**Introducing Microsoft SQL Server 2019**

**Reliability, scalability, and security both on premises and in the cloud**

**Kellyn Gorman, Allan Hirt, Dave Noderer, James Rowland-Jones, Arun Sirpal, Dustin Ryan, and Buck Woody**

**Introducing Microsoft SQL Server 2019**

Copyright © 2019 Packt Publishing

All rights reserved. No part of this course may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this course to ensure the accuracy of the information presented. However, the information contained in this course is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this course.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this course by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Authors: Kellyn Gorman, Allan Hirt, Dave Noderer, James Rowland-Jones, Arun Sirpal, Dustin Ryan, and Buck Woody

Additional material: Mitchell Pearson

Managing Editor: Alexander Mazonowicz

Acquisitions Editor: Alicia Wooding

Production Editor: Nitesh Thakur

Editorial Board: Shubhopriya Banerjee, Bharat Botle, Ewan Buckingham, Mahesh Dhyani, Taabish Khan, Manasa Kumar, Alex Mazonowicz, Pramod Menon, Bridget Neale, Dominic Pereira, Shiny Poojary, Erol Staveley, Ankita Thakur, Nitesh Thakur, and Jonathan Wray

First Published: December 2019

Production Reference: 1111219

ISBN: 978-1-83882-621-5

Published by Packt Publishing Ltd.

Livery Place, 35 Livery Street

Birmingham B3 2PB, UK

**About the Authors**

**Kellyn Gorman** is an Azure Data Platform Architect for Microsoft with a long history in multi-platform technology. She spends a 60/40% split between Oracle on Azure and Analytics with her present team at Microsoft. A recent Idera ACE, a current friend of Redgate in the Microsoft community and an Oracle ACE Director alumnus, she has been awarded numerous awards over the years for her technical contributions and community volunteerism. She is one of only six women part of the Oak Table, a network for the Oracle scientist. She has extensive experience in environment optimization, automation and architect of robust environments, specializing in multi-terabyte management of OLAP/DSS systems. A consistent advocate for logical tuning of code and design before hardware solutions. She's recently become known for her expertise in DevOps, Enterprise Manager, AWR, (Automatic Workload Repository) Warehouse and virtualization of database environments with complex cloud management. The technical knowledge required to support these features offers great educational opportunities to learn by attending her technical presentations, engaging with her on social media presence as DBAKevlar or reading her blog, https://dbakevlar.com/.

Boston-based Cloud and Data Center and Data Plaform Dual MVP **Allan Hirt** has been working with SQL Server since 1992 and clustering in Windows Server since the days just after Wolfpack. He got his start with databases as an intern at SQL Solutions, which then got purchased by Sybase where he remained an intern until the end of college. Allan has used every version of SQL Server that Microsoft has released for Windows. He founded his own company, Megahirtz, in 2007 and is now partners with Max Myrick in SQLHA.

You will often find Allan speaking at local user groups, SQL Saturdays, and various conferences like PASS Summit, SQLBits, and TechEd as well as doing various webcasts during the years. He has authored quite a bit of content over the years including articles for SQL Server Magazine and whitepapers for Microsoft that are up on TechNet and MSDN. He is the author or co-author of a quite a few books, and is working on his latest, *Mission Critical SQL Server*, which will be due out soon.

**Dave Noderer** is the CEO / President and founder of Computer Ways, Inc., a software development company in Deerfield Beach, FL. Dave is an electrical engineer by training, designed computers for 20 years and has been writing software since founding Computer Ways, Inc. in 1994. Dave spent three years as an officer and director of INETA (International .NET Association) where he oversaw the enrollment and support of hundreds of user groups worldwide and 16 years as a Microsoft MVP. He co-founded Florida .NET User groups in 2001 and has been holding meetups in South Florida ever since. Since 2005, he has led the annual, free South Florida Code Camp. This event attracts over 1000 developer attendees. Dave is involved in local community activities as a board member of the Deerfield Beach Historical Society, the Hillsboro Lighthouse Society, and TechLauderdale.org.

**James Rowland-Jones (JRJ)** is a Principal Program Manager at Microsoft. He is currently part of the SQL Server team working on SQL Server 2019 Big Data Clusters and data virtualization. Prior to joining SQL Server, JRJ worked extensively on Azure SQL Data Warehouse. He helped the team launch Gen 1 of the service and led the product management effort to bring Gen 2 into preview.

JRJ is passionate about delivering highly scalable solutions that are creative, simple and elegant. He is also a keen advocate for the worldwide SQL community; previously serving on the Board of Directors for PASS while also helping to build SQLBits—Europe's largest data event. JRJ was awarded Microsoft's MVP accreditation from 2008 to 2015 for his services to the community.

*For Jane, Lucy, Kate, and Oliver. Forever x.*

**Arun Sirpal** is a SQL Server consultant and currently a Microsoft Data Platform MVP. Specializing in both SQL Server and Microsoft Azure, he has over 12 years' experience architecting, designing, and performance tuning physical and virtualized SQL Servers

and has a wealth of experience with designing solutions using the Azure Data Platform including Azure SQL Database, Azure SQL Database Managed Instances, elastic pools, and hybrid concepts. Arun is also a frequent writer, speaker, and technical reviewer and a member of Microsoft's Azure Advisors and SQL Advisors groups.

**Dustin Ryan** is a Senior Cloud Solution Architect at Microsoft. Dustin has worked in the business intelligence and data warehousing field since 2008, has spoken at community events such as SQL Saturday, SQL Rally, and PASS Summit, and has a wide range of experience designing and building solutions featuring SQL Server and Azure. Prior to his time at Microsoft, Dustin worked as a business intelligence consultant and trainer for Pragmatic Works. Dustin is also an author, contributor, and technical editor of books.

Dustin resides outside Jacksonville, Florida with his wife, three children, and a three-legged cat and enjoys spending time with his family and serving at his local church.

**Buck Woody** works on the Azure Data Services team at Microsoft and uses data and technology to solve business and science problems. With over 35 years of professional and practical experience in computer technology, he is also a popular speaker at conferences around the world; author of over 700 articles and eight books on databases, machine learning, and R, he also sits on various Data Science Boards at two US Universities, and specializes in advanced data analysis techniques.

**Table of Contents**

**About the Authors ................................................................................................... c Preface i**

**Chapter 1: Optimizing for performance,**

**scalability and real-time insights 1**

**Hybrid transactional and analytical processing (HTAP) ...................................... 2 Clustered Columnstore Indexes ............................................................................ 3 Adding Clustered Columnstore Indexes to memory-optimized tables .................... 5 Disk-based tables versus memory-optimized tables .......................................... 5 In-memory OLTP ...................................................................................................... 6 Planning data migration to memory-optimized tables ....................................... 7 Natively compiled stored procedures ................................................................. 18 TempDB enhancements ........................................................................................ 21 Enabling memory-optimized TempDB metadata .................................................... 21 Limitations of memory-optimized TempDB metadata ........................................... 21 Intelligent Query Processing ................................................................................ 22 Hybrid Buffer Pool ................................................................................................. 24 Query Store ............................................................................................................. 25 Changes to default parameter values ....................................................................... 27 QUERY\_CAPTURE\_MODE ............................................................................................. 27 QUERY\_CAPTURE\_MODE: CUSTOM ............................................................................ 27 Support for FAST\_FORWARD and STATIC Cursors .................................................... 29 Automatic tuning ................................................................................................... 29 Automatic plan correction .......................................................................................... 29**

**Lightweight query profiling .................................................................................. 32 New functionality in 2019 ........................................................................................... 32 sys.database\_scoped\_configurations ........................................................................ 33 Activity monitor ........................................................................................................... 34**

**Columnstore stats in DBCC CLONEDATABASE ................................................... 34 Columnstore statistics support .................................................................................. 35 DBCC CLONEDATABASE validations .......................................................................... 35 Understanding DBCC CLONEDATABASE syntax ....................................................... 35**

**Estimate compression for Columnstore Indexes .............................................. 36 sp\_estimate\_data\_compression\_savings Syntax ...................................................... 37 Troubleshooting page resource waits ................................................................. 39 sys.dm\_db\_page\_info ................................................................................................... 39 sys.fn\_pagerescracker ................................................................................................. 41**

**Chapter 2: Enterprise Security 45**

**SQL Data Discovery and Classification ................................................................ 46 SQL Vulnerability Assessment .............................................................................. 51 Transparent Data Encryption ..................................................................................... 55 Setup .............................................................................................................................. 57 New features – suspend and resume ........................................................................ 59 Extensible Key Management ...................................................................................... 60 Always Encrypted ................................................................................................... 60 Algorithm types ............................................................................................................ 61 Setup .............................................................................................................................. 61 Confidential computing with secure enclaves ................................................... 61 Dynamic Data Masking ......................................................................................... 64 Types ............................................................................................................................. 64 Implementing DDM ..................................................................................................... 64**

**Row-Level Security ................................................................................................. 67 Auditing ................................................................................................................... 71 Securing connections ............................................................................................ 73**

**Configuring the MMC snap-in ..................................................................................... 73 Enabling via SQL Server Configuration Manager ..................................................... 74 Azure SQL Database .............................................................................................. 74 SSL/TLS .......................................................................................................................... 75 Firewalls ........................................................................................................................ 75 Azure Active Directory (AD) authentication ............................................................. 75 Advanced data security ............................................................................................... 77 Advanced threat detection ......................................................................................... 78**

**Chapter 3: High Availability and Disaster Recovery 81**

**SQL Server availability feature overview ............................................................ 82 Backup and restore ..................................................................................................... 82 Always On features ...................................................................................................... 83 Log shipping .................................................................................................................. 91**

**What About Database Mirroring and Replication? ............................................ 92 Availability improvements in SQL Server 2019 .................................................. 92 Accelerated database recovery .................................................................................. 92 Configuration-only replica .......................................................................................... 92 Certificate management in SQL Server Configuration Manager ........................... 94 Clustered columnstore index online rebuild ............................................................ 95 Database scoped default setting for online and resumable DDL operations ...... 95 Failover Cluster Instance Support for Machine Learning Services ........................ 96 Increased number of synchronous replicas in the Enterprise edition .................. 96 Online builds or rebuilds for Clustered Columnstore Indexes .............................. 97**

**Read-only routing configuration in SQL Server Management Studio ................... 98 Replication for Linux-based configurations ............................................................. 99 Secondary-to-primary read/write connection redirection ................................... 100**

**Windows Server 2019 availability enhancements ........................................... 102 Changing domains for a Windows Server Failover Cluster ................................... 103**

**Cluster Shared Volumes support for Microsoft Distributed**

**Transaction Coordinator ........................................................................................... 103 File share witness without a domain ...................................................................... 103 Improved Windows Server Failover Cluster security ............................................ 104 Storage Replica in the Standard edition ................................................................. 104 Storage Spaces Direct two-node configuration ..................................................... 106 Windows Server Failover Cluster improvements in Azure ................................... 107**

**Chapter 4: Hybrid Features – SQL Server and**

**Microsoft Azure 111**

**Backup to URL ...................................................................................................... 112 Benefits ....................................................................................................................... 112 Requirements ............................................................................................................. 112**

**The storage account ............................................................................................ 113 Setup ............................................................................................................................ 114 SQL Server data files in Azure ............................................................................ 118 Setup and concepts ................................................................................................... 119 Considerations ........................................................................................................... 121 File-snapshot backups ......................................................................................... 123 Setup ............................................................................................................................ 123 Extending on-premises Availability Groups to Azure ...................................... 125 Replication to Azure SQL Database ................................................................... 126 Classic approach ........................................................................................................ 127**

**Transactional replication .................................................................................... 127 Prerequisites .............................................................................................................. 129 Setup ............................................................................................................................ 129**

**Chapter 5: SQL Server 2019 on Linux 143**

**2019 platform support ........................................................................................ 144 Why move databases to SQL Server on Linux? ................................................ 145 Installation and configuration .................................................................................. 146 Improvements in SQL Server 2019 .......................................................................... 150 Machine Learning Services on Linux ................................................................. 150 Kubernetes ........................................................................................................... 152 Working with Docker and Linux ......................................................................... 154 Change data capture ........................................................................................... 155 Hybrid Buffer Pool and PMEM ............................................................................ 155 Distributed Transaction Coordinator on Linux ................................................ 157 Replication ............................................................................................................ 158 SQL Server tools ................................................................................................... 159 Azure Data Studio ...................................................................................................... 159 Command-line query tools for SQL in Linux .................................................... 163 SQLCMD ...................................................................................................................... 163 MSSQL-CLI ................................................................................................................... 164 Enhanced focus on scripting .............................................................................. 165 The SQL DBA in the Linux world ........................................................................ 165 Users and groups ....................................................................................................... 166 Azure Cloud Shell ....................................................................................................... 166 Windows Subsystem for Linux ........................................................................... 167 Root, the super-user .................................................................................................. 167**

**Chapter 6: SQL Server 2019 in Containers and Kubernetes 171**

**Why containers matter ....................................................................................... 172 Container technical fundamentals .................................................................... 173 Deploying an SQL Server container using Docker ........................................... 174**

**Using Docker and Bash ............................................................................................. 179 Using local SQL Server utilities ................................................................................. 179 Customizing SQL Server containers .................................................................. 180 Availability for SQL Server containers ............................................................... 180**

**Chapter 7: Data Virtualization 185**

**Data integration challenges ............................................................................... 186 Introducing data virtualization .......................................................................... 186 Data virtualization use cases ............................................................................. 188**

**Data virtualization and hybrid transactional analytical processing ................... 188 Data virtualization and caching ............................................................................... 188 Data virtualization and federated systems ............................................................ 188 Data virtualization and data lakes ........................................................................... 189**

**Contrasting data virtualization and data movement ...................................... 189 Data virtualization in SQL Server 2019 .............................................................. 190 Secure data access ............................................................................................... 190**

**The database master key .......................................................................................... 191 Database scoped credentials ................................................................................... 191 External data sources .......................................................................................... 192 Supported data sources ............................................................................................ 193 Extending your environment using an ODBC external data source ................... 194 Accessing external data sources in Azure .............................................................. 196 External file formats ............................................................................................ 197**

**PolyBase external tables ..................................................................................... 198 Creating external tables with Azure Data Studio .................................................. 200 Contrasting linked servers and external tables ..................................................... 201**

**Installing PolyBase in SQL Server 2019 ............................................................. 202 General pre-installation guidance ........................................................................... 203 Installing PolyBase on Windows .............................................................................. 204 Installing PolyBase on Linux ..................................................................................... 205 Installing PolyBase on SQL Server running in Docker ........................................... 206 Post-installation steps ............................................................................................... 208**

**Installing PolyBase as a scale-out group ........................................................... 209 Tip #1: Use different resource groups for each part of the architecture ........... 210**

**Tip #2: Create the virtual network and secure subnets before**

**building virtual machines ......................................................................................... 210 Tip #3: Place your scale-out group SQL Server instances inside one subnet ..... 210 Tip #4: Complete this pre-installation checklist! .................................................... 211 Scale-out group installation ..................................................................................... 212 Bringing it all together: your first data virtualization query .......................... 215**

**Chapter 8: Machine Learning Services**

**Extensibility Framework 219**

**Machine learning overview ................................................................................ 220 How machine learning works ................................................................................... 220 Use cases for machine learning ............................................................................... 221 Languages and tools for machine learning ............................................................ 222**

**SQL Server 2019 Machine Learning Services architecture and components .......................................................................................................... 224**

**Components ............................................................................................................... 226 Configuration ............................................................................................................. 228**

**Machine learning using the Machine Learning Services extensibility framework ............................................................................................................ 230**

**R for machine learning in SQL Server 2019 ............................................................ 230 Python for machine learning in SQL Server 2019 .................................................. 232 Java and machine learning in SQL Server ......................................................... 233 Machine learning using the PREDICT T-SQL command ................................... 237 Machine learning using the sp\_rxPredict stored procedure .......................... 239 Libraries and packages for machine learning .................................................. 240 Management ........................................................................................................ 241 Security ....................................................................................................................... 242 Monitoring and Performance ................................................................................... 242 Using the team data science process with Machine Learning Services ........ 244 Understanding the team data science process ..................................................... 244 Phase 1: Business understanding ............................................................................ 245 Phase 2: Data acquisition and understanding ....................................................... 245 Phase 3: Modeling ...................................................................................................... 245 Phase 4: Deployment ................................................................................................. 245 Phase 5: Customer acceptance ................................................................................ 246**

**Chapter 9: SQL Server 2019 Big Data Clusters 249**

**Big data overview ................................................................................................ 250 Applying scale-out architectures to SQL Server ............................................... 250 Containers ................................................................................................................... 251 Kubernetes ................................................................................................................. 253 SQL Server on Linux ................................................................................................... 254 PolyBase ...................................................................................................................... 255 SQL Server 2019 big data cluster components ................................................ 256**

**Installation and configuration ............................................................................ 257 Platform options ........................................................................................................ 258 Using a Kubernetes service ...................................................................................... 258 Using an on-premises Kubernetes installation ...................................................... 259 Working with a Dev/Test environment ................................................................... 259 Deploying the big data clusters on a Kubernetes cluster ..................................... 260**

**Programming SQL Server 2019 big data clusters ............................................ 262 Azure Data Studio ...................................................................................................... 262 Relational operations ................................................................................................ 264 Creating scale-out tables .......................................................................................... 266 Creating a data lake ................................................................................................... 268 Working with Spark ................................................................................................... 269 Submitting a job from Azure Data Studio ............................................................... 270 Submitting a Spark job from IntelliJ ........................................................................ 272 Spark job files and data locations ............................................................................ 273**

**Management and monitoring ............................................................................ 273 SQL Server components and operations ................................................................ 273 Kubernetes operations ............................................................................................. 273 SQL Server 2019 big data cluster operations ......................................................... 274 Monitoring performance and operations with Grafana ....................................... 275 Monitoring logs with Kibana .................................................................................... 276 Spark operations ........................................................................................................ 277**

**Security ................................................................................................................. 277 Access .......................................................................................................................... 278 Security setup and configuration ............................................................................ 278 Authentication and authorization ........................................................................... 280**

**Chapter 10: Enhancing the Developer Experience 283**

**SQL Graph Database ........................................................................................... 285 Why use SQL Graph? .................................................................................................. 287 Edge constraints ......................................................................................................... 287 SQL Graph data integrity enhancements ............................................................... 290 SQL Graph MATCH support in MERGE ..................................................................... 290 Using a derived table or view in a graph MATCH query ........................................ 294**

**Java language extensions ................................................................................... 296 Why language extensions? ....................................................................................... 296 Installation .................................................................................................................. 297 Sample program ........................................................................................................ 300**

**JSON ....................................................................................................................... 307 Why use JSON? ........................................................................................................... 307 JSON example ............................................................................................................. 308**

**UTF-8 support ....................................................................................................... 309 Why UTF-8? ................................................................................................................. 309 Temporal tables ................................................................................................... 310 Why temporal tables? ................................................................................................ 311 Temporal table example ........................................................................................... 311 Spatial data types ................................................................................................ 314 Why spacial data types? ............................................................................................ 315 Dealer locator example ............................................................................................. 315**

**Chapter 11: Data Warehousing 319**

**Extract-transform-load solutions with SQL Server Integration Services ...... 320 Best practices for loading your data warehouse with SSIS .................................. 321 Clustered Columnstore Indexes ........................................................................ 322**

**Partitioning ........................................................................................................... 324 Online index management ................................................................................. 325 Enabling online DML processing .............................................................................. 326 Resuming online index create or rebuild ................................................................ 327 Build and rebuild online clustered columnstore indexes ..................................... 329 Using ALTER DATABASE SCOPE CONFIGURATION ................................................. 329 Creating and maintaining statistics ................................................................... 330 Automatically managing statistics ........................................................................... 331 The AUTO\_CREATE\_STATISTICS option ................................................................... 331 The AUTO\_UPDATE\_STATISTICS option .................................................................. 331 The AUTO\_UPDATE\_STATISTICS\_ASYNC option .................................................... 331 Statistics for columnstore indexes .................................................................... 332 Modern data warehouse patterns in Azure ..................................................... 332 Introduction to Azure SQL Data Warehouse .................................................... 333 Control node ............................................................................................................... 333 Compute nodes .......................................................................................................... 334 Storage ........................................................................................................................ 334 Data movement services (DMSes) ........................................................................... 334 Best practices for working with Azure SQL Data Warehouse ........................ 334 Reduce costs by scaling up and down .................................................................... 335 Use PolyBase to load data quickly .......................................................................... 335 Manage the distributions of data ............................................................................ 336 Do not over-partition data ....................................................................................... 336 Using Azure Data Factory ................................................................................... 337 New capabilities in ADF ............................................................................................. 337 Understanding ADF .................................................................................................... 338 Copying data to Azure SQL Data Warehouse ................................................... 340**

**Hosting SSIS packages in ADF ............................................................................. 343 Azure Data Lake Storage ..................................................................................... 344 Key features of Azure Data Lake Storage Gen2 ..................................................... 345 Azure Databricks .................................................................................................. 346 Working with streaming data in Azure Stream Analytics ............................... 347 Analyzing data by using Power BI – and introduction to Power BI ............... 349 Understanding the Power BI ecosystem ................................................................. 349 Connecting Power BI to Azure SQL Data Warehouse ............................................ 352**

**Chapter 12: Analysis Services 355**

**Introduction to tabular models ......................................................................... 356 Introduction to multidimensional models ....................................................... 358 Enhancements in tabular mode ......................................................................... 359**

**Query interleaving with short query bias ............................................................... 360 Memory settings for resource governance ............................................................ 360 Calculation groups ..................................................................................................... 361 Dynamic format strings ............................................................................................ 362 DirectQuery ................................................................................................................ 363 Bidirectional cross-filtering ...................................................................................... 366 Many-to-many relationships .................................................................................... 367 Governance settings for Power BI cache refreshes ............................................... 368 Online attach .............................................................................................................. 368**

**Introducing DAX ................................................................................................... 369 Calculated columns ................................................................................................... 370 Calculated measures ................................................................................................. 370 Calculated tables ........................................................................................................ 372 Row filters ................................................................................................................... 373 DAX calculation best practices ................................................................................. 375**

**Writing DAX queries ............................................................................................ 376 Using variables in DAX ........................................................................................ 379 Introduction to Azure Analysis Services ........................................................... 380**

**Selecting the right tier ............................................................................................... 381 Scale-up, down, pause, resume, and scale-out ...................................................... 382 Connecting to your data where it lives ................................................................... 382 Securing your data ..................................................................................................... 383 Using familiar tools .................................................................................................... 383 Built-in monitoring and diagnostics ........................................................................ 384**

**Provisioning an Azure Analysis Services server and**

**deploying a tabular model ........................................................................................ 384 Chapter 13: Power BI Report Server 389**

**SSRS versus Power BI Report Server ................................................................. 389 Report content types ........................................................................................... 391 Migrating existing paginated reports to Power BI Report Server .................. 392 Exploring new capabilities .................................................................................. 395**

**Performance Analyzer ............................................................................................... 396 The new Modeling view ............................................................................................. 398 Row-level security for Power BI data models ......................................................... 398 Report theming .......................................................................................................... 400**

**Managing parameter layouts ............................................................................. 401 Developing KPIs .................................................................................................... 402 Publishing reports ............................................................................................... 405 Managing report access and security ............................................................... 406 Publishing mobile reports .................................................................................. 409 Viewing reports in modern browsers ................................................................ 409 Viewing reports on mobile devices .................................................................... 412**

**Exploring Power BI reports ................................................................................. 415 Using the FILTERS panel ............................................................................................ 416 Crossing-highlighting and cross-filtering ................................................................ 416 Sorting a visualization ............................................................................................... 417 Displaying a visualization's underlying data .......................................................... 417 Drill-down in a visualization ..................................................................................... 418**

**Automating report delivery with subscriptions ............................................... 418 Pinning report items to the Power BI service .................................................. 420**

**Chapter 14: Modernization to the Azure Cloud 423**

**The SQL data platform in Azure ......................................................................... 424 Azure SQL Database managed instance ................................................................. 424 Deployment of a managed instance in Azure .................................................. 425 Managed instance via the Azure portal .................................................................. 426 Managed instance via templates ............................................................................. 427 Migrating SQL Server to Managed Instance ..................................................... 430 Azure Database Migration Service (DMS) ............................................................... 431 Application Connectivity ........................................................................................... 431 Requirements for the DMS ....................................................................................... 432 Data Migration Assistant .......................................................................................... 433 Managed Instance Sizing .......................................................................................... 433 Migration ..................................................................................................................... 433 Monitoring Managed Instance ................................................................................. 434 SQL Server in Azure virtual machines ..................................................................... 435 Creating an Azure VM from the Azure portal ................................................... 436 Storage options for VMs ...................................................................................... 438 Diagnostics and advanced options .................................................................... 438 Creating a SQL Server 2019 VM from the command line in Azure ................ 440**

**Security for SQL Server on an Azure VM ........................................................... 443 Backups of Azure VM SQL Server instances ........................................................... 444 Built-in security for Azure VMs ................................................................................. 444**

**SQL Server IaaS agent extension ....................................................................... 446 Disaster Recovery environment in the cloud ................................................... 447 Azure Site Recovery ................................................................................................... 447 Extended support for SQL 2008 and 2008 R2 ......................................................... 448**

**Index 453**

**Be the data hero**

Get insights from all your data. At scale. Put your database skills to work in the cloud. Get free database and AI services—and a $200 credit. Start free >

Get help with your project. Talk to a sales specialist >

**>**

**Preface**

**About**

This section briefly introduces the coverage of this book, the technical skills you'll need to get started, and the hardware and software required to complete the book.

**ii** | Preface

**About Microsoft SQL Server 2019**

From its humble beginnings in OS/2 with version 1.1, SQL Server has proved over and over that it is a database that data professionals love to use. The engine is reliable, and the T-SQL dialect has everything the developer needs to quickly write resilient, high performing applications.

With every release, SQL Server has improved on performance, functions, reliability, and security. As the releases progressed, more features were added, and then entirely new capabilities—a job engine, a reporting server, business intelligence, and data mining systems. Groundbreaking technologies, such as in-memory databases and columnstore indexes, made SQL Server one of the most installed Relational Database Management Systems (RDBMSes) in the world.

In Spring of 2016, Microsoft announced that SQL Server would be made available on the Linux operating system—something unbelievable to many technical professionals. Addition of Platform Abstraction Layer (PAL) in SQL Server allowed it to run on Linux operating systems such as Ubuntu, Red Hat Enterprise Linux, and SUSE. It also added support for Linux containers, opening up amazing new possibilities for deployment and operation.

SQL Server 2019 represents not only an evolutionary release, but a revolutionary release. The promise of containers is completely realized with support for Kubernetes. The new SQL Server 2019 Big Data Clusters leverages Kubernetes as the deployment platform and adds the power of Spark and Apache Hadoop File System (HDFS). Additionally, SQL Server 2019 supports Data Virtualization and workloads with deployable applications running on-premises, in the cloud, and even in hybrid configurations. This allows SQL Server 2019 to modernize your data estate and applications with intelligence over any data—structured and unstructured.

Like the releases before it, SQL Server 2019 isn't limited to just the Windows platform. In addition to SQL Server 2019 on Windows, Linux, and containers, Microsoft has also announced a new product—Azure SQL Database Edge—which is a small-footprint SQL Server engine that runs on Edge devices and the ARM platform. This allows a consistent

developer experience from the ground to the cloud and the edge. Add to this the choice of platform and the choice of programming languages such as T-SQL, Java, C/C++, Scala, Node.js, C#/VB.NET, Python, Ruby, and .NET Core. Need more? You can add your own languages as well.

SQL Server 2019 supports machine learning and extensibility with R, Python, Java, and Microsoft .NET. You're able to operationalize your machine learning models close to the data, and developers can leverage Java and other languages server-side.

About Microsoft SQL Server 2019 | **iii**

But it's not just about new features. SQL Server maintains its high standards in performance and security. This release boasts industry-leading performance. It has the #1 OLTP performance benchmarks, and #1 DW performance on 1 TB, 3 TB, and 10 TB non-clustered DW workloads. It supports in-memory across all workloads and is the most consistent on-premises data platform—in both IaaS and PaaS. SQL Server 2019 has intelligent query processing features that improve the performance of mission-critical queries. They also support in-memory transactions and in-memory analytics for hybrid transactional and analytical processing.

Security is essential in any data storage and processing system, and SQL Server has prided itself on being the most secure database over the last eight years according to the National Institute of Standards and Technology's (NIST's) Comprehensive Vulnerability Database. SQL Server supports enterprise security and compliance with security features such as Transparent Data Encryption, Auditing, Row-Level Security, Dynamic Data Masking and Always Encrypted. SQL Server 2019 adds support for secure enclaves in Always Encrypted to enable rich computations on encrypted data.

SQL Server 2019 allows you to solve modern data challenges. Data virtualization with PolyBase allows you to use SQL Server 2019 as a data hub, directly querying data from data sources. These sources include Oracle, SAP HANA, MongoDB, Hadoop clusters, Cosmos DB, and SQL Server—all using T-SQL, and without separately installing client connection software. SQL Server 2019 also gives you insights and rich new reports, even for mobile BI with Power BI Report Server.

SQL Server 2019 improves reliability with several features in the High Availability and Disaster Recovery architecture and works with the built-in availability features in Kubernetes. It recovers faster with Accelerated Database Recovery.

This book covers these features, giving you a tour of each of them and diving in with real-world examples and sample code you can try out on your own. Put together by recognized experts and members of the team that wrote the software, we'll get you up to speed quickly and ready to start your own adventure with this latest release of the world's best data platform.

**About the chapters**

*Chapter 1*, *Optimizing for performance and real-time insights*, explains how SQL Server 2019 gets the most out of your hardware and empowers your analytics with features such as Hybrid Buffer Pool, and hybrid transactional and analytical processing.

*Chapter 2*, *Enterprise Security and Compliance*, covers the essential elements in SQL Server 2019 to ensure your operations are not compromised and that they stay compliant with industry regulations for data usage.

**iv** | Preface

*Chapter 3*, *High Availability and Disaster Recovery*, covers SQL Server 2019's built-in methods to increase availability, minimize downtime for maintenance, and assist when outages occur.

*Chapter 4*, *Hybrid Features—SQL Server and Microsoft Azure*, looks at how SQL Server 2019 and Azure Storage work together to offer enterprise-ready, highly scalable, and flexible storage solutions at competitive prices.

*Chapter 5*, *SQL Server 2019 on Linux*, looks at how SQL Server 2019 is building on the Linux features in the 2017 release to offer even more functionality.

*Chapter 6*, *SQL Server 2019 in Containers and Kubernetes*, explains how virtualization features have evolved and how you can deploy SQL Server across Docker and Kubernetes.

*Chapter 7*, *Data Virtualization*, highlights SQL Server 2019's position as a modern enterprise data hub and how you can use features such as hybrid transactional and analytical processing to query across disparate systems.

*Chapter 8*, *Machine Learning Services Extensibility Framework,* explores machine learning, the components and architectures in SQL Server 2019 you can use to implement such services, and the process you follow for your solutions.

*Chapter 9*, *SQL Server 2019 Big Data Clusters*, builds on the concepts covered in the previous chapter to show how SQL Server 2019 can be leveraged to handle scaled-out datasets.

*Chapter 10*, *Enhancing the Developer Experience*, covers the tools to develop and manage SQL Server projects, including Visual Studio, SQL Server Management Studio, and—especially for cross-platform development—Visual Studio Code.

*Chapter 11*, *Data Warehousing,* highlights mission-critical security features such as Row-Level Security, Always Encrypted, and data masking.

*Chapter 12*, *Analysis Services*, looks at how SQL Server 2019 provides superior performance for decision support and business analytics workloads via multidimensional mode and tabular mode.

*Chapter 13*, *Power BI Report Server,* looks at new features that are included in the latest releases of Power BI Report Server, as well as key differences between Power BI Report Server and SSRS.

*Chapter 14*, *Modernization to the Azure Cloud*, finishes the book with a discussion of Azure's role regarding modernization and the data platform.

About Microsoft SQL Server 2019 | **v**

**Conventions**

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "A non-durable table is declared with **DURABILITY=SCHEMA\_Only**."

A block of code is set as follows:

USE master;

GO

BACKUP CERTIFICATE MyServerCert

TO FILE = 'C:\SQLSERVER\MyServerCert.cer'

WITH PRIVATE KEY

(FILE = 'C:\SQLSERVER\certificate\_Cert.pvk',

ENCRYPTION BY PASSWORD = '!£$Strongpasswordherewelikesqlver#')

New terms and important words are shown like this: "Most Windows Server-based WSFCs (and SQL Server deployments) use **Active Directory Domain Services** (**AD DS**)."

Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Go to **Actions** and select **Get Shared Access Signature** as shown."

**System requirements**

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

• SQL Server 2019 Developer edition or higher with SQL Server Management Studio. • A computer that has a 1.4 GHz or faster x64 processor (2 GHz recommended) • 1 GB of memory (4 GB recommended)

• 6 GB of available hard-disk space

• Super VGA 800 x 600 or higher resolution display

• Internet connection to download software, as described in applicable chapters.

• For non-Windows platforms such as Linux or virtual machines, please refer to the release documentation.

Depending on your Windows configuration, you might require local administrator rights to install or configure SQL Server 2019 and related products.

**vi** | Preface

**Prerelease software**

To help you become familiar with SQL Server 2019 as soon as possible after its release, we wrote this book by using examples that worked with SQL Server 2019 Release Candidate. Consequently, the final version might include new features, the user interface might change, or features that we discuss might change or disappear. Refer to *What's New in SQL Server 2019* at https://docs.microsoft.com/en-us/sql/sql-server/ what-s-new-in-sql-server-ver15?view=sqlallproducts-allversions for the most up-to date list of changes to the product.

**AdventureWorks Database**

Some demonstrations make a reference to the **AdventureWorks** database. This is a sample database published by Microsoft and used to demonstrated SQL Server 2019's new features. The database, along with download and setup instructions, can be found at https://docs.microsoft.com/en-us/sql/samples/adventureworks-install configure?view=sql-server-ver15.

**1**

**Optimizing for**

**performance,**

**scalability and**

**real‑time insights**

Companies are optimizing their computing resources to get more transactional performance out of the same hardware resources. At the same time, the demand and pace of business and customer focus is increasing; they need real-time insights on the transactional data.

In recent years, many companies have turned to No-SQL solutions that allow very high write performance of transactions while allowing eventual consistency, but that later require data mining and analysis.

**2** | Optimizing for performance, scalability and real‑time insights

Microsoft SQL Server has taken on this challenge and, with every release, continues to expand the workloads in many dimensions. This chapter will discuss many of the features that allow both high-performance transaction processing while simultaneously allowing real-time analytics on transactional data without the need for a separate set of ETL processes, a separate data warehouse, and the time to do that processing.

Microsoft SQL Server 2019 is built on a database engine that is number one for TPC-E (On-Line Transaction Processing Benchmark) and TCP-H (Decision Support Benchmark). See http://www.tpc.org for more information.

Changes in hardware architecture allow dramatic speed increases with Hybrid Buffer Pool, which utilizes persistent memory (PMEM), also known as **Storage Class Memory** (**SCM**).

Microsoft SQL Server 2019 can be used in the most demanding computing environments required today. Using a variety of features and techniques, including in-memory database operations, can make dramatic increases in your transaction processing rate while still allowing near-real-time analysis without having to move your transaction data to another "data warehouse" for reporting and analysis.

Microsoft SQL Server 2019 has also expanded the number of opportunities to tune database operations automatically, along with tools and reports to allow monitoring and optimization of queries and workloads. Comprehensive diagnostic features including Query Store allow SQL Server 2019 to identify performance issues quickly.

By upgrading to SQL Server 2019, the customer will be able to boost query performance without manual tuning or management. **Intelligent Query Processing** (**IQP**) helps many workloads to run faster without making any changes to the application.

**Hybrid transactional and analytical processing (HTAP)**

**Hybrid transactional and analytical processing** (**HTAP**), is the application of tools and features to be able to analyze live data without affecting transactional operations.

In the past, data warehouses were used to support the reporting and analysis of transactional data. A data warehouse leads to many inefficiencies. First, the data has to be exported from the transactional database and imported into a data warehouse using ETL or custom tools and processes. Making a copy of data takes more space, takes time, may require specialized ETL tools, and requires additional processes to be designed, tested, and maintained. Second, access to analysis is delayed. Instead of immediate access, business decisions are made, meaning the analysis may be delayed by hours or even days. Enterprises can make business decisions faster when they can get real-time operational insights. In some cases, it may be possible to affect customer behavior as it is happening.

Clustered Columnstore Indexes | **3**

Microsoft SQL Server 2019 provides several features to enable HTAP, including memory-optimized tables, natively compiled stored procedures, and Clustered Columnstore Indexes.

This chapter covers many of these features and will give you an understanding of the technology and features available.

A more general discussion of HTAP is available here: https://en.wikipedia.org/wiki/ Hybrid\_transactional/analytical\_processing\_(HTAP).

**Clustered Columnstore Indexes**

Clustered Columnstore indexes can make a dramatic difference and are the technology used to optimize real-time analytics. They can achieve an order of magnitude performance gain over a normal row table, a dramatic compression of the data, and minimize interference with real-time transaction processing.

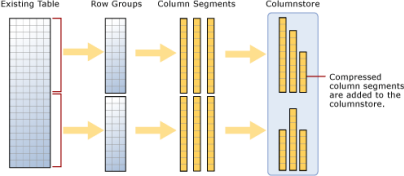
A columnstore has rows and columns, but the data is stored in a column format.

A rowgroup is a set of rows that are compressed into a columnstore format — a maximum of a million rows (1,048,576).

There are an optimum number of rows in a rowgroup that are stored column-wise, and this represents a trade-off between large overhead, if there are too few rows, and an inability to perform in-memory operations if the rows are too big.

Each row consists of column segments, each of which represents a column from the compressed row.

Columnstore is illustrated in *Figure 1.1*, showing how to load data into a non-clustered columnstore index:

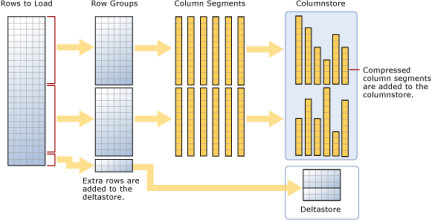
**Figure 1.1: Loading data into a non-clustered columnstore index**

**4** | Optimizing for performance, scalability and real‑time insights

A clustered columnstore index is how the columnstore table segments are stored in physical media. For performance reasons, and to avoid fragmenting the data, the columnstore index may store some data in a deltastore and a list of the IDs of deleted rows. All deltastore operations are handled by the system and not visible directly to the user. Deltastore and columnstore data is combined when queried.

A delta rowgroup is used to store columnstore indexes until there are enough to store in the columnstore. Once the maximum number of rows is reached, the delta rowgroup is closed, and a background process detects, compresses, and writes the delta rowgroup into the columnstore.

There may be more than one delta rowgroup. All delta rowgroups are described as the deltastore. While loading data, anything less than 102,400 rows will be kept in the deltastore until they group to the maximum size and are written to the columnstore.

Batch mode execution is used during a query to process multiple rows at once. Loading a clustered columnstore index and the deltastore are shown in *Figure 1.2*. **Figure 1.2: Loading a clustered columnstore index**

Further information can be found here: https://docs.microsoft.com/en-us/sql/ relational-databases/indexes/get-started-with-columnstore-for-real-time operational-analytics?view=sql-server-2017.

Disk-based tables versus memory-optimized tables | **5**

**Adding Clustered Columnstore Indexes to memory-optimized tables**

When using a memory-optimized table, add a non-clustered columnstore index. A clustered columnstore index is especially useful for running analytics on a transactional table.

A clustered columnstore index can be added to an existing memory-optimized table, as shown in the following code snippet:

-- Add a clustered columnstore index to a memory-optimized table

ALTER TABLE MyMemOpttable

ADD INDEX MyMemOpt\_ColIndex clustered columnstore

**Disk-based tables versus memory-optimized tables** There are several differences between memory-optimized and disk-based tables.

One difference is the fact that, in a disk-based table, rows are stored in 8k pages and a page only stores rows from a single table. With memory-optimized tables, rows are stored individually, such that one data file can contain rows from multiple memory optimized tables.

Indexes in a disk-based table are stored in pages just like data rows. Index changes are logged, as are data row changes. A memory-optimized table persists the definition of the index but is regenerated each time the memory-optimized table is loaded, such as restarting the database. No logging of index "pages" is required.

Data operations are much different. With a memory-optimized table, all operations are done in memory. Log records are created when an in-memory update is performed. Any log records created in-memory are persisted to disk through a separate thread. Disk-based table operations may perform in-place updates on non-key-columns, but key-columns require a delete and insert. Once the operation is complete, changes are flushed to disk.

With disk-based tables, pages may become fragmented. As changes are made, there may be partially filled pages and pages that are not consecutive. With memory optimized tables, storing as rows removes fragmentation, but inserts, deletes, and updates will leave rows that can be compacted. Compaction of the rows is executed by means of a merge thread in the background.

**6** | Optimizing for performance, scalability and real‑time insights

Additional information can be found at this Microsoft docs link:

https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/ comparing-disk-based-table-storage-to-memory-optimized-table-storage?view=sql server-2017.

**In-memory OLTP**

In-memory **on-line transaction processing** (**OLTP**) is available in Microsoft SQL Server for optimizing the performance of transaction processing. In-memory OLTP is also available for all premium Azure SQL databases. While dependent on your application, performance gains of 2-30x have been observed.

Most of the performance comes from removing lock and latch contention between concurrently executing transactions and is optimized for in-memory data. Although performed in-memory, changes are logged to disk so that once committed, the transaction is not lost even if the machine should fail.

To fully utilize in-memory OLTP, the following features are available: • Memory-optimized tables are declared when you create the table.

• Non-durable tables, basically in-memory temporary tables for intermediate results, are not persisted so that they do not use any disk I/O. A non-durable table is declared with **DURABILITY=SCHEMA\_ONLY**.

• Table values and table-valued parameters can be declared as in-memory types as well.

• Natively compiled stored procedures, triggers, and scalar user-defined functions are compiled when created and avoid having to compile them at execution time, thereby speeding up operations.

Additional information can be found at the following links:

• https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/ in-memory-oltp-in-memory-optimization?view=sql-server-2017

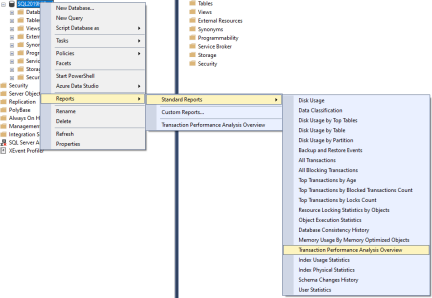
• https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/ survey-of-initial-areas-in-in-memory-oltp?view=sql-server-2017

Planning data migration to memory-optimized tables | **7**

**Planning data migration to memory-optimized tables**

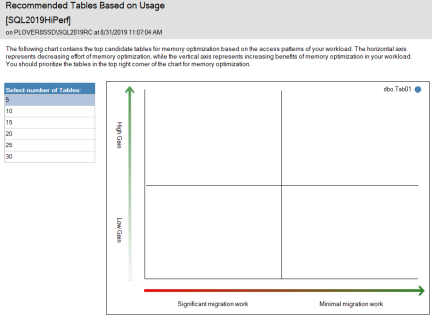
Microsoft **SQL Server Management Studio** (**SSMS**) contains tools to help analyze and migrate tables to memory-optimized storage.

When you right-click on a database in SSMS and click on **Reports** | **Standard Reports** | **Transaction Performance Analysis Overview**, a four-quadrant report of all tables in the database will be made:

**Figure 1.3: Choosing Transaction Performance Analysis**

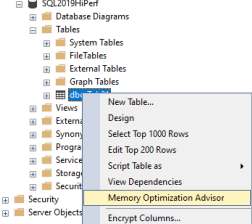
**8** | Optimizing for performance, scalability and real‑time insights

The report will look at each table and place it on the chart to show the ease of migration versus the expected gain by migrating the table to be memory-optimized:

**Figure 1.4: Recommended Tables Based on Usage**

Planning data migration to memory-optimized tables | **9**

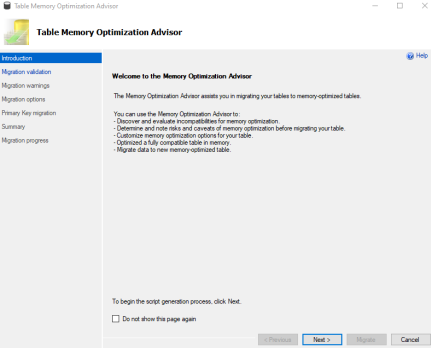
Once you have identified tables that might benefit, you can right-click on individual tables and run the Memory Optimization Advisor:



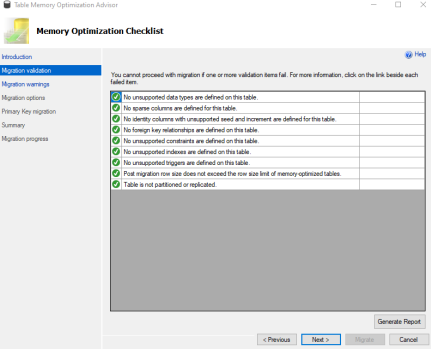
**Figure 1.5: Selecting the Memory Optimization Advisor**

**10** | Optimizing for performance, scalability and real‑time insights

The **Table Memory Optimization Advisor** is a "wizard" style of user interface that will step you through the configurations:

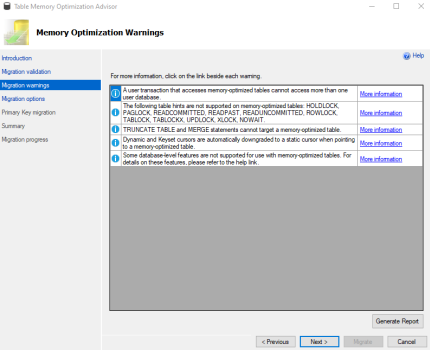
**Figure 1.6: The Table Memory Optimization Advisor dialogue**

Planning data migration to memory-optimized tables | **11**

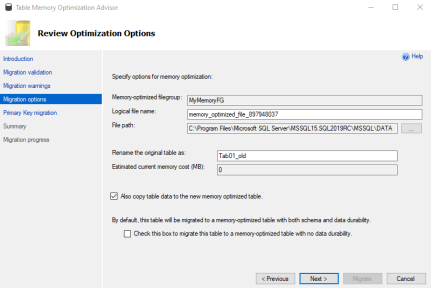
The wizard will take you through a checklist with any failed issues: **Figure 1.7: Memory Optimization Checklist**

**12** | Optimizing for performance, scalability and real‑time insights

The warnings dialogue will flag up other important issues.

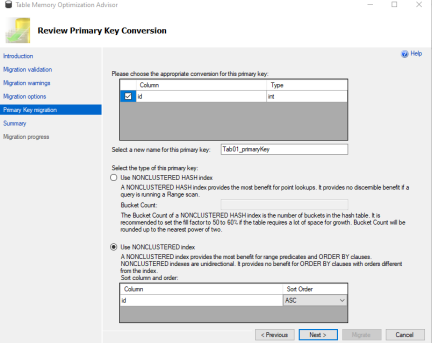
**Figure 1.8: Memory Optimization Warnings**

Planning data migration to memory-optimized tables | **13**

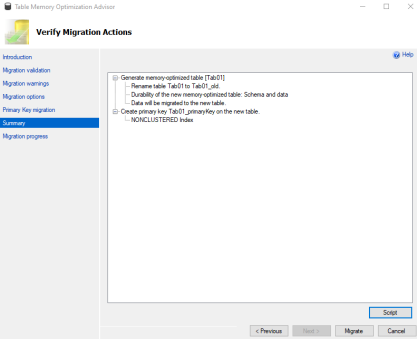
Next enter file names and check paths in the migration option dialogue. **Figure 1.9: Review Optimization options**

**14** | Optimizing for performance, scalability and real‑time insights

The wizard will detect the primary keys and populates the list of columns based on the primary key metadata. To migrate to a durable memory-optimized table, a primary key needs to be created. If there is no primary key and the table is being migrated to a non-durable table, the wizard will not show this screen.

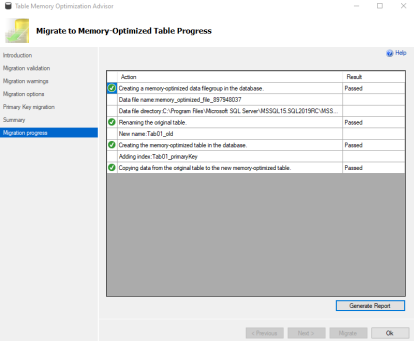
**Figure 1.10: Review Primary Key Conversion**

Planning data migration to memory-optimized tables | **15**

By clicking **Script** you can generate a Transact-SQL script in the summary screen. **Figure 1.11: Verify Migration Actions Summary Screen**

**16** | Optimizing for performance, scalability and real‑time insights

The wizard will the display a report as the table migrates.

**Figure 1.12: Migration progress report**

Memory-optimized tables are a great feature, but you will need to plan carefully to make sure you get the performance and transactional reliability you require.

You can create a new database specifying memory-optimized, or alter an existing database to handle memory-optimized data. In either case, a filegroup for containing the memory-optimized data must be created.

Planning data migration to memory-optimized tables | **17**

In the following sample, we will create a memory-optimized database using SQL script: -- Create Memory-Optimized Database

USE MASTER;

GO

CREATE DATABASE MemOptDB

ON (Name = MemOptDB\_Data, FileName = 'c:\sqldata\memoptdb\_data.mdf', size = 10 mb, maxsize = 20 mb, filegrowth = 5 mb)

  LOG ON (Name = MemOptDB\_Log, FileName = 'c:\sqldata\memoptdb\_log.ldf', size = 2 mb, maxsize = 10 mb, filegrowth = 1 mb);

GO

-- Must declare a memory-optimized filegroup

ALTER DATABASE MemOptDB

ADD FILEGROUP MemOptDB\_FG contains MEMORY\_OPTIMIZED\_DATA;

ALTER DATABASE MemOptDB

  ADD FILE (Name = 'MemOptDB\_MOFG', FileName = 'c:\sqldata\memoptdb\_mofg') TO FILEGROUP MemOptDB\_FG;

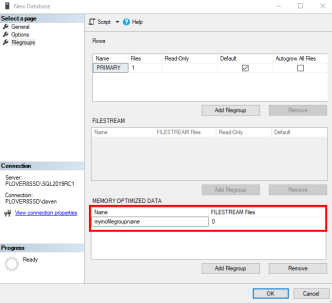
ALTER DATABASE MemOptDB

SET MEMORY\_OPTIMIZED\_ELEVATE\_TO\_SNAPSHOT = ON;

GO

**18** | Optimizing for performance, scalability and real‑time insights

You can also make a memory-optimized database by using SQL Server Management Studio and adding a memory-optimized filegroup:



**Figure 1.13: The new database dialogue window**

**Natively compiled stored procedures**

Natively compiled stored procedures are compiled when created and bypass the query execution engine. The procedure is compiled when created, and also manually or when the database or server are restarted.

A few additional concepts are introduced here, including **SCHEMABINDING** and **BEGIN ATOMIC**, both of which are required for natively compiled stored procedures.

Natively compiled stored procedures | **19**

SCHEMABINDING locks the table definition to prevent alteration after the stored procedure is created. SCHEMABINDING allows the compiled stored procedure to be certain of the data types involved. The tables involved in the natively compiled stored procedure cannot be altered without dropping the SCHEMABINDING, making changes and then reapplying the SCHEMABINDING. SHEMABINDING also requires that explicit field names are used in the query; "**select \*…**" will not work.

BEGIN ATOMIC is required in a natively compiled stored procedure and is only available for a natively compiled stored procedure. In interactive (non-natively compiled) procedures, you would use a BEGIN TRAN statement block. Using the ATOMIC block and transaction settings will be independent of the current connection/settings as the stored procedure may be used in different execution sessions.

If there is an existing active transaction, BEGIN ATOMIC will set a save point and roll back to that if it fails. Otherwise, a new transaction is created and completed or rolled back.

You indicated a natively compiled stored procedure in the create declaration of the stored procedure using the "NATIVE\_COMPILATION" directive.

In the following sample, we will create a memory-optimized table and a natively stored procedure. Note that memory-optimized tables cannot have clustered indexes. Memory-optimized tables are stored as rows, not in pages, as with a disk-based table:

-- Create Memory-Optimized Table

USE MemOptDB;

GO

CREATE TABLE dbo.MyMemOptTable

(

id int not null,

dtCreated datetime not null,

**20** | Optimizing for performance, scalability and real‑time insights

  orderID nvarchar(10) not null

  CONSTRAINT pk\_id PRIMARY KEY NONCLUSTERED (id)

  )

  WITH (MEMORY\_OPTIMIZED = ON, DURABILITY = SCHEMA\_AND\_DATA) GO

-- Create Natively Stored Procedure

CREATE PROCEDURE dbo.myNativeProcedure (@id int)

WITH NATIVE\_COMPILATION, SCHEMABINDING

AS BEGIN ATOMIC WITH ( TRANSACTION ISOLATION LEVEL = SNAPSHOT, LANGUAGE = N'us\_english' )

SELECT id, dtCreated, orderID

FROM dbo.MyMemOptTable

WHERE id = @id

END

GO

The table schema is locked due to the reference to a natively compiled stored procedure. If you try to alter the table, an exception will be thrown, as shown here:

-- Try to alter the schema!

ALTER TABLE [dbo].[MyMemOpttable]

  ALTER COLUMN orderId nvarchar(20)

GO

Msg 5074, Level 16, State 1, Line 55

The object 'myNativeProcedure' is dependent on column 'orderId'. Msg 4922, Level 16, State 9, Line 55

ALTER TABLE ALTER COLUMN orderId failed because one or more objects access this column.

TempDB enhancements | **21**

More information on natively compiled procedures can be found here:

https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/ creating-natively-compiled-stored-procedures?view=sql-server-2017.

**TempDB enhancements**

We have introduced another scalability enhancement with memory-optimized TempDB metadata. Historically, TempDB metadata contention has been a bottleneck to scalability for workloads running on SQL Server.

The system tables used for managing temp table metadata can be moved into latch-free non-durable memory-optimized tables.

**Enabling memory-optimized TempDB metadata**

Enabling this feature in SQL Server is a two-step process:

• First, alter the server configuration with T-SQL

• Restart the service

ALTER SERVER CONFIGURATION SET MEMORY\_OPTIMIZED tempdb\_METADATA = ON

The following T-SQL command can be used to verify whether **tempdb** is memory optimized:

SELECT SERVERPROPERTY('IsTempdbMetadataMemoryOptimized')

**Limitations of memory-optimized TempDB metadata**

There are a few limitations associated with using this new feature.

• Toggling the feature on and off requires a service restart.

• A single transaction may not access memory-optimized tables in more than one database. This means that any transactions that involve a memory-optimized table in a user database will not be able to access TempDB System views in the same transaction. If you attempt to access TempDB system views in the same transaction as a memory-optimized table in a user database, you will receive the following error:

A user transaction that accesses memory-optimized tables or natively compiled modules cannot access more than one user database or databases model and msdb, and it cannot write to master.

**22** | Optimizing for performance, scalability and real‑time insights

• Queries against memory-optimized tables do not support locking and isolation hints, so queries against memory-optimized TempDB catalog views will not honor locking and isolation hints. As with other system catalog views in SQL Server, all transactions against system views will be in READ COMMITTED (or, in this case, READ COMMITTED SNAPSHOT) isolation.

• There may be some issues with columnstore indexes on temporary tables when memory-optimized TempDB metadata is enabled. It is best to avoid columnstore indexes on temporary tables when using memory-optimized TempDB metadata.

**Intelligent Query Processing**

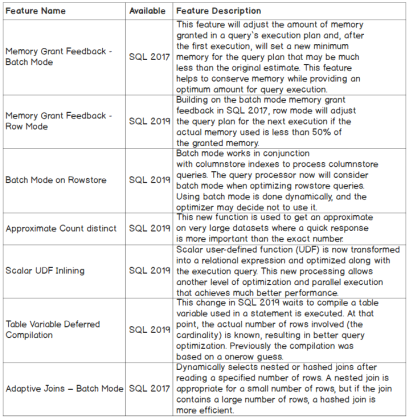
**Intelligent Query Processing** (**IQP**) is a family of features that were introduced in Microsoft SQL Server 2017 as adaptive query processing and has been expanded with new features in Microsoft SQL Server 2019. By upgrading to SQL Server 2019 and with compatibility level 150, most workloads will see performance improvements due to added intelligence in the query optimizer.

Intelligent Query Processing features are automatically enabled based on the "COMPATIBLITY\_LEVEL" of the database. To take advantage of the latest IQP features, set the database compatibility to 150.

Most of these are also available in Azure SQL, but it is best to check current documentation on exactly what is available there as this changes.

Intelligent Query Processing | **23**

The following table summarizes some of the IQP features.

**Table 1.14: Table summarizing IQP features**

• These features can be disabled and monitored.

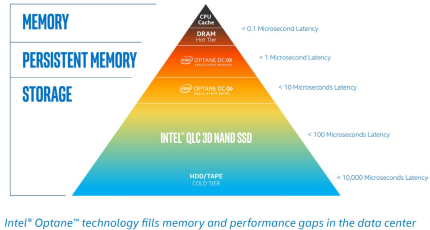
• For more information, refer to https://docs.microsoft.com/en-us/sql/relational databases/performance/intelligent-query-processing?view=sql-server-2017.

**24** | Optimizing for performance, scalability and real‑time insights

**Hybrid Buffer Pool**

Microsoft SQL Server 2019 introduces Hybrid Buffer Pool. This feature allows access to **Persistent MEMory** (**PMEM**) devices. These persistent memory devices add a new layer to server memory hierarchy and filling the gap between high performance / high cost of DRAM (Dynamic Random Access Memory) and the lower cost lower performance of file storage drives using SSD.

This memory architecture has been implemented by Intel as Intel® Optane™ Technology; refer to https://www.intel.com/content/www/us/en/products/docs/ storage/optane-technology-brief.html for more information:

**Figure 1.15: Intel memory architecture**

Persistent memory is integrated at the memory controller level of the CPU chip and will retain data even when the server is powered off.

While many aspects of persistent memory devices can be realized without any software changes, features such as Hybrid Buffer Pool can take advantage of the new storage hierarchy and provide direct memory access to files.

For clean database pages, those that have not been modified, SQL server can directly access them as memory. When an update is made, and then marked as dirty, the page is copied to DRAM, changes persisted, and the page is then written back into the persistent memory area.

Query Store | **25**

To enable Hybrid Buffer Pool, the feature must be enabled at the instance level of SQL Server. It is off by default. After enabling, the instance must be restarted:

ALTER SERVER CONFIGURATION

SET MEMORY\_OPTIMIZED HYBRID\_BUFFER\_POOL = ON;

Furthermore, the Hybrid Buffer Pool will only operate on memory-optimized databases: ALTER DATABASE <databaseName> SET MEMORY\_OPTIMIZED = ON;

Or, in order to disable, execute the following command:

ALTER DATABASE <databaseName> SET MEMORY\_OPTIMIZED = OFF;

To see the Hybrid Buffer Pool configurations and memory-optimized databases on an instance, you can run the following queries:

SELECT \* FROM sys.configurations WHERE name = 'hybrid\_buffer\_pool';

SELECT name, is\_memory\_optimized\_enabled FROM sys.databases;

There are many considerations when configuring a server with persistent memory, including the ratio of DRAM to PMEM. You can read more here:

• https://docs.microsoft.com/en-us/windows-server/storage/storage-spaces/ deploy-pmem

• https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/ hybrid-buffer-pool?view=sql-server-2017

**Query Store**

The Query Store in SQL Server, first introduced in SQL Server 2016, streamlines the process of troubleshooting query execution plans. The Query Store, once enabled, automatically captures query execution plans and runtime statistics for your analysis. You can then use the **sys.dm\_db\_tuning\_recommendations** view to discover where query execution plan regression has occurred and use the stored procedure, **sp\_query\_store\_ force\_plan**, to force a specific plan that performs better.

In SQL Server 2019, we now have made some additional enhancements to the default Query Store features. In this section, we will discuss the following topics:

• Changes to default parameter values when enabling Query Store • A new **QUERY\_CAPTURE\_MODE** custom

• Support for fast forward and static cursors

**26** | Optimizing for performance, scalability and real‑time insights

You can configure Query Store with SQL Server Management Studio (SSMS) or with T-SQL statements. SSMS configuration includes turning it on and off by setting the operation mode (off, read-only, or read/write), the Query Store size, and other settings. You can find Query Store parameters in the properties of a database by right-clicking on the database and selecting Query Store:

**Figure 1.16: Database properties dialogue window**

Query Store | **27**

**Changes to default parameter values**

Two of the existing parameters have new default values compared to SQL Server 2017. These parameters are **MAX\_STORAGE\_SIZE\_MB** and **QUERY\_CAPTURE\_MODE**. The new default values as of SQL Server 2019 are listed here:

• **MAX\_STORAGE\_SIZE\_MB** has a default value of 1000 (MB)

• The **QUERY\_CAPTURE\_MODE** has a default value of AUTdO

**QUERY\_CAPTURE\_MODE**

In previous versions of SQL Server, the default value for the **QUERY\_CAPTURE\_MODE** was set to ALL, and therefore all query plans were captured and stored. As mentioned in the previous section, the default value has now been changed to **AUTO**.

Setting the **QUERY\_CAPTURE\_MODE** to **AUTO** means that no query plans or associated runtime statistics will be captured for the first 29 executions in a single day. Query plans and runtime statistics are not captured until the 30th execution of a plan. This default setting can be changed by using the new custom mode.

**QUERY\_CAPTURE\_MODE: CUSTOM**

Before 2019, there were three available values for the **query\_capture\_mode;** those values were **NONE**, **ALL**, and **AUTO**. We have now added a fourth option, which is **CUSTOM**.

The **CUSTOM** mode provides you with a mechanism for changing the default settings of the Query Store. For example, the following settings can be modified when working in CUSTOM mode:

• **EXECUTION\_COUNT**

• **TOTAL\_COMPILE\_CPU\_TIME\_MS**

• **TOTAL\_EXECUTION\_CPU\_TIME\_MS**

• **STALE\_CAPTURE\_POLICY\_THRESHOLD**

**28** | Optimizing for performance, scalability and real‑time insights

First, you can verify and validate the current Query Store settings by using the **sys. database\_query\_store\_options** view:

SELECT actual\_state\_desc, stale\_query\_threshold\_days, query\_capture\_mode\_ desc,

capture\_policy\_execution\_count, capture\_policy\_total\_compile\_cpu\_time\_ms, capture\_policy\_total\_execution\_cpu\_time\_ms

FROM sys.database\_query\_store\_options

The output is as follows:

**Figure 1.17: Verifying and validating the Query Store settings**

To modify the default settings, you will first change the query capture mode to custom and then apply changes to the default values. Look at the following code by way of an example:

ALTER DATABASE AdventureWorks2017

SET QUERY\_STORE = ON

(

QUERY\_CAPTURE\_MODE = CUSTOM, QUERY\_CAPTURE\_POLICY =

(

EXECUTION\_COUNT = 20,

TOTAL\_COMPILE\_CPU\_TIME\_MS = 1000,

TOTAL\_EXECUTION\_CPU\_TIME\_MS = 100,

    STALE\_CAPTURE\_POLICY\_THRESHOLD = 7 DAYS

  )

);

Automatic tuning | **29**

The output is as follows:

**Figure 1.18: Modifying the default settings**

**Support for FAST\_FORWARD and STATIC Cursors**

We have added another exciting update to the Query Store. You can now force query execution plans for fast forward and static cursors. This functionality supports T-SQL and API cursors. Forcing execution plans for fast forward and static cursors is supported through SSMS or T-SQL using **sp\_query\_store\_force\_plan**.

**Automatic tuning**

Automatic tuning identifies potential query performance problems, recommends solutions, and automatically fixes problems identified.

By default, automatic tuning is disabled and must be enabled. There are two automatic tuning features available:

• Automatic plan correction

• Automatic index management

**Automatic plan correction**

To take advantage of automatic plan correction, the Query Store must be enabled on your database. Automatic plan correction is made possible by constantly monitoring data that is stored by the Query Store.

Automatic plan correction is the process of identifying regression in your query execution plans. Plan regression occurs when the SQL Server Query Optimizer uses a new execution plan that performs worse than the previous plan. To identify plan regression, the Query Store captures compile time and runtime statistics of statements being executed.

**30** | Optimizing for performance, scalability and real‑time insights

The database engine uses the data captured by the Query Store to identify when plan regression occurs. More specifically, to identify plan regression and take necessary action, the database engine uses the **sys.dm\_db\_tuning\_recommendations** view. This is the same view you use when manually determining which plans have experienced regressions and which plans to force.

When plan regression is noticed, the database engine will force the last known good plan.

The great news is that the database engine doesn't stop there; the database engine will monitor the performance of the forced plan and verify that the performance is better than the regressed plan. If the performance is not better, then the database engine will unforce the plan and compile a new query execution plan.

**Enabling automatic plan correction**

Automatic plan correction is disabled by default. The following code can be used to verify the status of automatic plan correction on your database:

SELECT name, desired\_state\_desc, actual\_state\_desc

FROM sys.database\_automatic\_tuning\_options

The output is as follows:



**Figure 1.19: Automatic plan correction is turned off**

You enable automatic plan correction by using the following code: ALTER DATABASE current

SET AUTOMATIC\_TUNING ( FORCE\_LAST\_GOOD\_PLAN = ON )

Automatic tuning | **31**

If you have not turned the Query Store on, then you will receive the following error: **Figure: 1.20: Error report if the Query Store is off**

**Automatically forced plans**

The database engine uses two criteria to force query execution plans: • Where the estimated CPU gain is higher than 10 seconds

• The number of errors in the recommended plan is lower than the number of errors in the new plan

Forcing execution plans improves performance where query execution plan regression has occurred, but this is a temporary solution, and these forced plans should not remain indefinitely. Therefore, automatically forced plans are removed under the following two conditions.

• Plans that are automatically forced by the database engine are not persisted between SQL Server restarts.

• Forced plans are retained until a recompile occurs, for example, a statistics update or schema change.

The following code can be used to verify the status of automatic tuning on the database: SELECT name, desired\_state\_desc, actual\_state\_desc

FROM sys.database\_automatic\_tuning\_options;



**Figure 1.21: Verifying the status of automatic tuning on the database**

**32** | Optimizing for performance, scalability and real‑time insights

**Lightweight query profiling**

**Lightweight query profiling** (**LWP**) provides DBAs with the capability to monitor queries in real time at a significantly reduced cost of the standard query profiling method. The expected overhead of LWP is at 2% CPU, as compared to an overhead of 75% CPU for the standard query profiling mechanism.

For a more detailed explanation on the query profiling infrastructure, refer to https:// docs.microsoft.com/en-us/sql/relational-databases/performance/query-profiling infrastructure?view=sqlallproducts-allversions.

**New functionality in 2019**

In SQL Server 2019, we have now improved LWP with new features and enhancements to the existing capabilities.

• In SQL Server 2016 and 2017, lightweight query profiling was deactivated by default and you could enable LWP at the instance level by using trace flag **7412**. In 2019, we have now turned this feature ON by default.

• You can also now manage this at the database level through Database Scoped Configurations. In 2019, you have a new database scoped configuration, **lightweight\_query\_profiling**, to enable or disable the **lightweight\_query\_ profiling** infrastructure at the database level.

• We have also introduced a new extended event. The new **query\_post\_execution\_ plan\_profile** extended event collects the equivalent of an actual execution plan based on lightweight profiling,unlike **query\_post\_execution\_showplan**, which uses standard profiling.

• We also have a new DMF **sys.dm\_exec\_query\_plan\_stats;** this DMF returns the equivalent of the last known actual execution plan for most queries, based on lightweight profiling.

Lightweight query profiling | **33**

The syntax for **sys.dm\_exec\_query\_plan\_stats** is as follows:

sys.dm\_exec\_query\_plan\_stats(plan\_handle)

For a more detailed analysis, refer to this online documentation: https://docs.microsoft. com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm exec-query-plan-stats-transact-sql?view=sql-server-2017.

**sys.database\_scoped\_configurations**

If you are not certain of the current status of LWP, you can use the following code to check the status of your database scoped configurations. The value column is 1; therefore, using the sys.database\_scoped\_configurations view, you see that Query Plan Stats is currently enabled:

SELECT \* FROM sys.database\_scoped\_configurations

WHERE name = 'LAST\_QUERY\_PLAN\_STATS'

The output is as follows:

**Figure 1.22: Check the status of the database scoped configurations**

To enable or disable LWP, you will use the database scoped configuration lightweight\_ query\_profiling. Refer to the following example:

ALTER DATABASE SCOPED CONFIGURATION

SET LIGHTWEIGHT\_QUERY\_PROFILING = OFF;

**34** | Optimizing for performance, scalability and real‑time insights

**Activity monitor**

With LWP enabled, you can now look at active expensive queries in the activity monitor. To launch the activity monitor, right-click on the instance name from SSMS and select Activity Monitor. Below Active Expensive Queries, you will see currently running queries, and if you right-click on an active query, you can now examine the Live Execution Plan!



**Figure 1.23: The activity monitor**

**Columnstore stats in DBCC CLONEDATABASE**

**DBCC CLONEDATABASE** creates a clone of the database that contains a copy of the schema and statistics for troubleshooting and diagnostic purposes. More specifically, with **DBCC CLONEDATABASE**, you have a lightweight, minimally invasive way to investigate performance issues related to the query optimizer. In SQL Server 2019, we now extend the capabilities of **DBCC CLONEDATABASE** by adding support for columnstore statistics.

Columnstore stats in DBCC CLONEDATABASE | **35**

**Columnstore statistics support**

In SQL Server 2019, support has been added for columnstore statistics. Before SQL Server 2019, manual steps were required to capture these statistics (refer to the following link). We now automatically capture stats blobs, and therefore, these manual steps are no longer required:

https://techcommunity.microsoft.com/t5/SQL-Server/Considerations-when-tuning your-queries-with-columnstore-indexes/ba-p/385294.

**DBCC CLONEDATABASE validations**

DBCC CLONEDATABASE performs the following validation checks. If any of these checks fail, the operation will fail, and a copy of the database will not be provided.

• The source database must be a user database.

• The source database must be online or readable.

• The clone database name must not already exist.

• The command must not be part of a user transaction.

**Understanding DBCC CLONEDATABASE syntax**

DBCC CLONEDATABASE syntax with optional parameters:

DBCC CLONEDATABASE

(

    source\_database\_name, target\_database\_name

)

[ WITH { [ NO\_STATISTICS ] [ , NO\_QUERYSTORE ]

[ , VERIFY\_CLONEDB | SERVICEBROKER ] [ , BACKUP\_CLONEDB ] } ]

The following T-SQL script will create a clone of the existing database. The statistics and Query Store data are included automatically.

DBCC CLONEDATABASE ('Source', 'Destination');

**36** | Optimizing for performance, scalability and real‑time insights

The following messages are provided upon completion:

**Figure 1.24: Cloned database output**

To exclude statistics, you rewrite the code to include **WITH NO\_STATISTICS**: DBCC CLONEDATABASE ('Source', 'Destination\_NoStats')

WITH NO\_STATISTICS;

To exclude statistics and Query Store data, execute the following code: DBCC CLONEDATABASE ('Source', 'Destination\_NoStats\_NoQueryStore') WITH NO\_STATISTICS, NO\_QUERYSTORE;

**Making the clone database production-ready**

Thus far, the database clones provisioned are purely for diagnostic purposes. The option **VERIFY\_CLONEDB** is required if you want to use the cloned database for production use. **VERIFY\_CLONEDB** will verify the consistency of the new database.

For example:

DBCC CLONEDATABASE ('Source', 'Destination\_ProdReady')

WITH VERIFY\_CLONEDB;

The output is as follows:

**Figure 1.25: Verifying the cloned database**

**Estimate compression for Columnstore Indexes**

The stored procedure **sp\_estimate\_data\_compression\_savings** estimates the object size for the requested compression state. Furthermore, you can evaluate potential compression savings for whole tables or parts of tables; we will discuss the available options shortly. Prior to SQL Server 2019, you were unable to use **sp\_estimate\_data\_ compression\_savings** for columnstore indexes and, thus, we were unable to estimate compression for columnstore or **columnstore\_archive**.

Estimate compression for Columnstore Indexes | **37**

We have extended the capability for **sp\_estimate\_data\_compression\_savings** to include support for **COLUMNSTORE** and **COLUMNSTORE\_ARCHIVE**.

**sp\_estimate\_data\_compression\_savings Syntax**

Look at the following T-SQL syntax:

sp\_estimate\_data\_compression\_savings

     [ @schema\_name = ] 'schema\_name'

   , [ @object\_name = ] 'object\_name'

, [@index\_id = ] index\_id

, [@partition\_number = ] partition\_number

, [@data\_compression = ] 'data\_compression'

[;]

The following argument descriptions are provided by docs.microsoft.com: https:// docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/ sp-estimate-data-compression-savings-transact-sql?view=sql-server-2017.

**Table 1.26: Description of the arguments**

**38** | Optimizing for performance, scalability and real‑time insights

There are currently eight available outputs; you will primarily focus on the four outputs related to size.

Output:

object\_name

schema\_name

index\_id

partition\_number

size\_with\_current\_compression\_setting (KB)

size\_with\_requested\_compression\_setting (KB)

sample\_size\_with\_current\_compression\_setting (KB)

sample\_size\_with\_current\_requested\_setting (KB)

The following is an example of the procedure in action, followed by a comparison of the space savings for page and columnstore compression:

EXEC sp\_estimate\_data\_compression\_savings

    @schema\_name = 'dbo',

    @object\_name = 'MySourceTable',

@index\_id = NULL,

@partition\_number = NULL,

@data\_compression = 'PAGE'

Example with PAGE Compression:

**Figure 1.27: PAGE Compression**

EXEC sp\_estimate\_data\_compression\_savings

    @schema\_name = 'dbo',

    @object\_name = 'MySourceTable',

@index\_id = NULL,

@partition\_number = NULL,

@data\_compression = 'COLUMNSTORE'

Troubleshooting page resource waits | **39**

Example with COLUMNSTORE compression:

**Figure 1.28: COLUMNSTORE compression**

In this example, page compression has estimated space savings of roughly 45%, and columnstore compression has estimated space savings of 68%.

**Troubleshooting page resource waits**

A new and exciting feature in SQL Server 2019 is sys.dm\_db\_page\_info. This new **dynamic management function** (**DMF**) retrieves useful page information, such as **page\_ id**, **file\_id**, **index\_id**, **object\_id**, and **page\_type**, that can be used for troubleshooting and debugging performance issues in SQL Server. Historically, troubleshooting has involved the use of DBCC Page and the undocumented DMF **sys.dm\_db\_page\_ allocations**.

Unlike DBCC Page, which provides the entire contents of a page, **sys.dm\_db\_page\_info** only returns header information about pages. Fortunately, this will be sufficient for most troubleshooting and performance tuning scenarios.

This section will discuss the following topics:

• Database State permissions

• **sys.dm\_db\_page\_info** parameters

• New column page\_resource in (**sys.dm\_exec\_requests**, **sys.processes**) • **sys.fn\_PageResCracker**

**sys.dm\_db\_page\_info**

First, to leverage this new DMF, we require the VIEW DATABASE STATE permission. The following code can be used to provide access:

GRANT VIEW DATABASE STATE TO [login]

There are four required parameters:

sys.dm\_db\_page\_info ( DatabaseId, FileId, PageId, Mode )

**40** | Optimizing for performance, scalability and real‑time insights

The following argument descriptions are provided by docs.microsoft.com: **Table 1.29: The description of the arguments**

You can execute the function by itself if you have all the requisite parameters. The mode is set to **Limited** in this example, and this will return **NULL** values for all description columns:

SELECT OBJECT\_NAME(object\_id) as TableName,\*

FROM SYS.dm\_db\_page\_info(6, 1, 1368, 'Limited')

The output is as follows:

**Figure 1.30: Output with LIMITED mode**

Using the **Detailed** mode, you will get much more descriptive information than provided in the previous example. In this example, you can see that the **NULL** values have been replaced with descriptive information.

SELECT OBJECT\_NAME(object\_id) as TableName,\*

FROM SYS.dm\_db\_page\_info(6, 1, 1368, 'Detailed')

Troubleshooting page resource waits | **41**

The output is as follows:

**Figure 1.31: Output with Detailed mode**

To see a full list of all the columns returned, go to https://docs.microsoft.com/en-us/ sql/relational-databases/system-dynamic-management-views/sys-dm-db-page-info transact-sql?view=sqlallproducts-allversions.

**sys.fn\_pagerescracker**

In the previous example, you saw how to pass parameters to this new function manually. Fortunately, the parameters can be directly retrieved from **sys.dm\_exec\_ requests** or **sys.processes**. To make this work, we added a new column called page\_ resource. The page\_resource column returns the page ID, the file ID, and the database ID. It is also important to highlight that the new page\_resource column in **sys.dm\_exec\_ request** will be **NULL** when **WAIT\_RESOURCE** does not have a valid value.

However, the page\_resource column stores the data as an 8-byte hexadecimal value that needs to be converted. Therefore, we have added a new function called **sys.fn\_ pagerescracker**. This function returns the page ID, the file ID, and the database ID for the given page\_resource value.

It is important to note that we require the user to **have VIEW SERVER STATE** permission on the server to run **sys.fn\_PageResCracker**.

In this example, the page\_resource column is being passed into the **sys.fn\_ PageResCracker** function, and then the database ID, file ID, and Page ID are passed to sys.dm\_db\_page\_info:

SELECT OBJECT\_NAME(page\_info.object\_id) AS TableName,page\_info.\* FROM sys.dm\_exec\_requests AS d

CROSS APPLY sys.fn\_PageResCracker (d.page\_resource) AS r

CROSS APPLY sys.dm\_db\_page\_info(r.db\_id, r.file\_id, r.page\_id, 'Detailed') AS page\_info

**42** | Optimizing for performance, scalability and real‑time insights

The output is as follows:

**Figure 1.32: Page resource column is being passed into a function**

You can read more here: https://docs.microsoft.com/en-us/sql/relational-databases/ system-functions/sys-fn-pagerescracker-transact-sql?view=sql-server-2017.

**2**

**Enterprise Security**

Securing sensitive data and staying compliant with industry regulations such as **PCI-DSS** (**Payment Card Industry Data Security Standard**) and **GDPR** (**General Data Protection Regulation**) is very important. A compromised database system can lead to a loss of revenue, regulatory fines, and a negative impact on the reputation of your business.

Tracking compliance and maintaining database security requires significant admin resources. SQL Server 2019 has tools such as Data Discovery and Classification, and SQL Vulnerability Assessment tools that allow DBAs to identify compliance issues and tag and classify specific datasets to ensure compliance.

SQL Server 2019 offers many security features that address these challenges, such as TDE (Transparent Data Encryption), Always Encrypted, Auditing, Dynamic Data Masking and Row-Level Security.

**46** | Enterprise Security

Combined with further enhancements to certificate management in SQL Server 2019, support for TLS 1.2, and confidential computing initiatives such as secure enclaves, you can be sure that you can build and deploy solutions to the highest security standards while becoming GDPR and PCI-DSS compliant. All these features are also available within Azure SQL Database.

**SQL Data Discovery and Classification**

The Data Discovery and Classification feature enables you to identify, classify, and label data held across your SQL Server estate. The sheer volume of data now held within databases makes this a challenging process, coupled with the fact that regulatory mandates such as GDPR, SOX, and PCI demand that businesses protect sensitive data. So you can see how this feature will help. Before you can develop a security strategy for your SQL Server databases, it makes logical sense to know what data you hold, and from this you can then classify and label the more sensitive data and implement the relevant security controls, therefore minimizing potential sensitive data leaks.

Key components for this feature include two metadata attributes, labels and information types. Labels are used to define the sensitivity of data. Information types are used to provide additional granularity into the types of data stored in a column. As you can see in *Figure 2.1*, email addresses and phone numbers have been classified as contact information under the GDPR label.

**Figure 2.1: Classification confirmation**

SQL Data Discovery and Classification | **47**

To start the classification process, you will need to right-click on the database and find the **Data Discovery and Classification** option (*Figure 2.2*).

**Figure 2.2: Accessing the Classify Data... option from the menu**

While you are connected to the database via **SSMS** (**SQL Server Management Studio**), you can issue the following query to get a really good summary of the classification that has just taken place:

SELECT

schema\_name(O.schema\_id) AS schema\_name,

O.NAME AS table\_name,

C.NAME AS column\_name,

information\_type,

sensitivity\_label

FROM

(

SELECT

IT.major\_id,

**48** | Enterprise Security

IT.minor\_id,

IT.information\_type,

L.sensitivity\_label

FROM

(

SELECT

major\_id,

minor\_id,

value AS information\_type

FROM sys.extended\_properties

WHERE NAME = 'sys\_information\_type\_name'

) IT

FULL OUTER JOIN

(

SELECT

major\_id,

minor\_id,

value AS sensitivity\_label

FROM sys.extended\_properties

WHERE NAME = 'sys\_sensitivity\_label\_name'

) L

ON IT.major\_id = L.major\_id AND IT.minor\_id = L.minor\_id ) EP

JOIN sys.objects O

ON EP.major\_id = O.object\_id

JOIN sys.columns C

ON EP.major\_id = C.object\_id AND EP.minor\_id = C.column\_id **Figure 2.3: Successfully connected to the database**

SQL Data Discovery and Classification | **49**

You can delegate this to SQL Server and let it carry out a review of the data and an automatic implementation of the classification process.

**Figure 2.4: Classification changes been implemeted**

**Note**

With SQL Server 2019, is it not possible to use T-SQL to add metadata about the sensitivity classification, such as the following:

**ADD SENSITIVITY CLASSIFICATION TO**

**<object\_name> [, ...n ]**

**WITH ( <sensitivity\_label\_option> [, ...n ]**

This is only possible with Azure SQL Database.

**50** | Enterprise Security

Another advantage of this feature is the visibility of the classification states in the form of a report, which you can then export to different formats as required. This will benefit you regarding compliance and auditing. The following screenshot shows a copy of a report in Excel format:

**Figure 2.5: SQL Data Classification Report**

Once you understand your data via the classification processes, you can then leverage different features from SQL Server 2019, such as Always Encrypted or Data Masking, to protect these sensitive columns.

SQL Vulnerability Assessment | **51**

**SQL Vulnerability Assessment**

While we're thinking about a sound security strategy for SQL Server, it is important to address current security issues that exist within your database estate. Where should you start? What technical work is required to address the issues found? SQL Vulnerability Assessment is the tool for this task. It will allow you to improve your internal processes and harden your security across a dynamic and ever-changing database environment.

**Note**

Vulnerability Assessment is supported for SQL Server 2012 and later and requires SSMS 17.4+.

This feature carries out a scan against the database(s) using a pre-built knowledge base of rules that will flag security concerns such as elevated accounts and security misconfigurations. To start this assessment, you will need to right-click on the database and click on **Vulnerability Assessment** (as shown in the following screenshot) and start a scan:

**Figure 2.6: Accessing the vulnerabilities scan from the Tasks menu**