I have seven more gray hairs now. Seven!

I would like to thank Melody for recommending me in her stead, my husband for giving me space at the kitchen counter to write, and my dog Trixie for much needed distraction.

Trina, William, Louis, Sven and Mindy have been a great support as well, especially during the Dark Times.

This book would not be possible without the contributions of everyone else behind the scenes, too. Writing a book of this magnitude is a huge endeavour. (So help me if "endeavour" is the one word I get to spell the Canadian way!)

From Sven Aelterman:

I met William Assaf several years ago when I spoke at the Baton Rouge SQLSaturday. I have been back to this event many times since then and enjoyed preceding the Troy University Trojans' victory over Louisiana State University. (This just added in case the actual college football game doesn't make it in the history books. At least it will be recorded here.)

I am grateful for William's invitation to contribute two chapters to this book. William made a valiant attempt to prepare me for the amount of work "just" two chapters would be. Yet, I underestimated the effort. If it weren't for his support and that of Randolph West, technical editor Louis Davidson, editor Trina Macdonald, and even more people behind the scenes, the space for this acknowledgment might have been saved. They were truly a great team and valued collaborators. Without hesitation, I would go on the journey of book writing again with each of them.

My children, Edward and Sofia, and my wife, Ebony, have experienced firsthand that SQL Server can slow down time. "About two months" must have felt to them like months with 60 days each. I thank them for their patience while they had to share me with Azure and various table types. I hope that maybe my children will be inspired one day to become authors in their career fields.

Finally, I'd like to thank my coworkers at Troy University for inspiring me to do my best work. Working in a public higher education institution has some challenges, but the environment is so conducive to intellectual growth that it makes up for each challenge and then some.

From Mindy Curnutt:

I would like to thank Patrick LeBlanc for inviting me to participate in the creation of this book. Thanks also to Tracy Boggiano, for an amazing amount of help pulling together much of the chapter about automating administration. She's an MVP in my eyes! To everyone in the 2016-2017 TMW DBA Services "Team Unicorn": Eric Blinn, Lisa Bohm, Dan Andrews, Vedran Ikonic, and Dan Clemens, thank you for your proof reading and feedback. Thanks to my mom Barbara Corry for always swooping in to help with just about anything I needed. Of course, I couldn't have done any of this without the support of my husband, Chris Curnutt. He is always supportive despite long work hours, phone conversations with strange acronyms, and travel, he's also the love of my life. Last but not least, thanks to our two children, Riley and Kimball, who have supported and encouraged me in more ways than I can count.

Support and feedback

The following sections provide information on errata, book support, feedback, and contact information.

Errata & 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:

https://aka.ms/SQLServ2017Admin/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*.

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 *https://support.microsoft.com*.

Stay in touch

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

CHAPTER 4

Provisioning databases

In this chapter, we review the process of installing and configuring a Microsoft SQL Server instance as well as the creation or migration of databases. We pay special attention to new features introduced in SQL Server 2017 and even some added since SQL Server 2016 Service Pack 1, including those features that have been expanded for the first time from the Enterprise edition to the Standard edition of SQL Server. We review some basic checklists for you to verify every time and, when necessary, direct you to where you can find more details on critical steps elsewhere in this book as well as other sources of information.

What to do before installing SQL Server

Before you run the SQL Server installer, there are a number of factors and settings that you should consider, some of which you cannot easily change after installation. Pay special attention to sections in this chapter regarding server volume alignment (whether this is a physical or virtual server, or whether the volumes are physical drives or Storage-Area Network–based), version and edition choices, and new features of the SQL Server 2017 installer.

CAUTION

We recommended that you do not install SQL Server on the same server as a domain controller. In some scenarios, it is not supported and can even cause Setup to fail.

Deciding on volume usage

When you're configuring a Microsoft Windows Server, before starting the SQL Server installer, consider the volumes. Although you can move user and system database data and log files to other locations after installation, it's best to plan your volumes prior to installation.

NOTE

The examples in this chapter assume that your Windows operating system installation is on the C volume of your server.

One of the basic guiding principles for a SQL Server installation is that anywhere you see "C:\", change it to another volume. This helps minimize SQL Server's footprint on the operating system (OS) volume, especially if you install multiple SQL Server instances, which can have potential disaster recovery implications in terms of volume-level backup and restores.

NOTE

For Microsoft Azure SQL Database virtual machines (VMs), do not set the installation directories for any settings on the D:\ "Temporary Storage" volume. This folder is wiped upon server restart! The only exception is that the TempDB data files can exist on the D drive if certain other considerations are taken. For more about this, see Chapter 3.

If this is the first SQL Server instance you are installing on a server, you will have the opportunity to change the location of shared features files, the data root directory for the instance (which contains the system databases), default database locations for user database files, and their backups. If this is not the first SQL Server 2017 instance installation on this server, the shared features directory locations (for Program Files and Program Files x86) will already be set for you, and you cannot change it.

Inside OUT

What if I am tight on space on the C drive when installing SQL Server?

There are some easy ways and some tricky ways to minimize the footprint of a SQL Server installation on the OS volume of your server (typically the C drive, as it is for this example). In general, SQL Server Setup and cumulative updates will delete temporary files involved in their installation, but not log files or configuration files, which should have minimal footprint. Outside of log files, we recommend that you do not delete any files installed by SQL Server Setup or cumulative updates. Instead, let's take a look at some proactive steps to move these files off of the C volume.

Some parts of SQL Setup will install on the OS volume (typically, and in this and future examples, the Windows C volume). These files, which are staging areas for SQL Server Setup, are created on the OS volume in a C:\Program Files\Microsoft SQL Server\140\Setup Bootstrap\ subfolder structure, where 140 is specific to the internal version number (14.0) of SQL 2017. This folder is used for future cumulative updates or feature changes.

If you're extremely tight on space before installing SQL Server, you will also find that the root binaries installation directory will be, by default, C:\Program Files\Microsoft SQL Server\. When you're using the SQL Server Setup user interface, there is no option to change this. You will, however, find this installation directory folder path listed as the `INSTANCEDIR` parameter in the config file that is generated by SQL Server Setup. We talk more about how to use the config file to install SQL Server in the section "Automating SQL Server Setup by using configuration files" later in the chapter.

You should place as much of the installation as possible on other volumes. Keep in mind that a full-featured installation of SQL Server 2017 can consume more than 7 GB. You will want to move some of those binaries for feature installations to other folders.

The following sample scenario is a good starting point (the volume letters don't matter):

- **Volume C.** OS only
- **Volume E.** SQL Server installation files, SQL Server data files
- **Volume F.** SQL Server log files
- **Volume G.** SQL Server TempDB data and log files (we look at TempDB data files in more detail later in the chapter)
- **Volume H.** SQL Server backups

Where do you go from here? Here are some avenues that you might take with respect to volumes:

- Use additional volumes for your largest data files (larger than 2 TB) for storage manageability
- Use an additional volume for your most active databases and their log files
- Use an additional volume for large amounts of FILESTREAM data
- Use an additional volume for large replicated database snapshot files

Inside OUT

Why separate files onto different volumes?

There are reasons to separate your SQL Server files onto various volumes, and not all of them are related to performance. You should separate your files onto different volumes even if you exclusively use a Storage-Area Network (SAN).

We know that more discrete Input/Output (I/O) on a physical server with dedicated drives means better performance. But even in a SAN, separating files onto different volumes is also done for stability. If a volume fills and has no available space, files cannot be allocated additional space. On volume C, 0 bytes free could mean Windows Server stability issues at worst, user profile and remote desktop problems at least, and possible impact to other applications.

In the aforementioned scenario, if the E or F volumes fill up because of unmonitored SQL Server file growth over time, the problems presented would be limited to SQL Server and, likely, only to the database(s) whose data or log files that have filled.

Important SQL Server volume settings

There are some settings that you need to consider for volumes that will host SQL Server data and log files, and this guidance applies specifically to these volumes (for other volumes—for example, those that contain the OS, application files, or backup files—the default Windows settings are acceptable unless otherwise specified):

- When adding these volumes to Windows, there are three important volume configuration settings that you should check for yourself or discuss with your storage administrator. When creating new drives, opt for GUID Partition Table (GPT) over Master Boot Record (MBR) drive types for new SQL Server installations. GPT is a newer drive partitioning scheme than MBR, and GPT drives support files larger than 2 TB, whereas the older MBR drive type is capped at 2 TB.

- The appropriate file unit allocation size for SQL Server volumes is 64 KB, with few exceptions. Setting this to 64 KB for each volume can have a significant impact on storage efficiency and performance. The Windows default is 4 KB, which is not optimal.

  To check the file unit allocation size for an NT File System (NTFS) volume, run the following from the Administrator: Command Prompt, repeating for each volume:

  ```
  Fsutil fsinfo ntfsinfo d:
  Fsutil fsinfo ntfsinfo e:
  ...
  ```

The file unit allocation size is returned with the Bytes Per Cluster; thus 64 KB would be displayed as 65,536 (bytes).

Correcting the file unit allocation size requires formatting the drive, so it is important to check this setting prior to installation.

If you notice this on an existing SQL Server instance, your likely resolution steps are to create a new volume with the proper file unit allocation size and then move files to the new volume during an outage. Do *not* format or re-create the partition on volumes with existing data: you will lose the data.

Note that new Azure VM drives follow the Windows default of 4 KB; thus, you must reformat them to 64 KB.

- There is a hardware-level concept related to file unit allocation size called "disk starting offset" that deals with how Windows, storage, disk controllers, and cache segments align their boundaries. Aligning disk starting offset was far more important prior to Windows Server 2008. Since then, the default partition offset of 1,024 KB has been sufficient to align with the underlying disk's stripe unit size, which is a vendor-determined value. This should be verified in consultation with the drive vendor's information.

  To access the disk starting offset information, run the following from the Administrator: Command Prompt:

  ```
  wmic partition get BlockSize, StartingOffset, Name, Index
  ```

  A 1024 KB starting offset is a Windows default; this would be displayed as 1048576 (bytes) for Disk #0 Partition #0.

  Similar to the file unit allocation size, the only way to change a disk partition's starting offset is destructive—you must re-create the partition and reformat the volume.

SQL Server editions

NOTE

This book is not intended to be a reference for licensing or sales-related documentation; rather, editions are a key piece of knowledge for SQL administrators to know.

Following are brief descriptions for all of the editions in the SQL Server family, including past editions that you might recognize. It's important to use the appropriate licenses for SQL Server even in preproduction systems.

- **Enterprise edition.** Appropriate for production environments. Not appropriate for preproduction environments such as User Acceptance Testing (UAT), Quality Assurance (QA), testing, development, and sandbox. For these environments, instead use the free Developer edition.

- **Developer edition.** Appropriate for all preproduction environments. Not allowed for production environments. This edition supports the same features and capacity as Enterprise edition and is free.

- **Standard edition.** Appropriate for production environments. Lacks the scale and compliance features of Enterprise edition that might be required in some regulatory environments. Limited to the lesser of 4 sockets or 24 cores and also 128 GB of buffer pool memory, whereas Enterprise edition is limited only by the OS for compute and memory.

  NOTE

  In case you missed it, in Service Pack 1 of SQL Server 2016, a large number of features in Enterprise edition features were moved "down" into Standard, Web, and Express editions, including database snapshots, Columnstore indexes (limited), table partitioning, data compression, memory-optimized OLTP, PolyBase, SQL Audit, and the new Always Encrypted feature. Standard and Web edition also gained the ability to use the Change Data Capture (CDC) feature.

- **Web edition.** Appropriate for production environments, but limited to low-cost server environments for web applications.

- **Express edition.** Not appropriate for most production environments or preproduction environments. Appropriate only for environments in which data size is small, is not expected to grow, and can be backed up with external tools or scripts (because Express edition has no SQL Server Agent to back up its own databases). The free Express edition is ideal for proof-of-concepts, lightweight, or student applications. It lacks some critical features and is severely limited on compute (lesser of 1 socket or 4 cores), available buffer pool memory (1,410 MB), and individual database size (10 GB cap).

- **Express with Advanced Services.** Similar to Express edition in all caveats and limitations. This edition includes some features related to data tools, R integration, full-text search, and distributed replay that are not in Express edition.

- **Business Intelligence edition.** This edition was a part of the SQL Server 2012 and SQL Server 2014 products but was removed in SQL Server 2016.

- **Datacenter edition.** This edition was part of SQL Server until SQL 2008 R2 and has not been a part of the SQL Server product since SQL Server 2012.

NOTE

When you run the SQL Server 2017 installer, you are prompted to install a number of features outside of the core database features. Installing SQL Server features on multiple Windows servers requires multiple licenses, even if you intend to install each SQL Server instance's features only once.

There is an exception to this rule, and that is if you have licensed all physical cores a virtual host server with SQL Server Enterprise and purchased Software Assurance. Then, you can install any number or combination of SQL Server instances and their standalone features on virtual guests.

Changing SQL Server editions and versions

Upgrading editions in-place is supported by a feature of the SQL Server 2017 installer. You can upgrade in the following order: Express, Web, Standard, Enterprise. You also can upgrade Developer edition to Enterprise, Standard, or Web edition in SQL Server 2017.

It is important to note that you cannot downgrade a SQL Server version or licensed edition. This type of change requires a fresh installation and migration. For example, you cannot downgrade in-place from SQL Server 2017 Enterprise edition to Standard edition.

Upgrading versions in-place is supported but not recommended, if at all possible. Instead, we strongly recommend that you perform a fresh installation of the newer version and then move from old to new instances. This method offers major advantages in terms of duration of outage, rollback capability, and robust testing prior to migration. And, by using DNS aliases or SQL aliases, you can ease the transition for dependent application and connections.

Although in-place upgrades to SQL Server 2017 are not recommended, they are supported from as early as SQL Server 2008 on Windows Server 2012, assuming that the earlier versions are not 32-bit installations. Beginning with SQL Server 2016, SQL Server is available only for 64-bit platforms. SQL Server 2017 requires Windows Server 2012 or later.

You cannot perform an in-place upgrade from SQL Server 2005 to SQL Server 2017; however, you can attach or restore its databases to SQL Server 2017, though they will be upgraded to compatibility level 100, which is the version level for SQL 2008.

Installing a new instance

In this section, we discuss how to begin a new SQL Server 2017 instance installation, upgrade an existing installation, or add features to an existing instance.

It's important to note that even though you can change *almost* all of the decisions you make in SQL Server Setup after installation, those changes potentially require an outage or a server restart. Making the proper decisions at installation time is the best way to ensure the least administrative effort. Some security and service account decisions should be changed only via the SQL Server Configuration Manager application, not through the Services console (services.msc). This guidance will be repeated elsewhere for emphasis.

We begin by going through the typical interactive installation. Later in this chapter, we go over some of the command-line installation methods that you can use to automate the installation of a SQL Server instance.

Planning for multiple SQL Server instances

You can install as many as 50 SQL Server instances on a Windows Server; however, we do not recommend this. In a Windows failover cluster, the number of SQL Server instances is reduced by half if you're using shared cluster drives.

Only one of the SQL Server instances on a server can be the default instance. All, or all but one, of the SQL Server instances on a SQL Server will be named instances. The default instance is addressable by the name of the server alone, whereas named instances require an instance name. The SQL Browser service then is required to handle traffic between multiple instances on the SQL Server.

➤ **For more information about the SQL Server browser, go to Chapter 7.**

For example, you can reach the default instance of a SQL Server by connecting to *servername*. All named instances would have a unique instance name, such as *servername/instancename*.

In application connection strings, *servername/instancename* should be provided as the Server or Data Source parameter.

Installing a SQL Server instance

The instructions in this chapter are the same for the first installation or any subsequent installations, whether for the default or any named instances of SQL Server 2017. As opposed to an exhaustive step-by-step instruction list for installations, we've opted to cover the important decision points and the information you need, plus new features introduced in SQL Server 2016 and 2017.

Inside OUT

What if I have a new Azure VM?

You do not need to install SQL Server on new Azure VMs, because provisioning new Windows Servers with SQL Server are easily available in the Azure Marketplace.

There are two types of SQL Server licensing agreements for Azure VMs:

- **SQL Server VM images in the Azure Marketplace contain the SQL Server licensing costs as an all-in-one package. The SQL Server license is included in the per-minute pricing of the VM, is billed regularly along with other Azure assets, and does not need to be purchased separately.**
- **If you'd like to bring your existing Enterprise licensing agreement, there are three options:**
  - **Bring-your-own-license (BYOL) VM images available for you to provision using the same process and then later associate your existing Enterprise license agreements. The image names you're looking for here are prefixed with BYOL.**
  - **Manually upload an .iso to the VM and install SQL Server 2017 as you would on any other Windows Server.**
  - **Upload an image of an on-premises VM to provision the new Azure VM.**

It is important for you to keep in mind that you cannot change from the built-in licensing model to the BYOL licensing model after the VM has been provisioned. You need to make this decision prior to creating your Azure VM.

Inside the SQL Server Installation Center

While logged in as a local Windows administrator, begin by mounting the installation .iso to the Windows server. These days, this rarely involves inserting a physical disc or USB flash drive; although you can use them if necessary. Unpacking the contents of the .iso file to a physical file folder over the network would also provide for a faster SQL Setup experience.

You should not run Setup with the installation media mounted over a remote network connection, via a shared remote desktop drive, or any other high-latency connection. It would be faster to copy the files locally before running Setup.

Start Setup.exe on the SQL Server Setup media, running the program as an administrator. If AutoPlay is not turned off (it usually is), Setup.exe will start when you first mount the media or double-click to open the .iso. Instead, as a best practice, right-click Setup.exe and then, on the shortcut menu that appears, click Run As Administrator.

We'll review here a few items (not all) in the SQL Server Installation Center worth noting before you begin an installation.

In the tab pane on the left, click Planning to open a long list of links to Microsoft documentation websites. Most helpful here might be a standalone version of the System Configuration Checker, which you run during SQL Server Setup later, but it could save you a few steps if you review it now. A link to download the Upgrade Advisor (now renamed to the Data Migration Assistant) is also present, a helpful Microsoft-provided tool when upgrading from prior versions of SQL Server.

On the Maintenance page, you will find the following:

- A link to launch the relatively painless Edition Upgrade Wizard. This is for changing your existing installation's edition, not its version (SQL 2012, 2014, 2016, 2017) or platform (x86 versus 64-bit). You can upgrade from "lower" editions to higher editions, but keep in mind that downgrading editions is not possible and would require that you install a new SQL Server. For example, you cannot downgrade from Enterprise to Standard edition without a new installation. For complete details on Edition upgrade paths, visit *https://docs.microsoft.com/sql/database-engine/install-windows/supported-version-and-edition-upgrades#includesscurrentincludessscurrent-mdmd-edition-upgrade*

- The Repair feature is not a commonly used feature. It's use is necessitated by a SQL Server with a corrupted Windows installation. You might also need to repair an instance of SQL Server when the executables, .dll files, or registry entries have become corrupted before repair. A failed SQL Server in-place upgrade or cumulative update installation might also require a Repair, which could be better than starting from scratch.

- Whereas adding a node to an existing SQL Server failover cluster is an option in the Installation menu, removing a node from an existing SQL Server failover cluster is an option in the Maintenance menu.

- A link to the Update Center for SQL Server (Technet Article ff803383) provides information on the latest cumulative updates for each version of SQL Server.

- On the Tools menu, there is a link to the Microsoft Assessment and Planning (MAP) Toolkit for SQL Server, which is a free download that can be invaluable when you're performing an inventory of your SQL Server presence in a network. It's also capable of searching for SQL Server instances by a variety of methods and generating CIO-level reports.

- On the Advanced menu, there is a link to perform an installation based on a configuration file, which we discuss later in this chapter in the section "Automating SQL Server Setup by using configuration files."

There are also links to wizards for advanced failover cluster installations.

➤ **We discuss Failover Cluster Instances (FCIs) in Chapter 12.**

Installing SQL Server

In the SQL Server Installation Center, in the pane on the left side, click Installation. Although what follows in this chapter is not a step-by-step walk-through, we'll cover key new features and decision points.

Inside OUT

Where are SQL Server Management Studio, SQL Server Data Tools, and SQL Server Reporting Services?

SQL Server Management Studio, SQL Server Data Tools (for Visual Studio 2015 and higher), and SQL Server Reporting Services are no longer installed with SQL Server's traditional setup media. These products are now updated regularly (monthly usually) and available for download.

As of SQL Server 2016, Management Tools is no longer an option on the Feature Selection page. Although both tools are listed in the SQL Server Installation Center, they are simply links to free downloads.

Remember to keep up-to-date versions of SQL Server Management Studio and SQL Server Data Tools on administrator workstations and laptops. Communicate with your team so that everyone is using the same releases of these free tools for on-premises SQL Server as well as Azure SQL Database administration.

The standalone version of SQL Server Reporting Services is now a 90-MB download that launches its own installer, but it still needs a SQL Server Database Engine instance as part of the license and to host the two Report Server databases. Note that this isn't a licensing change and that SQL Server Reporting Services isn't free, you will need to provide a license key or choose a nonproduction edition to install (Evaluation, Developer, or Express).

Installing options and features

In this section, we discuss the installation details of other features of SQL Server, including new options in Setup that were not available in previous versions.

Data analytics and artificial intelligence features

The Feature Selection page has a pair of new options in SQL Server 2017, both greatly expanding SQL Server's footprint into the big data field. Let's take a look at each option.

PolyBase Query Service For External Data The PolyBase Query Engine makes it possible to query Hadoop nonrelational data or Azure Blob Storage files by using the same Transact-SQL (T-SQL) language with which SQL Server developers are already familiar. You may have only one SQL Server Instance with the PolyBase feature installed on a Windows Server.

As strange as this sounds, this functionality of SQL Server requires that you install the Oracle SE Java Runtime Environment (JRE). If while on the Feature Rules page you encounter the error "Oracle JRE 7 Update 51 (64-bit) or higher is required," you can ignore the message and, in the background, proceed with the installation of Oracle JRE 7. You do not need to restart Windows or SQL Server Setup after a successful installation of the Oracle JRE, and on the Feature Rules page, the Re-Run button should clear that error so that you can proceed with SQL Server Setup.

This is a manual installation that is not contained in the Setup components or automatically downloaded by the Smart Setup (more on that later). To download the latest version, visit the Oracle JRE site at *http://www.oracle.com/technetwork/java/javase/downloads/index.html*.

Choose the JRE download because the Server JRE does not include an installer for Windows Platforms. Choose the Windows x64 Offline installer, which is a self-installing .exe file, and then complete the installation.

You must configure at least six individual ports in a range (with TCP 16450 through 16460 the default) for the PolyBase Engine. You can move forward with the defaults if these ranges are unclaimed in your network.

Later, on the Server Configuration page, you will choose a service account for the SQL Server PolyBase Engine service and the SQL Server PolyBase Data Movement service. We recommend a dedicated Windows Authenticated service account, with a special note that if it is a part of a scale-out of PolyBase instances, all of the instances should use the same service account.

Finally, after SQL Server Setup is complete, if you installed both the Database Engine and PolyBase at the same time, the two PolyBase services will be in the "Change Pending" state. They are unable to connect because, by default, TCP is not an activated protocol for the SQL Server instance. This is a common post-installation to-do item for other reasons, so turning on the TCP protocol for the new SQL Server instance, followed by restarting the SQL Server service, is required.

After the new SQL Server installation is complete, review the "Hadoop Connectivity" setting by using `sp_configure`. The setting ranges from 0 to 7, with options 1 through 6 dealing mostly with older versions of Hortonworks Data Platform. Setting 7 allows for the connectivity with

recent Hortonworks HDP versions as well as Azure Blob storage. To change this `sp_configure` setting, you must run the `RECONFIGURE` step and also restart the SQL Server service, and then manually start the two SQL Server PolyBase services.

> NOTE
>
> **Complete information on the settings for Hadoop connectivity are available in the documentation at *https://docs.microsoft.com/sql/database-engine/configure-windows/polybase-connectivity-configuration-transact-sql*.**

Additional steps, including a firewall change, are needed to install this feature as part of a PolyBase Scale-Out Group of multiple SQL Server instances, with one PolyBase Engine service per Windows Server.

Machine learning features The newly named Machine Learning Services (In-Database or the standalone Machine Learning Server) feature makes it possible for developers to integrate with R language and/or Python language extensions using standard T-SQL statements. Data scientists can take advantage of this feature to build advanced analytics, data forecasting, and algorithms for machine learning. Formerly called the Revolution R engine, SQL Server 2017 installs version 9 of the Microsoft R Open Server, supported for both Windows and Linux.

This feature adds the SQL Server Launchpad service. You cannot configure the service account for the Launchpad service; it will run as a dedicated NT Service\MSSQLLaunchpad virtual account. The standalone installation of Machine Learning Services does not create the SQL Server Launchpad service and is intended for models that do not need a SQL Server.

After the new SQL Server installation is complete, you must turn on a security option to allow external scripts. This makes it possible for you to run non-T-SQL language scripts, and in the SQL Server 2017 release of this feature, R and Python are the only languages supported. (In SQL Server 2016, only R was supported, thus the name change from R Services to Machine Learning Services.) Use `sp_configure` to select the External Scripts Enabled option, reconfigure, and restart the SQL Server service.

Grant Perform Volume Maintenance Tasks feature of SQL Server Setup

On the same Server Configuration Setup page on which service accounts are set, you will see a check box labeled Grant Perform Volume Maintenance Task Privilege To The SQL Server Database Engine Service. This option was added in SQL Server 2016.

This automates what used to be a standard post-installation checklist step for SQL DBAs since Windows Server 2003. The reason to grant this permission to use instant file initialization is to speed the allocation of large database data files, which could dramatically reduce the Recovery Time Objective (RTO) capacity for disaster recovery.

This can mean the difference between hours and minutes when restoring a very large database. It also can have a positive impact when creating databases with large initial sizes, or in large autogrowth events; for example, with multiple data files in the TempDB (more on this next). It is recommended that you allow SQL Setup to turn on this setting.

➤ **For more information on instant file initialization, see Chapter 3.**

Default settings for the TempDB database

Starting with SQL Server 2016, SQL Server Setup provides a more realistic default configuration for the number and size of TempDB data files. This was a common to-do list for all post-installation checklists for DBAs since the early days of SQL Server.

The TempDB database page in SQL Server Setup provides not only the ability to specify the number and location of the TempDB's data and log files, but also their initial size and autogrowth rates. The best number of TempDB data files is almost certainly greater than one, and less than or equal to the number of logical processor cores. Adding too many TempDB data files could in fact severely degrade SQL Server performance.

➤ **For more information on the best number of TempDB data files, see Chapter 3.**

Specifying TempDB's initial size to a larger, normal operating size is important and can improve performance after a SQL Server restart when the TempDB data files are reset to their initial size. Setup accommodates an individual TempDB data file initial size up to 256 GB. For data file initial sizes larger than 1 GB, you will be warned that Setup could take a long time to complete if instant file initialization is turned on by granting the Perform Volume Maintenance Task for the SQL Server Service Account. (This should be accomplished automatically by Setup; see the previous section.)

All TempDB files autogrow at the same time, keeping file sizes the same over time. Previously only available as a server-level setting via trace flag 1117, TempDB data files have behaved in this way by default since SQL Server 2016.

Note also the new naming convention for the second TempDB data file and beyond: tempdb_mssql_*n*.ndf. A SQL Server uninstallation will automatically clean up TempDB data files with this naming convention—for this reason, we recommend that you follow this naming convention for TempDB data files.

➤ **TempDB is discussed in greater detail in Chapter 3.**

Figure 4-1 depicts a VM with four logical processors. Note that the number of files is by default set to the number of logical processors. The sizes, autogrowth settings, and data directories have been changed from their defaults, you should consider doing the same.

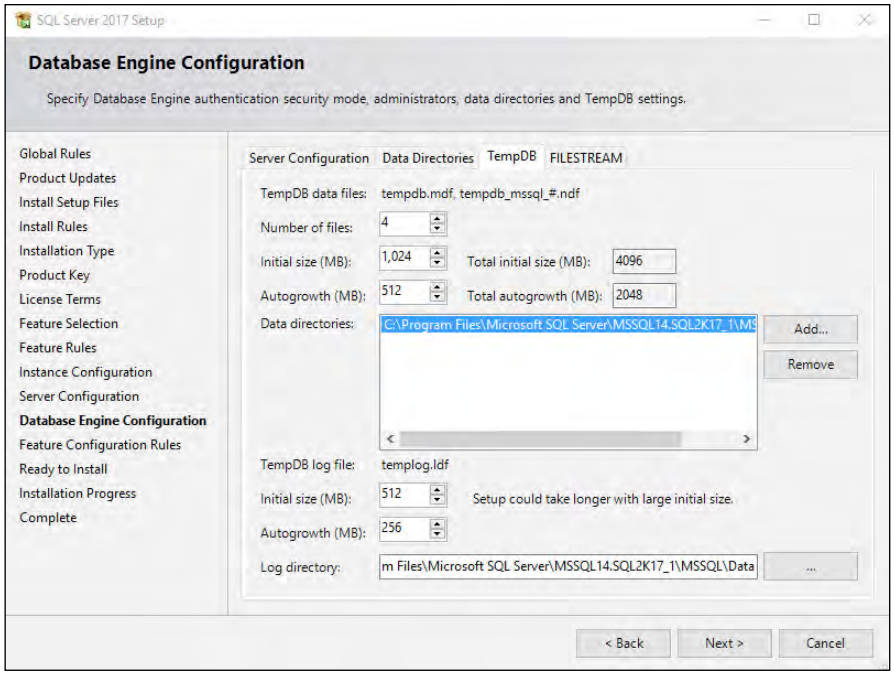

Figure 4-1 The Database Engine Configuration page in SQL Server 2017 Setup, including the TempDB tab.

Mixed Mode authentication

SQL Server supports two modes of authentication: Windows and SQL Server.

➤ **You can read more on this topic in Chapter 7, but it is important to note this decision point here.**

Ideally, all authentication is made via Windows Authentication, through types of server principals called *logins*, that reference Domain accounts in your Enterprise edition. These domain accounts are created by your existing enterprise security team, which manages password policy, password resets, password expiration, and so on.

A redundant security model for connecting to SQL Server also exists within each instance: SQL Server Authenticated logins. Logins are maintained at the SQL Server level, are subject to local policy password complexity requirements, are reset/unlocked by SQL DBAs, have their own password change policy, and so forth.

Turning on Mixed Mode (SQL Server and Windows Authentication Mode) activates SQL Server Authenticated login. It is important to note that it is not on by default and not the preferred method of connection. By default, only Windows Authentication is turned on and cannot be turned off. When possible, applications and users should use Windows Authentication.

Turning on Mixed Mode also activates the "sa" account, which is a special built-in SQL Server Authentication that is a member of the server sysadmin role. Setup will ask for a strong password to be provided at this time.

➤ **You can learn more about the "sa" account and server roles in Chapter 7.**

It is important to keep in mind that you do not need to turn on SQL Server Authentication in Setup; you can do this later on by connecting to the SQL Server via Object Explorer in SQL Server Management Studio. To do so, right-click the server name and then, on the shortcut menu that opens, click Properties, and then click the Security page. You must perform a service restart to make this change effective.

Installing other core features

Aside from the SQL Server service itself, three common core features of the product might be common to your installations. SQL Server Analysis Services, SQL Server Integration Services, and SQL Server Reporting Services are part of the license and are provided at no additional cost. If you need them, this section covers installing these features using Setup. Later in this chapter, we review the post-installation steps necessary to use them.

Installing SQL Server Analysis Services

Installing SQL Server Analysis Services requires you to make a decision at installation time regarding the mode in which it can be installed. Each instance of SQL Server Analysis Services can be in only one mode, which means that with a single license, you can run only the classic Multidimensional mode, the newer Tabular mode (introduced in SQL 2012), or the Power Pivot mode. Ask your business intelligence decision makers which platform you should use. Following are brief descriptions of each mode:

- **Multidimensional mode.** This is the familiar SQL Server Analysis Services setup that was first introduced in SQL 2000. This is also the mode to support data mining.
- **Tabular mode.** This is the newer SQL Server Analysis Services setup that was first introduced in SQL 2012, using the in-memory VertiPaq processing engine. For the first time in SQL Server 2017, this is the default installation mode selected on the Analysis Services Configuration page of Setup.
- **Power Pivot mode.** This mode installs SQL Server Analysis Services in the Power Pivot for SharePoint mode.

Inside OUT

What if you choose the wrong SQL Server Analysis Services mode?

If you choose one SQL Server Analysis Services mode at installation but your business intelligence developers want another mode, the supported option is to uninstall and reinstall the SQL Server Analysis Services feature. Changing the SQL Server Analysis Services mode from Multidimensional to Tabular, or vice versa, after installation is not supported and administrators are specifically warned not to do this.

Packages developed for each mode are not supported for the other. If no databases have been deployed to the SQL Server Analysis Services server instance, changing the `DeploymentMode` property in the MSMDSRV.ini file should make it possible to change an existing instance, but, again, this is not a supported change. The file is located in *%Programfiles%\Microsoft SQL Server\MSAS13.TABULAR\OLAP\Config*.

Installing SQL Server Integration Services

The SQL Server Integration Services instance for SQL Server 2017 is installed once per server per version, not once per instance, like other features. However, starting in SQL Server 2017, a new Integration Services Scale Out Configuration is available. We discuss this new feature further in the next section.

Also unlike other features, you can install SQL Server Integration Services on a 32-bit OS; however, we do not recommend this. A 64-bit version of SQL Server Integration Services is installed on 64-bit operating systems. If you worry about connecting to 32-bit servers, data sources, or applications installations (such as Microsoft Office), don't—those connections are not dependent on the 32-bit/64-bit installation and are handled at the package or connection-string level.

A standalone installation of SQL Server Integration Services without a matching SQL Server Database Engine is possible but not recommended. For the modern Project Deployment model of SQL Server Integration Services, the storage and logging of packages will still be dependent on a SQL Server Database Engine, and the execution of packages on a schedule would still require a SQL Agent service. Isolation of the SQL Server Integration Services workload is not best isolated in this way. A dedicated installation including the SQL Server Database Engine and SQL Server Agent is a better configuration to isolate SQL Server Integration Services package runtime workloads from other database workloads.

Installations of different versions of SQL Server Integration Services are installed side by side on a server; specifically, the service SQL Server Integration Services 14.0 is compatible with prior versions.

Outside of configuring the service account, you do not need any additional configuration when installing SQL Server Integration Services during SQL Server Setup. The default virtual service account is NT Service\MsDtsServer140. Note that this account is different from the Scale Out Master and Scale Out Worker service accounts, and is used differently. Let's talk about the Scale Out feature now.

Installing SQL Server Integration Services Scale Out configuration

A new feature in SQL Server 2017, Integration Services now supports a Scale Out configuration by which you can run a package on multiple SQL Server instances. This also allows for high availability of SQL Server Integration Services, with the secondary.

The master node talks to worker nodes in a SQL Server Integration Services Scale Out, with the communication over a port (8391 by default) and secured via a new Secure Sockets Layer (SSL) certificate. The SQL Server installer can automatically create a 10-year self-signed certificate and endpoint for communication at the time the master node is set up.

When adding another SQL Server Integration Services installation as a Scale Out Worker, start the new SQL Server Integration Services Manage Scale Out window via SQL Server Management Studio. Right-click the Catalog you have created, and then click Manage Scale Out. At the bottom of the page, click the + button to add a new Scale Out Worker node. Provide the server name to which to connect. If using a named instance, still provide only the server name of the node; do not include the instance name. A dialog box confirms the steps taken to add the Worker node, including copying and installing certificates between the Worker node and Master node, updating the endpoint and HttpsCertThumbprint of the worker, and restarting the Worker's Scale Out service. After the worker node is added, refresh the Worker Manager page, and then click the new Worker node entry, which will be red. You must turn on the Worker Node by clicking Enable Worker.

You also can copy and install the certificates manually between servers. You will find them in %program files%\Microsoft SQL Server\140\DTS\Binn\.

The Worker and Master nodes do not appear in SQL Server Configuration Manager (as of SQL 2017 RC2) but do appear in Services.msc.

One major security difference with Scale Out is that even though the SQL Server Integration Services Service Account doesn't run packages or need permission to do very much, the Scale Out Master and Worker service accounts actually do run packages. By default, these services run under virtual accounts NT Service\SSISScaleOutMaster140 and NT Service\SSISScaleOutWorker140, but you might want to change these to a Windows-authenticated Domain service account that will be used to run packages across the Scale Out.

Installing SQL Server Reporting Services

Starting with SQL Server 2017, SQL Server Reporting Services is no longer found in the SQL Server Setup media; it is instead available as a simplified, unified installer and a small download. SQL Server Reporting Services is now a 90-MB download that launches its own installer but still needs a SQL Server Database Engine instance as part of the license and to host the two Report Server databases. Note that SQL Server Reporting Services isn't free, and that the separate installer isn't a licensing change. You will need to provide a license key upon installation or choose a nonproduction edition to install (Evaluation, Developer, or Express).

The "native" mode of SQL Server Reporting Services is now the only mode in SQL Server 2017. If you are familiar with Reporting Services Report Manager in the past, accessible via the URL http://servername/Reports, that is the "native mode" installation of Reporting Services.

You'll notice the Report Server Configuration Manager in a new location, in its own Program Files menu, Microsoft SQL Server Reporting Services. After installation, start the Report Server Configuration Manager (typically installed in a path like *\Program Files (x86)\Microsoft SQL Server\140\Tools\Binn\RSConfigTool.exe*). The Report Server Configuration Manager application itself is largely unchanged since SQL 2008.

The virtual service account "NT SERVICE\SQLServerReportingServices" is the default SQL Server Reporting Services service account. It is a second-best option, however: we recommend that you create a new domain service account to be used only for this service; for example, "*Domain*\svc_*ServerName*_SSRS" or a similar naming convention. You will need to use a domain account if you choose to configure report server email with "Report server service account (NTLM)" authentication.

If you choose to change the SQL Server Reporting Services service account later, use only the Reporting Services Configuration Manager tool to make this change. Like other SQL Server services, never use the Services console (services.msc) to change service accounts.

After installation, you will need to follow-up on other changes and necessary administrative actions; for example, configuring the SQL Server Reporting Services Execution Account, email settings, or backing up the encryption key using Reporting Services Configuration Manager.

SQL Server 2017 Reporting Services also can integrate with Microsoft Power BI dashboards. A page in the Report Server Configuration Manager supports registering this installation of SQL Server Reporting Services with a Power BI account. You will be prompted to sign into Azure Active Directory. The account you provide must be a member of the Azure tenant where you intend to integrate with Power BI. The account should also be a member of the system administrator in SQL Server Reporting Services, via Report Manager, and a member of the sysadmin role in the SQL Server that hosts the Report Server database.

Inside OUT

Where is SQL Server Reporting Services SharePoint Integrated mode?

There is no more SharePoint Integrated mode, the simplified "native" mode download is the only installation available. This matches the moves that Microsoft has made in other areas that step away from the SharePoint on-premises product in favor of SharePoint Online features and development.

Instead, you can integrate SQL Server Reporting Services native mode with on-premises SharePoint sites via embedded SQL Server Reporting Services reports, including SQL Server Reporting Services reports stored in the Power BI Report Server.

Similarly, there is no future support for SQL Server Reporting Services integration with SharePoint Online.

"Smart Setup"

Since SQL Server 2012, the SQL Server installer has had the ability to patch itself while within the Setup wizard. The Product Updates page is presented after the License Terms page, and, after you accept it, it is downloaded from Windows Update (or Windows Server Update Services) and installed along with other SQL Server Setup files.

This is recommended, and so a SQL Server 2017 Setup with internet connectivity is the easiest way to carry out installation. This also could be described as a way to "slip-stream" updates, including hotfixes and cumulative updates, into the SQL Server installation process, eliminating these efforts post-installation.

For servers without internet access, there are two Setup.exe parameters that support downloading these files to an accessible location and making them available to Setup. When starting the SQL Server 2017 .iso's Setup.exe from Windows PowerShell or the command line (you can read more about this in the next section), you can set the `/UpdateEnabled` parameter to FALSE to turn off the download from Windows Update. The `/UpdateSource` parameter can then be provided as an installation location of unpacked .exe files. Note that the `/UpdateSource` parameter is a folder location, not a file.

Setting up logging

SQL Server Setup generates a large number of logging files for diagnostic and troubleshooting purposes. These logs should be the first place you go when you have an issue with Setup.

After you proceed past the Ready To Install page, and regardless of whether Setup was a complete success, it generates a number of log files in the following folder:

%programfiles%\Microsoft SQL Server\140\Setup Bootstrap\Log*YYYYMMDD_HHMMSS*\

Here's an example:

C:\Program Files\Microsoft SQL Server\140\Setup Bootstrap\Log\20170209_071118\

However, when you run Setup using the /Q or /QS parameters for unattended installation, the log file is written to the Windows %temp% folder.

A log summary file of the installation is created that uses the following naming convention:

Summary*_instancename_YYYYMMDD_HHMMSS*.txt

Setup generates similar files for the Component and Global Rules portions of Setup as well as a file called Detail.txt. These files might contain the detailed error messages you are looking for when troubleshooting a failed installation. The Windows Application Event log might also contain helpful information in that situation.

Finally, a System Configuration Check report .htm file is generated each time you run Setup, as well.

You'll also find the new SQL Server instance's first error log encoded at UTC time in this folder, showing the log from startup, similar to the normal SQL Server Error Log.

Automating SQL Server Setup by using configuration files

Let's dig more into what we can do with Setup.exe outside of the user interface. You can use configuration files to automate the selection process when installing SQL Server, which helps to create a consistent configuration.

Values provided in configuration files can prepopulate or override Setup settings. They also can configure Setup to run with the normal user interface or silently without any interface.

Starting Setup from prompt

You can start setup.exe from either Windows PowerShell or the command prompt, providing repeatability and standardization of parameter options. You also can use it to prefill sections of the Setup wizard or to change the default behavior of Setup.

For the purposes of the installer, ensure that you always use the Administrator level for these two prompts. The title on each page should be preceded by "Administrator: "; for example, Administrator: Windows PowerShell.

To start Windows PowerShell or command prompt as Administrator, in the Start menu, search for the desired application, right-click it, and then, on the shortcut menu that opens, select Run As Administrator.

Figure 4-2 shows an example of starting Setup.exe from the Windows PowerShell prompt, and Figure 4-3 shows starting it from the command prompt.

```
PS E:\> .\setup.exe /ACTION=install /UPDATEENABLED=false /UPDATESOURCE="d:\downloaded install files"
Microsoft (R) SQL Server 2017 RC2 14.00.900.75
Copyright (c) 2017 Microsoft.  All rights reserved.
```

Figure 4-2 Starting Setup.exe from the Windows PowerShell prompt.

```
E:\>setup.exe /ACTION=install /UPDATEENABLED=false /UPDATESOURCE="D:\downloaded install files"
Microsoft (R) SQL Server 2017 RC2 14.00.900.75
Copyright (c) 2017 Microsoft.  All rights reserved.
```

Figure 4-3 Starting Setup.exe from the command prompt.

Sometimes, you also might find it necessary to start Setup from the command line or Windows PowerShell because of a workaround for a specific problem.

Generating a configuration file

Writing a configuration file by hand is not necessary and can be tedious. Instead of going through that effort, you can let SQL Server Setup create a configuration file for you. Here's how to do that. Work your way through the normal SQL Server Setup user interface, completing everything as you normally would, but pause when you get to the Ready To Install page. Near the bottom of this page is a path (see Figure 4-4). At that location, you'll find a generated configuration file, ready for future use.

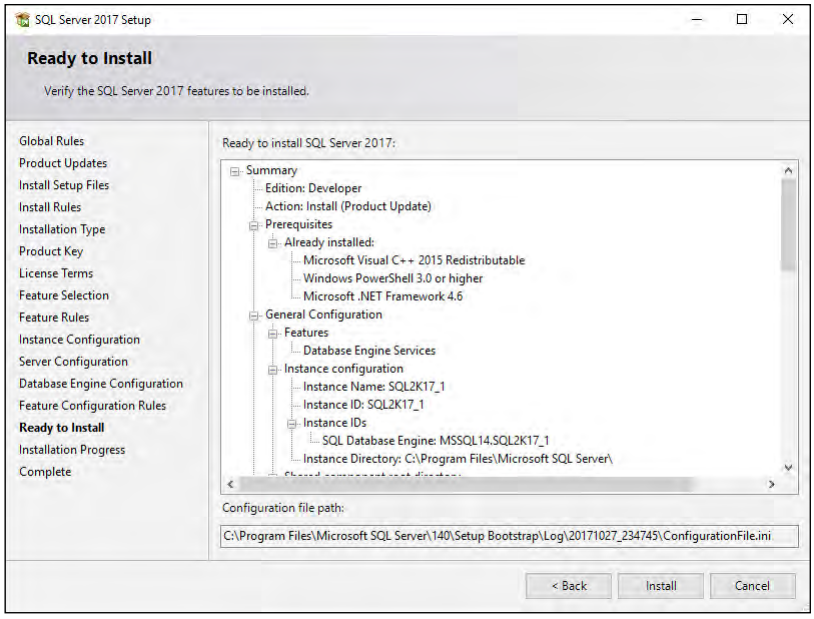

Figure 4-4 The Ready To Install page displays a summary of the installation steps that are about to begin as well as the Configuration File Path that has been prepared based on the selections.

Installing by using a configuration file

Now that you have a configuration file that you either prepared yourself or generated by using a previous walk-through of Setup, you can take the next step to automating or standardizing your installation.

You can start Setup.exe with a configuration file by using the `/CONFIGURATIONFILE` parameter of Setup.exe, or by navigating to the Advanced page of the SQL Server Installation Center that starts with Setup.exe in Windows. Then, start Setup.exe with a configuration file by selecting the Install Based On A Configuration File check box. A message appears, asking you to browse to the .ini file. After you select the appropriate file, Setup.exe will start with those options.

One thing to keep mind, however, is that configuration files generated by Setup.exe do not store the passwords you provided for any service accounts. If you do want to configure service account credentials in your configuration file, for security reasons, do not store the service account passwords in plain text in a configuration file. You should instead store them securely and provide them when you run Setup.exe. Each service's account information is available in a Setup.exe runtime parameter, which are listed in Table 4-1.

Table 4-1 Common Setup.exe parameters and their purposes

Service	Parameter name	Description
SQL Server Database Engine	/SQLSVCPASSWORD	Password for the main SQL Server Database Engine Services service account. This is the service account for sqlservr.exe. It is required if a domain account is used for the service.
SQL Server Agent	/AGTSVCPASSWORD	Password for the SQL Server Agent service account. This is the service account for sqlagent.exe. It is required if a domain account is used for the service.
sa password	/SAPWD	Password for the sa account. It is required if Mixed Mode is selected, or when /SECURITYMODE=SQL is used.
Integration Services	/ISSVCPASSWORD	Password for the Integration Services service. It is required if a domain account is used for the service.
Reporting Services (Native)	/RSSVCPASSWORD	Password for the Reporting Services service. It is required if a domain account is used for the service.
Analysis Services	/ASSVCPASSWORD	Password for the Analysis Services service account. It is required if a domain account is used for the service.
PolyBase	/PBDMSSVCPASSWORD	Password for the PolyBase engine service account.
Full-Text filter launcher service	/FTSVCPASSWORD	Password for the Full-Text filter launcher service.

For example, in the snippet that follows, the PROD_ConfigurationFile_Install.INI has provided the account name of the of the SQL Server Database Engine service account, but the password is provided when Setup.exe runs:

```
Setup.exe /SQLSVCPASSWORD="securepasswordhere"
/ConfigurationFile="d:\SQLInstaller\Configuration
Files\PROD_ConfigurationFile_Install.INI"
```

You can provide further parameters like passwords when you run Setup. Parameter settings provided will override any settings in the configuration file, just as the configuration file's settings will override any defaults in the Setup operation. Table 4-2 lists and describes the parameters.

Table 4-2 Common Setup.exe parameters of which you should be aware

Parameter usage	Parameter	Description
Unattended installations	/Q	Specifies Quiet Mode with no user interface and user interactivity allowed.
Unattended installations	/QS	Specifies Quiet Mode with user interface but no user interactivity allowed.
Accept license terms	/IACCEPTSQLSERVERLICENSETERMS	Must be provided in any Configuration File looking to avoid prompts for installation.
R open license accept terms	/IACCEPTROPENLICENSEAGREEMENT	Similarly, must provide this parameter for any unattended installation involving either of the two R Server options.
Configuration file	/CONFIGURATIONFILE	A path to the configuration .ini file to use.
Instant file initialization	/SQLSVCINSTANTFILEINIT	Set to `true` to Grant Perform Volume Maintenance Task privilege to the SQL Server Database Engine Service (recommended)
TempDB data file count	/SQLTEMPDBFILECOUNT	Set to the number of desired TempDB data files to be installed initially.

Post-installation server configuration

After you install SQL Server, there are a number of changes to make or confirm on the Windows Server and in settings for SQL Server.

Post-installation checklist

You should run through the following checklist on your new SQL Server instance. The order of these items isn't necessarily specific, and many deal with SQL server and/or Windows configuration settings. Evaluate whether these are appropriate for your environment, but you should consider and apply them to most SQL Server installations.

1. Check your SQL Server version.
2. Configure the Maximum Server Memory setting.

3. Surface Area Configuration.
4. Set up SQL Agent.
5. Turn on TCP/IP.
6. Verify power options.
7. Configure antivirus exclusions.
8. Optimize for ad hoc workloads.
9. Lock pages in memory.
10. System page file.
11. Backups, index maintenance, and integrity checks.
12. Backup service master and database master keys.
13. Default log retention.
14. Suppress successful backup messages.

Let's take a look at each of these in more detail in the subsections that follow.

Check your SQL Server version

After you install SQL Server, check the version number against the latest cumulative updates list, especially if you did not opt to or could not use Windows Update during SQL Server Setup. You can view the version number in SQL Server Management Studio's Object Explorer or via a T-SQL query on either the following two built-in functions:

```
SELECT @@VERSION;
select SERVERPROPERTY('ProductVersion');
```

NOTE

Take the opportunity before your SQL Server enters production to patch it. For information about the latest cumulative updates for SQL Server, search for KB321185 or visit *https://support.microsoft.com/help/321185/how-to-determine-the-version,-edition-and-update-level-of-sql-server-and-its-components*.

Configure the Maximum Server Memory setting

We discussed the Maximum Server Memory setting in Chapter 3, in the section "Configuration settings," but it is definitely on the list to set post-installation for any new SQL Server instance.

This setting is accessible via SQL Server Management Studio, in Object Explorer, on the Server Properties page. On the Memory page, look for the Maximum Server Memory (In MB) box. This value defaults to 2147483647, which does not limit the amount of memory SQL Server can access in the Windows server. This value is also available in `sp_configure`, when Show Advanced Options is turned on, under the configuration setting Max Server Memory (MB).

➤ **You can read more about why you should limit the Max Server Memory setting in Chapter 3.**

An example of configuring a Windows Server with one SQL Server instance and 64 GB of memory, to use a Max Server Memory setting of 58982 MB, as illustrated in Figure 4-5.

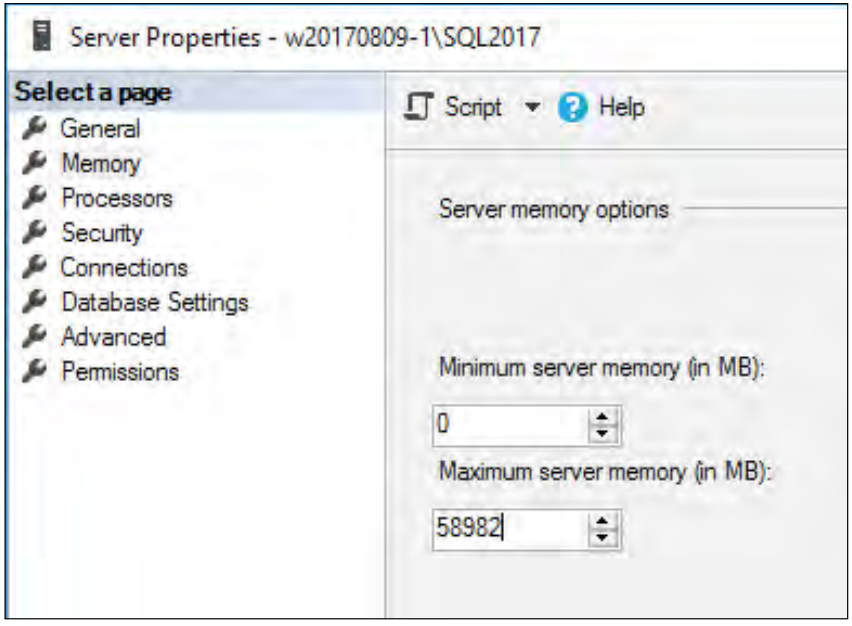

Figure 4-5 The Server Properties dialog box with the Memory page selected.

Keep lowering this number if you have other applications on the server that will be consuming memory, including other SQL Server instances. We discuss this more in the next section. If you observe over time that during normal operations your server's total memory capacity is nearly exhausted (less than 2 GB free), lower the Maximum Server Memory setting further. If there is sufficient padding available (more than 8 GB free), you can consider raising this setting.

Lowering this setting after installation and during operation does not return SQL Server memory back to the OS immediately; rather, it does so over time during SQL Server activity. Similarly, increasing this setting will not take effect immediately.

Just above the Maximum Server Memory setting is the Minimum Server Memory setting, which establishes a floor for memory allocation. It is not generally needed. You might find this setting useful for situations in which the total system memory is insufficient and many applications, including SQL Server instances, are present. The minimum server memory is not immediately allocated to the SQL Server instance upon startup; instead, it does not allow memory below this level to be freed for other applications.

Maximum server memory settings for other features

Other features of SQL Server have their own maximum server memory settings. As you will notice by their default settings, for servers on which both the SQL Server Database Engine and SQL Server Analysis Services and/or SQL Server Reporting Services are installed, competition for and exhaustion of memory is likely in environments with a significant amount of activity. It is recommended that you protect the Database Engine by lowering the potential memory impact of other applications.

Limiting SQL Server Analysis Services memory SQL Server Analysis Services has not just one maximum server memory limit, but four, and you can enforce limits by hard values in bytes or by a percentage of total physical memory of the server.

You can change these via SQL Server Management Studio by connecting to the SQL Server Analysis Services instance in Object Explorer. Right-click the server, and then, on the shortcut menu, click Properties. The memory settings described here are available and nearly identical for Multidimensional and Tabular installations:

- **LowMemoryLimit.** A value that serves as a floor for memory, but also the level at which SQL Server Analysis Services begins to release memory for infrequently used or low-priority objects in its cache. The default value is 65, or 65 percent of total server physical memory (or the Virtual Address Space technically, but Analysis Services, among other features, is not supported on 32-bit systems, and so this is not a concern).

- **TotalMemoryLimit.** A value that serves as a threshold for SQL Server Analysis Services to begin to release memory for higher priority requests. It's important to note that this is not a hard limit. The default is 80 percent of total server memory.

- **HardMemoryLimit.** This is a hard memory limit that will lead to more aggressive pruning of memory from cache and potentially to the rejection of new requests. By default, this is displayed as 0 and is effectively the midway point between the `TotalMemoryLimit` and the server physical memory. The `TotalMemoryLimit` must always be less than the `HardMemoryLimit`.

- **VertiPaqMemoryLimit.** For SQL Server Analysis Services installations in Tabular mode only, the Low Memory setting is similarly enforced by the `VertiPaqMemoryLimit`, which has a default of 60, or 60 percent of server physical memory. After this threshold is reached, and only if `VertiPaqPagingPolicy` is turned on (it is by default), SQL Server Analysis Services begins to page data to the hard drive using the OS page file. Paging to a drive can help prevent out-of-memory errors when the `HardMemoryLimit` is met.

Figure 4-6 shows the General page of the Analysis Server Properties dialog box, as started in Object Explorer, and the locations of the preceding memory configuration properties.

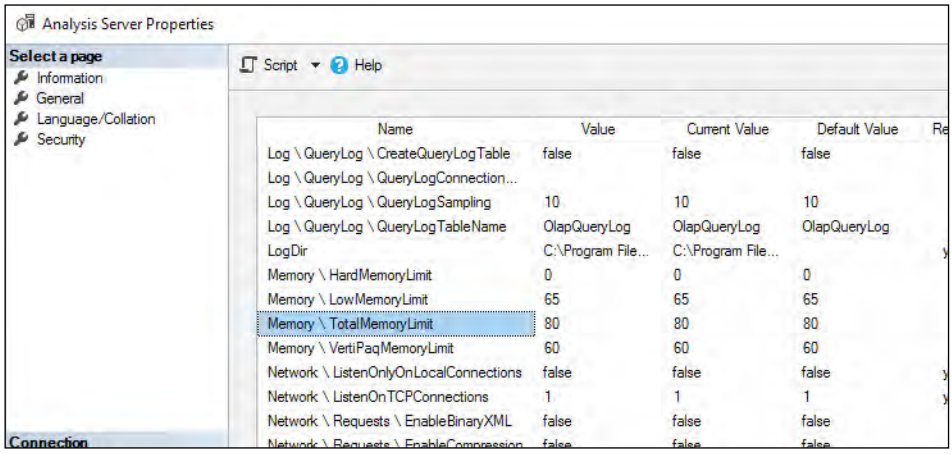

Figure 4-6 The General page in the Analysis Server Properties dialog box showing the default settings.

Limiting SQL Server Reporting Services memory You can configure memory settings only in the rsreportserver.config file, which is a text file that is stored at %ProgramFiles%\Microsoft SQL Server Reporting Services\SSRS\ReportServer.

> NOTE
>
> **This location has changed from previous versions, but the config file name has not.**

Four options are available for limiting SQL Server Reporting Services memory utilization, and all are based on numbers contained in tags within this .config file, so be sure to make a backup of it before editing.

Two of the settings are in the .config file by default; two more are available to administrators to use in advanced scenarios.

Let's take a look at each one:

- **MemorySafetyMargin.** The percentage of `WorkingSetMaximum` that SQL Server Reporting Services will use before taking steps to reduce background task memory utilization and prioritize requests coming from the web service, attempting to protect user requests. User requests could still be denied.

- **MemoryThreshold.** The percentage of `WorkSetMaximum` at which SQL Server Reporting Services will deny new requests, slow down existing requests, and page memory to a hard drive until memory conditions improve.

Two more settings are instead given values automatically upon service startup, but you can override them in the .config file. Two older memory settings from SQL Server 2005 with which SQL DBAs might be familiar are `MemoryLimit` and `MaximumMemoryLimit`, but those two values have been ignored since SQL Server 2008.

- **WorkingSetMaximum.** By default, this is the total server physical memory. This setting does not appear by default in the .config file, but you can override it to reduce the amount of memory of which SQL Server Reporting Services will be aware. This value is expressed in kilobytes of memory.

- **WorkingSetMinimum.** By default, this value is 60 percent of the `WorkingSetMaximum`. If SQL Server Reporting Services needs memory below this value, it will use memory and not release it due to memory pressure. This setting does not appear by default in the .config file, but you can override it to increase the variability of SQL Server Reporting Services' memory utilization.

These four settings can appear in the rsreportserver.config file. As demonstrated here, you should override the default settings to 4 GB maximum and 2 GB minimum (each expressed in KB):

```
<MemorySafetyMargin>80</MemorySafetyMargin>
<MemoryThreshold>90</MemoryThreshold>
<WorkingSetMaximum>4194304</WorkingSetMaximum>
<WorkingSetMinimum>2097152</WorkingSetMinimum>
```

Limiting Machine Learning Server memory Similar to SQL Server Analysis Services and SQL Server Reporting Services, the Machine Learning Server has a .config file at %ProgramFiles%\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\Binn\rlauncher.config.

By default, Machine Learning Server is similar to 20 percent of total server memory. You can override this by adding a tag to the .config file to provide a value for `MEMORY_LIMIT_PERCENT`. This value is not in the .config file by default.

Remember to make a backup of this config file before editing. Following is an example of the contents of the rlauncher.config file, with the default memory limit changed to 25 percent:

```
RHOME=C:\PROGRA~2\MICROS~1\MSSQL1~4.SQL\R_SERV~2
MPI_HOME=C:\Program Files\Microsoft MPI
INSTANCE_NAME=SQL2K17
TRACE_LEVEL=1
JOB_CLEANUP_ON_EXIT=1
USER_POOL_SIZE=0
WORKING_DIRECTORY=C:\Program Files\Microsoft SQL
Server\MSSQL14.SQL2K17\MSSQL\ExtensibilityData
PKG_MGMT_MODE=0
MEMORY_LIMIT_PERCENT=25
```

Surface Area Configuration

If you are a veteran SQL Server DBA, you will remember when SQL Server Surface Area Configuration was a separate application. Surface Area Configuration was moved to the Facets menu starting with SQL Server 2008.

To view Surface Area Configuration in SQL Server Management Studio, in Object Explorer, connect to the SQL Server, right-click the server, and then, on the shortcut menu, click Facets. (Note that this window sometimes takes a moment to load.) In the dialog box that opens, change the value in the list box to Surface Area Configuration.

Keep in mind that all of these options should remain off unless needed and properly configured because they present a specific potential for misuse by an administrator or unauthorized user. In typical installations of SQL Server 2017, however, you will need three of these options:

- Database Mail (more about this in Chapter 14. You also can turn this setting on or off via the Database Mail XPs option in `sp_configure`.

  NOTE

  Keep in mind that enterprise Simple Mail Transfer Protocol (SMTP) servers have an "allow list" of IP addresses; you will need to add this server's IP to this list to send email.

- Remote Dedicated Admin Connection (more on this in Chapter 7). You also can turn this setting on or off via the remote admin connections option in `sp_configure`.

- CLR Integration, which you will need to turn on to use SQL Server Integration Services. You also can turn this setting on or off via the `clr enabled` option in `sp_configure`.

You should turn on other options in Surface Area Configuration only if they are specifically required by a third-party application and you are aware of the potential security concerns.

Setting up SQL Agent

There are a number of post-installation tasks to set up in SQL Agent before SQL Server can begin to help you automate, monitor, and back up your new instance.

➤ **Chapter 7 and Chapter 14 cover these topics in greater detail.**

You need to do the following:

1. Change the SQL Agent service from Manual to Automatic startup.
2. To send email notifications for alerts or job status notifications, you must set up a Database Mail account and profile (see Chapter 14).
3. Set up an Operator for a distribution group of IT professionals in your organization who would respond to a SQL Server issue.
4. Configure SQL Server Agent to use Database Mail.
5. Set up SQL Server Alerts for desired errors and high severity (Severity 21+) errors.

At the very least, these steps are put in place so that SQL Server can send out a call for help. Even if you have centralized monitoring solutions in place, the most severe of errors should be important enough to warrant an email.

You can choose to configure a large number of Windows Management Instrumentation (WMI) conditions, Perfmon counter conditions, SQL Server Error messages by number or severity in SQL Server Alerts. However, do not overcommit your inboxes, and do not set an inbox rule to Mark As Read and file away emails from SQL Server. By careful selection of emails, you can assure yourself and your team that emails from SQL Server will be actionable concerns that rarely arrive.

Turning on TCP/IP

The common network protocol TCP/IP is off by default, and the only protocol that is on is Shared Memory, which allows only local connections. You will likely not end up using Shared Memory to connect to the SQL Server for common business applications; rather, you'll use it only for local connections in the server.

When you connect to SQL Server using SQL Server Management Studio while signed in to the server, you connect to the Shared_Memory endpoint whenever you provide the name of the server, the server\instance, localhost, the dot character (“.”), or (local).

TCP/IP, however, is ubiquitous in many SQL Server features and functionality. Many applications will need to use TCP/IP to connect to the SQL Server remotely. Many SQL Server features require it to be turned on, including the Dedicated Admin Connection (DAC), the AlwaysOn availability groups listener, and Kerberos authentication.

In the SQL Server Configuration Manager application, in the left pane, click SQL Server Network Configuration. Browse to the protocols for your newly installed instance of SQL Server. The default instance of SQL Server, here and in many places, will appear as MSSQLSERVER.

After turning on the TCP/IP protocol, you need to do a manual restart of the SQL Server service.

Turning on Named Pipes is not required or used unless an application specifically needs it.

Verifying power options

The Windows Server Power Options setting should be set to High Performance for any server hosting a SQL Server instance.

Windows might not operate the processor at maximum frequency during normal or even busy periods of SQL Server activity. This applies to physical or virtual Windows servers.

Review the policy and ensure that the group policy will not change this setting back to Balanced or another setting.

Configuring antivirus exclusions

Configure any antivirus software installed on the SQL Server to ignore scanning extensions used by your SQL Server data and log files. Typically, these will be .mdf, .ldf, and .ndf.

Also, configure any antivirus programs to ignore folders containing full-text catalog files, backup files, replication snapshot files, SQL Server trace (.trc) files, SQL Audit files, Analysis Services database, log and backup files, FILESTREAM and FileTable folders, SQL Server Reporting Services temp files and log files.

Processes might also be affected, so set antivirus programs to ignore the programs for all instances of the SQL Server Database Engine service, Reporting Services service, Analysis Services service, and R Server (RTerm.exe and BxlServer.exe).

In SQL Server FCIs (and also for availability groups), also configure antivirus software to exclude the MSCS folder on the quorum drive, the MSDTC directory on the MSDTC share, and the Windows\Cluster folder on each cluster node, if they exist.

Optimizing for ad hoc workloads

The server-level setting to Optimize For Ad Hoc Workloads doesn't have the most intuitive name.

We are not optimizing ad hoc queries; we are optimizing SQL Server memory usage to prevent ad hoc queries from consuming unnecessary cache.

➤ **For more about the Optimize For Ad Hoc Workloads setting, see Chapter 2.**

For the unlikely scenario in which a large number of queries are called only two times, setting this option to True would be a net negative for performance.

However, like other design concepts in databases, we find that there are either one or many. There is no two.

➤ **To read more about cached execution plans, see Chapter 9.**

Lock pages in memory

You should consider using this setting for environments in which instances of SQL Server are expected to experience memory pressure due to other applications, server limitations, or over-allocated virtualized systems; however, this is an in-depth topic to be considered carefully.

➤ **For more about the Lock pages in memory setting, see Chapter 2.**

➤ **For more about the Windows page file, see Chapter 3.**

Inside OUT

How do I know if the permission to Lock pages in memory is in effect?

Starting with SQL Server 2016 SP1, you can check whether the Lock pages in memory permission has been granted to the SQL Server Database Engine service. Here's how to do that:

```
select sql_memory_model_desc
--Conventional = Lock pages in memory privilege is not granted
--LOCK_PAGES = Lock pages in memory privilege is granted
--LARGE_PAGES = Lock pages in memory privilege is granted in Enterprise mode
--with Trace Flag 834 ON
from sys.dm_os_sys_info;
```

Backups, index maintenance, and integrity checks

Backups are a critical part of your disaster recovery, and they should begin right away.

Begin taking database backups, at least of the master and msdb databases immediately. You should also back up other Setup-created databases, including ReportServer, ReportServer-TempDB, and SSISDB right away.

➤ **For more information on backups, index maintenance, and monitoring, see Chapter 12.**

As soon as your new SQL Server instance has databases in use, you should be performing selective index maintenance and integrity checks, regularly.

➤ **For more information on automating maintenance, see Chapter 13.**

Backing up service master and database master keys

You also should back up service master keys and any database master keys as they are created, storing their information securely.

➤ **For more information on service master and database master keys, see Chapter 6.**

To back up the instance service master key, use the following:

```
BACKUP SERVICE MASTER KEY TO FILE = 'localfilepath_or_UNC' ENCRYPTION BY PASSWORD =
'complexpassword'
```

And as soon as they come into existence as needed, in each user database, back up individual database master keys, as follows:

```
BACKUP MASTER KEY TO FILE = 'localfilepath_or_UNC' ENCRYPTION BY PASSWORD =
'complexpassword'
```

Increasing default error and agent history retention

By default, SQL Server maintains the current SQL Server error log plus six more error logs of history. Logs are cycled each time the SQL Server service is started, which should be rare, but you also can manually cycle them via the `sp_cycle_errorlog`.

However, one eventful, fun weekend of server troubleshooting or maintenance could wipe out a significant amount of your error history. This could make the task of troubleshooting periodic or business-cycle related errors difficult or impossible. You need visibility into errors that occur only during a monthly processing, monthly patch day, or periodic reporting.

In SQL Server Management Studio, in Object Explorer, connect to the SQL Server instance. Expand the Management folder, right-click SQL Server Logs, and then, on the shortcut menu, click Configure. Select the Limit The Number Of Error Logs Before They Are Recycled check box and type a value larger than 6. You might find that a value between 25 and 50 will result in more useful log history contained for multiple business cycles.

Similarly, you might find that the SQL Server Agent history is not sufficient to cover an adequate amount of job history, especially if you have frequent job runs.

In SQL Server Management Studio, in Object Explorer, connect to the SQL Server instance. Right-click SQL Server Agent, and then click Properties. Click the History page. This page is not intuitive and can be confusing. The first option, Limit Size Of The Job History Log, is a rolling job history retention setting. You might find it a good start to simply add a 0 to each value, increasing the maximum log history size in rows from the default of 1,000 to 10,000, and also increase the maximum job history per job in rows from the default of 100 to 1,000.

The second option, Remove Agent History, along with its companion Older Than text box is not a rolling job history retention setting; rather, it is an immediate and manual job history pruning. Select this second check box, and then click OK and return to this page; you will find the second check box is cleared. Behind the scenes, SQL Server Management Studio ran the msdb.dbo.sp_purge_jobhistory stored procedure to remove job history manually.

➤ **For more information about SQL Server Agent job history, see Chapter 13.**

Suppress successful backup messages

By default, SQL Server writes an event to the error log upon a successful database backup, whether it be FULL, DIFFERENTIAL, or TRANSACTION LOG.

On instances with many databases and with many databases in FULL recovery mode with regular transaction log backups, the amount of log activity generated by just their successful frequent log backups could flood the log with clutter, lowering log history retention.

NOTE

It is important to note that you can review successful backup history by querying the msdb system database, which has a series of tables dedicated to storing the backup history for all databases, including msdb.dbo.backupset and msdb.dbo.backupmediafamily. The built-in "Backup and Restore Events" report in SQL Server Management Studio provides access to this data, as well.

➤ **For more on backups, see Chapter 11.**

SQL Server Trace Flag 3226 is an option that you can turn on at the instance level to suppress successful backup notifications.

There are many trace flags available to administrators to alter default behavior—many more options than there are user interfaces to accommodate them in SQL Server Management Studio. Take care when turning them on and understand that many trace flags are intended only for temporary use when aiding troubleshooting.

Because Trace Flag 3226 should be a permanent setting, simply starting the trace by using DBCC TRACEON is not sufficient, given that the trace flag will no longer be active following a SQL Server service restart. Instead, add the trace flag as a startup parameter to the SQL Server Database Engine service by using SQL Server Configuration Manager.

Use the syntax `-T`*`flagnumber`*, as illustrated in Figure 4-7.

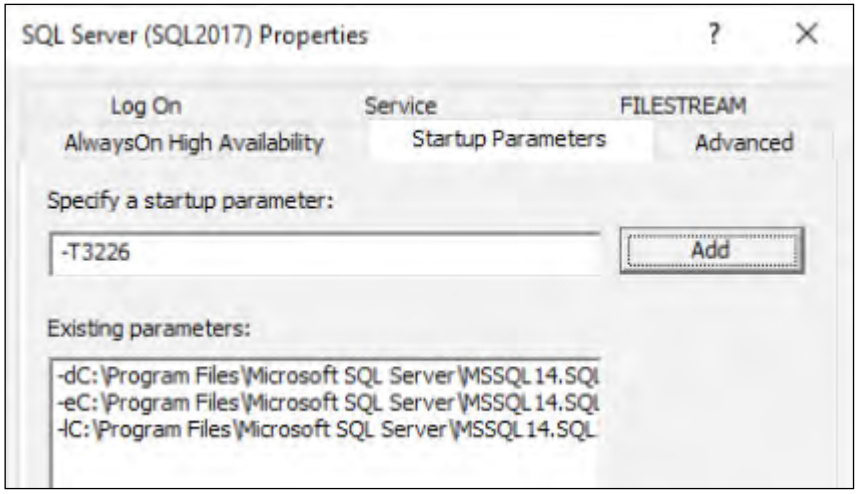

Figure 4-7 Specifying a startup parameter in the SQL Server Properties dialog box.

After you specify the trace flag, click Add. The change will not take effect until the SQL Server Database Engine service is restarted. Figure 4-8 shows that Trace Flag 3226 is now a part of the startup.

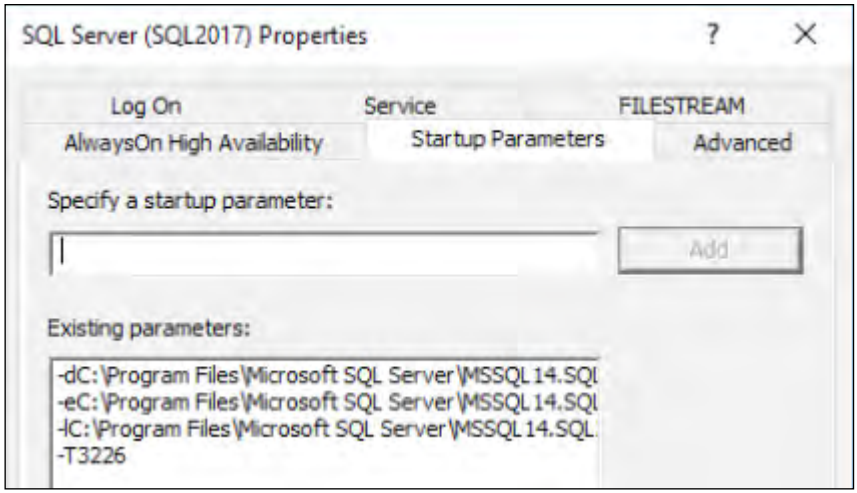

Figure 4-8 The Startup Parameters tab in the SQL Server Properties dialog box, with Trace Flag 3226 now appearing in the Existing Parameters box.

➤ **For more information on SQL Server Configuration Manager, see Chapter 1.**

Installing and configuring features

SQL Server installation is now complete, but three main features require post-installation configuration, including SQL Server Integration Services, SQL Server Reporting Services, and SQL Server Analysis Services. You will need to perform the steps detailed in this section before use if these features were installed.

SSISDB initial configuration and setup

Among the best features added by SQL Server 2012 were massive improvements to SQL Server Integration Services, specifically with a new server-integrated deployment, built-in performance data collector, environment variables, and more developer quality-of-life improvements.

When the Integration Services Catalog is created, a new user database called SSISDB is also created. You should back it up and treat it as an important production database.

You should create the SSISDB catalog database soon after installation and before a SQL Server Integration Services development can take place. You will need to do this only once. Because this will involve potential Surface Area Configuration changes and the creation of a new strong password, a SQL DBA, not a SQL Server Integration Services developer, should perform this and store the password securely alongside others generated at the time of installation.

In Object Explorer, connect to your instance, right-click Integration Services Catalog, and then, on the shortcut menu, click Create Catalog. In this single-page setup, you must select the Enable CLR Integration check box, decide whether SQL Server Integration Services packages should be allowed to be run at SQL Server Startup (we recommend this due to maintenance and cleanup performed then), and provide an encryption password for the SSISDB database.

The encryption password is for the SSISDB database master key. After you create it, you should then back up the SSISDB database master key.

➤ **For more information on database master keys, see Chapter 7.**

The SSISDB database will contain SQL Server Integration Services packages, their connection strings, and more data about the packages. The encryption would not allow these sensitive contents to be decrypted by a malicious user who gains access to the database files or backups. This password would be required if the database were moved to another server, so you should store it in a secure location within your enterprise.

NOTE

If you receive an error when creating the SSSIDB catalog that reads "The catalog backup file 'C:\Program Files\Microsoft SQL Server\140\DTS\Binn\SSISDBBackup.bak' could not be accessed" or similar, it is likely because SQL Server Integration Services was not actually installed, so the template database backup was not copied from the SQL Server media. You can run SQL Server Setup again or copy the SSISDBBackup.bak file from another SQL Server installation of the same version.

SQL Server Reporting Services initial configuration and setup

If you did not select the Install And Configure check box in SQL Server Setup, you will have more tasks to complete here, but there are still tasks to perform upon first installation of a SQL Server Reporting Services native-mode installation.

Open the Reporting Services Configuration Manager application, connect to the newly installed SQL Server Reporting Services instance, and then review the following options, from top to bottom:

- **Service Account.** You can change the SQL Server Reporting Services service account here. Remember that you should use only the Reporting Services Configuration Manager tool to make this change.
- **Web Service URL.** The web service URL is not for user interaction; rather, it is for the Report Manager and custom applications to programmatically connect to the SQL Server Reporting Services instance.

By default, a web service on TCP Port 80 is created called ReportServer. For named instances, the web service will be called ReportServer_*instancename*. The URL for the webservice would then be

http://*servername*/ReportServer

or:

http://*servername*/ReportServer_*instancename*

To accept defaults, at the bottom of the application window, click Apply.

You can optionally configure an SSL certificate to a specific URL here, and the Reporting Services Configuration Manager will make the changes necessary.

- **Database.** Each reporting service requires a pair of databases running on a SQL Server instance. If you selected the Install And Configure check box in SQL Server Setup, the databases will have been created for you with the names ReportServer and ReportServerTempDB, or, for a named instance, ReportServer$*InstanceName* and ReportServer$*InstanceName*TempDB. Both of these databases are important and you should create backups. The ReportServerTempDB is not a completely transient database like the SQL Server instance's TempDB system database.

  To set the databases for a new instance, click Change Database, and then follow the Create A New Report Server Database Wizard. This will add both databases to the instance.

- **Web Portal URL.** The web portal URL is the user-friendly website that hosts links to reports and provides for administrative features to the SQL Server Reporting Services instance. This is the link to share with users if you will be using the SQL Server Reporting Services portal.

  By default, the URL for the web portal is /Reports

  Http://*servername*/Reports for the default instance

  or:

  Http://*servername*/Reports_*InstanceName* for named instances

  You can change the name from the default here if desired. To proceed, at the bottom of the application window, click Apply.

- **Email Settings.** You use these email settings are used for sending reports to user subscribers via email. SQL Server Reporting Services uses its own Email Settings and does not inherit from the SQL Server instance's Database Mail settings. This setting is optional if you do not intend to send reports to subscribers via email.

  First introduced in SQL Server 2016 was the ability for SQL Server Reporting Services to authenticate to an external SMTP server using anonymous (No Authentication), Basic,

or NT LAN Manager (NTLM) authentication, which will use the SQL Server Reporting Services service account to authenticate to the SMTP server.

Prior to SQL Server 2016, authentication to external SMTP servers required the installation of a local SMTP server to provide a relay to the external, Azure-based (such as SendGrid) or cloud-based SMTP server. With SQL Server 2016 and 2017, SMTP connections can be made directly to these external SMTP servers.

NOTE

Keep in mind that enterprise SMTP servers have an allow list of IP addresses, and you will need to add this server's IP to this list to send email.

- **Execution Account.** You can provide this domain account optionally to be used when reports are configured to run on a schedule, to run without credentials (select the Credentials Are Not Required Option) or to connect to remote servers for external images.

  The execution account should not be the same as the SQL Server Reporting Services service account.

  This account should have minimal read-only access to any data sources or remote connections that will require it. You also can give it EXECUTE permissions for data sources that use stored procedures, but you should never give it any additional SQL Server permissions or let it be a member of any SQL Server server roles, including sysadmin.

- **Encryption Keys.** Immediately after installation and after the two SQL Server Reporting Services databases have been created, back up this instance's encryption keys. This key is used to encrypt sensitive information such as connection strings in the two databases. If the databases are restored to another server and this key is not available from the source server, credentials in connection strings will not be usable and you will need to provide them again for the reports to run successfully on a new server.

  If you can no longer locate the backup of a key, use the Change operation on this page to replace the key, and then back it up.

  To restore the original key to a new server to which the databases have been moved, use the Restore operation on this page.

- **Subscription Settings.** Use this page to specify a credential to reach file shares to which report subscriptions can be written. Reports can be dropped in this file share location in PDF, Microsoft Excel, or other formats for consumption.

  Multiple subscriptions can use this file share credential, which can be used on this page in a central location.

  This account should be different from the SQL Server Reporting Services execution account to serve its purpose appropriately.

- **Scale-Out Deployment.** Visit this page on multiple SQL Server Reporting Services instances to join them together. By using the same SQL Server Reporting Services databases for multiple SQL Server Reporting Services instances, multiple front ends can provide processing for heavy reporting workloads, including heavy subscription workloads. The server names can be used in a network load balancer such as a Network Load Balancing Cluster.

  Upon first installation, the Scale-Out Deployment page will show that the instance is "Joined" to a single server scale-out.

  Each scale-out instance of SQL Server Reporting Services must use the same settings on the Database page of the Reporting Services Configuration Manager. Connect to each instance in the scale-out and visit this page by opening it on each SQL Server Reporting Services instance to view the status, add servers to the scale-out, or remove servers.

- **PowerBI Integration.** Visit this page to associate the SQL Server Reporting Services instance to a Microsoft Power BI account, specifically to an account in Azure Active Directory. The administrator joining the Power BI instance to the SQL Server Reporting Services instance must be a member of the Azure Active Directory, a system administrator of the SQL Server Reporting Services instance, and a sysadmin on the SQL Server instance that hosts the SQL Server Reporting Services databases.

➤ **For the latest information on Power BI/SQL Server Reporting Services integration and the latest Azure authentication features, search for MT598750 or visit *https://docs.microsoft.com/sql/reporting-services/install-windows/power-bi-report-server-integration-configuration-manager*.**

SQL Server Analysis Services initial configuration and setup

No additional steps are required after setup to begin using a new SQL Server Analysis Services instance.

Because of the nature of SQL Server Analysis Services databases, their size, and how they are populated, typically they are not updated on a schedule, but you can do so by passing an XMLA command via a SQL Server Agent job step: type **SQL Server Analysis Services Command**. You also can initiate manual backups of SQL Server Analysis Services databases in Object Explorer in SQL Server Management Studio as well as restore SQL Server Analysis Services databases.

When installing SQL Server Analysis Services, a security group should have been chosen to grant permissions to SQL Server Analysis Services server administrators, granting a team full access to the server.

If you need to add a different group to the administrator role of the SQL Server Analysis Services instance, open SQL Server Management Studio, and then, in Object Explorer, connect to the Analysis Services instance. Right-click the server, and then, on the shortcut menu, click Properties. On the Security page, you can add additional windows-authenticated accounts or groups to the administrator role.

Adding databases to a SQL Server instance

Now that your SQL Server is installed and SQL Server features are configured, it is time to put your SQL Server instance to use. In SQL Server Management Studio, you'll see four system databases there already, plus additional databases for SQL Server Reporting Services and SQL Server Integration Services perhaps if you have installed these features. Now it is time to create user databases.

We have discussed a number of database configurations in Chapter 3, including the physical configuration of files and storage.

For information on Azure SQL Databases, refer to Chapter 5. The remainder of this chapter refers to SQL Server instance databases.

Considerations for migrating existing databases

As an administrator, you'll be faced with the need to move a database from one instance to another, perhaps for the purposes of refreshing a preproduction environment, moving to a new SQL Server instance, or promoting a database into production for the first time.

When copying a database into a new environment, keep in mind the following:

- Edition
- Version and compatibility mode
- SQL logins
- Database-scoped configurations
- Database settings
- Encryption

Let's look at each of these in more detail.

Edition

Generally speaking, databases progress upward in terms of cost and feature set, beginning with Express edition, Web, Standard, and finally Enterprise edition. (Developer edition is the same as Enterprise edition, except for the ability to use it in a production environment.) Moving a database up from Express, Web, or Standard edition expands the features available for use in the database.

The concern for DBAs is when databases need to move down from Enterprise, Standard, or Web edition. A large number of features that had historically been exclusive to Enterprise edition were included in Standard edition for the first time with SQL Server 2016 SP1, expanding what we can do with Standard edition as developers and administrators.

You will encounter errors related to higher-edition features when restoring or attaching to an instance that does not support those editions. For example, when attempting to restore a data-compressed database to tables to an instance that does not support data compression, you will receive an error message similar to "cannot be started in this edition of SQL Server because part or all of object '*somecompressedindex*' is enabled with data compression." In this case, you will need to manually remove data compression from the database in the source instance and then create a new backup or detach the database again before migrating to the lower-edition instance. You cannot turn off the use of higher-edition features on the lower-edition instance.

You can foresee this problem by using a dynamic management view that lists all edition-specific features in use. Keep in mind that some features are supported in all editions but are limited. For example, memory-optimized databases are supported even in Express edition but with only a small amount of memory that can be allocated.

For example,

```
USE [WIDEWORLDIMPORTERS];
SELECT FEATURE_NAME
FROM SYS.DM_DB_PERSISTED_SKU_FEATURES;
```

returns the following rows:

```
feature_name
Compression
Partitioning
ColumnStoreIndex
InMemoryOLTP
```

Version and compatibility mode

SQL Server databases upgraded from an older version to a new version will retain a prior compatibility mode. Compatibility mode is a database-level setting.

For example, restoring or attaching a database from SQL 2012 to SQL 2017 will result in the database assuming the SQL 2012 (11.0) compatibility mode inside the SQL 2017 environment. This is not necessarily a problem, but it does have consequences with respect to how you can use features and improvements and potentially how it performs.

You can view the compatibility level of a database in SQL Server Management Studio. To do so, in Object Explorer, right-click a database, and then, on the shortcut menu, click Properties. In the pane on the left, click Options. On the Options page, the Compatibility Level list box displays the current setting. You can change that setting or use the ALTER DATABASE command to change the COMPATIBILITY_LEVEL setting. You can also view this setting for all databases in the system catalog via sys.databases; look for the compatibility_level column.

SQL Server provides database compatibility modes for backward compatibility to database-level features, including improvements to the query optimizer, additional fields in dynamic management objects, syntax improvements, and other database-level objects.

For the scenario in which a SQL Server 2017 (internal version 140) instance is hosting a database in SQL Server 2012 (internal version 110) compatibility mode, it is important to note that applications are still connecting to a SQL Server 2017 instance. Only database-level features and options are honored in the prior compatibility modes.

For example, some recent syntax additions such as the new `STRING_SPLIT()` or `OPENJSON` functions, added in SQL Server 2016, will not work when run in the context of a database in a prior compatibility mode. Some syntax improvements, such as `DATEFROMPARTS()` and `AT TIME ZONE`, will work in any database in any compatibility mode in SQL Server 2017.

SQL Server 2017 supports compatibility levels down to SQL Server 2008 (internal version 100).

Changing the database compatibility level does not require a service restart to take place, but we strongly recommend that you do *not* perform this during normal operating hours. Promoting the database compatibility mode should be thoroughly tested in preproduction environments. Even though syntax errors are extremely unlikely, other changes to the query optimizer engine from version to version could result in performance changes to the application that must be evaluated prior to rollout to a production system. When you do upgrade production from a prior compatibility level, you should do so during a maintenance period, not during user activity.

➤ **For more information on the differences between compatibility modes since SQL 2005, reference MSDN article bb510680 or visit *https://docs.microsoft.com/sql/t-sql/statements/alter-database-transact-sql-compatibility-level*.**

Inside OUT

When should I keep a database in a prior compatibility mode?

It is a common oversight to forget to promote the database compatibility level to the new SQL Server version level after a database upgrade. You are missing out on new database features, but there can be good reasons to keep a database in a prior compatibility mode, though you should consider all of them temporary.

The most common reason to run a database in prior compatibility mode is not technical at all; rather, the administrator might be handcuffed by vendor support or software certification. Many changes from version to version in SQL Server are additive and rarely regressive.

One notable exception is one of the new features introduced in SQL Server 2014: improvements to the Cardinality Estimator resulted in poor query performance in rare situations. In the case of complex, fine-tuned and/or large chunks of poor-performing code, reverting to the previous Cardinality Estimator is the most realistic near-term solution. Changing the database's compatibility mode down to SQL 2012 might have resolved the issue, but two less-drastic options are available.

Trace flag 9481 will force a database in SQL 2014 compatibility mode to use the legacy Cardinality Estimation model from SQL 2012 and earlier. The LEGACY_CARDINALITY_ESTIMATION database option is also available starting with SQL Server 2016, to force the old Cardinality Estimation model into use for that database only. It has the same effect in the database at Trace Flag 9481.

NOTE

You can upgrade the SSISDB database, which contains the SQL Server Integration Services Catalog, independently of other databases by using the SSISDB Database Upgrade Wizard. This makes it easier to move your SQL Server Integration Services packages and environments from server to server by restoring or attaching a database from a previous version to a SQL Server 2017 Server.

SQL logins

SQL-authenticated logins and their associated database users are connected via security identifier (SID), not by name. When moving a database from one instance to another, the SIDs in the SQL logins on the old instance might be different from the SIDs in the SQL logins on the new instance, even if their names match. After migration to the new instance, SQL-authenticated

logins will be unable to access databases where their database users have become "orphaned," and you must repair this. This does not affect Windows Authenticated logins for domain accounts.

This condition must be repaired before applications and end users will be able to access the database in its new location. Refer to the section "Solving orphaned SIDs" in Chapter 6.

The database owner should be included in the security objects that should be accounted for on the new server. Ensure that the owner of the database, listed either in the Database Properties dialog box or the `sys.databases owner_sid` field, is still a valid principal on the new instance.

For databases with Partial Containment, contained logins for each type will be restored or attached along with the database, and this should not be a concern.

Database-scoped configurations

Database-scoped configurations were introduced in SQL Server 2016 (and also in Azure SQL Database v12) and represent a container for a set of options available to be configured at the database level. in earlier versions, these settings were available only at the server or individual query, such as Max Degree of Parallelism (MaxDOP).

➤ **For more information on Parallelism and MaxDOP, go to Chapter 9.**

You should evaluate these options for each database after it is copied to a new instance to determine whether the settings are appropriate. The desired MaxDOP, for example, could change if the number of logical processors differs from the system default.

You can view each of these database-scoped configurations in SQL Server Management Studio. In Object Explorer, right-click a database, and then, on the shortcut menu, click Properties. In the pane on the left, click Options. A heading just for database-scoped configurations appears at the top of the Other Options list. You can also view database-scoped configurations in the dynamic management view sys.database_scoped_configurations.

Database configuration settings

You should review database-specific settings at the time of migration, as well. You can review them with a quick glance of the sys.databases catalog view, or from the database properties window in SQL Server Management Studio.

The following is not a list of all database settings, but we will cover these and many more later in the chapter. You should pay attention to these when restoring, deploying, or attaching a database to a new instance.

- **Read Only.** If the database was put in READ_ONLY mode before the migration to prevent data movement, be sure to change this setting back to READ_WRITE.

- **Recovery Model.** Different servers might have different backup and recovery methods. In a typical environment, the FULL recovery model is appropriate for production environments when the data loss tolerance of the database is smaller than the frequency of full backups, or when point-in-time recovery is appropriate. If you are copying a database from a production environment to a development environment, it is likely you will want to change the recovery model from FULL to SIMPLE. If you are copying a database from a testing environment to a production environment for the first time, it is likely you will want to change the recovery model from SIMPLE to FULL.

➤ **For more information about database backups and the appropriate recovery model, see Chapter 11.**

- **Page Verify Option.** For all databases, this setting should be CHECKSUM. The legacy TORN_PAGE option is a sign that this database has been moved over the years up from a pre-SQL 2005 version, but this setting has never changed. Since SQL 2005, CHECKSUM has the superior and default setting, but it requires an administrator to manually change.

- **Trustworthy.** This setting is not moved along with the database. If it was turned on for the previous system and was a requirement because of external assemblies, cross-database queries, and/or Service Broker, you will need to turn it on again. It is not recommended to ever turn on this setting unless it is made necessary because of an inflexible architecture requirement. It could allow for malicious activity on one database to affect other databases, even if specific permissions have not been granted. It is crucial to limit this setting and understand cross-database permission chains in a multitenant or web-hosted shared SQL Server environment.

➤ **For more on object ownership, see Chapter 6.**

Transparent data encryption

Transparent data encryption (TDE) settings will follow the database as it is moved from one instance to another, but the certificate and the certificate's security method will not. For example, the server certificate created to encrypt the database key and the private key and its password are not backed up along with the database. These objects must be moved to the new instance along with the database *prior to* any attempt to restore or attach the database.

CAUTION

Restoring an unencrypted database over an encrypted database is allowed. When would you inadvertently do this? If you restore a backup from the database before it was encrypted, you will end up with an unencrypted database. You must then reapply transparent data encryption.

➤ **For more information on transparent data encryption, see Chapter 7.**

Moving existing databases

There are a number of strategies for moving or copying a SQL Server database from one instance to another. You should consider each as it relates to necessary changes to application connection strings, DNS, storage, and security environments. We'll review a number of options for migration in this section.

Restoring a database backup

Restoring a backup is an easily understandable way to copy data from one instance to another. You can also carry out this method in such a way as to minimize the outage impact.

Let's compare two different simplified migration processes. Following is a sample migration checklist using a FULL backup/restore:

➤ **For more information on the types of database backups and database restores, see Chapter 11.**

1. Begin application outage.
2. Perform a FULL backup of the database.
3. Copy the database backup file.
4. Restore the FULL backup.
5. Resolve any SQL-authenticated login security issues or any other changes necessary before use.
6. In applications and/or DNS and/or aliases, change connection strings.
7. End application outage.

In the preceding scenario, the application outage must last the entire span of the backup, copy, and restore, which for large databases could be quite lengthy, even with native SQL Server backup compression reducing the file size.

Instead, consider the following strategy:

1. Perform a FULL compressed backup of the database.
2. Copy the database backup file.
3. Restore the FULL backup WITH NORECOVERY.
4. Begin application outage.
5. Take a differential backup and then a log backup of the database.
6. Copy the differential backup file and the log backup file to the new server.
7. Restore the differential backup file WITH NORECOVERY.
8. Restore the transaction log backup WITH RECOVERY.
9. Resolve any SQL-authenticated login security issues or any other changes necessary before use.
10. In applications and/or DNS and/or aliases, change the connection strings.
11. End application outage.

In this scenario, the application outage spans only the duration of the differential and transaction log's backup/copy/restore operation, which for large databases should be a tiny fraction of the overall size of the database. This scenario does require more preparation and scripting in advance, and it requires coordination with the usual backup system responsible for transaction log backups. By taking a manual transaction log backup, you can create a split transaction log backup chain for another system, for which you should take account.

Attaching detached database files

Detaching, copying, and attaching database files will also get the job of getting the database in place on a new instance. It is relatively straightforward to disassociate (detach) the files from the old SQL Server, copy the files to the new instance, and then attach the files to the new SQL Server. This is largely limited by the data transfer speed of copying the files. You might also consider moving the SAN drives to the new server to decrease the time spent waiting for files to copy.

Attaching copied database files can be faster than restoring a full database backup; however, it lacks the ability to minimize the outage by taking advantage of transaction log backups (see earlier).

Copying the full set of database files (remember that the database might contain many more files than just the .mdf and .ldf files, including secondary data files and FILESTREAM containers) is not faster than restoring a transaction log backup during the application outage, and it is not a true recovery method. Because database backup files can also be compressed natively by SQL Server, the data transfer duration between the Windows servers will be reduced by using the backup/restore strategy.

Moving data with BACPAC files

A BACPAC file is an open JSON-format file that contains the database schema and row data, allowing for the migration of databases, ideally, at the start of a development/migration phase and not for large databases. SQL Server Management Studio can both generate and import BACPAC files, and the Azure portal can import them when moving an on-premises SQL Server to an Azure SQL database.

Creating a database

In this section, we review the basics of database settings and configuration. As a DBA, you might not create databases from scratch regularly, but you should be familiar with all the settings and design decisions that go into their creation, including, adding database files and the tools involved.

Managing default settings

It is important to understand the role of the model database when creating new databases, regardless of the method of creation. The model database and its entire contents and configuration options are copied to any new database, even TempDB upon service restart. For this reason, never store any data (even for testing) in the model database. Similarly, do not grow the model database from its default size, because this will require all future databases to be that size or larger.

However, the location of the model database's files is not used as a default for new databases. Instead, the default location for database files is stored at the server level. You can view these default storage locations, which should be changed and must be valid, in the Server Properties dialog box in SQL Server Management Studio, on the Database Settings page. There you will find the default locations for Data, Log, and Backup files. These locations are stored in the registry.

On this page, you'll also see the default recovery interval setting, which is by default 0, meaning that SQL Server can manage the frequency of internal automatic CHECKPOINTs. This typically results in an internal checkpoint frequency of one minute. This is an advanced setting that can be (though rarely is) changed at each database level, though it should not be changed at the server level or database level except by experienced administrators.

Also on the Database Settings page of Server Properties you will find the default index fill factor and default backup compression setting. These are server-level defaults applied to each database, but you cannot configure them separately for each database. You can configure them independently at each index level and with each backup statement, respectively.

Inside OUT

Watch out for the default data and log file locations: they can cause future cumulative updates to fail!

Portions of cumulative updates reference the default file locations. You might see errors such as "operating system error 3 (The system cannot find the path specified.)" in the detailed log of the cumulative update.

The patches will fail if these default database locations change to an invalid path, if the complete subfolder path does not exist, or if SQL Server loses permissions to access the locations. You will need to restart the cumulative update after correcting the problem with the default locations.

Among the settings inherited by new databases from the model database unless overridden at the time of creation are the following:

- Initial data and log file size
- Data and log file Autogrowth setting
- Data and log file Maximum size
- Recovery model
- Target Recovery Time (which would override the system default recovery interval)
- All Database-Scoped Configurations including the database-level settings for Legacy Cardinality Estimation, Max DOP, Parameter Sniffing, and Query Optimizer Fixes.
- All the Automatic settings, including auto close, auto shrink, auto create/update statistics. (We discuss each of these later in the chapter.)

Inside OUT

Your SQL Server Management Studio connections to the model database will block CREATE DATABASE statements.

Close or disconnect SQL Server Management Studio query windows that use the model database context. If you are configuring the model database by using T-SQL commands, you can leave SQL Server Management Studio query windows open. Create database statements need to reference the model database. User connections, including query windows in SQL Server Management Studio with the model database context, can block the creation of user databases.

You might see the error "Could not obtain exclusive lock on database 'model'. Retry the operation later. CREATE DATABASE failed. Some file names listed could not be created. Check related errors. (Microsoft SQL Server, Error: 1807)".

For applications like SharePoint that create databases, this could lead to application errors.

Owning the databases you create

The login that runs the CREATE DATABASE statement will become the owner of any database you create, even if the account you are using is not a member of the sysadmin group. Any principal that can create a database becomes the owner of that database, even if, for example, they have only membership to the dbcreator built-in server role.

Ideally, databases are not owned by named individual accounts. You might decide to change each database to a service account specific to that database's dependent applications. You must do this after the database is created. You cannot change the database owner via SQL Server Management Studio; instead, you must use the ALTER AUTHORIZATION T-SQL statement.

➤ **For more information on the best practices with respect to database ownership and how to change the database owner, see Chapter 6.**

Creating additional database files

Every SQL Server database needs at least one data file and one log file. You can use additional data files to maximize storage performance. (We discuss physical database architecture in detail in Chapter 3.)

However, adding additional log files long term is not a wise decision. There is no performance advantage to be gained with more than one transaction log file for a database. SQL Server will not write to them randomly, but sequentially.

The only scenario in which a second transaction log file would be needed is if the first had filled up its volume. If no space can be created on the volume to allow for additional transaction log file data to be written, the database cannot accept new transactions and will refuse new application requests. In this scenario, one possible troubleshooting method is to temporarily add a second transaction log file on another volume to create the space to allow the database transactions to resume accepting transactions. The end resolution involves clearing the primary transaction log file, performing a one-time-only shrink to return it to its original size, and removing the second transaction log file.

Using SQL Server Management Studio to create a new database

You can create and configure database files, specifically their initial sizes, in SQL Server Management Studio. In Object Explorer, right-click Databases, and then, on the shortcut menu, click New Database to open the New Database dialog box.

After you have configured the new database's settings but before you click OK, you can script the T-SQL for the CREATE DATABASE statement.

Here are a few suggestions when creating a new database:

- Pregrow your database and log file sizes to an expected size. This avoids autogrowth events as you initially populate your database. You can speed up this process greatly by using the Perform Volume Maintenance Task permission for the SQL Server service account so that instant file initialization is possible.

➤ **We covered instant file initialization earlier in this chapter.**

- Consider the SIMPLE recovery model for your database until it enters production use. Then, the FULL or BULK_LOGGED recovery models might be more appropriate.

➤ **For more information database backups and the appropriate recovery model, see Chapter 11.**

- Review the logical and physical files names of your database and the locations. The default locations for the data and log files are a server-level setting but you can override them here. You also can move the files later on (we cover this later in this chapter).

- As soon as the database is created, follow-up with your backup strategy to ensure that it is covered as appropriate with its role.

Deploying a database via SQL Server Data Tools

You can also deploy developed databases to a SQL Server instance using a Database Project in SQL Server Data Tools. For databases for which objects will be developed by your team or another team within your enterprise, SQL Server Data Tools provides a professional and mature environment for teams to develop databases, check them into source control, generate change scripts for incremental deployments, and reduce object scripting errors.

SQL Server Data Tools can generate incremental change scripts or deploy databases directly. It also has the option to drop or re-create databases for each deployment, though this is turned off by default.

You might find it easiest to create the new database by using SQL Server Management Studio and then deploy incremental changes to it with SQL Server Data Tools.

Database properties and options

In this section, we review some commonly changed and managed database settings. There are quite a few settings on the Options page in Database Properties, many involving rarely changed defaults or ANSI-standard deviations for legacy support.

You can view each of these settings in SQL Server Management Studio via Object Explorer. To do so, right-click a database, and then, on the shortcut menu, click Properties. In the Database Properties dialog box, in the pane on the left, click Options. You also can review database settings for all databases in the sys.databases catalog view.

The subsections that follow discuss the settings that you need to consider when creating and managing SQL Server databases.

Collation

Collations exist at three levels in a SQL Server instance: the database, the instance, and TempDB. The collation of the TempDB database by default matches the collation for the instance and should differ only in otherwise unavoidable circumstances. Ideally, the collations in all user databases match the collation at the instance level and for the TempDB, but there are scenarios in which and individual database might need to operate in a different collation.

Oftentimes databases differ from the server-level collation to enforce case sensitivity, but you can also enforce language usage differences (such as kana or accent sensitivity) and sort order differences at the database level.

The default collation for the server is decided at installation and is preselected for you based on the regionalization settings of the Windows Server. You can override this during installation. Some applications, such as Microsoft Dynamics GP, require a case-sensitive collation.

Whereas the server-level collation is virtually unchangeable, databases can change collation. You should change a database's collation only before code is developed for the database or only after extensive testing of existing code.

Be aware that unmatched collations in databases could cause issues when querying across those databases, so you should try to avoid collation differences between databases that will be shared by common applications.

For example, if you write a query that includes a table in a database that's set to the collation SQL_Latin1_General_CP1_CI_AS (which is **c**ase **i**nsensitive and **a**ccent **s**ensitive) and a join to a table in a database that's also set to SQL_Latin1_General_CP1_CS_AS, you will receive the following error:

```
Cannot resolve the collation conflict between "SQL_Latin1_General_CP1_CI_AS" and
"SQL_Latin1_General_CP1_CS_AS" in the equal to operation.
```


Short of changing either database to match the other, you will need to modify your code to use the COLLATE statement when referencing columns in each query, as demonstrated in the following example:

```
… FROM
CS_AS.sales.sales s1
INNER JOIN CI_AS.sales.sales s2
ON s1.[salestext] COLLATE SQL_Latin1_General_CP1_CI_AS = s2.[salestext]
```

In contained databases, collation is defined at two different levels: the database and the catalog. You cannot change the catalog collation cannot from Latin1_General_100_CI_AS_WS_KS_SC. Database metadata and variables are always in the catalog's collation. The COLLATE DATABASE_DEFAULT syntax can also be a very useful tool if you know the collation before execution.

Recovery model

The FULL recovery model is appropriate for production environments when the data loss tolerance of the database is smaller than the frequency of full backups or when point-in-time recovery is appropriate. If you are copying a database from a production environment to a development environment, it is likely you will want to change the recovery model from FULL to SIMPLE. If you are copying a database from a testing environment to a production environment for the first time, it is likely that you will want to change the recovery model from SIMPLE to FULL.

➤ **For more information on database backups and the appropriate recovery model, see Chapter 11.**

Compatibility level

SQL Server provides database compatibility modes for backward compatibility to database-level features, including improvements to the query optimizer, additional fields in dynamic management objects, syntax improvements, and other database-level objects.

Compatibility mode is a database-level setting, and databases upgraded from an older version to a new version will retain a prior compatibility mode. For example, some new syntax additions in SQL Server 2016 such as the new `STRING_SPLIT()` or `OPENJSON` functions will not work when run in the context of a database in a prior compatibility mode. Other syntax improvements, such as `DATEFROMPARTS()` and `AT TIME ZONE`, will work in any database in any compatibility mode in SQL Server 2017.

SQL Server 2017 supports compatibility levels down to SQL Server 2008 (internal version 100), the same as SQL Server 2016.

Database compatibility does not require a service restart to take place, but we strongly recommend that you do *not* perform this during normal operating hours. Promoting the database compatibility mode should be thoroughly tested in preproduction environments. Even though syntax errors are extremely unlikely, other changes to the query optimizer engine from version to version could result in performance changes to the application that must be evaluated prior to rollout to a production system. When you do upgrade production from a prior compatibility level, you should do so during a maintenance period, not during user activity.

You should review database-specific settings at the time of migration, as well. You can review them from a quick scroll of the sys.databases catalog view or from the database properties window in SQL Server Management Studio.

The following is not a list of all database settings, but you should pay attention to these when restoring, deploying, or attaching a database to a new instance.

Containment type

Partially contained databases represent a fundamental change in the relationship between server and database. They are an architectural decision that you make when applications are intended to be portable between multiple SQL Server instances or when security should be entirely limited to the database context, not in the traditional server login/database user sense.

➤ **For more information about the security implications of contained databases, see Chapter 6.**

Azure SQL databases are themselves a type of contained database, able to move from host to host in the Azure platform as a service (PaaS) environment, transparent to administrators and users. You can design databases that can be moved between SQL Server instances in a similar fashion, should the application architecture call for such capability.

Changing the Containment Type from None to Partial converts the database to a partially contained database, and should not be taken lightly. We do not advise changing a database that has already been developed without the partial containment setting, because there are differences with how temporary objects behave and how collations are enforced. Some database

features, including Change Data Capture, Change Tracking, replication, and some parts of Service Broker are not supported in partially contained databases. You should carefully review, while logged in as a member of the sysadmin server role or the db_owner database role, the system dynamic management view sys.dm_db_uncontained_entities for an inventory of objects that are not contained.

Autoclose

You should turn on this setting only in very specific and resource-exhausted environments. It activates the periodic closure of connections and the clearing of buffer allocations, when user requests are done. When active, it unravels the very purpose of application connection pooling; for example, rendering certain application architectures useless and increasing the number of login events. You should never turn on this settings as part of performance tuning or trouble-shooting exercise of a busy environment.

Auto Create statistics

When you turn on this setting, the query optimizer automatically create statistics needed for runtime plans, even for read-only databases (statistics are stored in the tempdb for read-only databases). Some applications, such as SharePoint, handle the creation of statistics automatically: due to the dynamic nature of its tables and queries, SharePoint handles statistics creation and updates by itself. Unless an application like SharePoint insists otherwise, you should turn on this setting. You can identify autocreated statistics in the database as they will use a naming convention similar to _WA_Sys_<*column_number*>_<*hexadecimal*>.

Inside OUT

What are statistics?

SQL Server uses statistics to describe the distribution and nature of the data in tables. The query optimizer needs the Auto Create setting turned on so that it can create single-column statistics when compiling queries. These statistics help the query optimizer create optimal runtime plans. Without relevant and up-to-date statistics, the query optimizer may not choose the best way to execute queries. Unless an application has been specifically designed to replace the functionality of Auto Create and Auto Update statistics, such as SharePoint, these two settings should be turned on.

Autocreate incremental statistics

Introduced in SQL 2014, this setting allows for the creation of statistics that take advantage of table partitioning, reducing the overhead of statistics creation. This setting has no impact on nonpartitioned tables. Because it can reduce the cost of creating and updating statistics, you should turn it on.

This will have an effect only on new statistics created after this setting is turned on. When you turn it on, you should update the statistics on tables with partitions, including the INCREMENTAL = ON parameter, as shown here:

```
UPDATE STATISTICS [dbo].[HoriztonalPartitionTable] [PK_HorizontalPartitionTable] WITH
RESAMPLE, INCREMENTAL = ON;
```

You also should update any manual scripts you have implemented to update statistics to use the ON PARTITIONS parameter when applicable. In the catalog view sys.stats, the is_incremental column will equal 1 if the statistics were created incrementally, as demonstrated here:

```
UPDATE STATISTICS [dbo].[HoriztonalPartitionTable] [PK_HorizontalPartitionTable] WITH
RESAMPLE ON PARTITIONS (1);
```

Autoshrink

You should never turn on this setting. It will automatically return any free space of more than 25 percent of the data file or transaction log. You should shrink a database only as a one-time operation to reduce file size after unplanned or unusual file growth. This setting could result in unnecessary fragmentation, overhead, and frequent rapid log autogrowth events.

Auto Update Statistics

When turned on, statistics will be updated periodically. Statistics are considered out of date by the query optimizer when a ratio of data modifications to rows in the table has been reached. The query optimizer checks for and updates the out-of-date statistic before running a query plan and therefore has some overhead, though the performance benefit of updated statistics usually outweighs this cost. This is especially true when the updated statistics resulted in a better optimization plan. Because the query optimizer updates the statistics first and then runs the plan, the update is described as synchronous.

Auto Update Statistics Asynchronously

This changes the behavior of the Auto Update Statistics by one important detail. Query runs will continue even if the query optimizer has identified an out-of-date statistics object. The statistics will be updated afterward.

NOTE

It is important to note that you must turn on Auto Update Statistics for Auto Update Statistics Asynchronously to have any effect. There is no warning or enforcement in SQL Server Management Studio for this, and though a Connect Item with this concern was raised in 2011, it was marked Closed as "Won't Fix."

Inside OUT

Should I turn on Auto Update Statistics and Auto Update Statistics Asynchronously in SQL Server 2017?

Yes! (Unless the application specifically recommends not to, such as SharePoint.)

Starting in SQL Server 2016 (and with database compatibility mode 130), the ratio of data modifications to rows in the table that helps identify out-of-date statistics has been aggressively lowered, causing statistics to be automatically updated more frequently. This is especially evident in large tables in which many rows were regularly updated. In SQL Server 2014 and earlier, this more aggressive behavior was not on by default, but could be turned on via Trace Flag 2371 starting with SQL 2008 R2 SP1.

It is more important starting with SQL Server 2016 than in previous versions to turn on Auto Update Statistics Asynchronously, which can dramatically reduce the overhead involved in automatic statistics maintenance.

Allow Snapshot Isolation

This setting allows for the use of Snapshot Isolation mode at the query level. When you turn this on, the row versioning process begins in TempDB, though this setting does little more than allow for this mechanism to be used in this database. To begin to use Snapshot Isolation mode in the database, you would need to change code; for example, to include `SET TRANSACTION ISOLATION LEVEL SNAPSHOT`.

➤ **For much more on Snapshot Isolation and other isolation levels, see Chapter 9.**

Is Read Committed Snapshot On

Turning on this setting changes the default isolation mode of the database from `READ COMMITTED` to `READ COMMITTED SNAPSHOT`. You should not turn this on during regular business hours; instead, do it during a maintenance window. Ideally, however, this setting is on and accounted for during development.

There will be an impact to the utilization of the TempDB as well as a rise in the `IO_COMPLETION` and `WAIT_XTP_RECOVERY` wait types, so you need to perform proper load testing. This setting, however, is potentially a major performance improvement and the core of enterprise-quality concurrency.

Page Verify Option

For all databases, this setting should be CHECKSUM. The legacy TORN_PAGE option is a sign that this database has been moved over the years up from a pre-SQL 2005 version, but this setting has never changed. Since SQL 2005, CHECKSUM has the superior and default setting, but it requires an administrator to manually change.

Trustworthy

It is not recommended to ever turn on this setting unless it is made necessary because of an inflexible architecture requirement. Doing so could allow for malicious activity on one database to affect other databases, even if specific permissions have not been granted. Before turning on this setting, you should understand the implications of cross-database ownership chains in a multitenant or web-hosted shared SQL Server environment.

➤ **For more on object ownership, see Chapter 6.**

Database Read-Only

You can set an older database, or a database intended for nonchanging archival, to `READ_ONLY` mode to prevent changes. Any member of the server sysadmin role or the database db_owner role can revert this to `READ_WRITE`, so you should not consider this setting a security measure.

Database-Scoped Configurations

First introduced in SQL Server 2016 (and also in Azure SQL Database v12), Database-Scoped Configurations are a set of options previously available only at the server or individual query, such as Max Degree of Parallelism (MaxDOP). You can now change settings easily via database options that previously were available only via trace flags at the server level.

You can view each of these Database-Scoped Configurations in SQL Server Management Studio. In Object Explorer, right-click a database, and then, on the shortcut menu, click Properties. In the Database Properties dialog box, in the pane on the left, click Options. On the Options page, a heading just for Database-Scoped Configurations appears at the top of the Other Options list.

The current database context is important for determining which database's properties will be applied to a query that references objects in multiple databases. This means that the same query, run in two different database contents, will have different execution plans, potentially because of differences in each database's Max DOP setting, for example.

Query Store

Introduced in SQL Server 2016, the Query Store is a built-in reporting mechanism and data warehouse for measuring and tracking cached runtime plans. Though useful, it is not on by default, and you should turn it on as soon as possible if you intend to use it to aid performance tuning and troubleshooting cached runtime plans.

➤ **For more information on the Query Store, see Chapter 9.**

Indirect checkpoints

If your database was created in SQL Server 2016 or 2017, your database is already configured to use indirect checkpoint, which became the default for all databases in SQL Server 2016. However, databases created on prior versions of SQL Server will continue to use the classic automatic checkpoint, which has been in place since SQL Server 7.0 and tweaked only since.

This is an advanced topic, and one that we won't dive into too deeply, save for one configuration option that you should change on databases that have been upgraded from versions prior to SQL Server 2016.

What is a checkpoint? This is the process by which SQL Server writes to the drive both data and transaction log pages modified in memory, also known as "dirty" pages. Checkpoints can be issued manually by using the `CHECKPOINT` command but are issued in the background for you, so issuing `CHECKPOINT` is rarely necessary and is usually limited to troubleshooting.

What is automatic checkpoint? Prior to SQL Server 2016 and since SQL Server 7.0, by default all databases used automatic checkpoint. The rate with which dirty pages were committed to memory has increased with versions, as disk I/O and memory capacities of servers have increased. The goal of automatic checkpoint was to ensure that all dirty pages were managed within a goal defined in the server configuration option Recovery Interval. By default, this was 0, which meant it was automatically configured. This tended to be around 60 seconds, but was more or less unconcerned with the number of pages dirtied by transactions between checkpoints.

What is indirect checkpoint? This is a new strategy of taking care of "dirty pages" that is far more scalable and can deliver a performance difference especially on modern systems with a large amount of memory. Indirect checkpoints manage dirty pages in memory differently; instead of scanning memory, indirect checkpoints proactively gather lists of dirty pages. Indirect checkpoints then manage the list of dirty pages and continuously commit them from memory to the drive, on a pace to not exceed an upper bound of recovery time. This upper bound is defined in the database configuration option `TARGET_RECOVERY_TIME`. By default, in databases created in SQL Server 2016 or higher, this is 60 seconds. In databases created in SQL Server 2012 or 2014, this option was available but set to 0, which indicates that legacy automatic checkpoints are in use.

So, even though the recovery time goal hasn't really changed, the method by which it is achieved has. Indirect checkpoints are significantly faster than automatic checkpoints, especially as servers are configured with more and more memory. You might notice an improvement in the performance of backups specifically.

You can configure a database that was created on an older version of SQL Server to use indirect checkpoints instead of automatic checkpoints with a single command. The TARGET_RECOVERY_TIME will be 0 for older databases still using automatic checkpoint. The master database will also have a TARGET_RECOVERY_TIME of 0 by default, though msdb and model will be set to 60 starting with SQL Server 2016.

Consider setting the TARGET_RECOVERY_TIME database configuration to 60 seconds to match the default for new databases created in SQL Server 2016 or higher, as shown here:

```
ALTER DATABASE [olddb] SET TARGET_RECOVERY_TIME = 60 SECONDS WITH NO_WAIT;
```

You can check this setting for each database in the TARGET_RECOVERY_TIME_IN_SECONDS column of the system view sys.databases.

NOTE

There is a specific performance degradation involving nonyielding schedulers or excessive spinlocks that can arise because of this setting being applied to the TempDB by default, as of SQL Server 2016. It is not common. It is identifiable and resolvable with analysis and custom solution to disable indirect checkpoints on the TempDB, detailed in this blog post from the SQL Server Tiger Team: *https://blogs.msdn.microsoft.com/sql_server_team/indirect-checkpoint-and-tempdb-the-good-the-bad-and-the-non-yielding-scheduler/*.

Moving and removing databases

In this section, we review the steps and options to moving databases and the various methods and stages of removing databases from use.

Moving user and system databases

In this section, we discuss moving database files, which becomes necessary from time to time, either because of improper initial locations or the addition of new storage volumes to a server. Relocating system and user databases is similar to each other, with the master database being an exception. Let's look at each scenario.

Locating SQL Server files

As we discussed in our earlier checklist, you can review the location of all database files by querying the catalog view sys.master_files. If you did not specify the intended location for the data files while you were on the Data Directories page of the Database Engine Configuration step of SQL Server Setup, you will find your system database files on the OS volume at %program-files%\Microsoft SQL Server*instance*\MSSQL\Data.

NOTE

In sys.master_files, the physical_name of each database file, the logical name of each database file (in the Name field of this view), and the name of the database do not need to match. It is possible, through restore operations, to accidentally create multiple databases with the same logical file names.

Ideally, there should be no data or log files on the OS volume, even system database files. You can move these after SQL Server Setup is complete, however.

When you're planning to move your database data or log files, prepare their new file path location by granting FULL CONTROL permissions to the per-SID name for the SQL Server instance. (Note that this is not necessarily the SQL Server service account.) For the default instance, this will be NT SERVICE\MSSQLSERVER; for default instances, it will be NT SERVICE\MSSQL$*instancename*.

Inside OUT

Where does SQL Server keep track of the locations of database files?

When the SQL Server process is started, only three pieces of location information are provided to the service:

- **The location of the master database data file**
- **The location of the master database log file**
- **The location of the SQL Server error log**

You can find this information in the startup parameters of the SQL Server service in the SQL Server Configuration Manager application. All other database file locations are stored in the master database.

Database actions: offline versus detach versus drop

Earlier in this chapter, we discussed strategies to move user database files by using the OFFLINE status. Let's discuss the differences between various ways to remove a database from a SQL Server instance.

The OFFLINE option is one way to quickly remove a database from usability. It is also the most easily reversed, as demonstrated here:

```
SET ONLINE;
```

You should set maintenance activities to ignore databases that are offline because they cannot be accessed, maintained, or backed up. The data and log files remain in place in their location on the drive and can be moved. The database is still listed with its files in sys.master_files.

Taking a database offline is an excellent intermediate administrative step before you DETACH or DROP a database; for example, a database that is not believed to be used any more. Should a user report that she can no longer access the database, the administrator can simply bring the database back online—an immediate action.

You can separate a database's files from the SQL Server by using a DETACH. The data and log files remain in place in their location on the drive and can be moved. But detaching a database removes it from sys.master_files.

To reattach the database, in SQL Server Management Studio, in Object Explorer, follow the Attach steps. It is not as immediate an action and requires more administrative intervention than taking the database offline.

When reattaching the database, you must locate at least the primary data file for the database. The Attach process will then attempt to reassociate all the database files to SQL Server control, in their same locations. If their locations have changed, you must provide a list of all database files and their new locations.

> NOTE
>
> **If you are detaching or restoring a database to attach or copy it to another server, do not forget to follow-up by moving SQL Server logins and then potentially reassociating orphaned database users with their logins. For more information, review Chapter 7.**

Inside OUT

When moving user database files, why should I use offline/online instead of detach/attach?

There are a number of reasons you need to take a user database offline instead of the strategy of detaching, moving, and reattaching the files.

While the database is offline, database information remains queryable in sys.master_files and other system catalog views. You can still reference the locations of database files after taking the database offline to ensure that everything is moved. Also, it is not possible to detach a database when the database is the source of a database snapshot or part of a replication publication. Taking a database offline is the only method possible in these scenarios.

Note that you cannot detach or take system databases offline. A service restart is necessary to move system databases, including the master database.

Finally, a DROP DATABASE command, issued when you use the Delete feature of Object Explorer, removes the database from the SQL Server and deletes the database files on the drive. An exception to the delete files on drive behavior is if the destination database is offline. Deleting an offline database and detaching a database are therefore similar actions.

Dropping a database does not by default remove its backup and restore history from the msdb database, though there is a check box at the bottom of the Drop Database dialog box in SQL Server Management Studio that you can select for this action. The stored procedure msdb.dbo.sp_delete_database_backuphistory is run to remove this history. For databases with a long backup history that has not been maintained by a log history retention policy, the step to delete this history can take a long time and could cause SQL Server Management Studio to stop responding. Instead, delete old backup and restore history incrementally by using msdb.dbo.sp_delete_backuphistory and/or run the msdb.dbo.sp_delete_database_backuphistory procedure in a new SQL Server Management Studio query window.

➤ **For more information on backup and restore history, see Chapter 13.**

Moving user database files

You can move user databases without a SQL Server instance restart and without disrupting other databases by taking the database offline, updating the files, moving them, and then bringing the database online again.

Use the following steps to move user database files:

1. Perform a manual full backup of the soon-to-be affected databases.

2. During a maintenance outage for the database and any applications that are dependent, begin by taking the user database offline and then running a T-SQL script to alter the location of each database file.

3. Here's an example of the T-SQL statements required:

```
ALTER DATABASE database_name SET OFFLINE WITH ROLLBACK IMMEDIATE
ALTER DATABASE database_name MODIFY FILE ( NAME = logical_data_file_name,
FILENAME = 'location\physical_data_file_name.mdf' );
ALTER DATABASE database_name MODIFY FILE ( NAME = logical_log_file_name,
FILENAME = 'location\physical_log_file_name.ldf' );
ALTER DATABASE database_name SET ONLINE
```

4. While the database is offline, physically copy the database files to their new location. (You will delete the old copies when you've confirmed the new configuration.) When the file operation is complete, bring the database back online.

5. Verify that the data files have been moved by querying sys.master_files, which is a catalog view that returns all files for all databases. Look for the physical_name volume to reflect the new location correctly.

6. After you have verified that SQL Server is recognizing the database files in their new locations, delete the files in the original location to reclaim the drive space.

7. After you have successfully moved the database files, you should perform a manual backup of the master database.

Moving system database files, except for master

You cannot move system database files while the SQL Server instance is online; thus, you must stop the SQL Server service.

NOTE

If you plan to move all of the system databases to a different volume, you also will need to move the SQL Server Agent Error Log, or SQL Server Agent will not be able to start.

You can do this in SQL Server Management Studio. In Object Explorer, connect to the SQL Server instance, and then expand the SQL Server Agent folder. Right-click Error Logs, and then, on the shortcut menu that opens, click Configure. Provide a new Error Log File location for the SQLAGENT.OUT file.

Verify that the SQL Server Agent per-SID name for the SQL Server Agent service has FULL CONTROL permissions to the new folder. The per-service SID account will be NT Service\SQLSERVERAGENT for default instances or NT Service\SQLAgent$*instancename* for named instances.

When you later restart the SQL Server service and the SQL Server Agent service, the Agent error log will be written to the new location.

1. Begin by performing a manual full backup of the soon-to-be affected databases.
2. For model, msdb, and TempDB, begin by running a T-SQL script (similar to the script for moving user databases). SQL Server will not use the new locations of the system databases until the next time the service is restarted. You cannot set the system databases to offline.
3. During a maintenance outage for the SQL Server instance, stop the SQL Server instance, and then copy the database files to their new location. (You will delete the old copies when you've confirmed the new configuration.) The only exception here is that the TempDB data and log files do not need to be moved—they will be re-created automatically by SQL Server upon service start.
4. When the file operation is complete, start the SQL Server service again.
5. Verify that the data files have been moved by querying sys.master_files. Look for the physical_name volume to reflect the new location correctly.
6. After you have verified that SQL Server is recognizing the database files in their new locations, delete the files in the original location to reclaim the drive space.
7. After you have successfully moved the database files, perform a manual backup of the master database.

If you encounter problems starting SQL Server after moving system databases to another volume—for example if the SQL Server service account starts and then stops—check for the following:

1. Verify that the SQL Server service account and SQL Server Agent service account have permissions to the new folders location. Review the following link for a list of File System Permissions Granted to SQL Server service accounts: *https://docs.microsoft.com/sql/database-engine/configure-windows/configure-windows-service-accounts-and-permissions#Reviewing_ACLs*
2. Check the Windows Application Event Log and System Event Log for errors.

3. If you cannot resolve the issue, if necessary, start SQL Server with Trace Flag T3608, which does not start the SQL Server fully, only the master database. You then can move all other database files, including the other system databases, back to their original location by using T-SQL commands issued through SQL Server Management Studio.

Moving master database files

Moving the master database files is not difficult, but it is a more complicated process than that for the other system databases. Instead of issuing an ALTER DATABASE ... ALTER FILE statement, you must edit the parameters passed to the SQL Server service in SQL Server Configuration Manager.

1. On the Startup Parameters page, notice that there are three entries containing three files in their current paths. (If you have other startup parameters in this box, do not modify them now.)

   Edit the two parameters beginning with `-d` and `-l` (lowercase "L"). The `-e` parameter is the location of the SQL Server Error Log; you might want to move that, as well.

   After editing the master database data file (-d) and the master database log file (-l) locations, click OK. Keep in mind that the SQL Server service will not look for the files in their new location until the service is restarted.

2. Stop the SQL Server service, and then copy the master database data and log files to their new location. (You will delete the old copies when you've confirmed the new configuration.)

3. When the file operation is complete, start the SQL Server service again.

4. Verify that the data files have been moved by querying sys.master_files, a dynamic management view that returns all files for all databases. Look for the physical_name volume to reflect the new location correctly.

5. After you have verified that SQL Server is recognizing the database files in their new locations, delete the files in the original location to reclaim the drive space.

Single-user mode

By default, all databases are in MULTI_USER mode. Sometimes, it is necessary to gain exclusive access to a database with a single connection, typically in SQLCMD or in a SQL Server Management Studio query window.

For example, when performing a restore, the connection must have exclusive access to the database. By default, the restore will wait until it gains exclusive access. You could attempt to discontinue all connections, but there is a much easier way: setting a database to SINGLE_USER mode removes all other connections but your own.

Setting a database to SINGLE_USER mode also requires exclusive access. If other users are connected to the database, running the following statement will be unsuccessful:

```
ALTER DATABASE database_name SET SINGLE_USER;
```

It is then necessary to provide further syntax to decide how to treat other connections to the database.

- **WITH NO_WAIT.** The ALTER DATABASE command will fail if it cannot gain exclusive access to the database It is important to note that without this statement or any other WITH commands, the ALTER DATABASE command will wait indefinitely.
- **WITH ROLLBACK IMMEDIATE.** Rollback all conflicting requests, ending other SQL Server Management Studio Query window connections, for example.
- **WITH ROLLBACK AFTER n SECONDS.** Delays the effect of WITH ROLLBACK IMMEDIATE by *n* SECONDS, which is not particularly more graceful to competing user connections, just delayed.

For example:

```
ALTER DATABASE databasename
SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
```

Instead of issuing a WITH ROLLBACK, you might choose to identify other sessions connected to the destination database; for example, by using the following:

```
SELECT *
FROM sys.dm_exec_sessions
WHERE
db_name(database_id) = 'database_name';
```

And then evaluate the appropriate strategy for dealing with any requests coming from that session, including communication with that user and closing of unused connections to that database in dialog boxes, SQL Server Management Studio query windows, or user applications.

After you have completed the activities that required exclusive access, set the database back to MULTI_USER mode:

```
ALTER DATABASE database_name SET MULTI_USER;
```

You need to gain exclusive access to databases prior to a restore. This script to change the database to SINGLE_USER and back to MULTI_USER is a common step wrapped around a database restore.

➤ **For more information on database restores, see Chapter 11.**

Index

A

B

C

D

H

I

Q

R

S

T

U

V

W

X

This page intentionally left blank

About the authors

William Assaf

William Assaf, MCSE, is a Microsoft SQL Server consultant and manager and blogs about SQL at *sqltact.com*. William has been a designer, database developer, and admin on application and data warehousing projects for private and public clients. He has helped write the last two generations of Microsoft SQL Server certification exams since 2012 and has been a Regional Mentor for PASS since 2015. William and fellow author Patrick Leblanc worked together on *SQL Server 2012 Step by Step* (Microsoft Press, 2015), having met at and together led the SQL Server User Group and SQLSaturday in Baton Rouge. William and his high school sweetheart enjoy travelling to speak at SQLSaturdays around the south, and hope to see to see you there, too.

Randolph West

Randolph West is a Data Platform MVP from Calgary, Alberta, Canada. He is coorganizer of the Calgary SQL Server User Group and Calgary SQLSaturday. He speaks at various conferences around the world, and acts on stage and screen. Randolph specializes in implementing best practices, performance tuning, disaster recovery, and cloud migrations, through his company Born SQL. You can read his blog at *bornsql.ca*.

Sven Aelterman

Sven Aelterman started with SQL Server when he first deployed version 2000 in a failover cluster scenario. Since then, he has worked as IT manager, principal consultant, and IT director. He currently serves the Trojans (students) of Troy University as a lecturer in information systems in the Sorrell College of Business and as director of IT for the College. In addition, he is cloud software architect for Sorrell Solutions, a business services nonprofit through which Trojans can gain real-world business and IT experience. In a fledgling attempt to give back to the community, he has spoken at many SQLSaturdays and code camps in the southeastern United States since 2005. He spoke about SSIS 2012 at Microsoft TechEd 2011. In 2012, he coauthored a book dedicated to SQL Server FILESTREAM. His involvement with Microsoft Azure resulted in the organization of two Global Azure Bootcamp events at Troy University. Sven blogs about a variety of Microsoft technologies at *svenaelterman.wordpress.com* and tweets and retweets about technology @svenaelterman.

Mindy Curnutt

Mindy Curnutt, an independent consultant, is 4X Microsoft Data Platform MVP and Idera ACE. She has been actively involved in the SQL Server Community for more than a decade, presenting at various User Group Meetings, SQLPASS Summits, as well as SQLSaturdays across North America. For two years, she was a team lead for the SQLPASS Summit Abstract Review Process and since 2015 has served as one of the three SQLPASS Summit program managers. She was a SME for a couple of the SQL 2012 and 2014 Microsoft SQL Server Certification Exams and helped to author *SQL Server 2014 Step by Step*. Mindy currently serves on the board of directors for the North Texas SQL Server User's Group. She also serves as a mentor to others, helping to educate and promote scalable and sustainable SQL Server architecture and design. She is passionate about Data Security, Accessibility, Usability, Scalability and Performance. You can follow Mindy at her blog, *mindycurnutt.com* and on Twitter where she's known as @sqlgirl.

About the Foreword author

Patrick LeBlanc

Patrick LeBlanc is a data platform technical solution professional at Microsoft, working directly with customers on the business value of SQL Server. He coauthored *SharePoint 2010 Business Intelligence 24-Hour Trainer* (Wrox, 2011) and *Knight's Microsoft Business Intelligence 24-Hour Trainer* (Wrox, 2010), and founded *www.sqllunch.com*, a website devoted to teaching SQL Server technologies.