

OFFICIAL MICROSOFT LEARNING PRODUCT

20765B

Provisioning SQL Databases

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2017 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at <https://www.microsoft.com/en-us/legal/intellectualproperty/trademarks/en-us.aspx> are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

Product Number: 20765B

Part Number (if applicable): X21-31974

Released: 03/2017

Welcome!

Thank you for taking our training! We've worked together with our Microsoft Certified Partners for Learning Solutions and our Microsoft IT Academies to bring you a world-class learning experience—whether you're a professional looking to advance your skills or a student preparing for a career in IT.

- **Microsoft Certified Trainers and Instructors**—Your instructor is a technical and instructional expert who meets ongoing certification requirements. And, if instructors are delivering training at one of our Certified Partners for Learning Solutions, they are also evaluated throughout the year by students and by Microsoft.
- **Certification Exam Benefits**—After training, consider taking a Microsoft Certification exam. Microsoft Certifications validate your skills on Microsoft technologies and can help differentiate you when finding a job or boosting your career. In fact, independent research by IDC concluded that 75% of managers believe certifications are important to team performance¹. Ask your instructor about Microsoft Certification exam promotions and discounts that may be available to you.
- **Customer Satisfaction Guarantee**—Our Certified Partners for Learning Solutions offer a satisfaction guarantee and we hold them accountable for it. At the end of class, please complete an evaluation of today's experience. We value your feedback!

We wish you a great learning experience and ongoing success in your career!

Sincerely,

Microsoft Learning
www.microsoft.com/learning

Microsoft | Learning

¹ IDC, Value of Certification: Team Certification and Organizational Performance, November 2006

Acknowledgements

Microsoft Learning would like to acknowledge and thank the following for their contribution towards developing this title. Their effort at various stages in the development has ensured that you have a good classroom experience.

Aaron Johal – Content Developer

Aaron Johal is a Microsoft Certified Trainer who splits his time between training, consultancy, content development, contracting and learning. Since he moved into the non-functional side of the Information Technology business. He has presented technical sessions at SQL Pass in Denver and at sqlbits in London. He has also taught and worked in a consulting capacity throughout the UK and abroad, including Africa, Spain, Saudi Arabia, Netherlands, France, and Ireland. He enjoys interfacing functional and non-functional roles to try and close the gaps between effective use of Information Technology and the needs of the Business.

Ed Harper – Content Developer

Ed Harper is a database developer specializing in Microsoft SQL Server. Ed has worked with SQL Server since 1999, and has developed and designed transaction-processing and reporting systems for cloud security, telecommunications, and financial services companies.

John Daisley – Content Developer

John Daisley is a mixed vendor Business Intelligence and Data Warehousing contractor with a wealth of data warehousing and Microsoft SQL Server database administration experience. Having worked in the Business Intelligence arena for over a decade, John has extensive experience of implementing business intelligence and database solutions using a wide range of technologies and across a wide range of industries including airlines, engineering, financial services, and manufacturing.

Nick Anderson – Content Developer

Nick Anderson MBCS MISTC has been a freelance Technical Writer since 1987 and Trainer since 1999. Nick has written internal and external facing content in many business and technical areas including development, infrastructure and finance projects involving SQL Server, Visual Studio and similar tools. Nick provides services for both new and existing document processes from knowledge capture to publishing.

Geoff Allix – Technical Reviewer

Geoff Allix is a Microsoft SQL Server subject matter expert and professional content developer at Content Master—a division of CM Group Ltd. As a Microsoft Certified Trainer, Geoff has delivered training courses on SQL Server since version 6.5. Geoff is a Microsoft Certified IT Professional for SQL Server and has extensive experience in designing and implementing database and BI solutions on SQL Server technologies, and has provided consultancy services to organizations seeking to implement and optimize database solutions.

Lin Joyner – Technical Reviewer

Lin is an experienced Microsoft SQL Server developer and administrator. She has worked with SQL Server since version 6.0 and previously as a Microsoft Certified Trainer, delivered training courses across the UK. Lin has a wide breadth of knowledge across SQL Server technologies, including BI and Reporting Services. Lin also designs and authors SQL Server and .NET development training materials. She has been writing instructional content for Microsoft for over 15 years.

Contents

Module 1: SQL Server 2016 Components

Module Overview	1-1
Lesson 1: Introduction to the SQL Server Platform	1-2
Lesson 2: Overview of SQL Server Architecture	1-10
Lesson 3: SQL Server Services and Configuration Options	1-18
Module Review and Takeaways	1-23

Module 2: Installing SQL Server 2016

Module Overview	2-1
Lesson 1: Considerations for Installing SQL Server	2-2
Lesson 2: tempdb Files	2-11
Lesson 3: Installing SQL Server 2016	2-13
Lesson 4: Automating Installation	2-17
Lab: Installing SQL Server 2016	2-20
Module Review and Takeaways	2-25

Module 3: Upgrading SQL Server to SQL Server 2016

Module Overview	3-1
Lesson 1: Upgrade Requirements	3-2
Lesson 2: Upgrade of SQL Server Services	3-10
Lesson 3: Side-by-Side Upgrade: Migrating SQL Server Data and Applications	3-20
Lab: Upgrading SQL Server	3-31
Module Review and Takeaways	3-34

Module 4: Working with Databases

Module Overview	4-1
Lesson 1: Introduction to Data Storage with SQL Server	4-2
Lesson 2: Managing Storage for System Databases	4-9
Lesson 3: Managing Storage for User Databases	4-14
Lesson 4: Moving and Copying Database Files	4-25
Lesson 5: Configuring the Buffer Pool Extension	4-29
Lab: Managing Database Storage	4-33
Module Review and Takeaways	4-37

Module 5: Performing Database Maintenance

Module Overview	5-1
Lesson 1: Ensuring Database Integrity	5-2
Lesson 2: Maintaining Indexes	5-8
Lesson 3: Automating Routine Maintenance Tasks	5-18
Lab: Performing Ongoing Database Maintenance	5-21
Module Review and Takeaways	5-24

Module 6: Database Storage Options

Module Overview	6-1
Lesson 1: SQL Server Storage Performance	6-2
Lesson 2: SMB Fileshare	6-12
Lesson 3: SQL Server Storage in Microsoft Azure	6-15
Lesson 4: Stretch Database	6-19
Lab: Implementing Stretch Database	6-23
Module Review and Takeaways	6-25

Module 7: Planning to Deploy SQL Server on Microsoft Azure

Module Overview	7-1
Lesson 1: SQL Server on Virtual Machines and Azure SQL Database	7-2
Lesson 2: Azure Storage	7-11
Lesson 3: Azure SQL Server Authentication	7-15
Lesson 4: Deploying Databases in Azure SQL Database	7-20
Lab: Migrating SQL Server by Using Azure	7-23
Module Review and Takeaways	7-26

Module 8: Migrating Databases to Azure SQL Database

Module Overview	8-1
Lesson 1: Database Migration Testing Tools	8-2
Lesson 2: Database Migration Compatibility Issues	8-8
Lesson 3: Migrating an On-Premises Database to an Azure SQL Database	8-14
Lab: Migrating SQL Server with Azure	8-21
Module Review and Takeaways	8-23

Module 9: Deploying SQL Server on a Microsoft Azure Virtual Machine

Module Overview	9-1
Lesson 1: Deploying SQL Server on Azure Virtual Machines	9-2
Lesson 2: Migrating a Database to a Microsoft Azure Virtual Machine	9-10
Lab: Deploying SQL Server on an Azure Virtual Machine	9-16
Module Review and Takeaways	9-20

Module 10: Managing Databases in the Cloud

Module Overview	10-1
Lesson 1: Managing Security in Azure SQL Database	10-4
Lesson 2: Configuring Azure Storage	10-18
Lesson 3: Azure Automation	10-25
Lab: Managing Databases in the Cloud	10-33
Module Review and Takeaways	10-38

Lab Answer Keys

Module 2 Lab: Installing SQL Server 2016	L02-1
Module 3 Lab: Upgrading SQL Server	L03-1
Module 4 Lab: Managing Database Storage	L04-1
Module 5 Lab: Performing Ongoing Database Maintenance	L05-1
Module 6 Lab: Implementing Stretch Database	L06-1
Module 7 Lab: Migrating SQL Server by Using Azure	L07-1
Module 8 Lab: Migrating SQL Server with Azure	L08-1
Module 9 Lab: Deploying SQL Server on an Azure Virtual Machine	L09-1
Module 10 Lab: Managing Databases in the Cloud	L10-1

About This Course

This section provides a brief description of the course, audience, suggested prerequisites, and course objectives.

Course Description



Note: This B MOC version of the course has been developed on RTM software. It has had content added to cover SQL Azure that was not in the A MOC version.

This five-day instructor-led course provides students with the knowledge and skills to provision a Microsoft SQL Server 2016 database. The course covers SQL Server 2016 provision both on-premise and in Azure, and covers installing from new and migrating from an existing install.

Audience

The primary audience for this course is individuals who administer and maintain SQL Server databases. These individuals perform database administration and maintenance as their primary area of responsibility, or work in environments where databases play a key role in their primary job.

The secondary audiences for this course are individuals who develop applications that deliver content from SQL Server databases.

Student Prerequisites

This course requires that you meet the following prerequisites:

- Basic knowledge of the Microsoft Windows operating system and its core functionality.
- Working knowledge of Transact-SQL.
- Working knowledge of relational databases.
- Some experience with database design

These prerequisites can be achieved by attending course 20761A – Querying Data with Transact-SQL.

Course Objectives

After completing this course, students will be able to:

- Provision a Database Server
- Upgrade SQL Server
- Configure SQL Server
- Manage Databases and Files (shared)

Course Outline

The course outline is as follows:

Module 1, "SQL Server 2016 components"

Module 2, "Installing SQL server 2016"

Module 3, "Upgrading SQL Server to SQL Server 2016"

Module 4, "working with databases"

Module 5, "Performing Database Maintenance"

Module 6, "Database storage options"

Module 7, "Planning to Deploy SQL Server on Microsoft Azure"

Module 8, "Migrating Databases to Azure SQL Database"

Module 9, "Deploying SQL Server on a Microsoft Azure Virtual Machine"

Module 10, "Managing Databases in the Cloud"

Course Materials

The following materials are included with your kit:

- **Course Handbook:** a succinct classroom learning guide that provides the critical technical information in a crisp, tightly-focused format, which is essential for an effective in-class learning experience.
 - **Lessons:** guide you through the learning objectives and provide the key points that are critical to the success of the in-class learning experience.
 - **Labs:** provide a real-world, hands-on platform for you to apply the knowledge and skills learned in the module.
 - **Module Reviews and Takeaways:** provide on-the-job reference material to boost knowledge and skills retention.
 - **Lab Answer Keys:** provide step-by-step lab solution guidance.



Additional Reading: Course Companion Content on the

<http://www.microsoft.com/learning/en/us/companion-moc.aspx> **Site:** searchable, easy-to-browse digital content with integrated premium online resources that supplement the Course Handbook.

- **Modules:** include companion content, such as questions and answers, detailed demo steps and additional reading links, for each lesson. Additionally, they include Lab Review questions and answers and Module Reviews and Takeaways sections, which contain the review questions and answers, best practices, common issues and troubleshooting tips with answers, and real-world issues and scenarios with answers.
- **Resources:** include well-categorized additional resources that give you immediate access to the most current premium content on TechNet, MSDN®, or Microsoft® Press®.



Additional Reading: Student Course files on the

<http://www.microsoft.com/learning/en/us/companion-moc.aspx> **Site:** includes the Allfiles.exe, a self-extracting executable file that contains all required files for the labs and demonstrations.

- **Course evaluation:** at the end of the course, you will have the opportunity to complete an online evaluation to provide feedback on the course, training facility, and instructor.
 - To provide additional comments or feedback on the course, send an email to support@microsoft.com. To inquire about the Microsoft Certification Program, send an email to mcp@microsoft.com.

Virtual Machine Environment

This section provides the information for setting up the classroom environment to support the business scenario of the course.

Virtual Machine Configuration

In this course, you will use Microsoft® Hyper-V™ to perform the labs.



Note: At the end of each lab, you must revert the virtual machines to a snapshot. You can find the instructions for this procedure at the end of each lab

The following table shows the role of each virtual machine that is used in this course:

Virtual machine	Role
20765B-MIA-DC	Domain controller for the ADVENTUREWORKS domain.
20765B-MIA-SQL	SQL Server 2016 and SharePoint Server
20765B-MIA-DC-UPGRADE	Domain controller for the ADVENTUREWORKS domain.
20765B-MIA-SQL-UPGRADE	SQL Server 2014 and SharePoint Server

Software Configuration

The following software is installed:

- Microsoft Windows Server 2012 R2
- Microsoft SQL Server 2016
- Microsoft SQL Server 2014
- Microsoft Office 2016
- Microsoft SharePoint Server 2013
- Microsoft Visual Studio 2015
- Microsoft Visio 2013

Course Files

The files associated with the labs in this course are located in the D:\Labfiles folder on the 20765A-MIA-SQL virtual machine.

Classroom Setup

Each classroom computer will have the same virtual machine configured in the same way.

Course Hardware Level

To ensure a satisfactory student experience, Microsoft Learning requires a minimum equipment configuration for trainer and student computers in all Microsoft Certified Partner for Learning Solutions (CPLS) classrooms in which Official Microsoft Learning Product courseware is taught.

Hardware Level 6+

- Intel Virtualization Technology (Intel VT) or AMD Virtualization (AMD-V) processor
- Dual 120 GB hard disks 7200 RPM SATA or better*
- 8GB or higher
- DVD drive
- Network adapter with Internet connectivity
- Super VGA (SVGA) 17-inch monitor
- Microsoft Mouse or compatible pointing device
- Sound card with amplified speakers

*Striped

In addition, the instructor computer must be connected to a projection display device that supports SVGA 1024 x 768 pixels, 16 bit colors.

Module 1

SQL Server 2016 Components

Contents:

Module Overview	1-1
Lesson 1: Introduction to the SQL Server Platform	1-2
Lesson 2: Overview of SQL Server Architecture	1-10
Lesson 3: SQL Server Services and Configuration Options	1-18
Module Review and Takeaways	1-23

Module Overview

This module introduces the Microsoft® SQL Server® 2016 platform. It describes the components, editions, and versions of SQL Server 2016, and the tasks that a database administrator commonly performs to configure a SQL Server instance.

Objectives

At the end of this module, you should be able to:

- Describe SQL Server components, editions, and versions.
- Describe SQL Server architecture.
- Configure SQL Server services and networking.

Lesson 1

Introduction to the SQL Server Platform

As a DBA, it is important to be familiar with the database management system used to store your data. SQL Server is a platform for developing business applications that are data focused. Rather than being a single monolithic application, SQL Server is structured as a series of components. It is important to understand how you use each of these.

You can install more than one copy of SQL Server on a server. Each of these is referred to as an instance and can be configured and managed independently.

SQL Server ships in a variety of editions, each with a particular set of capabilities for different scenarios. It's important to understand the target business cases for each of the SQL Server editions—and how the evolution, through a series of improving versions over many years, results in today's stable and robust platform.

Lesson Objectives

After completing this lesson, you will be able to:

- Explain the role of each component that makes up the SQL Server platform.
- Describe the functionality that SQL Server instances provide.
- Explain the available SQL Server editions.
- Explain how SQL Server has evolved.

SQL Server Components

SQL Server includes a powerful relational database engine, but the scope of the SQL Server product is far broader than the database engine alone. It is a complete data platform built from a group of features offering a wide range of data processing functions:

- Not just a database engine
- Relational and business intelligence components

SQL Server Components	
Database Engine	Analysis Services
Integration Services	Reporting Services
Master Data Services	Data Quality Services
StreamInsight	Full-Text Search
Replication	PowerPivot
Power View	Polybase
R Services	

Component	Description
Database Engine	The SQL Server database engine is the heart of the SQL Server platform. It provides a high-performance, scalable relational database engine based on the SQL language that can be used to host online transaction processing (OLTP) databases for business applications and data warehouse solutions. SQL Server 2016 also includes a memory-optimized database engine that uses in-memory technology to improve performance for high-volume, short-running transactions.

Component	Description
Analysis Services	SQL Server Analysis Services (SSAS) is an online analytical processing (OLAP) engine that works with analytic cubes and tables. It's used to implement enterprise BI solutions for data analysis and data mining.
Integration Services	SQL Server Integration Services (SSIS) is an extract, transform, and load (ETL) platform tool for orchestrating the movement of data in both directions between SQL Server components and external systems.
Reporting Services	SQL Server Reporting Services (SSRS) is a reporting engine, based on web services, that provides a web portal and end-user reporting tools. It can be installed in native mode or integrated with Microsoft SharePoint® Server.
Master Data Services	SQL Server Master Data Services (MDS) provides tooling and a hub for managing master or reference data.
Data Quality Services	SQL Server Data Quality Services (DQS) is a knowledge-driven data quality tool for data cleansing and matching.
StreamInsight	SQL Server StreamInsight provides a platform for building applications that perform complex event processing for streams of real-time data.
Full-Text Search	Full-Text Search is a feature of the database engine that provides a sophisticated semantic search facility for text-based data.
Replication	The SQL Server database engine includes Replication, a set of technologies for synchronizing data between servers to meet data distribution needs.
PowerPivot for SharePoint	PowerPivot for SharePoint is a specialized implementation of SQL Server Analysis Services that you can install in a Microsoft SharePoint Server farm to enable tabular data modeling in shared Microsoft Excel® workbooks. PowerPivot is also available natively in Excel.
Power View for SharePoint	Power View for SharePoint is a component of SQL Server Reporting Services when using SharePoint-integrated mode. It provides interactive data exploration, visualization, and presentation experience that encourages intuitive, impromptu reporting. Power View is also available natively in Excel.
PolyBase	PolyBase is an extension to the database engine where you can query distributed datasets held in Hadoop or Microsoft Azure® Blob storage from Transact-SQL statements.
R Services	SQL Server R Services is an extension to the database engine where you can execute scripts written in the open source R language and access their results from Transact-SQL statements.

SQL Server Instances

It is sometimes useful to install more than one copy of a SQL Server component on a single server. Many SQL Server components can be installed more than once as separate instances.

SQL Server Instances

The ability to install multiple instances of SQL Server components on a single server is useful in various situations:

- You might want to have different administrators or security environments for sets of databases. You can manage and secure each instance of SQL Server separately.
- Some of your applications might require server configurations that are inconsistent or incompatible with the server requirements of other applications. You can configure each instance of SQL Server independently.
- Your application databases might need different levels of service, particularly regarding availability. You can use SQL Server instances to separate workloads with differing service level agreements (SLAs).
- You might need to support different versions or editions of SQL Server.
- Your applications might require different server-level collations. Although each database can have different collations, an application might be dependent on the collation of the **tempdb** database when the application is using temporary objects.

You can use multiple instances to install different versions of SQL Server side-by-side. This functionality can assist when testing upgrade scenarios or performing upgrades.

Default and Named Instances

Before SQL Server 2000, just one copy of SQL Server could be installed on a server system. SQL Server was addressed by the name of the Windows® server where it was hosted. To maintain backward compatibility, this mode of connection is still supported and is known as the “default instance”. For internal configuration tools, a default instance of the database engine is named **MSSQLSERVER**.

Additional instances of SQL Server require an instance name that you can use in conjunction with the server name and are known as named instances. If you want all your instances to be named instances, you simply provide a name for each one when you install them. You cannot install all components of SQL Server in more than one instance. To access a named instance, client applications use the address **Server-Name\Instance-Name**.

For example, a named instance called **Test** on a Windows server called **APPSERVER1** could be addressed as **APPSERVER1\Test**. The default instance on the same **APPSERVER1** server could be addressed as **APPSERVER1** or **APPSERVER1\MSSQLSERVER**.

There is no need to install SQL Server tools and utilities more than once on a server. You can use a single installation of the tools to manage and configure all instances.

- Many SQL Server components are instance-aware
- Instances enable isolation of:
 - Administration and security configuration
 - Performance and SLAs
 - Versions and collations
- Two types of instance:
 - Default instance
 - Named instance

SQL Server Editions

SQL Server 2016 is available in a variety of editions, with different price points and levels of capability.

All of the editions listed here are available for 32-bit and 64-bit architectures.

Principal Editions

Edition	Business Use Case
Enterprise	Premium offering. Provides the highest levels of reliability for demanding workloads.
Business Intelligence	Adds advanced Business Intelligence (BI) features to the offering from Standard Edition.
Standard	Delivers a reliable, complete data management platform.

- Principal Editions
 - Enterprise
 - Business Intelligence
 - Standard
- Specialized Editions
 - Web
- Breadth Editions
 - Developer
 - Express
- Cloud Editions
 - Microsoft Azure SQL Database

Specialized Editions

Specialized editions are targeted at specific classes of business workloads.

Edition	Business Use Case
Web	Provides a secure, cost effective, and scalable platform for public websites, website service providers, and applications.

Breadth Editions

Breadth editions are intended for customer scenarios and offered free or at low cost.

Edition	Business Use Case
Developer	You can build, test and demonstrate all SQL Server functionality.
Express	Provides a free, entry-level edition suitable for learning and lightweight desktop or web applications.

Cloud Editions

Edition	Business Use Case
Microsoft Azure SQL Database	You can build database applications on a scalable and robust cloud platform.

SQL Server Parallel Data Warehouse uses massively parallel processing (MPP) to execute queries against vast amounts of data quickly. Parallel Data Warehouse systems are sold as a complete hardware and software “appliance” rather than through standard software licenses.

A Compact Edition, for stand-alone and occasionally connected mobile applications, and optimized for a very small memory footprint, is also available.

For more information, see the *Features Supported by the Editions of SQL Server 2016* topic in the SQL Server Technical Documentation at:



Features Supported by the Editions of SQL Server 2016

<http://aka.ms/b37z3x>

SQL Server Versions

SQL Server has been available for many years, yet it is rapidly evolving with new capabilities and features. It is a platform with a rich history of innovation achieved while maintaining strong levels of stability.

Early Versions

The earliest versions (1.0 and 1.1) were based on the OS/2 operating system. SQL Server 4.2 and later moved to the Microsoft Windows operating system, initially on Windows NT.

Release Name	Version Number	Release Year
1.0	1.0	1989
1.1	1.1	1991
4.2	4.2	1992
4.21	4.21	1994
6.0	6.0	1995
6.5	6.5	1996
7.0	7.0	1998
2000	8.0	2000
2005	9.0	2005
2008	10.0	2009
2008 R2	10.5	2010
2012	11.0	2013
2014	12.0	2014
2016	13.0	2016

SQL Server 7.0 saw a significant rewrite of the product. Substantial advances were made in reducing the administration workload, and OLAP Services (which later became Analysis Services) was introduced.

SQL Server 2000 featured support for multiple instances and collations. It also introduced support for data mining. After the product release, SQL Server Reporting Services (SSRS) was introduced as an add-on enhancement to the product, along with support for 64-bit processors.

SQL Server 2005 and Beyond

SQL Server 2005 provided a significant rewrite of many aspects of the product:

- Support for nonrelational data stored and queried as XML.
- SQL Server Management Studio (SSMS) was released to replace several previous administrative tools.
- SSIS replaced Data Transformation Services (DTS).
- Support for compiled .NET objects created using the Common Language Runtime (CLR).
- The Transact-SQL language was substantially enhanced, including structured exception handling.
- Dynamic Management Views (DMVs) and Dynamic Management Functions (DMFs) were introduced for detailed health monitoring, performance tuning, and troubleshooting.
- High availability improvements were included in the product; in particular, database mirroring was introduced.
- Support for column encryption was introduced.

SQL Server 2008 also provided many enhancements:

- Filestream support improved the handling of structured and semi-structured data.
- Spatial data types were introduced.
- Database compression and encryption technologies were added.

- Specialized date- and time-related data types were introduced, including support for time zones within date and time data.
- Full-text indexing was integrated directly within the database engine. (Previously, full-text indexing was based on interfaces to the operating system level services.)
- A policy-based management framework was introduced to assist with a move to more declarative-based management practices, rather than reactive practices.
- A Windows PowerShell® provider for SQL Server was introduced.

Enhancements and additions to the product in SQL Server 2008 R2 included:

- Substantial enhancements to SSRS.
- The introduction of advanced analytic capabilities with PowerPivot.
- The adding of improved multiserver management capabilities.
- Support for managing reference data being provided with the introduction of Master Data Services.
- StreamInsight providing the ability to query data that is arriving at high speed, before storing it in a database.
- Data-tier applications assisting with packaging database applications as part of application development projects.

Enhancements and additions to the product in SQL Server 2012 included:

- Further substantial enhancements to SSRS.
- Substantial enhancements to SSIS.
- The introduction of tabular data models into SQL Server Analysis Services (SSAS).
- The migration of BI projects into Visual Studio® 2010.
- The introduction of AlwaysOn enhancements to SQL Server High Availability.
- The introduction of Data Quality Services.
- Enhancements to the Transact-SQL language, such as the addition of sequences, new error-handling capabilities, and new window functions.
- The introduction of the FileTable.
- The introduction of statistical semantic search.
- Many general tooling improvements.

Enhancements and additions to the product in SQL Server 2014 included:

- Memory-optimized tables.
- The capability to store database files in Azure.
- Support for managed backup to Azure.
- Support for Azure virtual machines in Availability Groups.
- A new cardinality estimator.
- Updatable columnstore indexes (previously, columnstore indexes could be created but not updated).

SQL Server 2016

SQL Server 2016 builds on the mission-critical capabilities of previous versions providing even better performance, availability, scalability, and manageability. It provides enhancements for in-memory OLTP, in addition to better integration with Azure.

Demonstration: Identify the Edition and Version of a Running SQL Server Instance

In this demonstration, you will see:

- Five methods to identify the edition and version of a running SQL Server instance.

Demonstration Steps

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are running and log on to MIA-SQL-20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. On the taskbar, click the **Microsoft SQL Server Management Studio** shortcut.
3. In the **Connect to Server** dialog box, click **Connect**.
4. On the **File** menu, point to **Open**, and then click **File**.
5. In the **Open File** dialog box, navigate to **D:\Demofiles\Mod01**, click **Demonstration A - VersionAndEdition.sql**, and then click **Open**.
6. In Object Explorer, point to the server name (**MIA-SQL**) and show that the server number is in parentheses after the server name.
7. In Object Explorer, right-click the server name **MIA-SQL**, and then click **Properties**.
8. On the **General** page, note **Product** and **Version** properties are visible, and then click **Cancel**.
9. Start **File Explorer**. Navigate to **C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Log**.
10. In the **Log** dialog box, click **Continue**.
11. Double-click the **ERRORLOG** file.
12. In the **How do you want to open this file?** dialog box, click **Notepad**.
13. The first entry in the file displays the version name, version number and edition, amongst other information. Close Notepad.
14. In SQL Server Management Studio, select the code under the comment **Method 4**, and then click **Execute**.
15. Select the code under the comment **Method 5**, and then click **Execute**.
16. Close SQL Server Management Studio without saving changes.

Categorize Activity

Place each piece of SQL Server terminology into the appropriate category. Indicate your answer by writing the category number to the right of each item.

Items	
1	SQL Server versions
2	SQL Server editions
3	SQL Server instances

Category 1		Category 2		Category 3
Major releases of SQL Server		Levels of capability within a major release of SQL Server		Installations of SQL Server

Lesson 2

Overview of SQL Server Architecture

Before you start looking at the resources that SQL Server requires from the underlying server platform, you need to know how SQL Server functions, so you can understand why each resource requirement exists. To interpret SQL Server documentation, you also need to become familiar with some of the terminology used when describing how the product functions.

The most important resources used by SQL Server from a server platform are CPU, memory, and I/O. In this lesson, you will see how SQL Server is structured and how it uses these resources.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe SQL Server architecture.
- Explain CPU usage by SQL Server.
- Describe the role of parallelism.
- Explain how 32-bit and 64-bit servers differ, with respect to SQL Server.
- Describe how SQL Server uses memory.
- Describe the difference between physical and logical I/O.

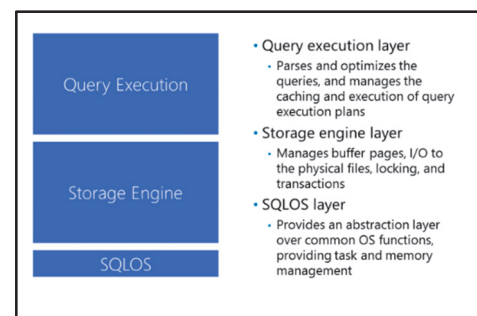
SQL Server Architecture

SQL Server comprises many components that work together. Three general categories of components exist within the database engine and are structured as layers—query execution (also referred to as the relational engine), storage engine, and SQLOS.

Query Execution Layer

In addition to managing the query optimization process, the query execution layer manages connections and security. It uses a series of subcomponents to help work out how to execute your queries:

- The parser checks that you have followed the rules of the Transact-SQL language and outputs a syntax tree, which is a simplified representation of the queries to be executed. The parser outputs what you want to achieve in your queries.
- The algebraizer converts the syntax tree into a relational algebra tree, where operations are represented by logic objects, rather than words. The aim of this phase is to take the list of what you want to achieve, and convert it to a logical series of operations representing the work that needs to be performed.
- The query optimizer then considers the different ways that your queries could be executed and finds an acceptable plan. Costs are based on the required operation and the volume of data that needs to be processed—this is calculated taking into account the distribution statistics. For example, the query optimizer considers the data that needs to be retrieved from a table and the indexes available on the



table, to decide how to access data in the table. It is important to realize that the query optimizer does not always look for the lowest cost plan because, in some situations, this might take too long. Instead, the query optimizer finds a plan that it considers satisfactory. The query optimizer also manages a query plan cache to avoid the overhead of performing all this work, when another similar query is received for execution.

Storage Engine Layer

The storage engine layer manages the data held within databases. The main responsibilities of the storage engine layer are to manage how data is stored, both on disk and in memory; to manage how the data is cached for reuse; and to manage the consistency of data through locking and transactions. The main subcomponents of the storage engine layer are as follows:

- The access methods component is used to manage how data is accessed. For example, the access methods component works with scans, seeks, and lookups.
- The page cache manages the storage of cached copies of data pages. Caching of data pages is used to minimize the time it takes to access them. The page cache places data pages into memory, so they are present when needed for query execution.
- The locking and transaction management components work together to maintain consistency of your data. This includes the maintenance of transactional integrity, with the help of the database log file.

SQLOS Layer

SQL Server Operating System (SQLOS) is the layer of SQL Server that provides operating system functionality to the SQL Server components. All SQL Server components use programming interfaces provided by SQLOS to access memory, schedule tasks, or perform I/O.

The abstraction layer provided by SQLOS avoids the need for resource-related code to be present throughout the SQL Server database engine code. The most important functions provided by this layer are memory management and scheduling. These two aspects are discussed in more detail later in this lesson.

CPU Usage by SQL Server

All work performed in the Windows operating system is based on the execution of threads. Windows uses preemptive scheduling of threads to maintain control. Rather than requiring threads to give up control of the CPU voluntarily, preemptive systems have a clock that they use to interrupt threads when the thread's allocated share of CPU time is complete. Threads are considered to have encountered a context switch when they are preempted.

- Windows uses preemptive scheduling of threads
- One scheduler for every logical CPU created in SQL OS:
 - Manages the threads retrieved from Windows and assigns tasks to threads
 - Minimizes context switches through cooperative scheduling
- CPU availability can be configured without restart:
 - Schedulers can be enabled or disabled
 - CPU affinity mask can be set
- Tasks waiting on a resource are moved to a waiting list:
 - Wait type and time are recorded
 - Details are useful for monitoring and troubleshooting

SQL Server Threads

SQL Server retrieves threads from Windows. The SQL Server configuration setting **max worker threads** (set at instance level) determines how many threads will be retrieved. SQL Server has its own internal scheduling system, independent of the scheduling performed by the operating system. Instead of using Windows threads directly, SQL Server creates a pool of worker threads that are mapped to Windows threads whenever work needs to be performed.

When a SQL Server component needs to execute code, the component creates a task that represents the unit of work to be done. For example, if you send a batch of Transact-SQL commands to the server, it's likely that the batch will be executed within a task.

When a SQL Server component creates a task, it is assigned the next available worker thread that is not in use. If no worker threads are available, SQL Server will try to retrieve another Windows thread, up to the point that the max worker threads configuration limit is reached. At that point, the new task would need to wait to get a worker thread. All tasks are arranged by the SQL Server scheduler until they are complete.

Affinity Mask

Schedulers can be enabled and disabled by setting the CPU affinity mask on the instance. The affinity mask is a configurable bitmap that determines which CPUs from the host system should be used for SQL Server—and can be changed without needing to reboot. By default, SQL Server will assume that it can use all CPUs on the host system. While you can configure the affinity mask bitmap directly by using `sp_configure`, this method is marked for deprecation in a future version of SQL Server. Use the Properties dialog box for the server instance in SQL Server Management Studio (SSMS) to modify processor affinity.

Processor affinity can be modified at the level of individual processors, or at the level of NUMA nodes.



Note: Whenever the term CPU is used here in relation to SQL Server internal architecture, it refers to any logical CPU, regardless of whether core or hyper-threading CPUs are being used.

Waiting for Resources

One concept that might be difficult to grasp at first is that most SQL Server tasks spend the majority of their time waiting for something external to happen. Most of the time, they are waiting for I/O or the release of locks, but this can involve other system resources.

When a task needs to wait for a resource, it is placed on a list until the resource is available; the task is then signaled that it can continue, though it still needs to wait for another share of CPU time. This allocation of CPUs to resources is a function of the SQLOS.

SQL Server keeps detailed internal records of how long tasks spend waiting, and of the types of resources they are waiting for. Wait statistics information can be a useful resource for troubleshooting performance problems. You can see these details by querying the following system views:

```
sys.dm_os_waiting_tasks;  
sys.dm_os_wait_stats;
```

Parallelism

All operations run against SQL Server are capable of running by using sequential elements on a single task. To reduce the overall run time (at the expense of additional CPU time), SQL Server can distribute elements of an operation over several tasks, so they execute in parallel.

Parallel Execution Plans

Parallel execution involves the overhead of synchronizing and monitoring the tasks. Because of this, SQL Server only considers parallel plans for expensive operations, where the advantages outweigh the additional overhead.

The query optimizer determines whether a parallel plan should be used, based on aspects of the operation and the system configuration:

- A configuration value, **maximum degree of parallelism** (MAXDOP), determines a limit for how many CPUs can be used to execute a query.
- Another configuration value, **cost threshold for parallelism**, determines the cost that a query must meet before a parallel query plan will even be considered.
- Query cost is determined based on the amount of data the query optimizer anticipates will need to be read to complete the operation. This information is drawn from table and index statistics.

If a query is expensive enough to consider a parallel plan, SQL Server might still decide to use a sequential plan that is lower in overall cost.

Controlling Parallelism

The query optimizer only creates a parallel plan and is not involved in deciding the MAXDOP value. This value can be configured at the server level and overridden at the query level via a query hint. Even if the query optimizer creates a parallel query plan, the execution engine might decide to use only a single CPU, based on the resources available when it is time to execute the query.

In earlier versions of SQL Server, it was common to disable parallel queries on systems that were primarily used for transaction processing. This limitation was implemented by adjusting the server setting for MAXDOP to the value 1. In current versions of SQL Server, this is no longer generally considered a good practice.

A better practice is to raise the value of **cost threshold for parallelism** so that a parallel plan is only considered for higher cost queries.

For further information, see the *Configure the cost threshold for parallelism Server Configuration Option* topic in the SQL Server 2016 Technical Documentation:



Configure the cost threshold for parallelism Server Configuration Option

<http://aka.ms/leh9eq>

- Parallelism refers to multiple processors cooperating to execute a single query at the same time
- SQL Server can decide to distribute queries to more than one task
 - Tasks can run in parallel
 - Overall execution is faster
 - Synchronization overhead is incurred
 - Parallelism is only considered for expensive plans
- **Max degree of parallelism** defines how many CPUs can be used for execution of a parallel query
 - Can be overridden using the MAXDOP query hint
- **Cost threshold for parallelism** defines minimal cost for considering parallel plans

32-bit and 64-bit Servers

Virtual Address Space (VAS) is the total amount of memory that an application like SQL Server could possibly refer to in Windows. The VAS depends upon the configuration of the server.

32-bit Systems

These systems have a VAS of 4 GB. By default, half the address space (2 GB) is reserved for the system and is known as kernel mode address space. The other half of the VAS (2 GB) is available for applications to use and is known as the user mode address space. You can change this proportion by using a /3 GB switch in the boot.ini file of the operating system (on earlier operating systems) or by using the bcdedit program to edit the Boot Configuration Datastore (on more recent operating systems). After the /3 GB switch has been configured, 1 GB of the VAS is allocated for the kernel, with the remaining 3 GB allocated for applications.

Note: More fine-grained control of the allocated space can be achieved by using the /USERVA switch instead of the /3 GB switch. The /USERVA switch means you can configure any value between 2 GB and 3 GB for user applications.

64-bit Systems

Database systems tend to work with large amounts of data and need substantial amounts of memory, so that the systems can serve significant numbers of concurrent users. SQL Server needs large amounts of memory for caching queries and data pages. A limit of 2 GB to 3 GB for VAS is a major constraint on most current systems. As is the case with most database engines, SQL Server is best installed on a 64-bit version of Windows, instead of a 32-bit version.

- It's best to install a 64-bit version of SQL Server on a 64-bit version of Windows. Full 64-bit systems offer a VAS of 8 TB.
- You can install a 32-bit version of SQL Server on a 64-bit version of Windows. This configuration provides a full 4 GB of VAS for the 32-bit applications and is based on the Windows on Windows (WOW) emulation technology.

64-bit System Limitations

Not all current systems can be implemented as 64-bit systems. The most common limitation is the availability of data providers. If special data providers are needed to connect SQL Server to other systems (particularly through the linked servers feature), it is important to check that the required data providers are available in 64-bit versions. For example, the Jet engine is currently only available as a 32-bit provider.

- Virtual Address Space is the memory that can be allocated to applications such as SQL Server
 - 4 GB on 32-bit systems (2-3 GB available for the application)
 - 4 GB for 32-bit applications running on WOW on 64-bit OS
 - 8 TB for 64-bit systems
- AWE extension can no longer be used to access additional memory on 32-bit systems
- Itanium processors are no longer supported
- SQL Server performance strongly depends on memory
 - Installing 64-bit versions is preferred
 - 64-bit options available for all editions of SQL Server

Overview of SQL Server Memory Management

The main memory object of SQL Server is known as the buffer pool. This is divided into 8 KB pages—the same size as a data page within a database. The buffer pool is comprised of three sections:

- **Free Pages.** Pages that are not yet used but are kept to satisfy new memory requests.
- **Stolen Pages.** Pages that are used (formally referred to as stolen) by other SQL Server components, such as the query optimizer and storage engine.
- **Data Cache.** Pages that are used for caching database data pages. All data operations are performed in the data cache. If a query wants to select data from a specific data page, the data page is moved from physical storage into the data cache first. Also, data modification is only ever performed in memory. Modifications are never performed directly on the data files. When a data page is changed, the page is marked as dirty. Dirty pages are written to data files by a background process known as a checkpoint.

- Buffer pool is the main memory object of SQL Server:
 - Holds data cache
 - Provides memory for other SQL Server components
 - Is divided into 8 KB pages
- SQL OS automatically allocates as much memory as is needed:
 - Has a mechanism to prevent memory shortage on the system
 - Can be configured using min and max server memory options

The data cache implements a least recently used (LRU) algorithm to determine candidate pages to be dropped from the cache as space is needed—after they have been flushed to disk (if necessary) by the checkpoint process. The process that performs the task of dropping pages is known as the lazy writer—this performs two core functions. By removing pages from the data cache, the lazy writer attempts to keep sufficient free space in the buffer cache for SQL Server to operate. The lazy writer also monitors the overall size of the buffer cache to avoid taking too much memory from the Windows operating system.

SQL Server 2014 and SQL Server 2016 include a feature where the buffer pool can be extended onto fast physical storage (such as a solid-state drive); this can significantly improve performance.

SQL Server Memory and Windows Memory

The memory manager within SQLOS allocates and controls the memory used by SQL Server. It does this by checking the available memory on the Windows system, calculating a target memory value—which is the maximum memory that SQL Server can use at that point—to avoid a memory shortage at the Windows operating system level. SQL Server is designed to respond to signals from the Windows operating system that indicate a change in memory needs.

Providing SQL Server stays within the target memory, it requests additional memory from Windows when required. If the target memory value is reached, the memory manager answers requests from components by freeing up the memory of other components. This can involve evicting pages from caches. You can control the target memory value by using the **min server memory** and **max server memory** instance configuration options.

It is good practice to reduce the **max server memory** to a value where a SQL Server instance will not attempt to consume all the memory available to the host operating system.

For more information, see the *Server Memory Server Configuration Options* topic in the SQL Server 2016 Technical Documentation:



Server Memory Server Configuration Options

<http://aka.ms/afm7fd>

Physical I/O and Logical I/O

A logical I/O occurs when a task can retrieve a data page from the buffer cache. When the requested page is not present in the buffer cache, it must first be read into the buffer cache from the database. This database read operation is known as a physical I/O.

Because access to data through physical I/O is typically much slower than access to logical I/O or CPU resources, it's important to optimize physical I/O as far as possible, to keep a SQL Server instance performing well.

I/O Type	Description
Physical I/O	Physical I/O occurs when the requested page is not available in buffer cache and must be read from the data file into the data cache before the requested page can be supplied or when a changed page is written to the data file.
Logical I/O	Logical I/O occurs when the requested page is available in the data cache.

Reducing Physical I/O

From an overall system performance point of view, two I/O aspects must be minimized:

- You need to minimize the number of physical I/O operations.
- You need to minimize the time taken by each I/O operation that is still required.

Minimizing the number of physical I/O operations is accomplished by:

- Providing enough memory for the data cache.
- Optimizing the physical and logical database layout, including indexes.
- Optimizing queries to request as few I/O operations as possible.

Reducing the time taken by physical I/O operations is accomplished by:

- Ensuring that SQL Server data files are held on sufficiently fast storage.

One of the major goals of query optimization is to reduce the number of logical I/O operations. The side effect of this is a reduction in the number of physical I/O operations.

Note: Logical I/O counts can be difficult to interpret as certain operations can cause the counts to be artificially inflated, due to multiple accesses to the same page. However, in general, lower counts are better than higher counts.

Monitoring Query I/O Operations

Both logical and physical I/O operation counts can be seen for each query by setting the following connection option:

```
SET STATISTICS IO ON;
```

The overall physical I/O operations occurring on the system can be seen by querying the `sys.dm_io_virtual_file_stats` system function. The values returned by this function are cumulative from the point that the system was last restarted.

Demonstration: CPU and Memory Configurations in SSMS

In this demonstration, you will see how to review and configure SQL Server CPU and memory by using SSMS.

Demonstration Steps

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are running and log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. On the taskbar, click the **SQL Server Management Studio** shortcut.
3. In the **Connect to Server** dialog box, click **Connect**.
4. On the **File** menu, point to **Open**, and then click **File**.
5. In the **Open File** dialog box, navigate to **D:\Demofiles\Mod01**, click **Demonstration B - CPUAndMemory.sql**, and then click **Open**.
6. In **Object Explorer**, right-click the **MIA-SQL** server and click **Properties**. Note the values for **Platform**, **Memory** and **Processors**.
7. Select the **Processors** tab, fully expand the tree under the Processor column heading. Note the setting for **Max Worker Threads**.
8. Select the **Advanced** tab and in the **Parallelism** group, review the default values for **Cost Threshold for Parallelism** and for **Max Degree of Parallelism**.
9. Select the **Memory** tab and review the default memory configurations. Click **Cancel** to close the Server Properties window.
10. Execute the query below **Step 5**.
11. Close SQL Server Management Studio without saving any changes.

Sequencing Activity

Put the following SQL Server architectural layers in order from highest (closest to the client application) to lowest (closest to the operating system) by numbering each to indicate the correct order.

	Steps
	Query Execution Layer
	Storage Engine
	SQLOS

Lesson 3

SQL Server Services and Configuration Options

A SQL Server instance consists of a number of services running on a Windows server. Whether you are managing the configuration of existing instances or planning to install new instances, it is important to understand the function of the various SQL Server services, in addition to the tools available to assist in configuring those services. The SQL Server Configuration Manager is the principal configuration tool you will learn about in this lesson.



Note: Many topics in this lesson do not apply to an Azure SQL Database. The configuration and management of service accounts for Azure SQL Database is not exposed to Azure users. However, these topics apply to SQL Server instances running on virtual machines in the Azure cloud.

Lesson Objectives

At the end of this lesson you will be able to:

- Describe the SQL Server services.
- Understand the requirements for service accounts for SQL Server services.
- Describe the configuration of SQL Server network ports, aliases, and listeners.

SQL Server Services

SQL Server services run on a Windows server. When you install SQL Server, you can choose to install some or all of the available features, which means that some or all of these will be installed. Only the services required to support the installed SQL Server features will be present.

The available SQL Server services include:

- **SQL Server.** This is the service for the SQL Server database engine.
- **SQL Server Agent.** This service is responsible for automation, including running jobs and issuing alerts.
- **SQL Server Analysis Services.** This service provides online analytical processing (OLAP) and data mining functionality.
- **SQL Server Reporting Services.** This service is involved in the creation and execution of reports.
- **SQL Server Integration Services 13.0.** This service performs the tasks associated with extract, transform, and load (ETL) operations.

- A SQL Server instance is made up of a number of Windows services
- Only the services required to support the installed SQL Server features will be present
- Many services are installed once per SQL Server instance
 - If a service is linked to an instance, the instance name will appear in brackets after the service name—SQL Server (MSSQLSERVER)
- Use SQL Server Configuration Manager to configure services

- **SQL Server Browser.** By default, named instances use dynamic TCP port assignment, which means that, when the SQL Server service starts, they select a TCP port from the available ports. With the SQL Browser service, you can connect to named SQL Server instances when the connection uses dynamic port assignment. The service is not required for connections to default SQL Server instances, which use port 1433, the well-known port number for SQL Server. For connections to named instances that use a TCP-specific port number, or which specify a port number in the connection string, the SQL Server Browser service is not required and you can disable it.
- **SQL Full-Text Filter Daemon Launcher.** This service supports the operation of full-text search; this is used for the indexing and searching of unstructured and semi-structured data.
- **SQL Server VSS Writer.** Use this to back up and restore by using the Windows Volume Shadow (VSS) copy service. This service is disabled by default; you should only enable it if you intend to use VSS backups.
- **SQL Server PolyBase Engine.** This service provides the engine used to query Hadoop and Azure Blob storage data from the SQL Server database engine.
- **SQL Server PolyBase Data Movement.** This service carries out the transfer of data between the SQL Server database engine and Hadoop or Azure Blob storage.
- **SQL Server Launchpad.** This service supports the integration of R language scripts into Transact-SQL queries.



Note: The service names in the above list match the names that appear in both the SQL Server Configuration and Windows Services tools for SQL Server 2016 services. In these tools, the names of services that are instance-aware will be followed by the name of the instance to which they refer in brackets—for example, the service name for the database engine service for an instance named MSSQLSERVER is **SQL Server (MSSQLSERVER)**.

Configuring Services

You use SQL Server Configuration Manager (SSCM) to configure SQL Server services, and the network libraries exposed by SQL Server services, in addition to configuring how client connections are made to SQL Server.

You can use SSCM to control (that is, start, stop, and configure) each service independently; to set the startup mode (automatic, manual, or disabled) of each service; and to set the service account identity for each service.

You can also set startup parameters to start SQL Server services with specific configuration settings for troubleshooting purposes.

Managing Service Accounts

Each SQL Server service must run in the security context of a service account. You can use several types of accounts for SQL Server services, so it is important to know the differences between them when planning a secure SQL Server installation. The account you choose for each service will require the privileges that the service needs to run; however, you should avoid using accounts, such as administrative accounts, that have excessive privileges and rights. Follow the principal of least privilege when choosing service accounts—never use an account that has greater privileges than it requires for doing its job. Ideally, each service should run under its own dedicated account. This minimizes risk because, if one account is compromised, other services remain unaffected.

When you use the SQL Server Setup installation program (or the SQL Server Configuration Manager) to specify a service account, the account is automatically added to the relevant security group to ensure that it has the appropriate rights.

- Assign service accounts during installation
 - After installation, configure SQL Server services by using the SQL Server Configuration Manager tool
- Choose an appropriate account type:
 - Domain user account
 - Local user account
 - Local service account
 - Local system account
 - Network service account
 - Managed service account
 - Virtual service account



Note: Because it automatically assigns accounts to the correct security groups, use the SQL Server Configuration Manager to amend SQL Server service accounts. You should not use the Windows Services tool to manage SQL Server service accounts.

The different types of accounts that you can use for SQL Server services include:

- **Domain user account.** A nonadministrator domain user account is a suitable choice for service accounts in a domain environment.
- **Local user account.** A nonadministrator local user account is a secure choice for service accounts in a nondomain environment, such as a perimeter network.
- **Local system account.** The local system account is a highly privileged account that is used by various Windows services. Consequently, you should avoid using it to run SQL Server services.
- **Local service account.** You use the local service account, a predefined account with restricted privileges, to access local resources. This account is used by Windows services and other applications that do not require access to remote resources; generally, a dedicated service account for each SQL Server service is preferred. If your database server runs on Windows Server 2008 R2 or later, you can use a virtual service account instead (see below).
- **Network service account.** The network service account has fewer privileges than the local system account, but it does give a service access to network resources. However, because this account is often used by multiple services, including Windows services, you should avoid using it where possible. If your database server runs on Windows Server 2008 R2 or later, you can use a virtual service account instead (see below).
- **Managed service account.** Managed service accounts are available if the host operating system is Windows Server 2008 R2 or later (Windows 7 also supports managed service accounts). SQL Server support for managed service accounts was introduced in SQL Server 2012. A managed service account is a type of domain account that is associated with a single server, and which you can use to manage services. You cannot use a managed service account to log on to a server, so it is more secure than a domain user account. Additionally, unlike a domain user account, you do not need to

manage passwords for managed service accounts manually. However, a domain administrator needs to create and configure a managed service account before you can use it.

- **Virtual service account.** Virtual service accounts are available if the host operating system is Windows Server 2008 R2 or later (Windows 7 also supports virtual service accounts). SQL Server support for virtual service accounts was introduced in SQL Server 2012. A virtual service account is similar to a managed service account, except that it is a type of local account that you can use to manage services, rather than a domain account. Unlike managed service accounts, an administrator does not need to create or configure a virtual service account. This is because a virtual service account is a virtualized instance of the built-in network service account with its own unique identifier.

Configuring Network Protocols

You use SQL Server Configuration Manager to configure SQL Server services and the network libraries exposed by SQL Server services, in addition to configuring how client connections are made to SQL Server.

Server Network Ports and Listeners

You can configure each network endpoint that is exposed by an instance of SQL Server. This includes the determination of which network libraries are enabled and, for each one, the configuration of the network library. Typically, this will involve settings such as protocol port numbers. You should discuss the required network protocol configuration of SQL Server with your network administrator.

- Three network protocols are supported:
 - TCP/IP
 - Named pipes
 - Shared Memory
- Network protocols can be defined for both server and client components
 - 32-bit and 64-bit server settings and client settings are configured independently
- You can use aliases to allow a single server instance to respond to multiple names

Three protocols are available:

- TCP/IP
- Named pipes
- Shared Memory

The configuration for the TCP/IP protocol provides different settings on each configured IP address if required—or a general set of configurations that are applied to all IP addresses.

By default, only the Shared Memory protocol (which means client applications can run on the same server to connect) is the only protocol available when a new SQL Server instance is installed. Other protocols, where client applications can connect over a network interface, must be enabled by an administrator; this is done to minimize the exposure of a new SQL Server instance to security threats.

Client Network Ports and Listeners

SQL Native Client (SNAC) is installed on both the server and client systems. When SQL Server management tools are installed on the server, they use the SNAC library to connect to the SQL Server services on the same system. Every computer with SNAC installed should have the ability to configure how that library will access SQL Server services. For this reason, in addition to server network configuration settings, SSCM includes client configuration nodes with which you can configure how client connections are made. Note that two sets of client configurations are provided and that they only apply to the computer where they are configured. One set is used for 32-bit applications; the other set is used for 64-bit applications. SSMS is a 32-bit application, even when SQL Server is installed as a 64-bit application.

Aliases

Connecting to a SQL Server service can involve multiple settings such as server address, protocol, and port. If you hard-code these connection details in your client applications—and then any of the details change—your application will no longer work. To avoid this issue and to make the connection process simpler, you can use SSCM to create aliases for server connections.

You create a server alias and associate it with a server, protocol, and port (if required). Client applications can then connect to the alias by name without any concern about how those connections are made. Aliases can provide a mechanism to move databases between SQL Server instances without having to amend client connection strings. You can configure one or more aliases for each client system that uses SNAC (including the server itself). Aliases for 32-bit applications are configured independently of those for 64-bit applications.

Check Your Knowledge

Question	
On a newly-installed SQL Server 2016 instance, which network protocol is enabled by default?	
Select the correct answer.	
☐	Shared Memory
☐	Named Pipes
☐	TCP/IP

Module Review and Takeaways

In this lesson, you have learned about the main components of a SQL Server installation, in addition to the differences between the available versions and editions of SQL Server 2016. You now have an overview of the SQL Server architecture, and how SQL Server interacts with the operating system to access CPU, I/O and memory. You also learned about how SQL Server services and networking are configured.

Review Question(s)

Question: On a single server, when might you use a multiple installation of SQL Server and when might you use multiple SQL Server instances?

Question: Which edition of SQL Server is most suitable in your organization?

Module 2

Installing SQL Server 2016

Contents:

Module Overview	2-1
Lesson 1: Considerations for Installing SQL Server	2-2
Lesson 2: tempdb Files	2-11
Lesson 3: Installing SQL Server 2016	2-13
Lesson 4: Automating Installation	2-17
Lab: Installing SQL Server 2016	2-20
Module Review and Takeaways	2-25

Module Overview

One of the key responsibilities of a database administrator (DBA) is to provision servers and databases. This includes planning and performing the installation of Microsoft SQL Server® on physical servers and virtual machines.

This module explains how to assess resource requirements for SQL Server 2016 and how to install it.

Objectives

After completing this module, you will be able to:

- Plan an installation of SQL Server 2016.
- Configure the **tempdb** database.
- Install SQL Server 2016.
- Describe how to automate SQL Server 2016 installations.

Lesson 1

Considerations for Installing SQL Server

This lesson covers the hardware and software requirements for installing SQL Server 2016. You will learn about the minimum requirements specified by Microsoft, in addition to tools and techniques to assess hardware performance before you install SQL Server.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe minimum hardware and software requirements for running SQL Server.
- Understand CPU and memory requirements for SQL Server.
- Understand considerations when designing I/O subsystems for SQL Server installations.
- Use the SQLIOSim tool to assess the suitability of I/O subsystems for use with SQL Server.
- Use the Diskspd tool for load generation and performance testing of storage I/O subsystems.

Minimum Hardware and Software Requirements

When planning an installation of SQL Server, consider the following factors:

- **Hardware resource requirements.** Depending on the components being installed and the workloads they must support, SQL Server can place substantial demands on the hardware resources of a server. A typical SQL Server database engine installation consumes CPU, memory, and storage I/O subsystem resources to meet the requirements of the applications using the database. In the case of most enterprise database solutions, the server (physical or virtual) is often dedicated to support the SQL Server installation. However, in some small organizations or departmental solutions, the database engine may need to coexist with other applications and software services on the same server, and compete for hardware resources. When planning an installation, you must be sure that the server where you intend to install SQL Server has enough spare capacity to support the database workload while providing the required levels of performance and concurrency.
- **Software resource requirements.** For SQL Server 2016, the host operating system has to meet certain software prerequisites.

- Hardware requirements:
 - Processors
 - Memory
 - Disk
- Software requirements:
 - Operating system
 - Prerequisite software
 - General software requirements

For more information, see the *Planning a SQL Server Installation* topic in the SQL Server 2016 Technical Documentation:



Planning a SQL Server Installation

<http://aka.ms/xedqba>

Hardware Requirements

In earlier versions of SQL Server, it was necessary to focus on minimum requirements for processor, disk space and memory. Nowadays, this is much less of a concern because the minimum hardware requirements for running SQL Server 2016 are well below the specification of most modern systems. However, SQL Server has limitations on the maximum CPU count and memory it will support. These limits vary between different SQL Server editions—they might also vary on the edition of Microsoft Windows® hosting the SQL Server instance.

Processors

The minimum processor requirements for x86 and x64 processor architectures are shown in the following table:

Architecture	Processor requirements
x86	x86 processor: Pentium III-compatible or faster Minimum speed: 1.4 GHz Recommended speed: 2.0 GHz or higher
x64	x64 processor: AMD Opteron, AMD Athlon 64, Intel Xeon with Intel EM64T support, Intel Pentium IV with EM64T support Minimum speed: 1.4 GHz Recommended speed: 2.0 GHz or higher

Processor core count, rather than processor speed, is more of a concern when planning a SQL Server installation. While you might want to add as many CPUs as possible, it is important to consider that there is a trade-off between the number of CPU cores and license costs. In addition, not all computer architectures support the addition of CPUs. Adding CPU resources might then require architectural upgrades to computer systems, not just the additional CPUs.

Memory

The minimum memory requirements are shown in the following table:

Edition	Minimum Memory	Recommended Memory
SQL Server 2016 Express	512 MB	1 GB
All other editions	1 GB	4 GB

Memory requirements for SQL Server are discussed further in the next topic.

Disk

SQL Server requires a minimum disk space of 6 GB to install (this might vary according to the set of features you install) but this is not the only value to consider; the size of user databases can be far larger than the space needed to install the SQL Server software.

Because of the large amount of data that must be moved between storage and memory when SQL Server is in normal operation, I/O subsystem performance is critical.

Tools for assessing I/O subsystem performance are discussed later in this module.

Software Requirements

Like any server product, SQL Server requires specific combinations of operating system and software before installation.

Operating System

Operating system requirements vary between different editions of SQL Server. The SQL Server 2016 Technical Documentation provides a precise list of supported versions and editions.

You can install some versions of SQL Server on the client operating systems, such as Windows 8.

It is strongly recommended that you do not install SQL Server on a domain controller.

Prerequisite Software

In earlier versions, the installer for SQL Server would preinstall most requirements as part of the installation process. This is no longer the case—the .NET Framework (3.5 SP1) needs to be preinstalled before running the setup program. The installer for SQL Server will install the SNAC and the SQL Server setup support files. However, to minimize the installation time for SQL Server, particularly in busy production environments, it is useful to preinstall these components during any available planned downtime. Components such as the .NET Framework often require a reboot after installation, so the preinstallation of these components can further reduce downtime during installations or upgrades.

General Software Requirements

The SQL Server installer is based on the Windows Installer 4.5 technology. Before the installation of SQL Server, you should consider installing Windows Installer 4.5—this will minimize SQL Server installation time.

For more information, see the *Hardware and Software Requirements for Installing SQL Server 2016* topic in the SQL Server 2016 Technical Documentation:



Hardware and Software Requirements for Installing SQL Server 2016

<http://aka.ms/usoqbl>

Assessing CPU and Memory Requirements

Planning server resource requirements is not an easy task. No simple formula enables you to calculate resource requirements, based on measures such as database size and the number of connections, even though you might sometimes see references to these.

CPU

CPU utilization for a SQL Server largely depends upon the types of queries that are running on the system. Processor planning is often considered as relatively straightforward, in that few system architectures provide fine-grained control of the available processor resources. Testing with realistic workloads is the best option.

Increasing the number of available CPUs will provide SQL Server with more scope for creating parallel query plans. Even without parallel query plans, SQL Server workloads will make good use of multiple processors when working with simple query workloads from a large number of concurrent users. Parallel query plans are particularly useful when large amounts of data must be processed to return output.

Whenever possible, try to ensure that your server is dedicated to SQL Server. Most servers that are running production workloads on SQL Server should have no other significant services running on the same system. This particularly applies to other server applications such as Microsoft Exchange Server.

- CPU:
 - Utilization dependent on types of queries running
 - Test against realistic workloads
- Memory:
 - Express Edition can only use 1 GB of memory
 - Cannot use AWE-based memory on 32-bit servers

Many new systems are based on Non-Uniform Memory Access (NUMA) architectures. In a traditional symmetric multiprocessing (SMP) system, all CPUs and memory are bound to a single system bus. The bus can become a bottleneck when additional CPUs are added. On a NUMA-based system, each set of CPUs has its own bus, complete with local memory. In some systems, the local bus might also include separate I/O channels. These CPU sets are called NUMA nodes. Each NUMA node can access the memory of other nodes but the local access to local memory is much faster. The best performance is achieved if the CPUs mostly access their own local memory. Windows and SQL Server are both NUMA-aware and try to make use of these advantages.

Optimal NUMA configuration is highly dependent on the hardware. Special configurations in the system BIOS might be needed to achieve optimal performance. It is crucial to check with the hardware vendor for the optimal configuration for a SQL Server on the specific NUMA-based hardware.

Memory

The availability of large amounts of memory for SQL Server to use is now one of the most important factors when sizing systems.

While SQL Server will operate in relatively small amounts of memory, when memory configuration challenges arise they tend to relate to the maximum, not the minimum, values. For example, the Express Edition of SQL Server will not use more than 1 GB of memory, regardless of how much memory is installed in the system.

The 64-bit operating system has a single address space that can directly access large amounts of memory.

SQL Server 2012, SQL Server 2014 and SQL Server 2016 no longer support the use of Address Windowing Extensions (AWE)-based memory to increase the address space for 32-bit systems. As a consequence, 32-bit installations of SQL Server are effectively limited to accessing 4 GB of memory.

Storage I/O and Performance

The performance of SQL Server is tightly coupled to the performance of the I/O subsystem it is using. Current I/O systems are complex, so planning and testing the storage is a key task during the planning stage.

Determining Requirements

In the first phase of planning, you should determine the requirements of the application, including the I/O patterns that must be satisfied. These include the frequency and size of reads and writes sent by the application. As a general rule, OLTP systems produce a high number of random I/O operations on the data files and sequential write operations on database log files. By comparison, data warehouse-based applications tend to generate large scans on data files, which are more typically sequential I/O operations on the data files.

- Plan and test your I/O requirements
- Considerations for storage:
 - Dedicated versus SAN storage
 - RAID systems
 - Number of drives
 - I/O caching configuration

Storage Styles

The second planning phase involves determining the style of storage to be used. With direct attached storage (DAS), it is easier to get good predictable performance. On storage area network (SAN) systems, more work is often required to get good performance; however, SAN storage typically provides a wide variety of management capabilities and storage consolidation.

One particular challenge for SQL Server administrators is that SAN administrators are generally more concerned with the disk space that is allocated to applications, rather than the performance requirements of individual files. Rather than attempting to discuss file layouts with a SAN administrator, try to concentrate on your performance requirements for specific files. Leave the decisions about how to achieve those goals to the SAN administrator. In these discussions, you should focus on what is needed rather than on how it can be achieved.

RAID Systems

In SAN-based systems, you will not often be concerned about the redundant array of independent disks (RAID) levels being used. If you have specified the required performance on a file basis, the SAN administrator will need to select appropriate RAID levels and physical disk layouts to achieve that.

For DAS storage, you should become aware of different RAID levels. While other RAID levels exist, RAID levels 1, 5, and 10 are the most common ones used in SQL Server systems.

Number of Drives

For most current systems, the number of drives (or spindles, as they are still sometimes called), matters more than the size of the disk. It is easy to find large disks that will hold substantial databases, but often a single large disk will not be able to provide sufficient I/O operations per second or enough data throughput (megabytes per second) to be workable. Solid state drive (SSD)-based systems are quickly changing the available options in this area.

Drive Caching

Disk and disk array read caches are unlikely to have a significant effect on I/O performance because SQL Server already manages its own caching system. It is unlikely that SQL Server will need to re-read a page from disk that it has recently written, unless the system is low on memory.

Write caches can substantially improve SQL Server I/O performance, but make sure that hardware caches guarantee a write, even after a system failure. Many drive write caches cannot survive failures and this can lead to database corruptions.

Assessing I/O by Using SQLIOSim

SQLIOSim is a utility designed to simulate the I/O activity generated by SQL Server without the need to install SQL Server. This capability makes SQLIOSim a good tool for pretesting server systems that are targets for running SQL Server.

SQLIOSim is a stand-alone tool that you can copy on to the server and run. You do not need to install it on the target system by using an installer. SQLIOSim has both GUI and command-line execution options.

While SQLIOSim is useful for stress-testing systems, it is not an appropriate tool for general performance testing. The tasks it performs vary between each execution of the utility, so don't attempt to compare the output of multiple executions directly, particularly in terms of timing.

- Used for reliability and integrity tests on disk subsystems
- Attempts to accurately simulate the I/O patterns of SQL Server
- Tests contain an element of randomness and should not be used for performance tuning

SQLIOSim is configured using an .ini file. The default configuration file, sqliosim.cfg.ini, is created the first time the GUI version of the tool is run. You can use the .ini file to configure the characteristics of the SQL Server disk activity the tool will simulate. These characteristics include:

- **Number and size of data files and log files.** You can configure the size of the files to change during the test.
- **Read-ahead activity.** I/O simulating SQLIOS activity.
- **Random user activity.** Random I/O simulating OLTP operations.
- **Bulk update user activity.** I/O simulating bulk data loads.
- **Administrative user activity.** I/O simulating administrative commands.

The settings for log and data files are accessible through the GUI version of the tool or command-line parameters. Simulated user activity can only be configured from an .ini file.

SQLIOSim is included on the SQL Server 2016 installation media, or you can download it from the Microsoft website. It is also included as part of a SQL Server installation. The download package includes several sample test configuration files for different usage scenarios; these are not included on the installation media.

For more information about SQLIOSim and to download the utility, see the *How to use the SQLIOSim utility to simulate SQL Server activity on a disk subsystem* topic on the Microsoft Support website:



How to use the SQLIOSim utility to simulate SQL Server activity on a disk subsystem

<http://aka.ms/itwi63>

Demonstration: Using SQLIOSim

In this demonstration, you will see how to use the SQLIOSim graphical user interface.

Demonstration Steps

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are running and log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Demofiles\Mod02** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and wait for the script to finish.
4. On the taskbar, click the **File Explorer** shortcut.
5. In File Explorer, browse to **C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Binn**, right-click **SQLIOSIM.EXE**, and then click **Run as administrator**.
6. In the **User Account Control** dialog box, click **Yes**.
7. In the **Files and Configuration** dialog box, in the **System Level Configurations** section, in the **Cycle Duration (sec)** box, type **30**.
8. In the **Test Cycles (0 - infinite)** box, type **1**.
9. In the **Error Log (XML)** box, type **D:\sqliosim.log.xml**, and then click **OK**.
10. In SQLIOSim, on the **Simulator** menu, click **Start**.
11. Allow the test to run for one or two minutes.

12. On the **Simulator** menu, click **Stop**.
13. In the **SQLIOSim** message box, click **OK**.
14. Examine the results of the test.
15. In File Explorer, go to **D:**, and open **sqliosim.log.xml** with **Office XML Handler**.
16. Review the test results in the XML file, and then close the file without saving changes.
17. Close SQLIOSim.

Assessing I/O by Using Diskspd

Diskspd is a command-line utility, developed by Microsoft, for load generation and performance testing of storage I/O subsystems. While it is not specifically designed for simulating SQL Server I/O activity, you can configure Diskspd for that purpose.

Diskspd is suitable for use when performance-tuning I/O subsystems because, for a given set of parameters, the load generated will always be the same.

Diskspd can be downloaded from the Microsoft website:

Diskspd Utility: A Robust Storage Testing Tool (superseding SQLIO)



<http://aka.ms/diskspd>

- A general purpose load generator for I/O subsystems
- Can be configured to mimic SQL Server I/O
- Suitable for use as a performance-tuning tool
- Replaces SQLIO

Detailed instructions on simulating SQL Server I/O activity, in addition to interpreting the results, are included in a document packaged with the Diskspd download (**UsingDiskspdforSQLServer.docx**).

Diskspd replaces an older utility with similar functionality, named SQLIO.

Demonstration: Using Diskspd

In this demonstration, you will see how to run the Diskspd utility.

Demonstration Steps

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are running and log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. On the taskbar, right-click the **Windows PowerShell®** shortcut and click **Run as Administrator**.
3. In the **User Account Control** dialog box, click **Yes**.
4. In Windows PowerShell, at the command prompt, type the following, and then press ENTER:

```
cd D:\DemoFiles\Mod02\amd64fre
```

5. Type the following, and then press ENTER:

```
.\diskspd.exe -c2G -r -t4 -w40 -o32 -b64K d:\test.dat; del d:\test.dat
```


6. Wait for the command to complete, and then review the output of the tool.
7. Close Windows PowerShell.

Virtual Servers and Containers

Traditionally, SQL Server was installed on dedicated hardware servers, often housed in a rack in a datacenter. To install a new server, you had to buy and set up new hardware, install an operating system, and then install the database software or use an imaging technique to deploy the operating system and database together.

In recent years, virtualization techniques have been developed to make server management easier and more dynamic. In virtualization, an application such as SQL Server executes in an environment that appears, from the application's point-of-view, to be a dedicated physical server with an operating system, CPU, memory, and storage resources. However, a single physical host can run several of these virtual execution environments within its own memory and by using its own resources. With virtualization, it's easy to create new virtual servers and easier to utilize the large resources available on modern server hardware. You can run SQL Server in two kinds of virtual environment.

- Virtualization:
 - Virtual Machines
 - Hyper-V
- Containerization:
 - Containers
 - Docker
- SQL Server Installation on VMs and Containers
- Cloud Virtualization and Containerization

Virtual Machines and Hyper-V®

A Virtual Machine (VM) is an isolated execution environment with an operating system and virtualized resources that are dedicated to that VM. A VM is stored as a hard disk image, which includes the operating system and applications such as SQL Server. When you run the virtual machine, changes are saved to the hard disk image file, which has a .vhd extension. The VM configuration is stored in a separate file.

Hyper-V is the server component that hosts the VMs and enables them to access real resources, such as the network interface card. Hyper-V is available as a Windows component in Windows Server® and Windows 10. Hyper-V is only one virtualization solution. Others include VMWare and KVM.

Hyper-V also support snapshots of each image. A snapshot records the complete status of the image, including the contents of all virtual hard drives and the virtual memory. You can, for example, use a snapshot to trial a new configuration or custom code. If a problem arises, you can apply the snapshot to return to the previous state.

You can use the same virtual hard disk image to create multiple VMs providing the image has been generalized—for example, by using the sysprep.exe tool. In a generalized image, unique values such as computer security identifiers (SIDs) have been removed. Multiple VMs created from such an image can coexist on the same network without conflicts. You can use a single generalized image that includes SQL Server, to deploy multiple SQL Server instances quickly.

For more information about Hyper-V and VMs, see:



Hyper-V on Windows Server 2016

<https://aka.ms/qatdzv>

Containers and Docker

Containerization is similar to virtualization except that running images share more of the resources. For example, if you execute two Windows 10 VMs in Hyper-V, they each run a separate copy of the Windows kernel because they share no virtual memory. By contrast, if you execute two Windows 10 containers, they can share a single instance of the Windows kernel and so save physical memory and CPU cycles.

Docker is the server component that hosts the containers in which applications execute. Docker manages containers and enables containers to access shared resources including network cards and the single kernel image.

Windows Server 2016 includes the Windows Containers server feature for container and docker support.

There are two principal advantages to using containers over VMs:

- Each image shares resources so you can execute more containers than VMs on a single physical server.
- Containers can be started more quickly than VMs, because a full boot-up sequence is unnecessary.

For more information about Windows Containers, see:



Windows Containers on Windows Server

<https://aka.ms/e8oaq4>

SQL Server Installation on VMs and Containers

You use Hyper-V to connect to a VM, and interact with its GUI, in much the same way as you do for a physical server. Therefore, the installation process for SQL Server on a VM is the same as for a physical server. You can use SQLIOSim and Diskspd to assess the capacity of the VM.

A container does not provide a user interface. Instead, you must interact with the container and its applications by using the command prompt and the docker.exe executable. For an example of SQL Server installation in a container, see:



SQL Server in Windows Containers

<https://aka.ms/oimhx9>

Cloud Virtualization and Containerization

Microsoft Azure® supports both VMs and Containers. This means you can dispense with an on-premises physical server and run an application such as SQL Server entirely in the cloud. Users can access servers over the Internet or through an encrypted Virtual Private Network (VPN). You will learn more about deploying SQL Server to Azure in Module 7.

Check Your Knowledge

Question	
If you need to run repeatable load tests on an I/O subsystem, which tool should you use?	
Select the correct answer.	
☐	Diskspd
☐	SQLIOSim

Lesson 2

tempdb Files

tempdb is a special database available as a resource to all users of a SQL Server instance; you use it to hold temporary objects that users, or the database engine, create.

Because it is a shared object, the performance of **tempdb** can have a significant impact on the overall performance of a SQL Server instance; you can take steps to improve the performance of **tempdb**.

Lesson Objectives

At the end of this lesson, you will be able to:

- Describe the purpose of **tempdb**.
- Describe special considerations needed when designing storage and allocating data files for **tempdb**.

What Are tempdb Files?

In many respects, **tempdb** files are identical to the files that make up other SQL Server databases. From the perspective of storage I/O, **tempdb** uses the same file structure as a user database—one or more data files and a log file. The arrangement of data pages within **tempdb** data files is also based on the same architecture as user databases.

Unlike all other databases, SQL Server recreates the **tempdb** database each time the SQL Server service starts. This is because **tempdb** is a temporary store.

There are three primary ways that the organization of **tempdb** files can affect system performance:

- Because users and the database engine both use **tempdb** to hold large temporary objects, it is common for **tempdb** memory requirements to exceed the capacity of the buffer pool—in which case, the data will spool to the I/O subsystem. The performance of the I/O subsystem that holds **tempdb** data files can therefore significantly impact the performance of the system as a whole. If the performance of **tempdb** is a bottleneck in your system, you might decide to place **tempdb** files on very fast storage, such as an array of SSDs.
- Although it uses the same file structure, **tempdb** has a usage pattern unlike user databases. By their nature, objects in **tempdb** are likely to be short-lived, and might be created and dropped in large numbers. Under certain workloads—especially those that make heavy use of temporary objects—this can lead to heavy contention for special system data pages, which can mean a significant drop in performance. One mitigation for this problem is to create multiple data files for **tempdb**; this is covered in more detail in the next topic.
- When SQL Server recreates the **tempdb** database following a restart of the SQL Server service, the size of the **tempdb** files returns to a preconfigured value. The **tempdb** data files and log file are configured to autogrow by default, so if subsequent workloads require more space in **tempdb** than is currently available, SQL Server will request more disk space from the operating system. If the initial size of **tempdb** and the autogrowth increment set on the data files is small, SQL Server might need to request additional disk space for **tempdb** many times before it reaches a stable size. Because a file-

- **tempdb** files are similar in architecture to data files for other SQL Server databases
- The usage pattern for **tempdb** is unlike other databases
 - Holds temporary objects created by users or by the database engine
- SQL Server recreates **tempdb** every time the instance restarts
- **tempdb** is a shared resource, so its performance can affect the performance of the whole system

growth operation requires an exclusive lock on the entire **tempdb** database, and **tempdb** is central to the function of SQL Server, each file-growth operation can pause the database engine for its duration. To avoid this problem, specify file size and autogrowth settings for **tempdb** that minimize the number of autogrowth operations during normal running.

For more information, see the *tempdb Database* topic in the SQL Server 2016 Technical Documentation:



tempdb Database

<http://aka.ms/d1jmp2>

Modifying the Number of tempdb Files During Installation

In SQL Server 2016, you can specify the number, size, and autogrowth characteristics of **tempdb** data files and log files during installation.

The setup program will determine the following default values:

- Number of **tempdb** data files: equal to the count of CPUs or eight, whichever is lower.
- Initial size of **tempdb** files: 8 MB (note that this value is per file).
- Autogrow increment for **tempdb** files: 64 MB.

• You can specify the following attributes of **tempdb** files during installation of SQL Server 2016:

- Location
- Number
- Initial size
- Autogrowth increment

This is a change from earlier versions of SQL Server, where configuring **tempdb** was a post-installation task.

With all versions of SQL Server, you can specify the location of the **tempdb** data files and log files during installation.

You can configure these settings on the **TempDB** tab in the **Database Engine Configuration** section of the installation process.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? Tables you create in tempdb will still be present after you restart the SQL Server instance.	

Lesson 3

Installing SQL Server 2016

After making the decisions about your SQL Server configuration, you can move to installation. In this lesson, you will see the phases that installation goes through and how SQL Server checks your system for compatibility by using the System Configuration Checker tool.

For most users, the setup program will report that all was installed as expected. For the rare situations where this does not occur, you will also learn how to carry out post-installation checks and troubleshooting.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe options for installing SQL Server.
- Install SQL Server.
- Perform post-installation checks.

Options for Installing SQL Server 2016

You can install SQL Server 2016 in different ways—by using the installation wizard, from the command prompt, by using a configuration file, and by using the SysPrep utility. You can choose the most appropriate installation process for your environment.

- Installation wizard
- Command prompt
- Configuration file
- SysPrep

Installation Wizard

The SQL Server installation wizard provides a simple user interface for installing SQL Server. You can use it to select all the components of SQL Server that you want to install. In addition to using it to create a new installation on the server, you can use it to add components to an existing one.



Note: You must be a local administrator to run the installation wizard on the local computer. When installing from a remote share, you need read and execute permissions.

Command Prompt

You can also run the SQL Server setup program from the command prompt, using switches to specify the options that you require. You can configure it for users to fully interact with the setup program, to view the progress without requiring any input, or to run it in quiet mode without any user interface. Unless you are using a volume licensing or third-party agreement, a user will always need to confirm acceptance of the software license terms.

Configuration File

In addition to using switches to provide information to the command prompt setup, you can use a configuration file. This can simplify the task of installing identically-configured instances across your enterprise.

The configuration file is a text file containing name/value pairs. You can manually create this file by running the installation wizard, selecting all your required options, and then, instead of installing the product, generating a configuration file of those options—or you could take the configuration file from a previously successful installation.

If you use a configuration file in conjunction with command prompt switches, the command prompt values will override any values in your configuration file.

SQL Server SysPrep

With SQL Server SysPrep, you can prepare an instance of SQL Server, and complete the installation later. This is useful when you want to prepare instances on multiple computers or multiple instances on one computer.

For more information, see the *Install SQL Server 2016* topic in the SQL Server 2016 Technical Documentation:



Install SQL Server 2016

<http://aka.ms/pzn817>

Performing an Installation of SQL Server 2016

To manually install the required components of SQL Server 2016, you should run the SQL Server 2016 setup program. There are many pages in the Installation Wizard, including:

- **Product Key.** If you are using anything other than the evaluation edition, you must enter the product key for your copy of SQL Server 2016.
- **License Terms.** To continue with the installation, you must accept the license terms.
- **Global Rules.** Verifies that the system is suitable for SQL Server installation. The process will only halt on this page if errors are detected.
- **Product Updates.** The installation process checks for any updates to prerequisite software.
- **Install Setup Files.** The installation process installs the setup files required to install SQL Server.
- **Install Rules.** The installation process checks for known potential issues that can occur during setup and requires you to rectify any that it finds before continuing.
- **Setup Role.** You must select the type of installation that you need to ensure that the process includes the correct feature components you require. The options are:
 - **SQL Server Feature Installation.** This option installs the key components of SQL Server, including the database engine, Analysis Services, Reporting Services, and Integration Services.

- Installation wizard:
 - Product Updates
 - Install Setup Files
 - Install Rules
 - Setup Role
 - Feature Selection
 - Instance Configuration
 - Server Configuration
 - <Component> Configuration
 - Ready to Install

- **All Features with Defaults.** This option installs all SQL Server features and uses the default options for the service accounts.

After you select one of these options, you can further customize the features to install on the next page of the wizard.

- **Feature Selection.** You can use this page to select the exact features you want to install. You can also specify where to install the instance features and the shared features.
- **Feature Rules.** The installation process checks for prerequisites of the features you have marked for installation.
- **Instance Configuration.** You must specify whether to install a default or named instance (if a default instance is already present, installing a named instance is your only option) and, if you opt for a named instance, the name that you want to use.
- **Server Configuration.** Specify the service account details and startup type for each service that you are installing.
- **<Component> Configuration.** You must configure component-specific settings for each component that you have selected to install.
- **Ready to Install.** Use this page to review the options you have selected throughout the wizard before performing the installation.
- **Complete.** When the installation is complete, you might need to reboot the server.



Note: This is the sequence of pages in the Installation Wizard when you install SQL Server 2016 on a server with no existing SQL Server 2016 instances. If SQL Server 2016 instances are already installed, most of the same pages are displayed; some pages appear at different positions in the sequence.

For more information, see the *Install SQL Server 2016 from the Installation Wizard (Setup)* topic in the SQL Server 2016 Technical Documentation:



Install SQL Server 2016 from the Installation Wizard (Setup)

<http://aka.ms/ug68a2>

Performing Post-Installation Checks

After installing SQL Server, the most important check is to make sure that all SQL Server services are running, by using the SQL Server services node in SSCM.



Note: SQL Server services have names that differ slightly to their displayed names. For example, the service name for the default **SQL Server** service is **MSSQLSERVER**. You can view the actual service name by looking on the properties page for the service.

- Verify that SQL Server services are running
- If necessary, view log file information at:
%ProgramFiles%\Microsoft SQL Server\130\Setup Bootstrap\Log

You do not need to check the contents of the SQL Server setup log files after installation, because the installer program will indicate any errors and attempt to reverse any of the SQL Server setup that has been completed to that point. When errors occur during the SQL Server setup phase, the installation of the SQL Server Native Access Client and the setup components is not reversed.

Typically, you only need to view the setup log files in two scenarios:

- If setup is failing and the error information displayed by the installer does not help you to resolve the issue.
- If you contact Microsoft Product Support and they ask for detailed information.

If you do require the log files, you will find them in the %Programfiles%\Microsoft SQL Server\130\Setup Bootstrap\Log folder.

Categorize Activity

Which of the following methods can be used to install SQL Server 2016? Indicate your answer by writing the category number to the right of each item.

Items	
1	Installation Wizard
2	Windows Update
3	Command Prompt
4	Command Prompt with a configuration file

Category 1		Category 2
Can be used to install SQL Server 2016		Cannot be used to install SQL Server 2016

Lesson 4

Automating Installation

After completing this lesson, you will be able to:

- Perform an unattended installation.
- Describe strategies for upgrading SQL Server.

SQL Server Servicing

As with all software products over time, issues can be encountered with SQL Server. The product group is very responsive in fixing any identified issues by releasing software updates.

SQL Server updates are released in several ways:

- Hotfixes (also known as QFE or Quick Fix Engineering) are released to address urgent customer concerns. Due to the tight time constraints, only limited testing can be performed on these fixes, so they should only be applied to systems that are known to have the issues that they address.
- Cumulative Updates (CUs) are periodic roll-up releases of hotfixes that have received further testing as a group.
- Service Packs (SPs) are periodic releases where full regression testing has been performed. Microsoft recommends applying SPs to all systems after appropriate levels of organizational testing.

- SQL Server updates are released in several ways
 - Hotfixes
 - Cumulative Updates
 - Service Packs
- SQL Server can receive automatic updates directly from Microsoft Update
 - In enterprise environments, updates should be tested before being applied to production servers

The simplest way to keep SQL Server up to date is to enable automatic updates from the Microsoft Update service. Larger organizations, or those with documented configuration management processes, should exert caution in applying automatic updates. It's likely that the updates should be applied to test or staging environments before being applied to production environments.

SQL Server 2016 can also have product SPs slipstreamed into the installation process to avoid having to apply them after installation.

For more information, see the *Overview of SQL Server Servicing Installation* topic in the SQL Server 2016 Technical Documentation:



Overview of SQL Server Servicing Installation

<http://aka.ms/yd8zlj>

Unattended Installation

In many organizations, senior IT administrators create script files for standard builds of software installations and use them to ensure consistency. Unattended installations can help with the deployment of multiple identical installations of SQL Server across an enterprise. Unattended installations can also facilitate the delegation of the installation to another person.

Unattended Installation Methods

One option for performing an unattended installation of SQL Server is to create an .ini file containing the required setup information and pass it as a parameter to setup.exe at a command prompt. An alternative is to pass all the required SQL Server setup details as parameters to the setup.exe program, rather than placing the parameters into an .ini file.

In both examples on the slide, the second method has been used. The first example shows a typical installation command and the second shows how an upgrade could be performed using the same method.

• Install SQL Server from the command line:

```
Setup.exe /q /ACTION=CompleteImage /INSTANCENAME=MYNEWINST
/INSTANCEID=<MYINST> /SQLSVCACCOUNT="<DomainName\UserName>"
/SQLSVCACCOUNT="<StrongPassword>"
/SQLSVCADMINACCOUNTS="<DomainName\UserName>"
/AGTSQLSVCACCOUNT="NT AUTHORITY\Network Service"
/ACCEPTSQLSERVERLICENSETERMS
```

• Upgrade SQL Server from the command line:

```
Setup.exe /q /ACTION=upgrade /INSTANCENAME=MSSQLSERVER
/RSUPGRADEDATABASEACCOUNT="<Provide a SQL Server login account
that can connect to the report server during upgrade>"
/RSUPGRADEPASSWORD="<Provide a password for the report server upgrade
account>" /ISSVCACCOUNT="NT Authority\Network Service"
/ACCEPTSQLSERVERLICENSETERMS
```

/q Switch

The "/q" switch shown in the examples specifies "quiet mode"—no user interface is provided. An alternative switch "/qs" specifies "quiet simple" mode. In the quiet simple mode, the installation runs and shows progress in the UI but does not accept any input.

Creating an .ini File

An .ini file for unattended installation can be created by using any text editor, such as Notepad. The SQL Server installation program creates a file called ConfigurationFile.ini in a folder that is named based upon the installation date and time, under the folder C:\Program Files\Microsoft SQL Server\130\Setup Bootstrap\Log. You can use this as a starting point for creating your own .ini file. The .ini file is composed of a single [Options] section containing multiple parameters, each relating to a different feature or configuration setting.



Note: Note that you can use the installation wizard to create new installation .ini files without carrying out a manual install. If you use the installation wizard all the way through to the **Ready to Install** stage, the wizard generates a new ConfigurationFile.ini with the settings you selected in C:\Program Files\Microsoft SQL Server\130\Setup Bootstrap\Log.

For more information, see the *Install SQL Server 2016 from the Command Prompt* topic in the SQL Server 2016 Technical Documentation:



Install SQL Server 2016 from the Command Prompt

<http://aka.ms/sjjkud>

For more information, see the *Install SQL Server 2016 Using a Configuration File* topic in the SQL Server 2016 Technical Documentation:



Install SQL Server 2016 Using a Configuration File

<http://aka.ms/suvulk>

Demonstration: Reviewing an Unattended Installation File

In this demonstration, you will review an unattended installation file.

Demonstration Steps

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are running and log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. On the taskbar, click the **File Explorer** shortcut.
3. In File Explorer, browse to **D:\Demofiles\Mod02**, and then double-click **ConfigurationFile.ini**.
4. Review the content in conjunction with the *Install SQL Server 2016 From the Command Prompt* topic in the SQL Server 2016 Technical Documentation. In particular, note the values of the following properties:
 - a. INSTANCEID
 - b. ACTION
 - c. FEATURES
 - d. QUIET
 - e. QUIETSIMPLE
 - f. INSTALLSHARED DIR
 - g. INSTANCEDIR
 - h. INSTANCENAME
 - i. AGTSVCSTARTUPTYPE
 - j. SQLCOLLATION
 - k. SQLSVCACCOUNT
 - l. SQLSYSADMINACCOUNTS
 - m. TCPENABLED
5. Close Notepad.
6. Close File Explorer.

Check Your Knowledge

Question	
What is the name of the configuration file generated by an installation of SQL Server 2016 using the Installation Wizard?	
Select the correct answer.	
☐	InstallationFile.ini
☐	ConfigurationFile.ini
☐	Wizard.ini
☐	Config.ini

Lab: Installing SQL Server 2016

Scenario

You have been tasked with creating a new instance of SQL Server that will be used by the IT department as a test server for new applications.

Objectives

After completing this lab, you will be able to:

- Assess resources available for a SQL Server installation.
- Install SQL Server 2016.
- Perform post-installation checks.
- Automate a SQL Server installation.

Estimated Time: 90 minutes

Virtual machine: **20765B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Preparing to Install SQL Server

Scenario

You are preparing to install SQL Server 2016 for the IT department in Adventure Works Cycles. Before installing, you want to find out if the server hardware provisioned for the instance is ready.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. View Hardware and Software Requirements
3. Run the System Configuration Checker

► Task 1: Prepare the Lab Environment

1. Ensure that the MSL-TMG1, 20765B-MIA-DC, and 20765B-MIA-SQL virtual machines are running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab02\Starter** folder as Administrator.

► Task 2: View Hardware and Software Requirements

1. Run **setup.exe** in the **X:** folder to run the SQL Server installation program.
2. In the SQL Server Installation Center, on the **Planning** page, view the **Hardware and Software Requirements**.

► Task 3: Run the System Configuration Checker

1. In the SQL Server Installation Center, on the **Tools** page, use the **System Configuration Checker** to assess the computer's readiness for a SQL Server installation.
2. Keep the SQL Server Installation Center window open. You will use it again in a later exercise.

Results: After this exercise, you should have run the SQL Server setup program and used the tools in the SQL Server Installation Center to assess the computer's readiness for SQL Server installation.

Exercise 2: Installing SQL Server

Scenario

The required configuration details for the new SQL Server 2016 instance you must install are described in the following table:

Item	Configuration value
Instance name	SQLTEST
Features	Database Engine only (excluding Replication, Full-Text, and DQS)
User database directory	M:\SQLTEST\Data
User database log directory	L:\SQLTEST\Logs
Service accounts	ADVENTUREWORKS\ServiceAcct / Pa\$\$w0rd for all services
Startup	Both SQL Server and SQL Server Agent should start manually
Server collation	SQL_Latin1_General_CP1_CI_AS
Authentication mode	Mixed
SA Password	Pa\$\$w0rd
Administrative user	ADVENTUREWORKS\Student
Filestream support	Disabled

The main tasks for this exercise are as follows:

1. Install the SQL Server Instance

► Task 1: Install the SQL Server Instance

- Install the required instance of SQL Server:
 - On the **Product Key** page, select **Evaluation** edition, which does not require a product key.
 - On the **Feature Selection** page, select only the features that are required.
 - On the **Server Configuration** page, configure the service account name and password, the startup type for the SQL Server Agent and SQL Server Database Engine services, and verify the collation.
 - On the **Database Engine Configuration** page, configure the authentication mode and the SA password; add the current user (Student) to the SQL Server administrators list, specify the required data directories, and verify that Filestream is not enabled.

Results: After this exercise, you should have installed an instance of SQL Server.

Exercise 3: Perform Post-Installation Checks

Scenario

In this exercise, you will start the SQL Server service for the new instance and connect to it using SSMS to make sure that the instance works.

The main tasks for this exercise are as follows:

1. Start the SQL Server Service
2. Configure Network Protocols and Aliases
3. Verify Connectivity to SQL Server

► Task 1: Start the SQL Server Service

1. Start SQL Server 2016 Configuration Manager, and view the properties of the **SQL Server (SQLTEST)** service.
2. Verify that the service is configured to log on as **ADVENTUREWORKS\ServiceAcct**, and then start the service.

► Task 2: Configure Network Protocols and Aliases

1. In SQL Server Configuration Manager, view the SQL Server network protocols configuration for the **SQLTEST** instance and verify that the **TCP/IP** protocol is enabled.
2. View the SQL Server Native Client 32-bit client protocols and verify that the **TCP/IP** protocol is enabled. Create an alias named **Test** that uses TCP/IP to connect to the **MIA-SQL\SQLTEST** instance from 32-bit clients.
3. View the SQL Server Native Client protocols and verify that the **TCP/IP** protocol is enabled. Create an alias named **Test** that uses TCP/IP to connect to the **MIA-SQL\SQLTEST** instance from 64-bit clients.

► Task 3: Verify Connectivity to SQL Server

1. Use `sqlcmd` to connect to the **MIA-SQL\SQLTEST** instance of SQL Server using a trusted connection, and run the following command to verify the instance name:

```
SELECT @@ServerName;  
GO
```

2. Use SQL Server Management Studio to connect to the **Test** alias.
3. In SQL Server Management Studio, in Object Explorer:
 - a. View the properties of the **Test** instance and verify that the value of the **Name** property is **MIA-SQL\SQLTEST**.
 - b. Stop the **Test** service.
4. Close SQL Server Management Studio.

Results: After this exercise, you should have started the SQL Server service and connected using SSMS.

Exercise 4: Automating Installation

Scenario

The Adventure Works development team want to install SQL Server database engine instances on several development servers.

The configuration of these instances should match the configuration of the SQLTEST instance you have just installed, with the following differences:

- The instance name should be SQLDEV.
- TCP/IP should be disabled.
- User database files and log files should all be placed in C:\devdb.
- The development servers have two CPU cores; you should configure **tempdb** accordingly.
- You must create a configuration file to standardize the installation process.

The main tasks for this exercise are as follows:

1. Review an Unattended Installation File
2. Update an Unattended Installation File

► Task 1: Review an Unattended Installation File

1. Open **D:\Labfiles\Lab02\Starter\ConfigurationFile.ini** using Notepad.
2. Review the content of the file, paying particular attention to the following properties:
 - a. INSTANCEID
 - b. INSTANCENAME
 - c. ACTION
 - d. FEATURES
 - e. TCPENABLED
 - f. SQLUSERDBDIR
 - g. SQLUSERDBLOGDIR
 - h. SQLTEMPDBFILECOUNT
3. Leave the file open in Notepad.

► Task 2: Update an Unattended Installation File

1. Return to the **ConfigurationFile.ini** file open in Notepad:
 - a. Amend the file to reflect the changes needed for this task.
 - b. The instance name should be SQLDEV.
 - c. TCP/IP should be disabled.
 - d. User database files and log files should all be placed in C:\devdb.
 - e. The development servers have two CPU cores; **tempdb** should be configured accordingly.
2. Save the file and compare your changes to the solution shown in the file D:\Labfiles\Lab02\Solution\SolutionConfigurationFile.ini.

Results: After this exercise, you will have reviewed and edited an unattended installation configuration file.

Module Review and Takeaways

In this module, you have learned about considerations for installing SQL Server, how to install SQL Server, and how to perform post-installation configuration tasks.

Review Question(s)

Question: What are the considerations for installing additional named instances on a server where SQL Server is already installed?

Module 3

Upgrading SQL Server to SQL Server 2016

Contents:

Module Overview	3-1
Lesson 1: Upgrade Requirements	3-2
Lesson 2: Upgrade of SQL Server Services	3-10
Lesson 3: Side-by-Side Upgrade: Migrating SQL Server Data and Applications	3-20
Lab: Upgrading SQL Server	3-31
Module Review and Takeaways	3-34

Module Overview

Occasionally, you might need to upgrade existing Microsoft® SQL Server® instances, services, and databases to a new version of SQL Server. This might arise for a number of reasons—for example:

- To replace existing server hardware.
- To consolidate existing SQL Server instances onto a single server.
- To use features only found in a new version of SQL Server.
- To continue to receive support and security patches for SQL Server from Microsoft (because mainstream support or extended support for your version is ending).
- To move a SQL Server instance between different editions of the same version.

This module will cover points you might need to consider when planning an upgrade to SQL Server 2016, and the different strategies you might use to deliver it.

For full details of the lifecycle dates for versions of SQL Server, see *Microsoft Support Lifecycle* on the Microsoft Support site:



Microsoft Support Lifecycle

<http://aka.ms/yyzozm>

Objectives

At the end of this module, you will be able to:

- Describe the different strategies available for upgrading existing SQL Server instances, services, and databases to SQL Server 2016.
- Explain the strengths and weaknesses of different migration strategies.
- Carry out an upgrade of an existing SQL Server instance to SQL Server 2016.

Lesson 1

Upgrade Requirements

Before beginning an upgrade of an existing SQL Server to SQL Server 2016, you should draw up an upgrade plan. This lesson covers the points you will need to consider.

Lesson Objectives

At the end of this lesson, you will be able to:

- Draw up an upgrade plan.
- Identify the versions and editions of SQL Server suitable for upgrading to SQL Server 2016.
- Describe the in-place and side-by-side strategies for upgrading to SQL Server 2016.
- Use the SQL Server 2016 upgrade advisor.
- Use the Distributed Replay Utility to evaluate server performance.

Creating an Upgrade Plan

Before you start an upgrade of a SQL Server installation, there are several points to consider.

Assess the characteristics of the system to be upgraded

- Which SQL Server features are installed?
- What are the usage characteristics of the installed features?
- Are all the features in use supported by SQL Server 2016?
 - Tools (covered in this module) are available to assist with this step.
- What are the performance characteristics of the existing hardware?

- Assess system to be upgraded
- Confirm that upgrade is supported
- Select an upgrade strategy:
 - In-place upgrade
 - Side-by-side upgrade
- Check hardware resources
- Identify acceptance criteria

You might decide to plan for application testing to confirm that your application will continue to operate correctly with SQL Server 2016. Changes to your application could be required to correct any issues discovered during testing.

Check your upgrade path

You should confirm whether a supported upgrade path exists between your current version of SQL Server and the proposed upgrade edition of SQL Server 2016. A direct upgrade is not possible between all previous versions and editions of SQL Server and SQL Server 2016. Whether or not a direct upgrade path is available might inform your choice of upgrade strategy.

Select an upgrade strategy

You will perform one of the following:

- An in-place upgrade.
- A side-by-side upgrade.

Upgrade strategies are covered in detail later in this module.

Check hardware resources

As part of this process, you must determine whether sufficient hardware resources are in place to support your chosen upgrade strategy, and to support the performance characteristics appropriate for your application. Other modules in this course cover tools and techniques for assessing hardware requirements.

Identify acceptance criteria

Clear acceptance criteria will help you determine whether the upgrade was successful and whether you need to roll back to the previous version. Your choice of upgrade strategy will influence the options for rolling back the upgrade.

The recommended practice is to treat a SQL Server upgrade as you would any other IT project by assembling a team with the appropriate skills and developing an upgrade plan.

For more information about all aspects of upgrading to SQL Server 2016, see *Upgrade to SQL Server 2016* in the SQL Server 2016 Technical Documentation:



Upgrade to SQL Server 2016

<http://aka.ms/dem1hx>

Supported Versions and Editions

You cannot upgrade every previous version of SQL Server to SQL Server 2016.

- There is a direct upgrade path to SQL Server 2016 from SQL Server 2008, 2008 R2, 2012, and 2014. These versions must be at a minimum patch level before upgrade can be attempted:
 - SQL Server 2008 SP3 or later.
 - SQL Server 2008 R2 SP2 or later.
 - SQL Server 2012 SP1 or later.
 - SQL Server 2014 or later.
- Limited support is available for upgrade from SQL Server 2005.
 - A direct upgrade from SQL 2005 is not possible.
 - SQL Server 2016 can restore database engine and Analysis Services backups from SQL Server 2005.
 - SQL Server 2016 can attach database data and log files from SQL Server 2005.
- No in-place upgrade path is available for SQL Server versions before 2005.
 - To upgrade in-place from an older version, an interim upgrade to a version where a supported upgrade path is available must be carried out. For example, a SQL Server 2000 instance could be upgraded to SQL Server 2008 R2 SP2, then upgraded to SQL Server 2016.

• In-place upgrades to SQL 2016 are supported from:

- SQL Server 2008 SP3 or later
- SQL Server 2008 R2 SP2 or later
- SQL Server 2012 SP1 or later
- SQL Server 2014 or later

• Older versions require an interim upgrade to allow upgrade in-place

• In-place upgrades must take place between compatible editions

In-place upgrades are permitted only between compatible editions of SQL Server. In general, you can move from a lower-featured edition to an equally featured or higher featured edition as part of an upgrade to SQL Server 2016; however, you cannot move from a higher featured edition to a lower featured edition.

For example:

- A SQL Server 2008 R2 Standard Edition instance could be upgraded to SQL Server 2016 Standard Edition, Enterprise Edition, or Business Intelligence Edition.
- A SQL Server 2008 R2 Enterprise Edition instance could be upgraded to SQL Server 2016 Enterprise Edition or Business Intelligence Edition. An upgrade to SQL Server 2016 Standard Edition would not be permitted.

For a full list of valid migration sources and targets, see *Supported Version and Edition Upgrades* in the SQL Server 2016 Technical Documentation:



Supported Version and Edition Upgrades

<http://aka.ms/ssdh5f>

For more information about upgrading to SQL Server 2016 from SQL Server 2005, see *Are you upgrading from SQL Server 2005?* in the SQL Server 2016 Technical Documentation:



Are you upgrading from SQL Server 2005?

<http://aka.ms/dc31wf>

In-Place and Side-by-Side Upgrades

There are two different ways to perform SQL Server upgrades. Each method has benefits and limitations, and is appropriate in certain circumstances.

In-Place Upgrade

An in-place upgrade occurs when you replace the installed version of SQL Server with a new version. This is a highly automated, and therefore easier, method of upgrading. However, an in-place upgrade is not without risks. If the upgrade fails, it is much harder to return to the previous operating state by reverting to the older version of SQL Server. For your organization, you will need to decide whether this risk outweighs the benefit of a more straightforward upgrade process.

When you are weighing up this risk, you need to consider that it might not be the SQL Server upgrade that fails. Even if the SQL Server upgrade works as expected, a client application might fail to operate as anticipated on the new version of SQL Server. In this case, the need to recover the situation quickly will be just as important as if the upgrade of SQL Server software had failed.

In-place upgrades have the added advantage of minimizing the need for additional hardware resources and avoiding the redirection of client applications that are configured to work with the existing server.

In-place upgrades have some restrictions:

- The upgrade must be between supported versions of SQL Server (see the previous topic for more details).
- All SQL Server components must be upgraded as part of an in-place upgrade. For instance, you cannot upgrade the database engine to SQL Server 2016 but leave SQL Server Analysis Services (SSAS) on the same server as an earlier version.

In-place	Side-by-side
Upgrade Benefits	Upgrade Benefits
Mostly automated	More granular control over process
System data upgraded	Use to perform test migration
No additional hardware	Relatively straightforward rollback
Apps pointing to same names	Can leverage failover/switchover

- Side-by-side upgrades can be carried out using one server or two servers
- Side-by-side upgrades are typical in virtualized/cloud environments

- An in-place upgrade of an instance from 32-bit to 64-bit versions of SQL Server (or vice versa) is not supported.

Side-by-Side Upgrade

In a side-by-side upgrade, databases are migrated from the existing SQL Server instance to a new SQL Server 2016 instance. The migration might be carried out using database backups, detach and reattach of data files, or through data transfer between databases using the Copy Database Wizard, BCP, SSIS, or another ETL tool.

A side-by-side upgrade is subject to less risk than an in-place upgrade, because the original system stays in place; it can be quickly returned to production should an upgrade issue arise. However, side-by-side upgrades involve extra work and more hardware resources.

There are two ways to carry out a side-by-side upgrade:

- **One server.** A SQL Server 2016 instance is installed alongside the instance to be upgraded on the same hardware. One-server side-by-side upgrades are less common in virtualized and cloud IT infrastructures.
- **Two servers.** SQL Server 2016 is installed on different hardware from the old instance.

A side-by-side upgrade offers a method to upgrade between versions and editions of SQL Server where no in-place upgrade path exists—such as moving a database from an Enterprise Edition instance to a Standard Edition instance.

Whether one or two servers are used, you will need enough hardware resources to provide for both the original and the new systems to perform a side-by-side upgrade. Common issues associated with side-by-side upgrades include:

- Configuration of server-level objects and services (for example, logins and SQL Server Agent jobs) on the new SQL Server 2016 instance.
- Time taken to copy all the user databases to a new location.
- Disk space required to hold data being transferred.

One-server side-by-side upgrades have some additional restrictions:

- Not all versions of SQL Server are supported when installed side-by-side on the same hardware.

For information on versions of SQL Server that might be installed side-by-side on the same server, see the topic *Using SQL Server Side-By-Side with Previous Versions of SQL Server*, in *Work with Multiple Versions and Instances of SQL Server*, in the SQL Server 2016 Technical Documentation:



Work with Multiple Versions and Instances of SQL Server

<http://aka.ms/r1oc97>

Hybrid Options

You can also use some elements of an in-place upgrade and a side-by-side upgrade together. For example, rather than copying all the user databases, after installing the new version of SQL Server beside the old version—and migrating all the server objects such as logins—you could detach user databases from the old server instance and reattach them to the new one.

After user databases have been attached to a newer version of SQL Server, they cannot be reattached to an older version, even if the database compatibility settings have not been upgraded. You need to consider this risk when you use a hybrid approach.

Rolling Upgrade

To maximize up time and minimize risk, a more complex approach might be required if you are upgrading an instance that employs High Availability (HA) functionality.

The details of your upgrade plan will vary, depending on which HA features you are using, including:

- Always-on availability groups
- Failover clustering
- Mirroring
- Log shipping
- Replication
- Reporting Services scale-out

To assist in your upgrade planning, see *Choose a Database Engine Upgrade Method* in the SQL Server 2016 Technical Documentation:



Choose a Database Engine Upgrade Method

<http://aka.ms/f2xd0w>

Upgrade Advisor

To help you identify areas of your application that might be affected by an upgrade to SQL Server 2016, Microsoft offers the Upgrade Advisor tool. The tool must be downloaded from the Microsoft website and is not included on the SQL Server 2016 installation media (there is a link to the Upgrade Advisor download page in the Planning section of the SQL Server Installation Center).

Upgrade Advisor runs a set of rules against your user databases to identify whether you are using features where behavior has changed between your current version of SQL Server and SQL Server 2016. This includes tests for features that are deprecated.

The tool generates a report that lists any issues found and rates them with a severity:

- **High.** The issue is a breaking change that will cause problems after migration.
- **Warning.** The issue can cause problems after migration.
- **Information.** For information only.

Where relevant, the report will include affected database object names, the affected line(s) of code, and a suggested resolution.

Lists of issues are generated for SQL Server 2016 and also for future versions (listing features that are still supported in SQL Server 2016 but marked for deprecation in a future version of SQL Server).

Reports generated by Upgrade Advisor can be exported to .html or .csv files.

If any high severity issues are reported by Upgrade Advisor for SQL Server 2016, you must modify your application to resolve them before proceeding with the upgrade.

- Assists in identifying breaking changes or deprecated features in SQL Server 2016 that might affect your application
- Available as a download from Microsoft—not included on the SQL Server 2016 installation media
- Upgrade Advisor cannot assess compatibility of T-SQL statements generated by client applications or user queries



Note: Upgrade Advisor can only check compatibility of code in database objects with SQL Server 2016 (user-defined functions, stored procedures, and so on). Database users and client applications might generate and execute Transact-SQL statements against the instance to be upgraded; Upgrade Advisor cannot assess the compatibility of SQL statements generated by users and applications.

Distributed Replay Utility

Upgrade Advisor cannot test the compatibility of client application code with SQL Server 2016 (as mentioned in the previous topic).

One way to test client application code compatibility is to carry out manual testing, but depending on the size and complexity of your production applications, this might not give an accurate representation of production activity. An alternative method to test client application code is to use the Distributed Replay Utility.

- Replays trace activity captured by SQL Server Profiler
 - Can be used to test code compatibility, performance, or both
- Replay is distributed over multiple clients (up to 16) to better simulate workloads

The Distributed Replay Utility operates by replaying client application activity you have captured from one SQL Server instance (typically a production instance) against a target SQL Server instance (typically an instance under test, not in production), reporting any errors or warnings this activity generates on the target SQL Server instance. This operation has several applications, including performance tuning for hardware and software, in addition to assessing client application code compatibility with SQL Server 2016.

The Distributed Replay Utility uses as its input a file of trace data captured by the SQL Server Profiler utility that records all client activity against the source SQL Server instance.



Note: The results from testing with the Distributed Replay Utility are only as representative as the content of the source trace data files. You will need to consider how long the trace that captures client application activity needs to run to capture a representative workload, and trade this off against the size of the captured data.



Additional Reading: For more information on the SQL Server Profiler utility and SQL Server Profiler traces, see course 20764: *Administering a SQL Database Infrastructure*.

SQL Server Profiler trace data files must be preprocessed before they are used by the Distributed Replay Utility.

The Distributed Replay Utility consists of two components: a server and a client. Either or both of these components can be installed during SQL Server installation.

When using the Distributed Replay Utility, you will have one server and one or more clients—up to a maximum of 16. The server coordinates test activity, whilst the clients replay commands as assigned to them by the server. The server and clients might be installed on the same or different hardware.

Distributing the replay across multiple clients results in a better simulation of activity on the source system. Workloads from source systems with high levels of activity, which would not be possible to replicate using a single replay client, can also be replayed.

At the end of the replay, you will manually review the output from each replay client, looking for commands that generated error messages. These errors could indicate client application code that is not compatible with the target server.

For more information on installing, configuring and using the Distributed Replay Utility, see *SQL Server Distributed Replay* in the SQL Server 2016 Technical Documentation:



SQL Server Distributed Replay

<http://aka.ms/l7pvpo>

Demonstration: Preparing for an Upgrade with Upgrade Advisor

In this demonstration, you will see:

- How to install Upgrade Advisor.
- How to run Upgrade Advisor.

Demonstration Steps

1. Ensure that the 20765B-MIA-DC-UPGRADE and 20765B-MIA-SQL-UPGRADE virtual machines are running and log on to 20765B-MIA-SQL-UPGRADE as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the **D:\Demofiles\Mod03** folder as Administrator.
3. When the script has completed, press any key to close the window.
4. Install **Upgrade Advisor** by executing **D:\Demofiles\Mod03\SqlAdvisor.msi**.
5. In the Microsoft SQL Server 2016 Upgrade Advisor Setup window, click **Next**.
6. Select the **I accept the terms in the License Agreement** check box, and then click **Next**.
7. Select the **I agree to the Privacy Policy** check box, and then click **Install**.
8. In the **User Account Control** dialog box, click **Yes**, and then click **Finish**.
9. On the **Start** screen, type **Microsoft SQL Server 2016 Upgrade Advisor**, and then click **Microsoft SQL Server 2016 Upgrade Advisor**.
10. Click **RUN DATABASE UPGRADE ANALYZER**.
11. Click **SELECT DATABASES TO ANALYZE**.
12. In the **SERVER NAME** box, type **MIA-SQL**, and then click **Connect**.
13. In the list of databases, click **TSQL** and **MDS**, click **Select**, and then click **Run**.
14. When analysis is complete, in the **Analyzed databases** blade, click **TSQL**.
15. In the **Analysis results** blade, click **130 - SQL Server 2016**.
16. In the **Compatibility results** blade, click **SET ROWCOUNT used in context of an INSERT / UPDATE DELETE**.
17. In the **Rule result** blade, in the element **IMPACTED OBJECTS** section, click **1**. Note that details of the object and line of code affected are reported.
18. Close SQL Server 2016 Upgrade Advisor.
19. In the **Microsoft SQL Server 2016 Upgrade Advisor** dialog box, click **Yes**.

Check Your Knowledge

Question	
Which of the following is not a version of SQL Server for which a direct upgrade path exists to SQL Server 2016?	
Select the correct answer.	
☐	SQL Server 2014
☐	SQL Server 2008
☐	SQL Server 2008 R2
☐	SQL Server 2012
☐	SQL Server 2000

Lesson 2

Upgrade of SQL Server Services

Your upgrade plan should take account of all the SQL Server features that are installed on an instance that you wish to upgrade to SQL Server 2016.

Some SQL Server features might require additional work to facilitate the upgrade process; the nature of this work might vary, depending on whether you are planning to use an in-place or side-by-side migration strategy.

Lesson Objectives

At the end of this lesson, you will be able to:

- Describe special considerations when upgrading Analysis Services instances.
- Describe special considerations when upgrading Database Engine instances.
- Describe special considerations when upgrading Data Quality Services.
- Describe special considerations when upgrading Integration Services.
- Describe special considerations when upgrading Master Data Services.
- Describe special considerations when upgrading Power Pivot for SharePoint®.
- Describe special considerations when upgrading Replicated Databases.
- Describe special considerations when upgrading and migrating Reporting Services.
- Describe special considerations when upgrading SQL Server management tools.
- Carry out an in-place upgrade of a SQL Server instance.

Upgrading Analysis Services

- Supports in-place upgrade: Yes.
- Supports side-by-side upgrade: Yes.

An in-place upgrade of Analysis Services can be carried out automatically using the SQL Server setup utility.

A side-by-side upgrade of Analysis Services requires that the Analysis Services databases be backed up from the source instance and restored on the target instance.

The target Analysis Services instance must use the same server mode (Tabular or Multidimensional) as the source instance.

For more information on this topic, see *Upgrade Analysis Services* in the SQL Server 2016 Technical Documentation:



Upgrade Analysis Services

<http://aka.ms/h1b1fn>

- In-place upgrade supported
- Side-by-side upgrade supported
 - Target server must be configured with the same server mode (Tabular or Multidimensional) as the source server

Upgrading Database Engine

- Supports in-place upgrade: Yes.
- Supports side-by-side upgrade: Yes.

An in-place upgrade of the database engine can be carried out automatically using the SQL Server setup utility.

A side-by-side upgrade of the database engine requires transfer of user databases from the source instance to the target instance, typically using database backup, or detach and attach data and log files.

- In-place upgrade supported
- Side-by-side upgrade supported
 - Objects stored in system databases must be migrated to the target instance, including:
 - Logins
 - SQL Server Agent jobs
 - Server-level triggers
 - Integration Services packages stored in MSDB
 - Reporting Services encryption keys
- After an upgrade, update database statistics



Additional Reading: For more information on working with database backups, data files, and log files, see the module *Working with Databases* later in this course, and course 20764: *Administering a SQL Database Infrastructure*.

Objects stored in system databases (master, msdb) must also be transferred to the target server. This might include:

- Logins.
- SQL Server Agent jobs.
- Server-level triggers.
- Integration Services packages stored in MSDB.
- Reporting Services encryption keys.

After an upgrade, you should update statistics on all user databases using **sp_updatestats**.

For more information on this topic, see *Upgrade Database Engine* in the SQL Server 2016 Technical Documentation:



Upgrade Database Engine

<http://aka.ms/ektsvf>

Upgrading Data Quality Services

- Supports in-place upgrade: Yes.
- Supports side-by-side upgrade: Yes.

Data Quality Services (DQS) databases are user databases, so are covered by the criteria in the *Upgrading Database Engine* topic earlier in this lesson.

- In-place upgrade supported
- Side-by-side upgrade supported
- DQS database is treated like any other user database
- DQS schema must be upgraded as a separate step
- All Data Quality Client instances must be upgraded using the setup utility

Whether you undertake an in-place or side-by-side upgrade, there are two additional steps you must carry out to upgrade DQS:

- Upgrade each computer where the Data Quality Client is installed.
- Upgrade the schema of the DQS databases using the command-line utility **dsqinstaller**.

For more information on this topic, see *Upgrade Data Quality Services* in the SQL Server 2016 Technical Documentation:



Upgrade Data Quality Services

<http://aka.ms/xcfpuo>

Upgrading Integration Services

- Supports in-place upgrade: No.
- Supports side-by-side upgrade: Yes.

A SQL Server Integration Services (SSIS) upgrade is always a side-by-side upgrade; in a one-server scenario, the new version of the SSIS components does not replace the old version; it is installed in addition to the old version.

- In-place upgrade not supported
- Side-by-side upgrade supported
- Behavior of the upgrade will vary when SSIS and the database engine are installed on the same machine and upgraded together
- Upgrade will not update SSIS packages to the SQL 2016 format



Note: SQL Server 2016 has no support for migrating or executing Data Transformation Services (DTS) packages. SQL Server 2012 and 2014 have the same restriction so upgrades to SQL Server 2016 from these versions will not be affected.

You should confirm whether you use SSIS packages that incorporate DTS packages, if you are upgrading to SQL Server 2016 from SQL Server 2008 or 2008 R2.

The behavior of the upgrade will vary, depending on whether SSIS is installed on the same machine as the database engine, and whether both the database engine and SSIS are upgraded at the same time.

- When SSIS and the database engine are upgraded together on the same machine, the upgrade process will remove the system tables used to store SSIS packages in earlier versions of SQL Server, and replace them with the SQL Server 2016 versions of these tables. This means that earlier versions of the SQL Server client tools cannot manage or execute SSIS packages stored in SQL Server.
- When only the database engine is upgraded, the old versions of the tables continue to be used in SQL Server 2016.

Regardless of whether SSIS and the database engine are upgraded at the same time or not, the upgrade process will *not*:

- Remove earlier versions of the Integration Services service.
- Migrate existing SSIS packages to the new package format used by SQL Server 2016.
- Move packages from file system locations other than the default location.
- Update SQL Agent job steps that call the **dtexec** utility with the file system path for the SQL Server 2016 version of **dtexec**.

For more information on this topic, see *Upgrade Integration Services* in the SQL Server 2016 Technical Documentation:



Upgrade Integration Services

<http://aka.ms/qhump>

Upgrading Master Data Services

- Supports in-place upgrade: Yes.
- Supports side-by-side upgrade: Yes.

The Master Data Services database (normally called MDS) is a user database, covered by the criteria in the Upgrading Database Engine topic earlier in this lesson.

Whether you undertake an in-place or side-by-side upgrade, you must carry out additional steps to upgrade Master Data Services:

- Upgrade the schema of the MDS database (using the Upgrade Database Wizard in Master Data Services Configuration Manager).
 - Any customizations you have made to objects in the MDS database will be overwritten during this upgrade.
- Install the SQL Server 2016 Master Data Services web application.
 - When the schema upgrade of the MDS database is complete, any old version of the MDS web application will no longer be able to connect to the database.
- Upgrade any clients using the Master Data Services Add-In for Excel® to the SQL Server 2016 version of the add-in.
 - When the schema upgrade of the MDS database is complete, clients using any old version of the Master Data Services Add-In for Excel can no longer connect to the database.

- In-place upgrade supported
- Side-by-side upgrade supported
- MDS database schema must be updated
- SQL 2016 MDS web application must be installed
- MDS Add-In for Excel must be updated
- You can in-place upgrade MDS to SQL 2016 without upgrading the database engine

You can carry out an in-place upgrade of Master Data Services to SQL Server 2016 without upgrading the database engine hosting the MDS database.

For more information on this topic, see *Upgrade Master Data Services* in the SQL Server 2016 Technical Documentation:



Upgrade Master Data Services

<http://aka.ms/e2vkak>

Upgrading Power Pivot for SharePoint

- Supports in-place upgrade: Partially.
- Supports side-by-side upgrade: Yes.
 - Servers running Analysis Services in SharePoint mode can undergo a side-by-side or in-place upgrade.
 - The Power Pivot for SharePoint Add-In is always installed side-by-side with earlier installations and cannot be upgraded in-place.

- In-place upgrade supported for Analysis Services in SharePoint mode
- Side-by-side upgrade supported
- Upgrade procedure different for SharePoint 2010 and SharePoint 2013
- Power Pivot for Excel workbooks will function without an upgrade, except for scheduled data refresh

When you upgrade Power Pivot for SharePoint to SQL Server 2016, you need to install or upgrade SharePoint components on your SharePoint server or server farm.

Upgrade prerequisites and the details of the upgrade procedure for Power Pivot for SharePoint vary, depending on whether you are using SharePoint 2010 or SharePoint 2013. However, the summary of the upgrade is identical for both versions of SharePoint:

- Upgrade all servers running Analysis Services in SharePoint mode (at a minimum, the POWERPivot instance must be upgraded).
- Install and configure the SQL Server 2016 Power Pivot for SharePoint Add-In on all servers in the SharePoint farm. In a multiserver farm, when the upgrade is completed on one server, the remaining servers in the farm become unavailable until they are upgraded.

The upgrade will not upgrade Power Pivot workbooks that run on the SharePoint servers, but workbooks created using previous versions of Power Pivot for Excel will continue to function.

- The exception to this is workbooks using scheduled data refresh. These workbooks must be using a version of Power Pivot for Excel that matches the server version. You must manually upgrade these workbooks, or use the auto-upgrade for data refresh feature in SharePoint 2010.

For more information, including the detailed steps required to upgrade SharePoint 2010 and SharePoint 2013 servers, see the topic *Upgrade Power Pivot for SharePoint* in the SQL Server 2016 Technical Documentation:



Upgrade Power Pivot for SharePoint

<http://aka.ms/uvim08>

Upgrading Replicated Databases

- Supports in-place upgrade: Yes.
- Supports side-by-side upgrade: Yes.



Note: To carry out a two-server side-by-side upgrade of databases involved in a replication topology without having to reconfigure replication or requiring a new snapshot to be delivered to subscribers, the server and database names on the new hardware should match the server and database names on the old hardware. This requirement will add complexity to your upgrade plan.

- In-place upgrade supported
- Side-by-side upgrade supported
- Nodes may run different versions of SQL Server
- Distributor version must be greater than or equal to the Publisher version
- Transactional replication subscribers must be within two versions of the Publisher

SQL Server supports replication of data between nodes running different versions of SQL Server. As a result, nodes in a SQL Server replication topology can be upgraded to SQL Server 2016 independently of one another, without halting activity across the whole topology.

Some limitations apply:

- A Distributor must be running a version of SQL Server greater than or equal to the Publisher version. Correspondingly, the Publisher must be running a lesser or equal version of SQL Server than its Distributor.
- Subscribers to a transactional publication must be running a version within two versions of the Publisher version. For example, a SQL Server 2016 publisher can have Subscribers running SQL Server 2012 or 2014.
- There are no version restrictions on Subscribers to a merge publication.

When upgrading instances where the log reader agent is running for transactional replication, you must stop activity on the database and allow the log reader agent to process any pending operations before stopping it. When this process is complete, you can upgrade to SQL Server 2016.

For merge replication instances, the Merge Agent and Snapshot Agent should be run for each subscription following an upgrade.

For more information, see the topic *Upgrade Replicated Databases* in the SQL Server 2016 Technical Documentation:



Upgrade Replicated Databases

<http://aka.ms/lako1d>

Upgrading and Migrating Reporting Services

- Supports in-place upgrade: Yes.
- Supports side-by-side upgrade: Yes.



Note: One-server side-by-side upgrades are only supported for Reporting Services running in native mode.



Note: An in-place upgrade cannot be used to switch a Reporting Services installation between native mode and SharePoint mode.

- In-place upgrade supported
- Side-by-side upgrade supported
 - One-server side-by-side upgrade only supported in native mode
- Reporting Services Add-In for SharePoint must also be upgraded when using SharePoint mode

Before embarking on an upgrade:

- Back up your Reporting Services encryption keys.
- Back up customizations to Reporting Services virtual directories in Internet Information Services (IIS).
- Remove invalid/expired SSL certificates from IIS. The presence of an invalid SSL certificate will cause the installation of Reporting Services to fail.

The upgrade process for Reporting Services will be different, depending on whether you use Reporting Services in native mode or in SharePoint mode. When Reporting Services is running in SharePoint mode, you will need to upgrade the Reporting Services Add-In for SharePoint after the Reporting Services service has been upgraded.

If you are using Reporting Services in native mode with a scale-out deployment (where the deployment includes more than one report server), you must remove all the members from the scaled-out deployment group before upgrading them. As servers are upgraded, they can be added back into the scaled-out deployment group.

For more information, see the topic *Upgrade and Migrate Reporting Services* in the SQL Server 2016 Technical Documentation:



Upgrade and Migrate Reporting Services

<http://aka.ms/vqkxy0>

For more information on upgrading the Reporting Services Add-In for SharePoint, see the topic *Install or Uninstall the Reporting Services Add-in for SharePoint* in the SQL Server 2016 Technical Documentation:



Install or Uninstall the Reporting Services Add-in for SharePoint

<http://aka.ms/x9ehkc>

For more information on the steps needed to complete a two-server side-by-side upgrade of Reporting Services, see the topic *Migrate a Reporting Services Installation (Native Mode)* in the SQL Server 2016 Technical Documentation:



Migrate a Reporting Services Installation (Native Mode)

<http://aka.ms/c6tn20>

Upgrading SQL Server Management Tools

- Supports in-place upgrade: No.
- Supports side-by-side upgrade: Yes.

SQL Server management tools (including SQLCMD, SQL Server Management Studio, and SQL Server Profiler) are always installed side-by-side with earlier versions of the tools. An in-place upgrade is not permitted.

When the upgrade to SQL Server 2016 is complete, you might opt to remove older versions of the management tools through the Windows® Control Panel's Programs and Features menu.

You should always manage SQL Server 2016 instances through the SQL Server 2016 management tools.

- In-place upgrade not supported
- Side-by-side upgrade supported
- PATH environment variable should be checked to confirm which version of the tools will be accessed by default



Note: You might need to update the PATH environment variable (or specify a full path for executable files) to ensure that you can use the new version of command-line tools and utilities.

For more information, see the topic *Upgrade SQL Server Management Tools* in the SQL Server 2016 Technical Documentation:



Upgrade SQL Server Management Tools

<http://aka.ms/hlqv5o>

Upgrading Using Setup

In-Place Upgrade to SQL Server 2016

SQL Server setup includes support for upgrading SQL Server in-place. In SQL Server Installation Center, use **Upgrade from a previous version of SQL Server** in the **Installation** section, and follow the upgrade wizard.



Note: Features cannot be added or removed during an in-place upgrade.

- SQL Server setup program has wizards for:
 - In-place upgrade
 - Changing the edition of an existing SQL 2016 instance
- Upgrade can also be performed from the command line using setup.exe

For more information on using SQL Server setup to upgrade to SQL Server 2016, see the topic *Upgrade to SQL Server 2016 Using the Installation Wizard (Setup)* in the SQL Server 2016 Technical Documentation:



Upgrade to SQL Server 2016 Using the Installation Wizard (Setup)

<http://aka.ms/dv3nvp>

Upgrading to a Different Edition of SQL Server 2016

SQL Server setup includes support for changing the edition of an installed instance of SQL Server 2016. In SQL Server Installation Center, use **Edition Upgrade** in the **Maintenance** section.

For a full list of supported upgrades between different editions of SQL Server 2016, see the *SQL Server 2016 Edition Upgrade* section of the *Supported Version and Edition Upgrades* topic in the SQL Server 2016 Technical Documentation:



Supported Version and Edition Upgrades

<http://aka.ms/ssdh5f>

For more information on using SQL Server setup to amend the edition of a SQL Server 2016 installation, see the topic *Upgrade to a Different Edition of SQL Server 2016 (Setup)* in the SQL Server 2016 Technical Documentation:



Upgrade to a Different Edition of SQL Server 2016 (Setup)

<http://aka.ms/itd5ma>



Note: You can carry out an upgrade or change of edition using the command-line interface for **setup.exe**.

Demonstration: Carry Out an In-Place Upgrade

In this demonstration, you will see how to carry out an in-place upgrade of a SQL Server 2014 installation to SQL Server 2016 using the upgrade wizard.

Demonstration Steps

1. Ensure that the 20765B-MIA-DC-UPGRADE and 20765B-MIA-SQL-UPGRADE virtual machines are running, and log on to 20765B-MIA-SQL-UPGRADE as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In File Manager, double-click **X:\setup.exe**.
3. In the **User Account Control** dialog box, click **Yes**.
4. In SQL Server Installation Center, click **Installation**, and then click **Upgrade from a previous version of SQL Server**.
5. On the **Product Key** page, click **Next**.
6. On the **License Terms** page, select **I accept the license terms**, and then click **Next**.
7. On the **Product Updates** page, click **Next**. Any error relating to a failure to search for updates through Windows Update can be ignored.
8. On the **Select Instance** page, set the value of the **Instance to upgrade** box to **MSSQLSERVER**, and then click **Next**.
9. On the **Select Features** page, click **Next**.
10. On the **Reporting Services SharePoint Mode** page, select **Continue with the upgrade**, and then click **Next**.
11. On the **Instance Configuration** page, click **Next**.
12. On the **Server Configuration** page, click **Next**.
13. On the **Full-text Upgrade** page, click **Next**.

14. The demonstration stops at this point because you cannot complete the upgrade with an Evaluation version of SQL Server.
15. On the **Feature Rules** page, click **Cancel**, and then click **Yes** to Cancel the installation.
16. Close the SQL Server Installation Center.

Categorize Activity

Place each SQL Server component into the appropriate category. Indicate your answer by writing the category number to the right of each item.

Items	
1	Database Engine
2	Integration Services
3	Reporting Services
4	SQL Server management tools
5	Analysis Services
6	Master Data Services

Category 1		Category 2
Supports side-by-side upgrade on a single server		Does not support side-by-side upgrade on a single server

Lesson 3

Side-by-Side Upgrade: Migrating SQL Server Data and Applications

In a side-by-side upgrade, user databases are transferred from an existing SQL Server instance to a new SQL Server 2016 instance. The new SQL Server 2016 instance can be installed on the same hardware as the instance you wish to upgrade (a one-server upgrade) or new hardware (a two-server upgrade). The points covered in this lesson will apply whether you are upgrading to SQL Server 2016 using a one-server upgrade or a two-server upgrade.

During a side-by-side upgrade to SQL Server 2016, much of your time and effort will be consumed by moving or copying databases between SQL Server instances.

However, it is common for a SQL Server application to consist of a mixture of database-level objects (tables and stored procedures, for example) and server-level objects (such as logins and SQL Server Agent jobs). Consequently, when you are carrying out a side-by-side migration to SQL Server 2016, you will need to take some manual steps to ensure that your applications continue to operate correctly after migration.



Note: The techniques for migrating databases discussed in this lesson are not only suitable for carrying out a side-by-side database upgrade; these same techniques can also be used to move databases between different instances of the same version of SQL Server.

Lesson Objectives

At the end of this lesson, you will be able to:

- Describe considerations for upgrading data and applications.
- Explain the difference between database migration and an in-place upgrade.
- Use backup and restore to migrate a database.
- Use detach and attach to migrate a database.
- Use the Copy Database Wizard to migrate a database.
- Explain why you might need to create logins on the SQL Server 2016 instance.
- Identify and repair orphaned users.
- Use **sp_configure** to compare configuration between SQL Server instances.

Upgrading Data and Applications

Application Downtime During an Upgrade

During a side-by-side upgrade, user databases are moved or copied from an instance of an older version of SQL Server to an instance of SQL Server 2016.

While the upgrade is underway, you must consider how to keep the original and migrated versions of the database synchronized. If you allow client applications to make changes to the original version of the database after the migration cut-off point, these changes will not be reflected in the SQL Server 2016 version of the database—whichever migration technique you use.

Restricting client application access to the databases is likely to cause a period of partial loss of function or complete downtime for those applications. You must determine the duration of loss of function that is acceptable to your organization.

- Plan for some application downtime
- Database file metadata is tightly linked to the version of SQL Server running the database
 - File metadata is automatically updated during a restore or attach operation
 - It's only possible to upgrade file metadata from an older version of SQL Server to a newer version
 - Rollback planning needs to take account of this
- Database compatibility level provides partial backward compatibility with previous versions of the database engine

Database File Metadata

SQL Server database files contain metadata that links them to a specific version and patch level of SQL Server with which the file is compatible. When the version of SQL Server hosting a database is upgraded, this metadata must be updated, and any other changes applied—such as the addition of new metadata items and the removal of metadata items that are no longer used.

SQL Server will automatically carry out these metadata changes whenever a database is restored, or a database file attached, to a new instance. However, upgrading a database from an older version of SQL Server to a newer version is the only supported path; you cannot downgrade a database from a newer version of SQL Server to an older version. You should consider this limitation when designing a rollback plan for an upgrade to SQL Server 2016.

Database Compatibility Level

SQL Server databases include a user-modifiable metadata attribute that indicates the version of the database engine with which the database is designed to be compatible: database compatibility level.

Database compatibility level means SQL Server 2016 can provide partial support for backward compatibility with previous versions of the database engine. Databases relying on features that have been deprecated or removed from SQL Server 2016 can be used in SQL Server 2016 without modification.

SQL Server 2016 supports the following database compatibility levels:

Compatibility Level	Database Engine Version
130	SQL Server 2016
120	SQL Server 2014
110	SQL Server 2012
100	SQL Server 2008 and 2008 R2



Note: SQL Server 2005 databases—compatibility level 90—are automatically upgraded to compatibility level 100 when they are restored or attached to a SQL Server 2016 instance.

When you restore or attach a database to a new SQL Server instance, the database compatibility level remains unchanged until you manually modify it; you might do this if you want to start using new features. An administrator can modify the compatibility level of a database, either up or down, at any time.

For more information on database compatibility level, see the topic *ALTER DATABASE Compatibility Level (Transact-SQL)* in the SQL Server 2016 Technical Documentation:



ALTER DATABASE Compatibility Level (Transact-SQL)

<http://aka.ms/dvkfr7>

Migrating SQL Server Databases

Before carrying out a side-by-side database upgrade to SQL Server 2016, you should perform the following steps on each of the user databases you plan to upgrade:

- Run Upgrade Advisor (as covered earlier in this module) to test for code compatibility with SQL Server 2016. You should assess the impact of any issues that Upgrade Advisor reports, and be prepared to make code or design changes if necessary.
- Run **DBCC CHECKDB** to ensure that the databases are in a consistent state.
- Confirm that the data files and log file are set to allow autogrowth. This is to allow for additional space required during upgrade; you can disable autogrowth after the upgrade completes.
- Confirm that database statistics are up to date, either by enabling `AUTO_UPDATE_STATS`, running **sp_updatestats**, or running **UPDATE STATISTICS** commands.
- Ensure that you have current valid backups.

Prepare user databases for migration:

- Run Upgrade Advisor and address any issues found
- Run DBCC CHECKDB
- Confirm that autogrowth for data files and log file is enabled
- Update statistics
- Verify backups

Before beginning the upgrade, you should also have installed and prepared the target SQL Server 2016 instance.



Additional Reading: For more information on installing SQL Server 2016, see Module 2 of this course: *Installing SQL Server 2016*.

When these steps are complete, you can proceed with the side-by-side migration using one of the following techniques. Each technique is discussed in more detail later in the lesson:

- Backup and restore.
- Detach/attach data files.
- Copy Database Wizard.

All these methods can be used to upgrade databases in scenarios that would be unsupported for an in-place upgrade, such as a switch between 32-bit and 64-bit versions of SQL Server.

Using Backup and Restore to Migrate Databases

You can upgrade a SQL Server 2005, 2008, 2008 R2, 2012, or 2014 database to SQL Server 2016 by carrying out a backup of the database, then restoring it to an instance of SQL Server 2016.

Both the backup operation and the restore operation can be carried out using any valid means:

- SQL Server Management Studio wizards.
- Transact-SQL (Transact-SQL) commands.
- Windows PowerShell® script.
- Third-party backup management tools.

Using backup and restore commands has the following advantages:

- Backups are usually smaller than the data files they contain, because free space is not backed up, and only the active tail of the log file is backed up. To further reduce file size, SQL Server can compress backups as they are taken (Standard, Business Intelligence and Enterprise Editions only), or you can compress them with a compression utility before transfer.
- If the database is in full recovery mode, incremental transaction log backups taken from the old instance of SQL Server can be applied to the SQL Server 2016 copy of the database while it is in NORECOVERY mode (after an initial full backup has been restored). Doing this can substantially reduce downtime during the upgrade.
- The original database remains unchanged and available for rollback if a problem is found with the upgrade.

Using backup and restore commands has the following disadvantages:

- Sufficient disk space must be available to store both the database backup and the database files, after they have been restored.
- If the database is not in full recovery mode, or you choose not to apply incremental backups, downtime for the upgrade will include time taken to run a final backup on the old version of SQL Server; transfer the backup to the new hardware (in a two-server side-by-side upgrade); and restore the backup to SQL Server 2016.
- Because the source database remains available after the backup has been taken, you must manage client application connections and activity to prevent data changes that will not be reflected in the upgraded database.

For more information, see the topic *Copy Databases with Backup and Restore* in the SQL Server 2016 Technical Documentation:



Copy Databases with Backup and Restore

<http://aka.ms/r6nsk4>

- Suitable for upgrading databases from SQL Server 2005, 2008, 2008 R2, 2012, and 2014
- Pros:
 - Backup will only include data; database free space is not backed up; backups can be compressed
 - Incremental restores of transaction log backups can be used to minimize downtime (full recovery)
 - Source database remains unchanged for use in rollback
- Cons:
 - Disk space required for backup file(s) and database file(s)
 - Downtime equals backup time *plus* transfer time *plus* restore time
 - Source database remains available—manage client application access

Using Detach and Attach to Migrate Databases

You can upgrade a SQL Server database taken from SQL Server 2005, 2008, 2008 R2, 2012, or 2014 to SQL Server 2016 by detaching the data and log files from the older SQL Server instance and attaching them to a SQL Server 2016 instance. The data files are upgraded to SQL Server 2016 at the point where they are attached to the new instance.

Using the detach and attach commands has the following advantages:

- The detach and attach commands run quickly under most circumstances. Most of the time taken by the migration will be in copying or moving files.
- Because the database on the old instance becomes unavailable when the data files are detached, the cut-off point between the old version of SQL Server and SQL Server 2016 is very clear. There is no risk of client applications continuing to make changes to the old version of the database after migration has started.
- If you copy (rather than move) the database files, you can roll back to the old version of SQL Server more quickly, without needing to wait for a database restore to complete, if there's an issue with migration.

Using detach and attach commands also has some disadvantages:

- The time taken to copy the data files might be considerable, especially if the files are large and you are in a two-server scenario where data must be copied across a network link.
- Database data files and log files commonly include a portion of empty space to allow for future expansion. When you use detach and attach, you must copy the whole data file (including the empty space). You cannot copy only the portions of the file containing data.
- If you move (rather than copy) the database files, you will not easily be able to roll back the upgrade without restoring a backup. Upgrading a database file to SQL Server 2016 is not a reversible process.



Note: If data files and log files for your SQL Server databases are held on a SAN volume, you might be able to save the time taken to copy files across the network by detaching or cloning the SAN volume from your old SQL Server and attaching it to your SQL Server 2016 hardware.

Cloning the SAN volume requires more storage space but will keep the old volume unchanged should you need to roll back the upgrade. You should discuss this with your SAN administrator.

For more information, see the topic *Database Detach and Attach (SQL Server)* in the SQL Server 2016 Technical Documentation:



Database Detach and Attach (SQL Server)

<http://aka.ms/nrxtnn>

- Suitable for upgrading databases from SQL Server 2005, 2008, 2008 R2, 2012, and 2014
- Pros:
 - Detach/attach commands are typically quick
 - Detaching the source database makes it unavailable—a clear cut-off point for client application changes
 - Copying the data files enables easy rollback (attach the files back to source server)
- Cons:
 - Large file sizes—database files typically include free space within the file
 - Moving data files (rather than copying) is a one-way process and cannot be rolled back—restore from backup would be required
 - Downtime equals detach time *plus* transfer time *plus* attach time

Using the Copy Database Wizard to Migrate Databases

The Copy Database Wizard is a tool offered by SQL Server Management Studio. It means you can copy or move SQL Server databases (the difference between copy and move being that, in a move, the source database is dropped by the wizard at the end of the process).

You can run a move or copy operation created by the Copy Database Wizard immediately, or schedule it for execution by the SQL Server Agent in the future.

The Copy Database Wizard can work in two ways:

- **Detach/attach.** This method uses detach and attach commands exactly as described earlier in this lesson. The source database files are detached and duplicated, and the duplicates attached to the target server. As this method has already been discussed, it is not covered further in this topic.
- **SQL Server Management Objects (SMO).** This method uses two steps:
 - a. The SMO API is used to generate scripts for objects in the database from the source database (tables, views, stored procedures, and so on). These scripts are applied to the target database to create copies of the database objects.
 - b. A SQL Server Integration Services (SSIS) package is generated to transfer data from the source database to the target database. As an option, the SSIS package can be saved for reference or modification.

Using the SMO method of the Copy Database Wizard has the following advantages:

- When the wizard is used to copy a database between SQL Server instances, options are available to create server-level objects related to the database being transferred (including logins, SQL Server Agent jobs, and SSIS packages).
- The source database remains available whilst the copy is being carried out.
- Storage space is only required for the source and target databases. No storage space is required for database files or backups.

Using the SMO method of the Copy Database Wizard has the following disadvantages:

- Copying data between databases like this will usually be significantly slower than using a backup or "detach and attach".
- The process of copying data is vulnerable to network connectivity problems.
- The source database remains available whilst the copy is being carried out. If client applications are permitted to make changes to the source database whilst the data transfer is taking place, the data copied to the target system might not be completely consistent; changes to one table could take place before it is copied and changes to another table might take place after it is copied.



Note: To carry out similar steps to the Copy Database Wizard manually, you could:

- Write your own SMO scripts or script database objects using Transact-SQL.
- Create your own SSIS package to transfer data, or use another tool such as bcp.exe.

- Detach/Attach provides a UI for detach/attach commands
- SQL Server Management Objects (SMO) method
- Pros:
 - Can copy server-level objects (logins, SQL Agent jobs)
 - Source database remains unchanged and available
- Cons:
 - Slower than backup/restore or detach/attach
 - Vulnerable to network problems
 - Client application activity must be managed

Similar advantages and disadvantages will apply for manually created steps, as for the Copy Database Wizard.

For more information, see the topic *Use the Copy Database Wizard* in the SQL Server 2016 Technical Documentation:



Use the Copy Database Wizard

<http://aka.ms/gmdzdz>

Transferring Logins and Passwords

A user's security context in a SQL Server database derives from two objects:

- **Server-level login.** A login controls access to a server instance. Depending on how your SQL Server instance is configured, this could be a Windows principal (a Windows user or an Active Directory® group) or a SQL Server login. Logins are stored in the **master** system database.
- **Database-level user.** A database user controls access to a database. Server logins are mapped to database users, or logins might be granted database access through membership of a database user group, or membership of a server role. Database users and groups are stored in the database to which they relate.

- SQL Server logins are not included in a database backup or database files
 - Logins must be created on the target SQL Server when migrating databases
- SQL Server logins require a password to be specified:
 - Specify the existing password (if known)
 - Use the source system password hash to create the new login
- Contained databases can break the dependency between logins and database users



Additional Reading: For a more detailed discussion of authentication and authorization in SQL Server, see course 20764: *Administering a SQL Database Infrastructure*.

When a database is migrated between SQL Server instances, as part of an upgrade or for any other reason, database users and groups are copied with it (because the users and groups are stored within the database). However, the server-level logins associated with the users will not be automatically transferred.

Unless you plan to migrate your database with the Copy Database Wizard SQL Server Management Objects method (which can optionally be configured to create logins), you will need to create the same logins on the target SQL Server instance as the ones used to access the database being transferred on the source SQL Server instance.

Logins can be created with Transact-SQL using the CREATE LOGIN command, or through SQL Server Management Objects (SMO) using the Login.Create method.

Scripts to recreate existing logins can be generated from SQL Server Management Studio (SSMS); in Object Explorer, under the server's **Security** node, expand **Logins**, right-click the login you want to script and click **Script Login as...** click **CREATE To**, and then click an output method.

SQL Server Logins

If your server is configured to allow SQL Server authentication, some of the logins you need to transfer might be SQL Server logins. These are logins whose password is stored in SQL Server (as opposed to Windows logins, where the user's Windows password is stored in Active Directory).

When generating a script for a SQL Server login, SSMS will not include the actual password value; this is a security feature designed to prevent compromise of SQL Server logins. Instead, the script generated by SSMS will have a random value for the login's password.

If you wish to use the same password for a SQL Server login on your source and target systems, two options are available:

- Determine the current password for the login on the source system and create the login on the target system using a `CREATE LOGIN WITH PASSWORD` Transact-SQL command.
- Identify the hashed password value SQL Server uses to store the password and use the password hash to create the login on the target system, using a `CREATE LOGIN WITH PASSWORD HASHED` Transact-SQL command.

Contained Databases

SQL Server 2012, 2014, and 2016 support contained databases. A contained database facilitates many features that are normally provided at instance level (such as user authentication) to be carried out at database level, rather than at instance level. Choosing to allow contained database users (that is, users with a password stored in the database) can be used to break the dependency between logins and database users. Contained databases might be partially or fully contained; partial containment allows some features to be contained in the database and others to be carried out at server level. You should confirm which features are contained when planning to upgrade a contained database.

Orphaned Users

If a database user is linked to a login that does not exist on the SQL Server instance hosting the database, the user is said to be orphaned, or an orphan.

Orphaned users are permitted to exist and might have object permissions assigned to them; however, you cannot log directly into a database as an orphaned user.

- Occurs when a database user has no corresponding login, typically when:
 - The login was deleted
 - The database has been restored/attached from another SQL Server instance
- Use `sp_change_users_login` to report on orphaned users
- To fix orphaned users:
 - Use `sp_change_users_login` to fix users with SQL Server logins
 - `ALTER USER WITH LOGIN` to fix users with SQL Server or Windows logins



Note: SQL Server 2012, 2014 and 2016 explicitly allow the creation of orphaned users—they are referred to as Users Without Login. These users can be employed to implement more complex authorization systems.

Users typically become orphaned for one of two reasons:

- The login was deleted after the user was created.
- The user belongs to a database that has been restored or attached from another SQL Server instance; the linked login has never been created on the instance now hosting the database.

Detecting Orphaned Users

A report of orphaned users in a database can be created using the system stored procedure `sp_change_users_login` with the `@Action` parameter set to 'Report'.

Repairing Orphaned Users

Orphaned users are repaired by associating them with a database login; the login must exist before this can take place.

- Users associated with a SQL Server login can be repaired using the system stored procedure **sp_change_users_login**.
- Users associated with a SQL Server login or a Windows login can be repaired with the ALTER USER WITH LOGIN Transact-SQL command.

An orphaned user can also be repaired by creating a SQL Server login with a security identifier (SID) that matches the SID found in the user's definition (see the system view sys.database_principals for details). The user definition is attached to the login's SID, not the login name. The CREATE LOGIN command can take a SID as an optional parameter.

For more information, see the topic *Troubleshoot Orphaned Users (SQL Server)* in the SQL Server 2016 Technical Documentation:



Troubleshoot Orphaned Users (SQL Server)

<http://aka.ms/grh3id>

Update Configuration with sp_configure

As part of a side-by-side upgrade, you should confirm that the features of the target SQL Server 2016 instance are configured in a way that will continue to support your applications after the upgrade is complete.

The system stored procedure sp_configure can be used to both view and amend configuration settings for a SQL Server instance. When executed without parameters, sp_configure will display a list of instance-level configuration settings. By default, only a partial list of settings is returned (simple settings). To view the complete list, the **show advanced settings** option must be set to 1. Making this change requires the ALTER SETTINGS permission.

The following example demonstrates how to enable the **show advanced settings** option:

Enabling Show Advanced Settings

```
EXEC sp_configure 'show advanced options',1;
RECONFIGURE;
--return a complete list of settings
EXEC sp_configure;
```

By running this command on the source and target SQL Server instances involved in your upgrade, you can compare the settings from both servers to determine whether any settings should be changed.



Note: Settings can be added and removed between versions of SQL Server. The list of settings returned by your source and target servers might not return an identical list of settings.

- sp_configure offers a convenient way to compare instance configuration settings between SQL Server instances
- **show advanced options** must be enabled for all settings to be displayed
- The list of settings might vary between versions of SQL Server
- sp_configure can also be used to change setting values

The value of an individual setting might be changed by executing `sp_configure` with two parameters—the first parameter being the setting to change, and the second parameter being the new value. After a setting has been amended, a `RECONFIGURE` or `RECONFIGURE WITH OVERRIDE` command must be issued to apply the new setting. Alternatively, the new setting can be applied by restarting the SQL Server instance.

For more information, see the topic *Server Configuration Options (SQL Server)* in the SQL Server 2016 Technical Documentation:



Server Configuration Options (SQL Server)

<http://aka.ms/qitsih>

Demonstration: Scripting SQL Server Logins

In this demonstration, you will see how to:

- Script SQL Server logins.
- Link logins to database users.

Demonstration Steps

1. Ensure that the 20765B-MIA-DC-UPGRADE and 20765B-MIA-SQL-UPGRADE virtual machines are running, and log on to 20765B-MIA-SQL-UPGRADE as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. On the taskbar, click the **SQL Server Management Studio** shortcut.
3. In the **Connect to Server** dialog box, click **Connect**.
4. On the **File** menu, point to **Open**, and then click **File**.
5. In the **Open File** dialog box, navigate to **D:\Demofiles\Mod03**, click **Demonstration - Login and User.sql**, and then click **Open**.
6. Select the code under the comment **Demonstration - Login and User**, and then click **Execute**.
7. Select the code under the comment **Step 1**, and then click **Execute**.
8. Select the code under the comment **Step 2**, and then click **Execute**.
9. Select the code under the comment **Step 3**, and then click **Execute**.
10. Select the code under the comment **Step 4**, and then click **Execute**.
11. In Object Explorer, expand **Security**, and then expand **Logins**.
12. Right-click **DemoLogin1**, point to **Script Login as**, point to **CREATE To**, and then click **New Query Editor Window**. If **DemoLogin1** is not visible, right-click the **Logins** node and click **Refresh**.
13. Examine the generated script. Note that the password is not correct, and then close the tab.
14. Select the code under the comment **Step 6**, and then click **Execute**.
15. Close SQL Server Management Studio without saving changes.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? During a side-by-side upgrade, if a login is created with the same SID on the new SQL Server instance, a database user will automatically be mapped to the login when a database is upgraded.	

Lab: Upgrading SQL Server

Scenario

You are a database administrator for Adventure Works. As part of an upgrade of company databases from SQL Server 2014 to SQL Server 2016, you need to complete a two-server side-by-side upgrade of a database called TSQL. The upgrade is using backup and restore to move the databases.

You have been given a full database backup and a transaction log backup taken from the SQL Server 2014 instance. You must restore the database to upgrade it to SQL Server 2016, and create any missing logins.

Objectives

After completing this lab, you will be able to:

- Upgrade a database to SQL Server 2016 by using backup and restore.
- Create a database login with SSMS and via the CREATE LOGIN command.
- Repair an orphaned database user.
- Change database compatibility level.

Estimated Time: 45 minutes

Virtual machine: **20765B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Create the Application Logins

Scenario

The TSQL database has two database users:

- appuser1, linked to a login appuser.
- reportuser1, linked to a login reportuser.

You should create the logins before the database is restored.

The SID and password hash for reportuser are available.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Create the appuser Login
3. Create the reportuser Login Using CREATE USER

► Task 1: Prepare the Lab Environment

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are both running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab03\Starter** folder as Administrator.

► Task 2: Create the appuser Login

1. Open SSMS and connect to the **MIA-SQL** database engine.
2. Using the SSMS UI, create a login with the following credentials:
 - Login: **appuser**
 - Password: **Pa\$\$w0rd1**

► Task 3: Create the reportuser Login Using CREATE USER

1. Open the project file **D:\Labfiles\Lab03\Starter\Project\Project.ssmssl** and the Transact-SQL **Lab Exercise 01 - create login.sql**. Ensure that you are connected to the master database.
2. Write a CREATE LOGIN statement to create a login with the name **reportuser** and the SID and password hash values specified in the Transact-SQL script.

Results: After this exercise, you should be able to create a login using SSMS and the CREATE USER command.

Exercise 2: Restore the Backups of the TSQL Database

Scenario

As the logins have been created, you can restore the database backups. You have been given a full backup of the TSQL database and a transaction log backup—both of these must be restored. When the restore is complete, you should update the database statistics.

The main tasks for this exercise are as follows:

1. Restore the Full Backup
2. Restore the Transaction Log Backup
3. Update Statistics

► Task 1: Restore the Full Backup

- Restore the backup **D:\Labfiles\Lab03\Starter\TSQL1.bak** to the **MIA-SQL** instance. Leave the database in the NORECOVERY state so that additional transaction logs can be restored.

► Task 2: Restore the Transaction Log Backup

- Restore the transaction log backup **D:\Labfiles\Lab03\Starter\TSQL1_trn1.trn** to the TSQL database you have restored on the **MIA-SQL** instance. Leave the database in the RECOVERY state so that the database can be used.

► Task 3: Update Statistics

- Update the database statistics for the database **TSQL** with the **up_updatestats** system stored procedure.

Results: At the end of this exercise, you should be able to:

Prepare for a migration by creating application logins.

Restore a database backup taken from one SQL Server instance and restore it to another.

Detect and repair orphaned users.

Exercise 3: Orphaned Users and Database Compatibility Level

Scenario

Now that the database has been restored, you should check for orphaned users. After this is complete, you have been asked to raise the database to the SQL Server 2016 database compatibility level.

The main tasks for this exercise are as follows:

1. Detect Orphaned Users
2. Repair Orphaned Users
3. Change Database Compatibility Level

► Task 1: Detect Orphaned Users

- Run a check for orphaned users in the newly restored TSQL database. Hint: use **sp_change_users_login** with the @Action parameter set to 'Report'.

The report should return one result - appuser1.

► Task 2: Repair Orphaned Users

- Repair the orphaned user you found in the first task by linking it to the appuser login.

► Task 3: Change Database Compatibility Level

1. Raise the database compatibility level to SQL Server 2016 (compatibility level = 130).
2. Close SQL Server Management Studio, without saving any changes.

Results: After this lab, you should be able to:

Identify orphaned database users.

Repair orphaned database users.

Update the database compatibility level.

Question: In the task where you ran a report for orphaned users, why was only one orphaned user found, even though the database had two users?

Module Review and Takeaways

In this module, you have learned about the different ways you can upgrade existing SQL Server instances and databases to SQL Server 2016. You should now be able to select an appropriate upgrade strategy to meet your organization's needs, and note any special considerations you should be aware of when upgrading some SQL Server features to SQL Server 2016.

You should be able to carry out an in-place or a side-by-side upgrade to SQL Server 2016, and be aware of any necessary post-upgrade tasks.

Review Question(s)

Question: Which upgrade strategy would best suit your organization? Why?

Module 4

Working with Databases

Contents:

Module Overview	4-1
Lesson 1: Introduction to Data Storage with SQL Server	4-2
Lesson 2: Managing Storage for System Databases	4-9
Lesson 3: Managing Storage for User Databases	4-14
Lesson 4: Moving and Copying Database Files	4-25
Lesson 5: Configuring the Buffer Pool Extension	4-29
Lab: Managing Database Storage	4-33
Module Review and Takeaways	4-37

Module Overview

One of the most important roles for database administrators who work with Microsoft® SQL Server® is the management of databases. This module provides information about how you can manage your system and user databases, and associated files.

Objectives

After completing this lesson, you will be able to:

- Describe how SQL Server stores data.
- Know how to create databases.
- Configure files and filegroups.
- Move database files.
- Configure the buffer pool extension.

Lesson 1

Introduction to Data Storage with SQL Server

To effectively create and manage databases, you must understand SQL Server files, file location, and planning for growth, in addition to how data is stored.

Lesson Objectives

After completing this lesson, you will be able to:

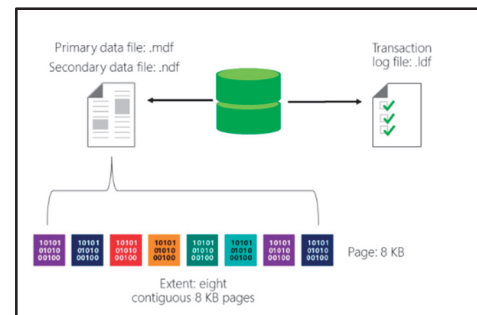
- Describe how data is stored in SQL Server.
- Describe the considerations for disk storage devices.
- Explain how specific redundant array of independent disks (RAID) systems work.
- Determine appropriate file placement and the number of files for SQL Server databases.
- Ensure sufficient file capacity and allow for ongoing growth.

How Data Is Stored in SQL Server

SQL Server databases consist of a logical schema of tables and other objects in which data structures are used to organize records. This logical schema is physically stored in a set of files allocated for the database to use—data records are written to pages within those files.

Database Files

Three types of database file are used by SQL Server—primary data files, secondary data files, and transaction log files.



Primary Data Files

The primary data file is the starting point of the database. Every database has a single primary data file. In addition to holding data pages, the primary data file holds pointers to the other files in the database. Primary data files typically use the file extension **.mdf**. Using this file extension is not mandatory but is highly recommended.

Secondary Data Files

Secondary data files are optional, user-defined, additional data files that can be used to spread the data across more storage locations for performance and/or maintenance reasons. You can use secondary files to spread data across multiple disks by putting each file on a different disk drive. Additionally, if a database exceeds the maximum size for a single Windows® file, you can use secondary data files so the database can continue to grow. The recommended extension for secondary data files is **.ndf**.

Transaction Log Files

Transaction log files (commonly referred to as log files) hold information you can use to recover the database when necessary. There must be at least one log file for each database. All transactions are written to the log file using the write-ahead logging (WAL) mechanism—this ensures the integrity of the database in case of a failure and also supports rollbacks of transactions. The recommended extension for log files is **.ldf**.

When data pages need to be changed, they are fetched into memory and changed there. The changed pages are marked as “dirty”. These “dirty pages” are then written to the transaction log in a synchronous manner. Later, during a background process known as a “checkpoint”, the dirty pages are written to the database files. For this reason, the pages in the transaction log are critical to the ability of SQL Server to recover the database to a known committed state. Transaction logs are discussed in detail in this course.

For more information about transaction log files, see *SQL Server Transaction Log Architecture and Management* in Technet:



SQL Server Transaction Log Architecture and Management

<http://aka.ms/a7mdkx>



Note: A **Logical** write occurs where data is changed in memory (buffer cache).

A **Physical** write occurs when data is changed on disk.



Note: The log file is also used by other SQL Server features, such as transactional replication, database mirroring, and change data capture. These are advanced topics and beyond the scope of this course.

Pages and Extents

Data files store data on pages, which are grouped into extents.

Data File Pages

Pages in a SQL Server data file are numbered sequentially, starting with zero for the first page. Each file in a database has a unique file ID number. To identify a page in a database uniquely, both the file ID and the page number are required. Each page is 8 KB in size. After allowing for header information for each page, there is a region of 8,096 bytes remaining for holding data. Data rows can hold fixed length and variable length column values. All fixed length columns of a data row need to fit on a single page, within an 8,060-byte limit. Data pages only hold data from a single database object, such as a table or an index.

Extents

Groups of eight contiguous pages are referred to as an extent. SQL Server uses extents to simplify the management of data pages. There are two types of extents:

- **Uniform Extents.** All pages within the extent contain data from only one object.
- **Mixed Extents.** The pages of the extent can hold data from different objects.

The first allocation for an object is at the page level, and always comes from a mixed extent. If they are free, other pages from the same mixed extent will be allocated to the object as needed. Once the object has grown bigger than its first extent, all future allocations come from uniform extents.

In both primary and secondary data files, a small number of pages are allocated to track the usage of extents in the file.

Considerations for Disk Storage Devices

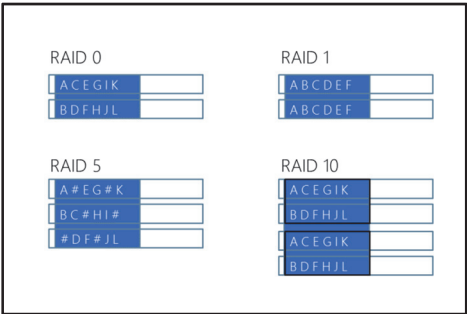
Typically, a database server will not have enough internal disks to deliver the required levels of performance—therefore, many servers use an external array of disks to store data. A disk array uses a combination of magnetic disks and/or solid-state devices to provide redundancy and improved performance. Commonly used types of disk array include:

- **Direct Attached Storage:** disks connected by a RAID controller
- **Storage Area Network:** disks connected by a network and available to multiple servers
- **Windows Storage Pools:** commodity disk drives grouped together to create one large storage space

- **Direct Attached Storage (DAS).** When using DAS, disks are stored in an enclosure and connected to the server by a RAID controller. You can use the controller to create RAID arrays from the DAS disks. The Windows Server® operating system will treat each RAID array as a storage volume, just as it would an internal disk. Typically, a DAS enclosure contains between eight and 24 drives. DAS offers very good levels of performance, particularly with throughput, and is relatively inexpensive. However, DAS can be limiting because you typically cannot share the storage between multiple servers.
- **Storage Area Network (SAN).** In a SAN, disks are stored in enclosures and connected by a common network. This can be either an Ethernet network, as is the case with Internet SCSI (iSCSI) SANs, or a fiber channel network. Servers connect to SAN storage through Host Bus Adapters (HBAs). Fiber channel SANs generally offer better performance, but are more expensive than iSCSI SANs. Although more expensive than DAS, a SAN's principal benefit is that it enables storage to be shared between servers, which is necessary for configurations such as server clustering. In a SAN, it is common practice to ensure that components such as HBAs, ports, and switches are duplicated. This removes single points of failure, helping to maintain service availability.
- **Windows Storage Pools.** In Windows storage pools, you can group drives together in a pool, and then create storage spaces which are virtual drives. You can then use commodity storage hardware to create large storage spaces and add more drives when you run low on pool capacity. You can create storage pools from internal and external hard drives (including USB, SATA, and SAS), and from solid-state drives.

RAID Levels

Many storage solutions use RAID hardware to provide fault tolerance through data redundancy, and in some cases, to improve performance. You can also implement software-controlled RAID 0, RAID 1, and RAID 5 by using the Windows Server operating system; other levels might be supported by third-party SANs. Commonly used types of RAID include:



- **RAID 0, Disk Striping.** A stripe set consists of space from two or more disks that is combined into a single volume. The data is distributed evenly across all of the disks, which improves I/O performance, particularly when each disk device has its own hardware controller. RAID 0 offers no redundancy, and if a single disk fails, the volume becomes inaccessible.

- **RAID 1, Disk Mirroring.** A mirror set is a logical storage volume based on space from two disks, with one disk storing a redundant copy of the data on the other. Mirroring can provide good read performance, but write performance can suffer. RAID 1 is expensive in terms of storage because 50 percent of the available disk space is used to store redundant data.
- **RAID 5, Disk Striping with Parity.** RAID 5 offers fault tolerance through the use of parity data that is written across all the disks in a striped volume, comprised of space from three or more disks. RAID 5 typically performs better than RAID 1. However, if a disk in the set fails, performance degrades. RAID 5 is less costly than RAID 1 in terms of disk space because parity data only requires the equivalent of one disk in the set to store it. For example, in an array of five disks, four would be available for data storage, which represents 80 percent of the total disk space.
- **RAID 10, Mirroring with Striping.** In RAID 10, a nonfault tolerant RAID 0 stripe set is mirrored. This arrangement delivers the excellent read/write performance of RAID 0, combined with the fault tolerance of RAID 1. However, RAID 10 can be expensive to implement because, like RAID 1, 50 percent of the total space is used to store redundant data.

When planning files on RAID hardware, consider the following points:

- Generally, RAID 10 offers the best combination of read/write performance and fault tolerance, but is the most costly solution.
- Write operations on RAID 5 can sometimes be relatively slow compared to RAID 1, because of the need to calculate parity data (RAID 5). If you have a high proportion of write activity, therefore, RAID 5 might not be the best candidate.
- Consider the cost per GB. For example, implementing a 500 GB database on a RAID 1 mirror set would require (at least) two 500 GB disks. Implementing the same database on a RAID 5 array would require substantially less storage space.
- Many databases use a SAN, and the performance characteristics can vary between SAN vendors. For this reason, if you use a SAN, you should consult your vendors to identify the optimal solution for your requirements.
- With Windows storage spaces, you can create extensible RAID storage solutions that use commodity disks. This solution offers many of the benefits of a specialist SAN hardware solution, at a significantly lower cost.

For more information about RAID levels (SQL Server 2008 notes), see:



RAID Levels and SQL Server

<http://aka.ms/A87k06>

Determining File Placement and Number of Files

When you create a database, you must decide where to store the database files. The choice of storage location for database files is extremely important, because it can have a significant effect on performance, resiliency, recoverability, and manageability.

Isolating Data and Log Files

For both performance and recovery reasons, it is important to isolate log and data files. This isolation needs to be at the physical disk level.

- Isolate log and data files at the physical disk level
- Determine the number and location of data files based on performance and maintenance considerations
 - Use additional files to spread data across storage locations
 - Use smaller data files when easier maintenance is needed
 - Use data files as units of backup and restore
- Determine log file requirements
 - Use a single log file in most situations, because log files are written sequentially

Access Patterns

The access patterns of log and data files are very different. Data access on log files consists primarily of sequential, synchronous writes, with occasional random disk access. Data access on data files predominantly offers asynchronous random disk access to the data files from the database. A single physical storage device does not tend to provide good response times when these types of data access are combined.

Recovery

While RAID volumes provide some protection from physical storage device failures, complete volume failures can still occur. If a SQL Server data file is lost, the database can be restored from a backup and the transaction log reapplied to recover the database to a recent point in time. If a SQL Server log file is lost, the database can be forced to recover from the data files, with the possibility of some data loss or inconsistency in the database. However, if both the data and log files are on a single disk subsystem that is lost, the recovery options usually involve restoring the database from an earlier backup and losing all transactions since that time. Isolating data and log files can help to avoid the worst impacts of drive subsystem failures.



Note: Storage solutions use logical volumes as units of storage; a common mistake is to place data files and log files on different volumes that are based on the same physical storage devices. When isolating data and log files, ensure that volumes on which you store data and log files are based on separate underlying physical storage devices.

Data File Management

Ideally, all data files that are defined for a database should be the same size. Data is spread evenly across all available data files. The main performance advantages are gained when the files are spread over different storage locations.

Allocating multiple data files provides a number of management advantages, including:

- The possibility of moving files and part of the data later.
- A reduction in recovery time when separately restoring a database file (for example, if only part of the data is corrupt).
- An increase in the parallelism in the I/O channel.
- The ability to have databases larger than the maximum size of a single Windows file.

Number of Log Files

Unlike the way that SQL Server writes to data files, the SQL Server database engine only writes to a single log file at any one time. Additional log files are only used when space is not available in the active log file.

Ensuring Sufficient File Capacity

Capacity planning helps to ensure that your databases have access to the space required as they grow. Calculating the rate of database growth means you can plan file sizes and file growth more easily and accurately.

When planning the capacity required, you should estimate the maximum size of the database, indexes, transaction log, and **tempdb**, through a predicted growth period.

For most sites, you should aim to create database files that are large enough to handle the data expected to be stored in the files over a 12-month period. If possible, base your capacity planning on tests with the actual application(s) that will store data in the database. If you cannot do this, consult the application developer or vendor to determine realistic data capacity requirements.

- Estimate the size of data, log files and tempdb:
 - Perform load testing with the actual application
 - Check with the application vendor
- Set the size to a reasonable size:
 - Leave enough space for new data, without the need to regularly expand
 - Monitor data and log file usage
 - Plan for manual expansion
 - Keep autogrowth enabled to allow for unexpected growth

Autogrowth vs. Planned Growth

SQL Server can automatically expand a database according to growth parameters that were defined when the database files were created. While the options for autogrowth should be enabled to prevent downtime when unexpected growth occurs, it's important to avoid the need for SQL Server to ever autogrow the files. Instead, you should monitor file growth over time and ensure that files are large enough for several months or years.

Many administrators are concerned that larger database files will somehow increase the time it takes to perform backups. The size of a SQL Server backup is not related directly to the size of the database files because only pages that actually contain data are backed up.

One significant issue that arises with autogrowth is a trade-off related to the size of the growth increments. If a large increment is specified, there might be a significant delay in the execution of the Transact-SQL statement that triggers the need for growth. If the specified increment is too small, the filesystem can become very fragmented and the database performance can suffer, because the data files have been allocated in small chunks all over a disk subsystem.

Log File Growth Planning

If the transaction log is not set up to expand automatically, it can quickly run out of space when certain types of activity occur in the database. For example, performing large-scale bulk operations, such as bulk import or index creation, can cause the transaction log to fill rapidly.

In addition to expanding the size of the transaction log, you can also truncate a log file. Truncating the log purges the file of inactive, committed, transactions and means the SQL Server database engine can reuse this part of the transaction log. However, you should be careful when truncating the transaction log, because doing so might affect the recoverability of the database in the event of a failure. Generally, log truncation is managed as part of a backup strategy.

Check Your Knowledge

Question	
Consider the following statement—to which of these storage options does it most closely refer? “Disks are stored in an enclosure and connected to the server by a RAID controller.”	
Select the correct answer.	
☐	SAN
☐	Windows Storage Pools
☐	External Multiple SSD on SATA
☐	DAS
☐	Nimble Storage Arrays

Question: When determining file placement and the number of files, what should you consider?

Lesson 2

Managing Storage for System Databases

SQL Server uses system databases to maintain internal metadata. Database administrators should be familiar with the SQL Server system databases and how to manage them.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe each of the system databases in a SQL Server instance.
- Move system database files.
- Configure **tempdb**.

SQL Server System Databases

In addition to the user databases that you create for applications, a SQL Server instance always contains five system databases—**master**, **msdb**, **model**, **tempdb**, and **resource**. These databases contain important metadata that is used internally by SQL Server—you cannot drop any of them.

master

The **master** database contains all system-wide information. Anything that is defined at the server instance level is typically stored in the **master** database. If the **master** database is damaged or corrupted, SQL Server will not start, so you must back it up on a regular basis.

msdb

The **msdb** database holds information about database maintenance tasks; in particular, it contains information used by the SQL Server Agent for maintenance automation, including jobs, operators, and alerts. It is also important to regularly back up the **msdb** database, to ensure that jobs, schedules, history for backups, restores, and maintenance plans are not lost. In earlier versions of SQL Server, SQL Server Integration Services (SSIS) packages were often stored within the **msdb** database. From SQL Server 2014 onward, you should store them in the dedicated SSIS catalog database instead.

model

The **model** database is the template on which all user databases are established. Any new database uses the **model** database as a template. If you create any objects in the **model** database, they will then be present in all new databases on the server instance. Many sites never modify the **model** database. Note that, even though the **model** database does not seem overly important, SQL Server will not start if the **model** database is not present.

tempdb

The **tempdb** database holds temporary data. SQL Server truncates or creates this database every time it starts, so there is no need to perform a backup. In fact, there is no option to perform a backup of the **tempdb** database.

System Database	Description
master	Stores all system-level configuration
msdb	Holds SQL Server Agent configuration data
model	Provides the template for new databases
tempdb	Holds temporary data
resource	Contains system objects that are mapped to the sys schema of databases

resource

The **resource** database is a read-only hidden database that contains system objects mapped to the **sys** schema in every database. This database also holds all system stored procedures, system views and system functions. In SQL Server versions before SQL Server 2005, these objects were defined in the **master** database.

Moving System Databases

All system databases, except the **resource** database, can be moved to new locations to help balance I/O load. However, you need to approach moving system databases with caution because, if this is performed incorrectly, it can stop the operation of SQL Server.

Moving the msdb, model, and tempdb Databases

To move the **msdb**, **model**, and **tempdb** databases, perform the following steps:

1. For each file to be moved, execute the ALTER DATABASE ... MODIFY FILE statement.
2. Stop the instance of SQL Server.
3. Move the files to the new location (this step is not necessary for **tempdb**, as its files are recreated automatically on startup—though you should delete the old **tempdb** files after restarting).
4. Restart the instance of SQL Server.

Moving the master Database

The process for moving the **master** database is different from the process for other databases. To move the **master** database, perform the following steps:

1. Open SQL Server Configuration Manager.
2. In the SQL Server Services node, right-click the instance of SQL Server, click **Properties**, and then click the **Startup Parameters** tab.
3. Edit the **Startup Parameters** values to point to the planned location for the **master** database data (-d parameter) and log (-l parameter) files.
4. Stop the instance of SQL Server.
5. Move the **master.mdf** and **mastlog.ldf** files to the new location.
6. Restart the instance of SQL Server.

- Moving **msdb** and **model**, and **tempdb**
 1. Execute ALTER DATABASE ... MODIFY FILE for each file.
 2. Stop the SQL Server service.
 3. Move the files.
 4. Restart the SQL Server service.
 - Moving **master**
 1. Change the -d and -l startup parameters for the SQL Server service.
 2. Stop the SQL Server service.
 3. Manually move the files while the instance is stopped.
 4. Restart the SQL Server service.
- Misconfiguration can prevent SQL Server from starting

Considerations for tempdb

The performance of the **tempdb** database is critical to the overall performance of most SQL Server installations. The database consists of the following objects:

- **Internal Objects.** Internal objects are used by SQL Server for its own operations. They include work tables for cursor or spool operations, temporary large object storage, work files for hash join or hash aggregate operations, and intermediate sort results.

- **tempdb:**
 - Contains temporary data for internal objects, row versioning, and user objects
 - Is truncated or rebuilt with every restart of the instance
 - Occupies varying amounts of space
 - Should be tested with real-life workloads
- Place **tempdb** on a fast and separate I/O subsystem to ensure good performance
- Split **tempdb** into data files of equal size per core



Note: Working with internal objects is an advanced concept beyond the scope of this course.

- **Row Versions.** Transactions that are associated with snapshot-related transaction isolation levels can cause alternate versions of rows to be briefly maintained in a special row version store within **tempdb**. Row versions can also be produced by other features, such as online index rebuilds, Multiple Active Result Sets (MARS), and triggers.
- **User Objects.** Most objects that reside in the **tempdb** database are user-generated and consist of temporary tables, table variables, result sets of multistatement table-valued functions, and other temporary row sets.

Planning tempdb Location and Size

By default, the data and log files for **tempdb** are stored in the same location as the files for all other system databases. If your SQL Server instance must support database workloads that make extensive use of temporary objects, you should consider moving **tempdb** to a dedicated volume—to avoid fragmentation of data files—and set its initial size based on how much it is likely to be used. You can leave autogrowth enabled, but set the growth increment to be quite large to ensure that performance is not interrupted by frequent growth of **tempdb**. You can choose the location of **tempdb** files during installation, and move them later if required.

Because **tempdb** is used for so many purposes, it is difficult to predict its required size in advance. You should carefully test and monitor the sizes of your **tempdb** database in real-life scenarios for new installations. Running out of disk space in the **tempdb** database can cause significant disruptions in the SQL Server production environment, in addition to preventing running applications from completing their operations. You can use the **sys.dm_db_file_space_usage** dynamic management view to monitor the disk space that the files are using. Additionally, to monitor the page allocation or deallocation activity in **tempdb** at the session or task level, you can use the **sys.dm_db_session_space_usage** and **sys.dm_db_task_space_usage** dynamic management views.

By default, the **tempdb** database automatically grows as space is required, because the MAXSIZE of the files is set to UNLIMITED. Therefore, **tempdb** can continue growing until space on the disk that contains it is exhausted.

Using Multiple Files

Increasing the number of files in **tempdb** can overcome I/O restrictions and avoid latch contention during page free space (PFS) scans as temporary objects are created and dropped, resulting in improved overall performance. However, do not create too many files, as this can degrade the performance. As a general rule, it is advised to have one file per core, with the ratio lower as the number of cores on the system increases. However, the optimal configuration can only be identified by doing real live tests.



Note: SQL Server setup adds as many **tempdb** files as the CPU count. If there are less than eight CPUs, setup defaults the number of files to eight.

For more information on **tempdb**, see:



tempdb Database

<http://aka.ms/oo0ysh>

Demonstration: Moving tempdb Files

In this demonstration, you will see how to move **tempdb** files.

Demonstration Steps

Move **tempdb** Files

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are running, and log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Demofiles\Mod04** folder, run **Setup.cmd** as Administrator. Click **Yes** when prompted.
3. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
4. In Object Explorer, expand **Databases**, expand **System Databases**, and then right-click **tempdb** and click **Properties**.
5. In the **Database Properties - tempdb** dialog box, on the **Files** page, note the current files and their location, and then click **Cancel**.
6. Open the **MovingTempdb.sql** script file in the **D:\Demofiles\Mod04** folder.
7. View the code in the script, and then click **Execute**. Note the message that is displayed after the code has run.
8. View the contents of **T:** and note that no files have been created in that location, because the SQL Server service has not yet been restarted.
9. In Object Explorer, right-click **MIA-SQL** and click **Restart**. Click **Yes** when prompted.
10. If the **User Account** prompt is displayed, click **Yes** to allow SQL Server 2016 to make changes. When prompted to allow changes, to restart the service, and to stop the dependent SQL Server Agent service, click **Yes**.
11. View the contents of **T:** and note that the **tempdb.mdf** and **templog.ldf** files have been moved to this location.
12. Keep SQL Server Management Studio open for the next demonstration.

Check Your Knowledge

Question	
Which of these is not a SQL Server system database?	
Select the correct answer.	
☐	master
☐	adventureworks
☐	model
☐	tempdb
☐	resource

Sequencing Activity

Put the following steps in order by numbering each to indicate the correct order.

Steps	
	Open SQL Server Configuration Manager.
	In the SQL Server Services node, right-click the instance of SQL Server, click Properties, and then click the Startup Parameters tab.
	Edit the Startup Parameters values to point to the planned location for the master database data (-d parameter) and log (-l parameter) files.
	Stop the instance of SQL Server.
	Move the master.mdf and mastlog.ldf files to the new location.
	Restart the instance of SQL Server.

Lesson 3

Managing Storage for User Databases

User databases are nonsystem databases that you create for applications. Creating databases is a core competency for database administrators working with SQL Server. In addition to understanding how to create them, you need to be aware of the impact of file initialization options and know how to alter existing databases.

When creating databases, you should also consider where the data and logs will be stored on the file system. You might also want to change this or provide additional storage when the database is in use. When databases become larger, the data should be allocated across different volumes, rather than storing it in a single large disk volume. This allocation of data is configured using filegroups and is used to address both performance and ongoing management needs within databases.

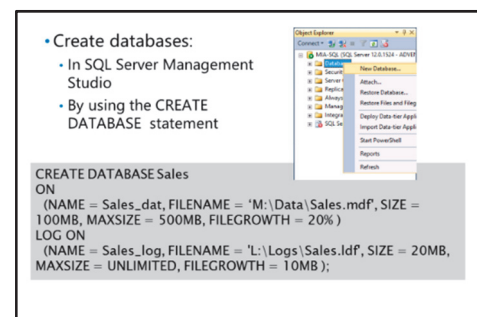
Lesson Objectives

After completing this lesson, you will be able to:

- Create user databases
- Configure database options
- Create databases
- Alter databases
- Manage database files
- Describe key features of filegroups
- Create and manage filegroups

Creating User Databases

You create databases by using either the user interface in SQL Server Management Studio (SSMS) or the CREATE DATABASE command in Transact-SQL. The CREATE DATABASE command offers more flexible options, but the user interface can be easier to use. This topic will concentrate on using the CREATE DATABASE command, but the information is equally applicable to the options available in the SSMS user interface.



CREATE DATABASE

Database names must be unique within an instance of SQL Server and comply with the rules for identifiers. A database name is of data type *sysname*, which is defined as *nvarchar(128)*. This means that up to 128 characters can be present in the database name and that each character can be chosen from the double-byte Unicode character set. Database names can be quite long, and you will find that these become awkward to work with.

Data Files

As discussed earlier in this module, a database must have at least one primary data file and one log file. The ON and LOG ON clauses of the CREATE DATABASE command specify the name and path to use.

In the following code, a database named **Sales** is being created, comprising of two files—a primary data file located at M:\Data\Sales.mdf and a log file located at L:\Logs\Sales.ldf.

```
CREATE DATABASE Sales
ON
    (NAME = Sales_dat,
    FILENAME = 'M:\Data\Sales.mdf', SIZE = 100MB, MAXSIZE = 500MB, FILEGROWTH = 20%)
LOG ON
    (NAME = Sales_log,
    FILENAME = 'L:\Logs\Sales.ldf', SIZE = 20MB, MAXSIZE = UNLIMITED, FILEGROWTH = 10MB);
```

Each file includes a logical file name in addition to a physical file path. Because operations in SQL Server use the logical file name to reference the file, the logical file name must be unique within each database.

In this example, the primary data file has an initial file size of 100 MB and a maximum file size of 500 MB. It will grow by 20 percent of its current size whenever autogrowth needs to occur. The log file has an initial file size of 20 MB and has no limit on maximum file size. Each time it needs to autogrow, it will grow by a fixed 10 MB allocation.

Collations and Default Values

If required, a specific collation can be allocated at the database level. If no collation is stipulated, it will default to the collation that was specified for the server instance during SQL Server installation. Keeping individual databases with the same collation as the server is considered a best practice.

While it is possible to create a database by providing just the database name, this results in a database that is based on the **model** database—with the data and log files in the default locations—which is unlikely to be the configuration that you require.

Deleting Databases

To **delete** (or “drop”) a database, right-click it in Object Explorer and click **Delete** or use the DROP DATABASE Transact-SQL statement. Dropping a database automatically deletes all of its files.

The following code example drops the **Sales** database:

Dropping a Database

```
DROP DATABASE Sales;
```

Configuring Database Options

Each database has a set of options that you can configure. These options are unique to each database so changing them for one database will not affect any others. Initially, all database options are set from the configuration of the **model** database when you create a database. You can change them by using the SET clause of the ALTER DATABASE statement or by using the Properties page for each database in SSMS.

- Database-level options are unique to each database

Option	Description
Auto options	Defines whether some operations should occur automatically within the database
Page verify	Defines how the page should be verified when read from disk; should be set to CHECKSUM
Recovery model	Defines the recovery model of the database
State options	Sets the state of the database, such as Online/Offline, Restricted Access or Read Only

Categories of Options

There are several categories of database options:

- **Auto Options.** They control certain automatic behaviors. As a general guideline, Auto Close and Auto Shrink should be turned off on most systems but Auto Create Statistics and Auto Update Statistics should be turned on.
- **Cursor Options.** They control cursor behavior and scope. In general, the use of cursors when working with SQL Server is not recommended, apart from for particular applications such as utilities. Cursors are not discussed further in this course but it should be noted that their overuse is a common cause of performance issues.
- **Database Availability Options.** They control whether the database is online or offline, who can connect to it, and whether or not it is in read-only mode.
- **Maintenance and Recovery Options.**
 - **Recovery Model.** For more information about database recovery models, refer to course 20764B: *Administering a SQL Server Database Infrastructure*.
 - **Page Verify.** Early versions of SQL Server offered an option called Torn Page Detection. This option caused SQL Server to write a small bitmap across each disk drive sector within a database page. There are 512 bytes per sector, meaning that there are 16 sectors per database page (8 KB). This was a fairly crude, yet reasonably effective, way to detect a situation where only some of the sectors required to write a page were in fact written. In SQL Server 2005, a new CHECKSUM verification option was added. The use of this option causes SQL Server to calculate and add a checksum to each page as it is written and to recheck the checksum whenever a page is retrieved from disk.



Note: Page checksums are only added the next time that any page is written. Enabling the option does not cause every page in the database to be rewritten with a checksum.

Demonstration: Creating Databases

In this demonstration, you will see how to:

- Create a database by using SQL Server Management Studio.
- Create a database by using the CREATE DATABASE statement.

Demonstration Steps

Create a Database by Using SQL Server Management Studio

1. Ensure that you have completed the previous demonstration. If not, start the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines, log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**, and run **D:\Demofiles\Mod04\Setup.cmd** as **Administrator**.
2. If SQL Server Management Studio is not open, start it and connect to the **MIA-SQL** database engine using Windows authentication.
3. In Object Explorer, right-click **Databases** and click **New Database**.
4. In the **Database name** box, type **DemoDB1**.

5. In the **Database files** list, note the default logical names, initial size, and autogrowth settings. Change the **Path** and **File Name** by typing the following values:
 - **DemoDB1:**
 - **Path:** D:\Demofiles\M od04
 - **File Name:** DemoDB1.mdf
 - **DemoDB1_log:**
 - **Path:** D:\Demofiles\Mod04
 - **File Name:** DemoDB1.ldf
6. Click **OK** to create the new database.
7. Expand the **Databases** folder, right-click **DemoDB1**, and then click **Properties**.
8. On the **Options** tab, review the database options, and then click **Cancel**.

Create a Database by Using the CREATE DATABASE Statement

1. In SQL Server Management Studio, open the **CreatingDatabases.sql** script file from the **D:\Demofiles\Mod04** folder.
2. Select the code under the comment **Create a database** and click **Execute** to create a database named **DemoDB2**.
3. Select the code under the comment **View database info** and click **Execute**. View the information that is returned.
4. Keep SQL Server Management Studio open for the next demonstration.

Altering User Databases

You might need to modify a database when it is in use; for example, you might need to change the name or some options. You can make such modifications by using the ALTER DATABASE Transact-SQL statement or by using SSMS. You can use the ALTER DATABASE statement to modify the files and filegroups of a database, in addition to the options, and the compatibility levels.

Altering Database Options

You can modify database options by using the ALTER DATABASE SET statement, specifying the option name and, where applicable, the value to use.

For example, you can set a database to read only or read/write.

ALTER DATABASE SET statement

```
ALTER DATABASE HistoricSales
SET READ_ONLY;
```

• Altering database options:

```
ALTER DATABASE HistoricSales
SET READ_ONLY;
```

• Altering database compatibility options:

```
ALTER DATABASE Sales
SET COMPATIBILITY_LEVEL = 100;
```



Note: Many of the database set options that you configure by using the ALTER DATABASE statement can be overridden using a session level set option. This means users or applications can execute a SET statement to configure the setting just for the current session.

For more information about database set options, see *ALTER DATABASE SET Options (Transact-SQL)* in the SQL Server 2016 Technical Documentation:



ALTER DATABASE SET Options (Transact-SQL)

<http://aka.ms/ei5fss>

Altering Database Compatibility Options

If you want your database to be compatible with a specific version of SQL Server, you can use the SET COMPATIBILITY_LEVEL option with the ALTER DATABASE statement. You can set compatibility to SQL Server 2000 and later versions.

The value that you specify for the compatibility level defines which previous versions it should be compatible with.

SET COMPATIBILITY_LEVEL option

```
ALTER DATABASE Sales
SET COMPATIBILITY_LEVEL = 100;
```

The values you can use are described in the following table:

Value	Version to be compatible with
80	SQL Server 2000
90	SQL Server 2005
100	SQL Server 2008 and SQL Server 2008 R2
110	SQL Server 2012
120	SQL Server 2014
130	SQL Server 2016

Managing Database Files

You might need to modify the structure of a database when it is in operation. The most common requirement is to add additional space by either expanding existing files or adding additional files. You might also need to drop a file, though SQL Server prevents you from dropping a file that is currently in use in the database. Dropping a file is a two-step process—first the file has to be emptied, and then it can be removed.

- Adding space to a database:
 - ALTER DATABASE ... ADD FILE
 - ALTER DATABASE ... MAXSIZE
- Dropping database files:
 - Empty file: DBCC SHRINKFILE ... EMPTYFILE
 - Drop file: ALTER DATABASE
- Shrinking databases:
 - DBCC SHRINKDATABASE
 - DBCC SHRINKFILE
 - TRUNCATE_ONLY

Adding Space to a Database

By default, SQL Server automatically expands a database according to growth parameters that you define when you create the database files. You can also manually expand a database by allocating additional space to an existing database file or by creating a new file. You might need to expand the data or transaction log space if the existing files are becoming full.

If a database has already exhausted the space allocated to it and cannot grow a data file automatically, error 1105 is raised. (The equivalent error number for the inability to grow a transaction log file is 9002.) This can happen if the database is not set to grow automatically, or if there is not enough disk space on the hard drive.

Adding Files

One option for expanding the size of a database is to add files. You can do this by using either SSMS or by using the ALTER DATABASE ... ADD FILE statement.

Expanding Files

When expanding a database, you must increase its size by at least 1 MB. Ideally, any file size increase should be much larger than this. Increases of 100 MB or more are common.

When you expand a database, the new space is immediately made available to either the data or transaction log file, depending on which file was expanded. When you expand a database, you should specify the maximum size to which the file is permitted to grow. This prevents the file from growing until disk space is exhausted. To specify a maximum size for the file, use the MAXSIZE parameter of the ALTER DATABASE statement, or use the Restrict filegrowth (MB) option when you use the Properties dialog box in SSMS to expand the database.

Transaction Log

If the transaction log is not set up to expand automatically, it can run out of space when certain types of activity occur in the database. In addition to expanding the size of the transaction log, the log file can be truncated. Truncating the log purges the file of inactive, committed transactions and allows the SQL Server database engine to reuse this unused part of the transaction log. If there are active transactions, the log file might not be able to be truncated—expanding it might be the only available option.

Dropping Database Files

Before you drop a database file, it must be empty of data. You can empty the file by using the EMPTYFILE option of the DBCC SHRINKFILE command, and then remove the file by using the ALTER DATABASE statement.

Shrinking a Database

You can reduce the size of the files in a database by removing unused pages. Although the database engine will reuse space effectively, there are times when a file no longer needs to be as large as it once was. Shrinking the file might then become necessary, but it should be considered a rarely used option. You can shrink both data and transaction log files—this can be done manually, either as a group or individually, or you can set the database to shrink automatically at specified intervals.

Methods for Shrinking

You can shrink a database or specific database files by using the DBCC SHRINKDATABASE and DBCC SHRINKFILE commands. The DBCC SHRINKFILE is preferred, because it provides much more control of the operation than DBCC SHRINKDATABASE.



Note: Shrinking a file usually involves moving pages within the files, which can take a long time.

Regular shrinking of files tends to lead to regrowth of files. For this reason, even though SQL Server provides an option to automatically shrink databases, this should only be rarely used. As in most databases, enabling this option will cause substantial fragmentation issues on the disk subsystem. It is best practice to only perform shrink operations if absolutely necessary.

Truncate Only

TRUNCATE_ONLY is an additional option of DBCC SHRINKFILE that releases all free space at the end of the file to the operating system, but does not perform any page movement inside the file. The data file is shrunk only to the last allocated extent. This option often does not shrink the file as effectively as a standard DBCC SHRINKFILE operation, but is less likely to cause substantial fragmentation and is much faster.

For more information on Database Files and Filegroups, see:



Database Files and Filegroups

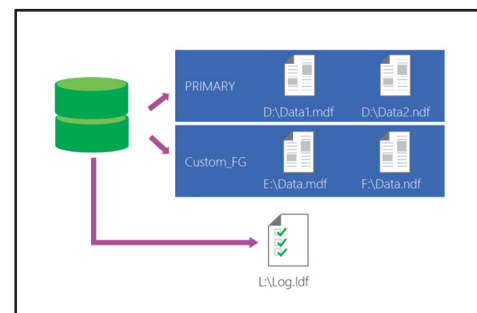
<http://aka.ms/eql7p2>

Introduction to Filegroups

As you have seen in this module, databases consist of at least two files: a primary data file and a log file. To improve performance and manageability of large databases, you can add secondary files.

Data files are always defined within a *filegroup*—these are named collections of data files you can use to simplify data placement and administrative tasks, such as backup and restore operations. Using filegroups might also improve database performance in some scenarios, because you can use them to spread database objects, such as tables, across multiple storage volumes.

Every database has a primary filegroup (named PRIMARY); when you add secondary data files to the database, they automatically become part of the primary filegroup, unless you specify a different filegroup.



When planning to use filegroups, consider the following facts:

- Database files can only belong to one filegroup.
- A filegroup can only be used by one database.

Using Filegroups for Data Manageability

You can use filegroups to control the placement of data, based on management considerations. For example, if a database contains tables of read-only data, you can place these tables in a dedicated filegroup that is set to be read-only.

Additionally, you can back up and restore files and filegroups individually. This means you can achieve faster backup times because you only need to back up the files or filegroups that have changed, instead of backing up the entire database. Similarly, you can achieve efficiencies when it comes to restoring data. SQL Server also supports partial backups. You can use a partial backup to separately back up read-only and read/write filegroups. You can then use these backups to perform a piecemeal restore—to restore individual filegroups one by one, and bring the database back online, filegroup by filegroup.



Note: You will learn more about partial backups and piecemeal restores later in this course.

Using Filegroups for Performance

When you create tables, you can specify a filegroup for the table data. If you do not specify a filegroup, the default filegroup will be used. The default filegroup is the primary filegroup, unless you configure a different filegroup as the default. By creating tables on specific filegroups, you can isolate heavily accessed tables from other tables, reducing contention and boosting performance.

When a filegroup contains multiple files, SQL Server can write to all of the files simultaneously, and it populates them by using a proportional fill strategy. Files of the same size will have the same amount of data written to them, ensuring that they fill at a consistent rate. Files of different sizes will have different amounts of data written to them to ensure that they fill up at a proportionally consistent rate. The fact that SQL Server can write to filegroup files simultaneously means you can use a filegroup to implement a simple form of striping. You can create a filegroup that contains two or more files, each of which is on a separate disk. When SQL Server writes to the filegroup, it can use the separate I/O channel for each disk concurrently, which results in faster write times.



Note: Generally, you should use filegroups primarily to improve manageability and rely on storage device configuration for I/O performance. However, when a striped storage volume is not available, using a filegroup to spread data files across physical disks can be an effective alternative.

For more information on database files and filegroups, see:



Database Files and Filegroups

<http://aka.ms/eqI7p2>

Creating and Managing Filegroups

As a database administrator, you might be required to manage databases that contain large numbers of files. Creating and managing filegroups makes it easier to control the physical storage of the database files and means you can keep files with similar manageability or access requirements together.

Creating Filegroups

You can create additional filegroups and assign files to them as you create a database; or you can add new filegroups and files to an existing database.

In the following code example, the **Sales** database includes a primary filegroup containing a single file named **sales.mdf**, a filegroup named **Transactions** containing two files named **sales_tran1.ndf** and **sales_tran2.ndf**, and a filegroup named **Archive** containing two files named **sales_archive1.ndf** and **sales_archive2.ndf**.

Creating Filegroups

```
CREATE DATABASE Sales
ON PRIMARY
(NAME = 'Sales', FILENAME = 'D:\Data\sales.mdf', SIZE = 5MB, FILEGROWTH = 1MB),
FILEGROUP Transactions
(NAME = 'SalesTrans1', FILENAME = 'M:\Data\sales_tran1.ndf', SIZE = 50MB, FILEGROWTH = 10MB),
(NAME = 'SalesTrans2', FILENAME = 'N:\Data\sales_tran2.ndf', SIZE = 50MB, FILEGROWTH = 10MB),
FILEGROUP Archive
(NAME = 'HistoricData1', FILENAME = 'O:\Data\sales_archive1.ndf', SIZE = 200MB, FILEGROWTH = 10MB),
(NAME = 'HistoricData2', FILENAME = 'P:\Data\sales_archive2.ndf', SIZE = 200MB, FILEGROWTH = 10MB)
LOG ON
(NAME = 'Sales_log', FILENAME = 'L:\Logs\sales.ldf', SIZE = 10MB, FILEGROWTH = 1MB);
```

To add filegroups to an existing database, you can use the ALTER DATABASE ... ADD FILEGROUP statement. To add files to the new filegroup, you can then use the ALTER DATABASE ... ADD FILE statement.

Setting the Default Filegroup

Unless you specify otherwise, the primary filegroup is the default filegroup for the databases. Any objects created without an explicit **ON <filegroup>** clause are created in the default filegroup. A recommended practice is to use the primary filegroup for internal system objects (which are created automatically with the database) and to add a secondary filegroup for user objects. If you adopt this practice, you should make the secondary filegroup the default filegroup for the database. You can do this by modifying the properties of the database in SSMS, or by using the ALTER DATABASE ... MODIFY FILEGROUP statement.

- Creating filegroups
 - CREATE DATABASE ... FILEGROUP (<files>)
 - ALTER DATABASE ... ADD FILEGROUP <filegroup>
- Setting the default filegroup
 - ALTER DATABASE ... MODIFY FILEGROUP <filegroup> DEFAULT
- Using read-only filegroups
 - ALTER DATABASE ... MODIFY FILEGROUP <filegroup> READONLY

The following code example changes the default filegroup in the **Sales** database to the **Transactions** filegroup:

MODIFY FILEGROUP statement

```
ALTER DATABASE Sales
MODIFY FILEGROUP Transactions DEFAULT;
```

Using Read-Only Filegroups

When a database contains a mixture of read/write and read-only data, you can use read-only filegroups to store tables containing data that will not be modified. This approach is particularly useful in large databases, such as data warehouses—it means you can employ a backup strategy that includes a single backup of read-only data, and regular backups that include only volatile data.

To make a filegroup read-only, use the ALTER DATABASE ... MODIFY FILEGROUP statement with the READONLY option.

The following code example makes the Archive filegroup read-only:

MODIFY FILEGROUP statement

```
ALTER DATABASE Sales
MODIFY FILEGROUP Archive READONLY;
```

To make a read-only filegroup writable, use the ALTER DATABASE ... MODIFY FILEGROUP statement with the READWRITE option.

Check Your Knowledge

Question	
Which of these options is incorrect with regard to SQL Server user databases?	
Select the correct answer.	
☐	A user database must have at least one primary data file and one log file.
☐	The default file extension for the primary data file is .mdf.
☐	To delete a database, you use the DROP DATABASE Transact-SQL statement.
☐	The maximum length of a database name is 128 characters.
☐	Logical file names for data and log files do not have to be unique.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? Database files can belong to many filegroups.	

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
When expanding a database, you must increase its size by at least 1 MB.	

Question: What Transact-SQL would you use to modify the default filegroup in the Customers database to the Transactions filegroup?

Lesson 4

Moving and Copying Database Files

Along with adding and removing files from a database, you might sometimes need to move database files, or even whole databases. You might also need to copy a database.

Lesson Objectives

After completing this lesson, you will be able to:

- Move user database files.
- Use the Copy Database Wizard.

Detach and attach databases.

Moving User Database Files

You can move database files to a different location within the same instance by using SSMS or the Transact-SQL ALTER DATABASE statement.



Note: Before you can move database files, you need to take the database offline.

When you move user database files, you should use the logical name of the file defined when you create the database. You can use the **sys.database_files** view to discover the logical name of the files in a database.

- Data and log files can be moved within the instance:
 - Database must be offline
- ALTER DATABASE statement:
 - For copying within an instance
 - Manually move files on the file system

Using the ALTER DATABASE Statement

You can use the ALTER DATABASE statement to move database files within the same instance of SQL Server by including the MODIFY FILE clause in the statement.

The following example shows how to move the data file for the AdventureWorks database:

ALTER DATABASE statement

```
ALTER DATABASE AdventureWorks SET OFFLINE;
// Move the files on the file system
ALTER DATABASE AdventureWorks MODIFY FILE (NAME = AWDataFile, FILENAME =
'C:\AWDataFile.mdf');
ALTER DATABASE AdventureWorks SET ONLINE;
```

Use the Copy Database Wizard

You can use the Copy Database Wizard to move and copy databases between servers with no downtime for your servers.



Note: You can also upgrade to the latest version of SQL Server. For more details, see Module 3.

You perform the following steps:

1. Select the source and destination servers.
2. Choose the databases to move or copy.
3. Specify where you want the databases located.
4. Create logins for the destination server.
5. Copy other database objects, jobs, user-defined stored procedures, and error messages.
6. Schedule the database move or copy operation.



Note: You can also copy database metadata, such as login information and other necessary **master** database objects.

For more information on using the Copy Database Wizard, see:



Use the Copy Database Wizard

<http://aka.ms/gmdzdz>

- The Database Copy Wizard
 1. Select the source and destination servers.
 2. Choose the databases to move or copy.
 3. Specify where you want the databases located.
 4. Create logins for the destination server.
 5. Copy other database objects, jobs, user-defined stored procedures, and error messages.
 6. Schedule the database move or copy operation.
- Also used for SQL Server upgrades

Detaching and Attaching Databases

SQL Server provides a functionality for attach and detach that you can use to add or remove a database from a particular instance of the database engine. This enables a commonly used technique to move a database from one instance to another.

Detaching Databases

You detach databases from an instance of SQL Server by using SSMS or the **sp_detach_db** stored procedure. Detaching a database does not remove the data from the data files or remove the data files from the server. It removes the metadata entries for that database from the system databases on that SQL Server instance. The detached database then no longer appears in the list of databases in SSMS or in the results of the **sys.databases** view. After you have detached a database, you can move or copy it, and then attach it to another instance of SQL Server.

- Detaching a database unhooks the database from the instance:
 - Data and log files are kept intact
 - Detached files can be attached again on the same or a different instance
- Use detach/attach to move databases to other instances
 - Detach/attach is useful in disaster recovery situations

UPDATE STATISTICS

SQL Server maintains a set of statistics on the distribution of data in tables and indexes. As part of the detach process, you can specify an option to perform an UPDATE STATISTICS operation on table and index statistics. While this is useful if you are going to reattach the database as a read-only database, in general it is not a good option to use while detaching a database.

Detachable Databases

Not all databases can be detached. Databases that are configured for replication, mirrored, or in a suspect state, cannot be detached.



Note: Replicated and mirrored databases are advanced topics beyond the scope of this course.

A more common problem that prevents a database from being detached when you attempt to perform the operation, is that connections are open to the database. You must ensure that all connections are dropped before detaching the database. SSMS offers an option to force connections to be dropped during this operation.

Attaching Databases

You can also use SSMS or the CREATE DATABASE ... FOR ATTACH statement to attach databases.



Note: You may find many references to the **sp_attach_db** and **bp_attach_single_file_db** stored procedures. These older system stored procedures are replaced by the FOR ATTACH option to the CREATE DATABASE statement. Note also that there is no equivalent replacement for the **sp_detach_db** procedure.



Note: A common problem when databases are reattached is that database users can become *orphaned*. For more information about this problem and its solutions, see Module 3.

Demonstration: Detaching and Attaching a Database

In this demonstration, you will see how to:

- Detach a database.

Attach a database.

Demonstration Steps

Detach a Database

1. Ensure that you have completed the previous demonstrations in this module, and that you have created a database named **DemoDB2**.
2. In Object Explorer, right-click the **Databases** folder and click **Refresh**; verify that the **DemoDB2** database is listed.
3. Right-click **DemoDB2**, point to **Tasks**, and click **Detach**.
4. In the **Detach Database** dialog box, select **Drop Connections** and **Update Statistics**, and then click **OK**.
5. View the **M:\Data** and **L:\Logs** folders and verify that the **DemoDB2.mdf** and **DemoDB2.ldf** files have not been deleted.

Attach a Database

1. In SQL Server Management Studio, in Object Explorer, in the **Connect** drop-down list, click **Database Engine**.
2. Connect to the **MIA-SQL\SQL2** database engine using Windows authentication.
3. In Object Explorer, under **MIA-SQL\SQL2**, expand **Databases** and view the databases on this instance.
4. In Object Explorer, under **MIA-SQL\SQL2**, right-click **Databases** and click **Attach**.
5. In the **Attach Databases** dialog box, click **Add**.
6. In the **Locate Database Files - MIA-SQL\SQL2** dialog box, select the **M:\Data\DemoBD2.mdf** database file, then click **OK**.
7. In the **Attach Databases** dialog box, after you have added the **master** databases file, note that all of the database files are listed, then click **OK**.
8. In Object Explorer, under **MIA-SQL\SQL2**, under **Databases**, verify that **DemoDB2** is now listed.

Check Your Knowledge

Question	
You can move database files to a different location within the same instance by using SSMS or which Transact-SQL statement?	
Select the correct answer.	
☐	MOVE DATABASE
☐	MODIFY FILE
☐	UPDATE PATH
☐	ALTER PATH
☐	ALTER DATABASE

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? Databases that are configured for replication, mirrored, or in a suspect state, cannot be detached.	

Lesson 5

Configuring the Buffer Pool Extension

The topics in this module so far have discussed the storage of system and user database files. However, SQL Server also supports the use of high performance storage devices, such as solid-state disks (SSDs), to extend the buffer pool (the cache used to modify data pages in-memory).

Lesson Objectives

After completing this lesson, you will be able to:

- Describe the buffer pool extension.
- Explain the considerations needed when working with the buffer pool extension.
- Configure the buffer pool extension.

Introduction to the Buffer Pool Extension

SQL Server uses a buffer pool of memory to cache data pages, reducing I/O demand and improving overall performance. As database workloads intensify over time, you can add more memory to maintain performance—but this solution isn't always practical. Adding storage is often easier than adding memory; with SQL Server, you can use fast storage devices for buffer pool pages with buffer pool extension.

The buffer pool extension is an extension for the SQL Server buffer pool that targets nonvolatile storage devices, such as SSDs. When the buffer pool extension is enabled, SQL Server uses it for data pages in a similar way to the main buffer pool memory.

Only clean pages, containing data that is committed, are stored in the buffer pool extension, ensuring that there is no risk of data loss in the event of a storage device failure. Additionally, if a storage device containing the buffer pool extension fails, the extension is automatically disabled. You can easily re-enable the extension when the failed storage device has been replaced.

The buffer pool extension provides the following benefits:

- Performance gains on online transaction processing (OLTP) applications with a high amount of read operations can be improved significantly.
- SSD devices are often less expensive per megabyte than physical memory, making this a cost-effective way to improve performance in I/O-bound databases.
- The buffer pool extension is easily enabled and requires no changes to existing applications.



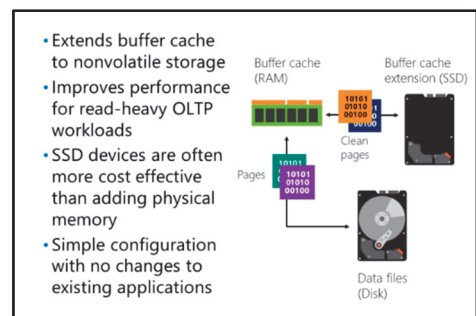
Note: The buffer pool extension is not supported in all SQL Server 2016 versions.

For more information about buffer pool extension, see:



Buffer Pool Extension

<http://aka.ms/pnqvzg>



Considerations for Using the Buffer Pool Extension

The buffer pool extension has been shown to improve the performance of OLTP databases. While database workloads can vary significantly, using the buffer pool extension is typically beneficial when the following conditions are true:

- The I/O workload consists of OLTP operations with a high volume of reads.
- The database server contains up to 32 GB of physical memory.
- The buffer pool extension is configured to use a file that is between four and 10 times the amount of physical memory in the server.
- The buffer pool extension file is stored on high throughput SSD storage.

Scenarios where the buffer pool extension is unlikely to significantly improve performance include:

- Data warehouse workloads.
- OLTP workloads with a high volume of write operations.
- Servers on which more than 64 GB of physical memory is available to SQL Server.

- Improves performance for OLTP databases where:
 - OLTP operations have a high volume of reads
 - There is up to 32 GB of physical memory
 - Buffer pool extension is 4x to 10x physical memory
 - Buffer pool extension is on high throughput SSD storage
- Unlikely to improve performance for:
 - Data warehouse workloads
 - OLTP workloads with a high volume of write operations
 - Servers with more than 64 GB of physical memory for SQL Server

Working with the Buffer Pool Extension

To resize or relocate the buffer pool extension file, you must disable the buffer pool extension, and then re-enable it with the required configuration. When you disable the buffer pool extension, SQL Server will have less buffer memory available, which might cause an immediate increase in memory pressure and I/O, resulting in performance degradation. You should therefore plan reconfiguration of the buffer pool extension carefully to minimize disruption to application users.

You can view the status of the buffer pool extension by querying the

sys.dm_os_buffer_pool_extension_configuration dynamic management view. You can monitor its usage by querying the **sys.dm_os_buffer_descriptors** dynamic management view.

Configuring the Buffer Pool Extension

To enable the buffer pool extension, you must use the ALTER SERVER CONFIGURATION statement and specify the file name and size to be used for the buffer pool extension file.

The following code sample enables the buffer pool extension with a size of 50 GB:

ALTER SERVER CONFIGURATION statement

```
ALTER SERVER CONFIGURATION
SET BUFFER POOL EXTENSION ON
(FILENAME = 'E:\SSDCACHE\MYCACHE.BPE', SIZE
= 50 GB);
```

- Enable using ALTER SERVER CONFIGURATION

```
ALTER SERVER CONFIGURATION
SET BUFFER POOL EXTENSION ON
(FILENAME = 'E:\SSDCACHE\MYCACHE.BPE',
SIZE = 50 GB);
```

- To reconfigure, disable and then re-enable

To disable the buffer pool extension, use the ALTER SERVER CONFIGURATION statement with the SET BUFFER POOL EXTENSION OFF clause.

To resize or relocate the buffer pool extension file, you must disable the buffer pool extension, and then re-enable it with the required configuration. When you disable the buffer pool extension, SQL Server will have less buffer memory available, which might cause an immediate increase in memory pressure and I/O, resulting in performance degradation. You should therefore carefully plan reconfiguration of the buffer pool extension to minimize disruption to application users.

You can view the status of the buffer pool extension by querying the **sys.dm_os_buffer_pool_extension_configuration** dynamic management view. You can monitor its usage by querying the **sys.dm_os_buffer_descriptors** dynamic management view.

Demonstration: Configuring the Buffer Pool Extension

In this demonstration, you will see how to:

- Enable the buffer pool extension.
- Verify buffer pool extension configuration.
- Disable the buffer pool extension.

Demonstration Steps

Enable the Buffer Pool Extension

1. Ensure that you have completed the previous demonstration. If not, start the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines, log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**, and then run **D:\Demofiles\Mod04\Setup.cmd** as Administrator.
2. If SQL Server Management Studio is not open, start it and connect to the **MIA-SQL** database engine using Windows authentication.
3. Open the script file **ConfiguringBPE.sql** in the **D:\Demofiles\Mod04** folder.
4. Review the code under the comment **Enable buffer pool extension**, and note that it creates a buffer pool extension file named **MyCache.bpe** on **T:**. On a production system, this file location would typically be on an SSD device.
5. Use File Explorer to view the contents of the **T:** folder and note that no **MyCache.bpe** file exists.
6. In SQL Server Management Studio, select the code under the comment **Enable buffer pool extension**, then click **Execute**.
7. Use File Explorer to view the contents of the **T:** folder and note that the **MyCache.bpe** file has been created.
8. Select the code under the comment **View buffer pool extension details**, click **Execute**, then review the output in the **Results** tab and note that buffer pool extension is enabled.
9. Select the code under the comment **Monitor buffer pool extension**, click **Execute**, then review the output in the **Results** tab.

Disable the Buffer Pool Extension

1. In SQL Server Management Studio, select the code under the comment **Disable buffer pool extension**, then click **Execute**.
2. Use File Explorer to view the contents of the **T:** folder and note that the **MyCache.bpe** file has been deleted.

3. In SQL Server Management Studio, select the code under the comment **View buffer pool extension details again**, and click **Execute**. Review the row in the **Results** tab, and note that the buffer pool extension is disabled.
4. Close SQL Server Management Studio, without saving any changes.

Check Your Knowledge

Question	
Which of these statements is not true about buffer pool extensions?	
Select the correct answer.	
☐	Extends buffer cache to nonvolatile storage.
☐	SSD devices are cheaper than conventional hard disk storage.
☐	Improves performance for read-heavy OLTP workloads.
☐	Simple configuration with no changes to existing applications.
☐	SSD devices are often more cost effective than adding physical memory.

Lab: Managing Database Storage

Scenario

As a database administrator at Adventure Works Cycles, you are responsible for managing system and user databases on the MIA-SQL instance of SQL Server. There are several new applications that require databases, which you must create and configure.

Objectives

After completing this lab, you will be able to:

- Configure **tempdb**.
- Create databases.
- Attach a database.

Estimated Time: 45 minutes

Virtual machine: **20765B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Configuring tempdb Storage

Scenario

The application development team has notified you that some of the new applications will make extensive use of temporary objects. To support this requirement while minimizing I/O contention, you have decided to move the **tempdb** database files to a dedicated storage volume and increase the size of the data and log files.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Configure tempdb Files

► Task 1: Prepare the Lab Environment

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are both running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab04\Starter** folder as **Administrator**.

► Task 2: Configure tempdb Files

1. Use SQL Server Management Studio to view the properties of the **tempdb** system database on the **MIA-SQL** database engine instance, and note the current location and size of the database files.
2. Alter **tempdb** so that the database files match the following specification:
 - **Tempdev:**
 - **Initial Size:** 10 MB
 - **File growth:** 5 MB
 - **Maximum size:** Unlimited
 - **File Name:** T:\tempdb.mdf

- **Templog:**
 - **Initial Size:** 5 MB
 - **File growth:** 1 MB
 - **Maximum size:** Unlimited
 - **File Name:** T:\templog.ldf

3. Restart the **SQL Server** service and verify that the changes have taken effect.

Results: After this exercise, you should have inspected and configured the **tempdb** database.

Exercise 2: Creating Databases

Scenario

The following two applications have been developed and both require databases:

- The Human Resources application is a simple solution for managing employee data. It is not expected to be used heavily or to grow substantially.
- The Internet Sales application is a new e-commerce website and must support a heavy workload that will capture a large volume of sales order data.

The main tasks for this exercise are as follows:

1. Create the HumanResources Database
2. Create the InternetSales Database
3. View Data File Information

► Task 1: Create the HumanResources Database

- Create a new database named **HumanResources** with the following database files:

Logical Name	Filegroup	Initial Size	Growth	Path
HumanResources	PRIMARY	50 MB	5 MB / Unlimited	M:\Data\HumanResources.mdf
HumanResources_log		5 MB	1 MB / Unlimited	L:\Logs\HumanResources.ldf

► Task 2: Create the InternetSales Database

- Create a new database named **InternetSales** with the following database files:

Logical Name	Filegroup	Initial Size	Growth	Path
InternetSales	PRIMARY	50 MB	1MB / Unlimited	M:\Data\InternetSales.mdf
InternetSales_data1	SalesData	100MB	10MB / Unlimited	N:\Data\InternetSales_data1.ndf

Logical Name	Filegroup	Initial Size	Growth	Path
InternetSales_data2	SalesData	100MB	10MB / Unlimited	N:\Data\InternetSales_data2.ndf
InternetSales_log		2Mb	10% / Unlimited	L:\Logs\InternetSales.ldf

► Task 3: View Data File Information

1. In SQL Server Management Studio, open the **ViewFileInfo.sql** script file in the **D:\Labfiles\Lab04\Starter** folder.
2. Execute the code under the comment **View page usage** and note the **UsedPages** and **TotalPages** values for the **SalesData** filegroup.
3. Execute the code under the comments **Create a table on the SalesData** filegroup and **Insert 10,000 rows**.
4. Execute the code under the comment **View page usage again** and verify that the data in the table is spread across the files in the filegroup.

Results: After this exercise, you should have created a new **HumanResources** database and an **InternetSales** database that includes multiple filegroups.

Exercise 3: Attaching a Database

Scenario

Business analysts at Adventure Works Cycles have developed a data warehouse that must be hosted on **MIA-SQL**. The analysts have supplied you with the database files so that you can attach the database.

The database has multiple filegroups, including a filegroup for archive data, which should be configured as read-only.

The main tasks for this exercise are as follows:

1. Attach the AWDDataWarehouse Database
2. Configure Filegroups

► Task 1: Attach the AWDDataWarehouse Database

1. Using File Explorer, move **AWDataWarehouse.ldf** from the **D:\Labfiles\Lab04\Starter** folder to the **L:\Logs** folder, and then move the following files from the **D:\Labfiles\Lab04\Starter** folder to the **M:\Data** folder:
 - AWDDataWarehouse.mdf
 - AWDDataWarehouse_archive.ndf
 - AWDDataWarehouse_current.ndf
2. Attach the **AWDataWarehouse** database by selecting the **AWDataWarehouse.mdf** file and ensuring that the other files are found automatically.

► Task 2: Configure Filegroups

1. View the properties of the **AWDataWarehouse** database and note the filegroups it contains.
2. Set the **Archive** filegroup to read-only.
3. View the properties of the **dbo.FactInternetSales** table and verify that it is stored in the **Current** filegroup.
4. View the properties of the **dbo.FactInternetSalesArchive** table and verify that it is stored in the **Archive** filegroup.
5. Edit the **dbo.FactInternetSales** table and modify a record to verify that the table is updateable.
6. Edit the **dbo.FactInternetSalesArchive** table and attempt to modify a record to verify that the table is read-only.
7. Close SQL Server Management Studio without saving any changes.

Results: After this exercise, you should have attached the **AWDataWarehouse** database to MIA-SQL.

Module Review and Takeaways

In this module, you have learned how to manage storage for SQL Server databases and the buffer pool extension.



Best Practice: When working with database storage, consider the following best practices:

- Carefully plan and test your file layout.
- Separate data and log files on the physical level.
- Keep the data files of a database at the same size.
- Create the database in an appropriate size so it doesn't have to be expanded too often.
- Shrink files only if absolutely necessary.
- Set a filegroup other than PRIMARY as the default filegroup.

Review Question(s)

Question: Why is it typically sufficient to have one log file in a database?

Question: Why should only temporary data be stored in the **tempdb** system database?

Module 5

Performing Database Maintenance

Contents:

Module Overview	5-1
Lesson 1: Ensuring Database Integrity	5-2
Lesson 2: Maintaining Indexes	5-8
Lesson 3: Automating Routine Maintenance Tasks	5-18
Lab: Performing Ongoing Database Maintenance	5-21
Module Review and Takeaways	5-24

Module Overview

The Microsoft® SQL Server® database engine is capable of managing itself for long periods with minimal ongoing maintenance. However, obtaining the best performance from the database engine requires a schedule of routine maintenance operations.

Database corruption is relatively rare but one of the most important tasks in the ongoing maintenance schedule is to ensure that no corruption has occurred in the databases. The likely success of recovering from corruption depends upon the amount of time between the corruption occurring, and the administrator identifying and addressing the issues that caused it. SQL Server indexes can also continue to work without any maintenance, but they will perform better if you periodically review their performance and remove any fragmentation that occurs within them. SQL Server includes a Maintenance Plan Wizard to assist in creating SQL Server Agent jobs that perform these and other ongoing maintenance tasks.

Objectives

After completing this module you will be able to:

- Maintain database integrity by using the Database Console Commands (DBCC) CHECKDB.
- Maintain indexes for optimum performance.
- Create Database Maintenance Plans for effective automation.

Lesson 1

Ensuring Database Integrity

It is rare for the database engine to cause corruption directly. However, the database engine depends upon the hardware platform that it runs on—and that can cause corruption. In particular, issues in the memory and I/O subsystems can lead to corruption within databases.

If you do not detect corruption soon after it has occurred, then further issues can arise. For example, it would be difficult to trust the recovery of a corrupt database from a set of backup copies that had themselves become increasingly corrupted.

You can use the DBCC CHECKDB command to detect, and in some circumstances correct, database corruption. It is therefore important that you are familiar with how DBCC CHECKDB works.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe database integrity.
- Use DBCC CHECKDB.
- Describe the most common DBCC CHECKDB options.
- Explain how to use the DBCC CHECKDB repair options.

Introduction to Database Integrity

There are two levels of database integrity, termed as follows:

- **Physical integrity:** this ensures that SQL Server can read and write data extents to the physical media.
- **Logical integrity:** this ensures that the data within the pages is logically correct; the pages can then refer to each other as required so that SQL Server can fetch related pages. For example, every index entry points to the correct data row and every data row points to the correct index entry. If there is no index, then the collection of pages is chained together into Index Allocation Map (IAM) pages instead.

- Physical integrity
 - Data pages are written to physical storage as SQL Server requested and can be read correctly
- Logical integrity
 - The data within the pages is logically correct so that they can refer to each other as required to instruct SQL Server to fetch related pages

Without regular checking of the database files, any lack of integrity of the database might lead to bad information derived from it. Backup does *not* check the integrity, it only checks the page checksums—and that is only when you use the WITH CHECKSUM option on the BACKUP command. Although the CHECKSUM database option is important, the checksum is only checked when SQL Server reads the data. The exception to this is when SQL Server is backing up the database and using the WITH CHECKSUM option. Archive data is, by its nature, not read frequently and this can lead to corrupt data within the database. If corrupt data is not checked as it's backed up, it may not be found for months.



DBCC CHECKDB (Transact-SQL)

<http://aka.ms/npdjvx>

Question: If you have a perfectly good data archiving process, and a regularly tested restoral system, do you still need the DBCC commands?

Overview of DBCC CHECKDB

SQL Server provides the DBCC utility, which supports a range of management facilities. In earlier documentation, you might see it referred to as the Database Consistency Checker. Although using the CHECKDB option to check the consistency of databases is a primary function of DBCC, it has many other uses.

In current versions of SQL Server it is known as the Database Console Commands utility, to more closely reflect the wider variety of tasks that it supports.

- Checks logical and physical database integrity
 - Allocation of all pages in the database
 - Consistency of tables and indexes
 - Consistency of the database catalog
- Offers repair options
 - Some options permit data loss
- Uses an internal database snapshot or table locks
- Should be run frequently
 - Synchronize executions with your backup strategy
- Optimised check might be possible with
 - DBCC CHECKFILEGROUP

DBCC CHECKDB

The DBCC CHECKDB command makes a thorough check of the structure of a database, to detect almost all forms of potential corruption. The functionality that DBCC CHECKDB contains is also available as a range of options if required. The most important of these are described in the following table:

Option	Description
DBCC CHECKALLOC	Checks the consistency of the database files disk space allocation units.
DBCC CHECKTABLE	Checks the pages associated with a specified table and the pointers between pages that are associated with the table. DBCC CHECKDB executes DBCC CHECKTABLE for every user table in the database.
DBCC CHECKCATALOG	Checks the database catalog schema by performing logical consistency checks on the metadata tables in the database. These tables hold information that describes both system and users' tables, in addition to other database objects. DBCC CHECKCATALOG does not check user tables.

DBCC CHECKDB also performs checks on other types of objects, such as the links for FILESTREAM objects and consistency checks on the Service Broker objects.



Note: FILESTREAM and Service Broker are advanced topics that are beyond the scope of this module.

Repair Options

Even though DBCC CHECKDB provides repair options, you cannot always repair a database without data loss. Usually, the best method for database recovery is to restore a backup of the database by synchronizing the execution of DBCC CHECKDB with your backup retention policy. This ensures that you can always restore a database from an uncorrupted database and that all required log backups from that time onward are available.

Online Concurrent Operations

DBCC CHECKDB can take a long time to execute and consumes considerable I/O and CPU resources, so you will often have to run it while the database is in use. Therefore, DBCC CHECKDB works using internal database snapshots to ensure that the utility works with a consistent view of the database. If the performance requirements for having the database activity running while DBCC CHECKDB is executing are too high, running DBCC CHECKDB against a restored backup of your database is an alternative option. It's not ideal, but it's better than not running DBCC CHECKDB at all.

Disk Space

The use of an internal snapshot causes DBCC CHECKDB to need additional disk space. DBCC CHECKDB creates hidden files (using NTFS Alternate Streams) on the same volumes where the database files are located. Sufficient free space on the volumes must be available for DBCC CHECKDB to run successfully. The amount of disk space required on the volumes depends on how much data is changed during the execution of DBCC CHECKDB.

DBCC CHECKDB also uses space in the **tempdb** database while executing. To provide an estimate of the amount of space required in **tempdb**, DBCC CHECKDB has an ESTIMATEONLY option.

Backups and DBCC CHECKDB

It's good practice to run DBCC CHECKDB on a database before performing a backup. This check helps to ensure that the backup will contain a consistent version of the database. In a system that is used 24 hours a day, seven days a week, this might not be convenient; in this case, you should run the checks against a restored backup of the system instead.

MAXDOP Override

You can reduce the impact of the DBCC utility on other services running on your server by setting the MAXDOP option to more than 0 and less than the maximum number of processors in your system. A configuration value of 0 for any server resource means the server can determine the maximum degree of parallelism, potentially affecting other tasks.

DBCC CHECKFILEGROUP

The DBCC CHECKFILEGROUP command runs checks against the user objects in the specified filegroup, as opposed to the complete database. Although this has the potential of saving you considerable checking of nonuser metadata objects, it is only available if you have created an advanced configuration of database files and filegroups. You can query the **sys.sysfilegroups** table to see if this is the case. If the query only returns the name of the PRIMARY filegroup, then you have a nonoptimal physical environment and you will not be able to use advanced recovery options. However, the command will still run against data objects stored in the default PRIMARY device.

DBCC CHECKDB Options

DBCC CHECKDB provides a number of options that alter its behavior while it is executing.

PHYSICAL_ONLY

Database administrators often use the **PHYSICAL_ONLY** option on production systems because it substantially reduces the time taken to run DBCC CHECKDB on large databases. If you regularly use the **PHYSICAL_ONLY** option, you still have to run the full version of the utility periodically. How often you should do this depends on your specific business requirements.

Option	Description
PHYSICAL_ONLY	Only checks the physical integrity to reduce overhead
NOINDEX	Does not perform logical checks on nonclustered indexes
EXTENDED_LOGICAL_CHECKS	Performs additional logical checks of indexed views, spatial, and XML indexes
TABLOCK	Uses locks instead of database snapshots
ALL_ERRORMSG	Returns all error messages instead of the default action that returns the first 200
NO_INFOMSGS	Returns only error messages and no informational message
ESTIMATEONLY	Estimates the amount of tempdb space that it requires to run

NOINDEX

You can use the **NOINDEX** option to specify not to perform the intensive checks of nonclustered indexes for user tables. This decreases the overall execution time but does not affect system tables because the integrity of system table indexes is always checked. The assumption that you are making when using the **NOINDEX** option is that you can rebuild the nonclustered indexes if they become corrupt.

EXTENDED_LOGICAL_CHECKS

You can only perform the **EXTENDED_LOGICAL_CHECKS** when the database is in database compatibility level 100 (SQL Server 2008) or above. This option performs detailed checks of the internal structure of objects such as CLR user-defined data types and spatial data types.

TABLOCK

You can use the **TABLOCK** option to request that DBCC CHECKDB takes a table lock on each table while performing consistency checks, rather than using the internal database snapshots. This reduces the disk space requirement at the cost of preventing other users from updating the tables.

ALL_ERRORMSG and NO_INFOMSGS

The **ALL_ERRORMSG** and **NO_INFOMSGS** options only affect the output from the command, not the operations that the command performs.

ESTIMATEONLY

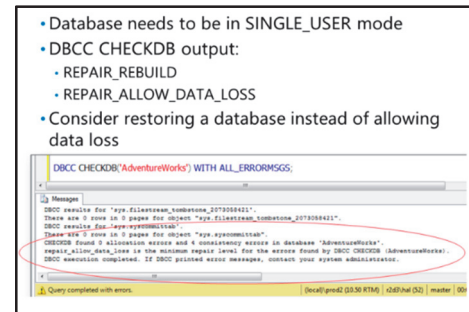
The **ESTIMATEONLY** option estimates the space requirements in the **tempdb** database if you were to run the DBCC CHECKDB command. You can then find out how much space the utility will need and avoid the risk of the process terminating partway through because **tempdb** is out of space.

DBCC CHECKDB Repair Options

Repairing a database should be a last resort. When a database is corrupt, it is typically better to restore it from a backup, after solving the cause of the corruption. If possible, you should back up a database before performing any repair option, and find and resolve the reason for the corruption—otherwise it could well happen again soon after.



Additional Reading: For more information about backing up and restoring a SQL Server database, see course 20754B: *Administering a SQL Database Infrastructure*.



In addition to providing details of errors, the output of DBCC CHECKDB shows the repair option that you have to use to correct the problem. DBCC CHECKDB offers two repair options. For both options, the database needs to be in single user mode. The options are:

- **REPAIR_REBUILD.** This rebuilds indexes and removes corrupt data pages. This option only works with certain mild forms of corruption and does not involve data loss.
- **REPAIR_ALLOW_DATA_LOSS.** This will almost certainly lose some data. It de-allocates any corrupt pages it finds and changes others that reference the corrupt pages to stop them trying to reach them. After the operation completes, the database will be consistent, but only from a physical database integrity point of view. Significant loss of data could have occurred. Repair operations do not consider any of the constraints that might exist on or between tables. If the specified table is involved in one or more constraints, you should execute DBCC CHECKCONSTRAINTS after running the repair operation.

In the example on the slide, SQL Server has identified several consistency check errors, and the REPAIR_ALLOW_DATA_LOSS option is required to repair the database.

If the transaction log becomes corrupt and no backups are available, you can use a special feature called an emergency mode repair. To do this, put the database in emergency mode and single-user mode, and then run DBCC CHECKDB with the REPAIR_ALLOW_DATA_LOSS option.

Demonstration: Using DBCC CHECKDB

In this demonstration, you will see how to use the DBCC CHECKDB command.

Demonstration Steps

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are running, and log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Demofiles\Mod05** folder, run **Setup.cmd** as Administrator. Click **Yes** when prompted.
3. Start Microsoft SQL Server Management Studio and connect to the **MIA-SQL** database engine instance using Windows authentication.
4. Open the **1 DemoDBCCRecovery.sql** file in the **D:\Demofiles\Mod05** folder.

5. Select the code under the comment -- **Run DBCC CHECKDB with default options** and click **Execute**. This checks the integrity of the **AdventureWorks** database and provides maximum information.
6. Select the code under the comment -- **Run DBCC CHECKDB without informational messages** and click **Execute**. This code checks the integrity of the **AdventureWorks** database and only displays messages if errors were present.
7. Select the code under the comment -- **Run DBCC CHECKDB against CorruptDB** and click **Execute**. This checks the integrity of the **CorruptDB** database and identifies some consistency errors in the **dbo.Orders** table in this database. The last line of output tells you the minimum repair level required.
8. Select the code under the comment -- **Try to access the Orders table** and click **Execute**. This attempts to query the **dbo.Orders** table in the **CorruptDB** database, and returns an error because of a logical consistency issue.
9. Select the code under the comment -- **Access a specific order** and click **Execute**. This succeeds, indicating that only "some data pages are affected by the inconsistency issue".
10. Select the code under the comment -- **Repair the database** and click **Execute**. Note that this technique is a last resort, when no valid backup is available. There is no guarantee of logical consistency in the database, such as the checking of foreign key constraints. These will need checking after running this command.
11. Select the code under the comment -- **Access the Orders table** and click **Execute**. This succeeds, indicating that the physical consistency is re-established.
12. Select the code under the comment -- **Check the internal database structure** and click **Execute**. Note that no error messages appear, indicating that the database structure is now consistent.
13. Select the code under the comment -- **Check data loss** and click **Execute**. Note that a number of order details records have no matching order records. The foreign key constraint between these tables originally enforced a relationship, but some data has been lost.
14. Close the file without saving changes.

Check Your Knowledge

Question	
Which of the following DBCC commands can you use to perform logical consistency checks on the metadata tables in the database?	
Select the correct answer.	
☐	DBCC CHECKTABLE
☐	DBCC CHECKALLOC
☐	DBCC CHECKCATALOG

Lesson 2

Maintaining Indexes

Another important aspect of SQL Server that requires ongoing maintenance for optimal performance is the management of indexes. The database engine can use indexes to optimize data access in tables on the physical media. Over time, through the manipulation of the data, indexed data becomes fragmented. This fragmentation reduces the performance of storage access operations at the physical level. Defragmenting or rebuilding the indexes will restore the performance of the database, as will using newer techniques such as memory optimized tables and indexed views, in addition to appropriate partitioning of the data as it is stored.

Index management options are often included in regular database maintenance plan schedules. Before learning how to set up the maintenance plans, it is important to understand more about how indexes work and how to maintain them.

Lesson Objectives

After completing this lesson, you will be able to:

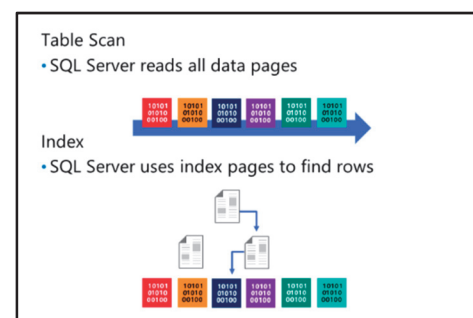
- Explain how indexes affect performance.
- Describe the different types of SQL Server indexes.
- Describe how indexes become fragmented.
- Use FILLFACTOR and PAD_INDEX.
- Explain the ongoing maintenance requirements for indexes.
- Implement online index operations.
- Describe how SQL Server creates and uses statistics.

How Indexes Affect Performance

Whenever SQL Server needs to access data in a table, it calculates whether to read all the pages (known as a table scan) or use a seek operation to pick up individual pages, reducing the amount of effort required to locate the relevant rows.

Consider a query that selects a single order based on the OrderID from the Orders table. Without an index, SQL Server has to scan all the orders in the table to find the particular one. With an index, SQL Server can find the row quickly by calculating which page stores the row with the relevant OrderID. The difference here could be between reading a few of the data pages compared with accessing all the pages, some multiple times.

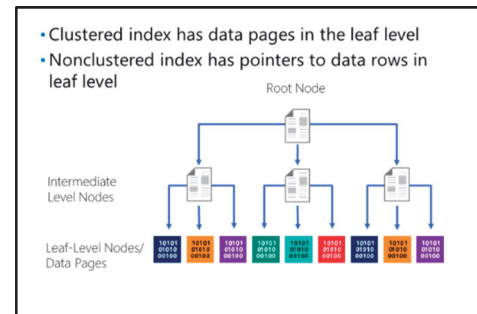
Indexes can help to improve searching, sorting, and join performance, but they can also impede the performance of data modification. They require ongoing management and demand additional disk space.



If the query optimizer calculates that it can improve performance, SQL Server will create its own temporary indexes to do so. However, this is beyond the control of the database administrator or programmer. An important side effect of the optimizer creating temporary indexes and you spotting this, is that you can determine whether those indexes would be beneficial for all users and create them yourself. This then means that the indexes are also available to the query optimizer when it needs them.

Types of Indexes

Rather than storing rows of data as a heap of data pages, where the data is not stored in any specific logical or physical order (merely appended to a chain of relevant pages as allocated), you can design tables with an internal logical and/or physical ordering. To implement physical ordering of the data pages, you create an index known as a clustered index. On top of a clustered index, you can also apply logical ordering by using another type of index called a nonclustered index. For each clustered index, you can have many nonclustered indexes without increasing the overhead of optimizing the data for different queries.



Clustered Indexes

A table with a clustered index has a predefined order for rows within a page and for pages within the table. The storage engine orders the data pages according to a key, made up of one or more columns. This key is commonly called a clustering key.

Because the rows of a table should only be in one physical order, there can only be one clustered index on a table. SQL Server uses an IAM entry to point to the root page of a clustered index; the remainder of the pages are stored as a balanced b-tree set of pages linked together from the root page outwards.

The physical order of data stored with a clustered index is only sequential at the data page level. When data changes, rows might migrate from one page to another, to leave space for other rows to be inserted in the original data page. This is how fragmentation occurs. It helps to bear in mind that fragmentation is not automatically bad. For a reporting system, high levels of fragmentation can impede performance, but for a transactional system, high levels of fragmentation can actually assist the functional requirements of the business.

Nonclustered Indexes

A nonclustered index does not affect the layout of the data in the table in the way that a clustered index does. If the underlying table has no clustered index, the data is stored as a heap of pages that the data storage engine allocates as the object grows. The leaf level of a nonclustered index contains pointers, either to where the data rows are stored in a heap, or to the root node of the corresponding clustered index for the object. The pointers include a file number, a page number, and a slot number on the page.

How These Indexes Are Used

Consider searching for a particular order with the OrderID of 23678 in an index of OrderID. In the root node (that is, the page level), SQL Server searches for the range of OrderID values that contains 23678. The entry for the range in the root page points to an index page in the next level. In this level, the range covers smaller ranges, again pointing to pages on the following level of the index. This continues to a point where every row has an entry for its location. Leaf-level nodes are the index pages at the final point

of the path of the index page chain. With a clustered index, the leaf level of pages/nodes also contain the actual data pages of the table.

Index and data pages are linked and double-linked within a logical hierarchy across all pages at the same level of the hierarchy. This helps SQL Server to perform partial scan operations across an index, from a given start point, that is located using a seek operation.

Other Types of Index

SQL Server includes other types of index:

- Integrated Full-Text Search (iFTS) uses a special type of index that provides flexible searching of text.
- The GEOMETRY and GEOGRAPHY data types use spatial indexes.
- Primary and secondary XML indexes assist when querying XML data.
- Large data warehouses or operational data stores can use columnstore indexes.

Dynamic Management Objects

SQL Server 2016 includes a large collection of Dynamic Management Objects (DMOs). These are functions and views that you can examine in Transact-SQL queries in the same way that you examine the contents of tables, views, and stored procedures. However, the data returned by these views does not present the contents of a database but instead presents the health and activity of the server.

- Dynamic Management Objects
 - Dynamic Management Views
 - Dynamic Management Functions
- Categories of DMO
 - Index-Related DMOs
 - sys.dm_db_index_usage_stats
 - sys.dm_db_index_operational_stats
 - sys.dm_db_index_physical_stats
 - sys.dm_db_missing_index_details

There are two types of DMO:

- **Dynamic Management Views (DMVs).** You can query DMVs in the same way as you call views in a user database by naming the DMV in the FROM clause of a SELECT statement.
- **Dynamic Management Functions (DMFs).** You can query DMFs in the same way as you call a table-valued function. That is to say, you must provide parameters to the DMF in brackets.

DMOs are classified into 26 categories. Some examples of these categories include:

- **Database-Related DMVs.** These DMVs provide information about the health of a specific database. For example, the **sys.dm_db_file_space_usage** DMV returns data about how each file in the database uses disk space.
- **Server-Related DMOs.** These DMVs and DMFs provide information about the SQL Server and the services that are installed on it. For example, the **sys.dm_server_services** DMV returns data about the status and configuration of services such as the Full-Text Service and the SQL Server Agent.
- **Transaction-Related DMOs.** These DMVs and DMFs provide information about the transactions that are in process on the server. For example, the **sys.dm_tran_active_transactions** DMV shows data about all the active transactions on the server.
- **Security-Related DMOs.** These DMVs and DMFs provide information about the status of SQL Server security features. For example, the **sys.dm_database_encryption_keys** DMV returns data about the encryption status of a database and the keys used for that encryption.

For more information and a complete list of SQL Server 2016 DMOs, see:



Dynamic Management Views and Functions (Transact-SQL)

<https://aka.ms/saz17l>

Index-Related DMOs

DMOs can be used to obtain information about all aspects of server health. One category of DMO, however, relates specifically to indexes and can be useful in their maintenance. Index-related DMVs and DMFs include the following:

- **sys.dm_db_index_usage_stats.** This DMV returns counts of different types of index operations and the time that each type of operation was last performed.
- **sys.dm_db_index_operational_stats.** This DMF returns current I/O, locking, latching and access method activity for each index in the database.
- **sys.dm_db_index_physical_stats.** This DMF returns size and fragmentation information for the indexes of a specified table or view.
- **sys.dm_db_missing_index_details.** This DMV returns information about missing indexes. You can use this information to help optimize the indexes you create.

For complete information on the index-related DMOs, see:



Index Related Dynamic Management Views and Functions (Transact-SQL)

<https://aka.ms/pd1eek>

You will learn more about specific index-related DMOs later in this module.

Index Fragmentation

Index fragmentation occurs over time, as you insert and delete data in the table. For operations that read data, indexes perform best when each page is as full as possible. However, if your indexes initially start full (or relatively full), adding data to the table can result in the index pages needing to be split. Adding a new index entry to the end of an index is easy, but the process is more complicated if the entry needs to be inserted in the middle of an existing full index page.

- Fragmentation occurs when data modification causes index pages to split:
 - Internal fragmentation when pages are not full
 - External fragmentation when pages are not in logical sequence
- Detecting fragmentation:
 - Index properties in SQL Server Management Studio
 - **sys.dm_db_index_physical_stats**

Internal vs. External Fragmentation

The two types of index fragmentation are:

Internal Fragmentation. This occurs when pages are not completely full of data and often when a page is split during an insert or update operation—in a case where the data could not fit back into the space initially allocated to the row.

External Fragmentation. This occurs when pages become out of sequence during Data Manipulation Language (DML) operations that split existing pages when modifying data. When the index requires and allocates a new page within an extent that is different to the one that contains the original page, extra links are required to point between the pages and extents involved. This means that a process needing to read the index pages in order must follow pointers to locate the pages. The process then involves accessing pages that are not sequential within the database.

Detecting Fragmentation

SQL Server provides a useful measure in the `avg_fragmentation_in_percent` column of the `sys.dm_db_index_physical_stats` DMV. You can use this to analyze the level of fragmentation in your index and to decide whether to rebuild it.

The following code shows two ways to use a stored procedure to detect physical fragmentation. The first sample shows all rows being returned and the second sample shows the particular rows that we are interested in, as defined by the `WHERE` clause.

Retrieving Physical Fragmentation Details for Database Objects

```
SELECT * FROM sys.dm_db_index_physical_stats
( DB_ID() --Current database
, OBJECT_ID('HumanResources.Employee') -- Object needing checking
, NULL -- Index_id
, NULL -- partition_number
, 'DETAILED' -- as opposed to LIMITED OR SAMPLED
);

-- Or

SELECT          DB_NAME(database_id)          AS [Database]
, OBJECT_NAME(object_id)          AS [Object]
, avg_fragmentation_in_percent      AS Fragmentation

FROM            sys.dm_db_index_physical_stats(DEFAULT,DEFAULT,DEFAULT,DEFAULT,DEFAULT)
WHERE           avg_fragmentation_in_percent > 10
ORDER BY avg_fragmentation_in_percent DESC
```



For more information about using this stored procedure, see `sys.dm_db_index_physical_stats` (Transact-SQL)

<http://aka.ms/lzvjmj>

SQL Server Management Studio (SSMS) also provides details of index fragmentation in the properties page for each index.

FILLFACTOR and PAD_INDEX

The availability of free space in an index page can have a significant effect on the performance of index update operations. If there is insufficient space for an index entry on the relevant page, there will be a new index page allocated to the index and the contents of the old page will be split across the two. This can impede performance if it happens too frequently.

You can alleviate the performance impact of page splits by leaving an empty space on each page when creating an index, including a clustered index. You can use the FILLFACTOR option when you create the index to define how full the index pages should be. The default value of the FILLFACTOR is 0, which actually means “fill the pages 100 percent”. Any other value indicates the percentage of each page filled, on the index creation.

The following example shows how to create an index that is 70 percent full, leaving 30 percent free space on each page:

Using FILLFACTOR

```
ALTER TABLE Person.Contact
  ADD CONSTRAINT PK_Contact_ContactID
    PRIMARY KEY CLUSTERED
  (
    ContactID ASC
  ) WITH (PAD_INDEX = OFF, FILLFACTOR = 70);
GO
```



Note: The same action of the values 0 and 100 can seem confusing. While both lead to the same outcome, 100 indicates that a specific FILLFACTOR value is being set. The value zero indicates that no FILLFACTOR has been specified and that the default server value should be applied.

By default, the FILLFACTOR option only applies to leaf-level pages of an index. You can use it in conjunction with the PAD_INDEX = ON option to cause the same free space to also be allocated in the nonleaf levels of the index.

Although you can use the fill factor to preallocate space in the index pages to help in some situations, it is not a panacea for good performance; it might even impede performance where all the changes are on the last page anyway. Consider the primary key index shown in the previous example. It is unlikely that this fill factor will improve performance. The primary key ensures that no duplicate rows occur; therefore, it is unlikely that you will insert values for such rows in the middle of existing ones, because the primary key is sequential. If you make sure you don't use variable sized columns, you can update rows in place—so there is no requirement for empty space for updates.

Make sure you consider the costs in addition to the benefits of setting the empty space in index pages. Both memory pages and disk space are wasted if the space adds no value. There will also be reduced network efficiency and an inability of the processing unit to obtain as much useful data when it needs to process it.

- Leave free space in indexes to avoid fragmentation:
 - FILLFACTOR (leaf level only)
 - PAD_INDEX (intermediate and root levels also)

```
ALTER TABLE Person.Contact
  ADD CONSTRAINT PK_Contact_ContactID
    PRIMARY KEY CLUSTERED
  (
    ContactID ASC
  ) WITH (PAD_INDEX = OFF, FILLFACTOR = 70);
GO
```

Ongoing Maintenance of Indexes

When SQL Server updates indexes during data modifications, the index storage can fragment. SQL Server provides two options for removing fragmentation from clustered and nonclustered indexes—rebuild and reorganize.

REBUILD

Rebuilding an index drops and recreates the index. This removes fragmentation, reclaims disk space by compacting the pages based on the specified or existing fill factor setting, and reorders the index entries in contiguous pages. When you use the ALL option, SQL Server drops all indexes on the table and rebuilds them in a single operation. If any part of that process fails, SQL Server rolls back the entire operation.

Because SQL Server performs rebuilds as logged, single operations, a single rebuild operation can use a large amount of space in the transaction log. To avoid this, you can change the recovery model of the database to use the BULK_LOGGED or SIMPLE recovery models before performing the rebuild operation, so that it is a minimally logged operation. A minimally logged rebuild operation uses much less space in the transaction log and takes less time. However, if you are working on a transactional database, remember to switch back to full logging mode and back up the database after you have completed the operation—otherwise, you will not have a robust backup strategy to insure against database corruption.

Use the ALTER INDEX statement to rebuild an index.

Rebuilding an Index

```
ALTER INDEX CL_LogTime ON dbo.LogTime REBUILD;
```

REORGANIZE

Reorganizing an index uses minimal system resources. It defragments the leaf level of clustered and nonclustered indexes on tables by physically reordering the leaf-level pages to match the logical, left to right order of the leaf nodes. Reorganizing an index also compacts the index pages. The compaction uses the existing fill factor value for the index. You can interrupt a reorganize without losing the work performed so far. This means that, on a large index, you could configure partial reorganization to occur each day.

For heavily fragmented indexes (those with fragmentation greater than 30 percent) rebuilding is usually the most appropriate option to use. SQL Server maintenance plans include options to rebuild or reorganize indexes. If you do not use maintenance plans, it's important to create a job that regularly performs defragmentation of the indexes in your databases.

You also use the ALTER INDEX statement to reorganize an index.

Reorganizing an Index

```
ALTER INDEX ALL ON dbo.LogTime REORGANIZE;
```

- **Rebuild:**
 - Rebuilds the whole index
 - Needs free space in database
 - Performed as a single transaction with potential requirement for a large amount of transaction log space

```
ALTER INDEX CL_LogTime ON dbo.LogTime REBUILD
```

- **Reorganize:**
 - Sorts the pages and is always online
 - Less transaction log usage
 - Can be interrupted but still retain work performed to that point

```
ALTER INDEX ALL ON dbo.LogTime REORGANIZE;
```


Online Index Operations

The Enterprise edition of SQL Server can perform index operations online while users are accessing the database. This is very important because many organizations have no available maintenance time windows during which to perform database maintenance operations, such as index rebuilds.

When performing an online index rebuild operation, SQL Server creates a temporary mapping index that tracks data changes taking place while the index rebuild operation is occurring. For consistency, SQL Server takes a very brief shared lock on the object at the beginning and end of the operation. During the online rebuild operation, the database engine applies schema locks to prevent metadata changes. This means that users cannot change the structure of the table using commands such as ALTER TABLE while the online index rebuild operation is occurring.

Because of the extra work that needs to be performed, online index rebuild operations are typically slower than offline ones.

The following example shows how to rebuild an index online:

Rebuilding an Index Online

```
ALTER INDEX IX_Contact_EmailAddress ON Person.Contact REBUILD
WITH ( PAD_INDEX = OFF, FILLFACTOR = 80, ONLINE = ON, MAXDOP = 4 );
```



Guidelines for Online Index Operations

<http://aka.ms/jvo7hw>

Updating Statistics

One of the main tasks that SQL Server performs when it is optimizing queries is to determine which indexes to use and which ones not to use. To make these decisions, the query optimizer uses statistics about the distribution of the data in the index and data pages. SQL Server automatically creates statistics for indexed and nonindexed columns when you enable the AUTO_CREATE_STATISTICS database option. You enable and disable this option by using the ALTER DATABASE statement.

Statistics are there to provide as much information as possible for the SQL Server optimizer. This is how we switch them on.

Switching On Automatic Statistics Creation for the Data Columns in the Current Database.

```
ALTER DATABASE CURRENT
SET AUTO_CREATE_STATISTICS ON
```

- Can create, rebuild, and drop indexes online:
 - Enables concurrent user access to the underlying table and indexes
 - Only needs short term shared locks at beginning and end of the operation and schema locks during the operation
- Typically slower than equivalent offline operation

```
ALTER INDEX IX_Contact_EmailAddress
ON Person.Contact REBUILD
WITH ( PAD_INDEX = OFF,
      FILLFACTOR = 80,
      ONLINE = ON,
      MAXDOP = 4 );
```

- Distribution statistics become outdated
 - They can be set to update automatically/manually
 - Do not disable auto_update_statistics

Option	Description
AUTO_CREATE_STATISTICS AUTO_UPDATE_STATISTICS	Database options that enable SQL Server to automatically create/update statistics
UPDATE STATISTICS <table>	Statement that updates all statistics on a table or specified subset of statistics on demand
SP_UPDATESTATS	System stored procedure that updates all statistics in a database

SQL Server automatically updates statistics when `AUTO_UPDATE_STATISTICS` is set. This is the default setting so it's best that you do not disable this option unless you really have to—it is necessary for a package solution you are implementing on top of SQL Server.

For large tables, the `AUTO_UPDATE_STATISTICS_ASYNC` option instructs SQL Server to update statistics asynchronously instead of delaying query execution, where it would otherwise update an outdated statistic before query compilation.

You can also update statistics on demand. Executing the Transact-SQL code `UPDATE STATISTICS <tablename>` causes SQL Server to refresh all statistics on the specified table. You can also run the **sp_updatestats** system stored procedure to update all statistics in a database. This stored procedure checks which table statistics are out of date and refreshes them.

Demonstration: Maintaining Indexes

In this demonstration, you will see how to maintain indexes.

Demonstration Steps

Maintain Indexes

1. If SQL Server Management Studio is not open, start it and connect to the MIA-SQL database engine instance using Windows authentication.
2. Open the **2 DemoIndexFragmentation.sql** script file in the **D:\Demofiles\Mod05** folder.
3. Select the code under the comment -- **Create a table with a primary key** and click **Execute**. This creates a table with a primary key, which by default creates a clustered index on the primary key field.
4. Select the code under the comment -- **Insert some data into the table** and click **Execute**. This inserts 10,000 rows into the table.
5. Select the code under the comment -- **Check fragmentation** and click **Execute**. In the results, note the `avg_fragmentation_in_percent` and `avg_page_space_used_in_percent` values for each index level.
6. Select the code under the comment -- **Modify the data in the table** and click **Execute**. This updates the table.
7. Select the code under the comment -- **Re-check fragmentation** and click **Execute**. In the results, note that the `avg_fragmentation_in_percent` and `avg_page_space_used_in_percent` values for each index level have changed because the data pages have become fragmented.
8. Select the code under the comment -- **Rebuild the table and its indexes** and click **Execute**. This rebuilds the indexes on the table.
9. Select the code under the comment -- **Check the fragmentation again** and click **Execute**. In the results, note that the `avg_fragmentation_in_percent` and `avg_page_space_used_in_percent` values for each index level indicate less fragmentation.
10. Close the file without saving changes.

Check Your Knowledge

Question	
Which of these observations indicate that you should reorganize your data pages, rather than rebuild them?	
Select the correct answer.	
☐	You are looking at an index, on a reporting table, with a fill factor of 0.
☐	You are looking at an index, on a reporting table, with a fragmentation of 0 percent.
☐	You are looking at an index, on a transactional table, with a fragmentation of 0 percent.
☐	You are looking at an index, on a transactional table, with fragmentation of less than 30 percent.
☐	You are looking at an index, on a transactional table, with a fragmentation of 50 percent.

Lesson 3

Automating Routine Maintenance Tasks

You have seen how you can manually perform some of the common database maintenance tasks that you will have to execute on a regular basis. SQL Server provides the Maintenance Plan Wizard that you can use to create SQL Server Agent jobs to perform the most common database maintenance tasks.

Although the Maintenance Plan Wizard makes this process easy to set up, it is important to realize that you can use the output of the wizard as a starting point for creating your own maintenance plans. Alternatively, you can create plans from scratch, using the SQL Server DMOs or system stored procedures.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe SQL Server maintenance plans.
- Monitor SQL Server database maintenance plans.

Overview of SQL Server Maintenance Plans

You can use the SQL Server Maintenance Plan Wizard to create SQL Server Agent jobs that perform routine database maintenance tasks, such as regularly backing up system and user databases. The user databases can be optimized and checked for consistency either at the same time, before, or after backups. The wizard creates SQL Server Integration Services (SSIS) packages for SQL Server Agent tasks to execute on the specified schedules. The wizard can also integrate with other features of the SQL Agent service to facilitate monitoring and control of the package outcomes.

- You can schedule many tasks:
 - Checking database integrity
 - Shrinking databases
 - Rebuilding and reorganizing indexes in a database
 - Updating database object statistics
 - Cleanup of task history
 - Activation of related SQL Agent jobs and alerts
- The SQL Agent uses SSIS packages to perform the tasks

You can schedule many maintenance tasks to run automatically, including:

- Checking database integrity.
- Shrinking databases.
- Rebuilding and reorganizing indexes in a database.
- Updating database object statistics.
- Cleanup of task history.
- Activation of related SQL Agent jobs and alerts.

You can create maintenance plans using one schedule for all tasks or individual schedules for each task. You use combinations of these to group and sequence tasks based on custom requirements.

Monitoring Database Maintenance Plans

SQL Server implements maintenance plans by using SQL Server Agent jobs that run SSIS packages. Because they become SQL Server Agent jobs, you can monitor them by using the standard Job Activity Monitor in SSMS.

The results of the maintenance tasks are stored in the maintenance plan tables in the **msdb** database. Like any other user table, you can inspect these tables, **dbo.sysmaintplan_log** and **dbo.sysmaintplan_log_detail**, by using the SELECT statement. You can also view the entries by querying them directly from the Log File Viewer.

In addition, tasks can generate text-based reports, write them to the file system, and send them automatically to operators defined in the SQL Server Agent service system.

- Real-time monitoring in Job Activity Monitor
- Execution results stored in **msdb** and can also be:
 - Written to a text file
 - Sent to an operator
- Cleanup tasks to implement retention policies



Note: You can use the cleanup tasks available in the maintenance plans to implement a retention policy for backup files, job history, maintenance plan report files, and **msdb** database table entries.

Demonstration: Configuring a Database Maintenance Plan

In this demonstration, you will see how to create a database maintenance plan.

Demonstration Steps

1. If SQL Server Management Studio is not open, start it and connect to the **MIA-SQL** database engine instance using Windows authentication.
2. In Object Explorer, under **MIA-SQL**, expand **Management**, right-click **Maintenance Plans**, and click **Maintenance Plan Wizard**.
3. In the SQL Server Maintenance Plan Wizard, click **Next**.
4. On the **Select Plan Properties** page, in the **Name** box, type **Maintenance Plan for Optimization AdventureWorks Database**, and then click **Next**.
5. On the **Select Maintenance Tasks** page, select the following tasks, and then click **Next**:
 - Shrink Database
 - Rebuild Index
 - Update Statistics
6. On the **Select Maintenance Task Order** page, change the order of the tasks to **Rebuild Index**, **Shrink Database**, then **Update Statistics**, and then click **Next**.
7. On the **Define Rebuild Index Task** page in the **Databases** list, click **AdventureWorks**, and then click **OK** to close the drop-down list box. Click **Next**.
8. On the **Define Shrink Database Task** page, in the **Databases** list, click **AdventureWorks**, click **OK** to close the drop-down list box, and then click **Next**.

9. On the **Define Update Statistics** page, in the **Databases** list, click **AdventureWorks**. Click **OK** to close the drop-down list box, and then click **Next**.
10. On the **Select Report Options** page, review the default settings, and then click **Next**.
11. On the **Complete the Wizard** page, click **Finish** to create the Maintenance Plan. Wait for the operation to complete, and then click **Close**.
12. Close SQL Server Management Studio without saving changes.

Categorize Activity

Your manager asks you to implement a maintenance solution that minimizes data loss and cost in addition to maximizing performance. Sort items by writing the appropriate category number to the right of each one.

Items	
1	Use multiple data maintenance plans combined with SQL Server Agent scheduled jobs
2	Use Transact-SQL statements to implement maintenance tasks
3	Use one data maintenance plan
4	Use jobs and schedules to implement multiple maintenance plans

Category 1		Category 2		Category 3
Definitely appropriate		May be appropriate		Not appropriate

Lab: Performing Ongoing Database Maintenance

Scenario

You are a database administrator at Adventure Works Cycles, with responsibility for databases on the **MIA-SQL** Server instance. You must perform the ongoing maintenance of the database on this instance—this includes ensuring database integrity and managing index fragmentation.

Objectives

After completing this lab, you will be able to:

- Use DBCC CHECKDB.
- Defragment indexes.
- Create and run database maintenance plans.

Estimated Time: 45 minutes

Virtual machine: **20765B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Use DBCC CHECK to Verify Data Integrity

Scenario

Junior colleagues ask you to check a query they are writing before they commit the transaction. Before you can point out an error in the script, a failure occurs on the disk storing the log file and the server stops. After restarting the server, you will review the state of the data, replace the log with a copy from a mirror server, and run DBCC CHECKDB to repair the database.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Manage Database Integrity

► Task 1: Prepare the Lab Environment

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are both running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab05\Starter** folder as Administrator.

► Task 2: Manage Database Integrity

1. In SQL Server Management Studio, connect to the **MIA-SQL** database engine using Windows authentication.
2. Open **Ex1.sln** from the **D:\Labfiles\Lab05\Starter\Ex1** folder and review the Transact-SQL code in the **Ex1-DBCC.sql** file.
3. Execute the code up to and including the CHECKPOINT statement.
4. In a new query window using a separate database connection, shut down the database server without performing checkpoints in every database.
5. In Windows Explorer, rename the **CorruptDB_log.ldf** file to simulate the file being lost.
6. Restart SQL Server and SQL Server Agent.

7. In SQL Server Management Studio, try to access the **CorruptDB** database, and then check its status.
8. Use emergency mode to review the data in the database and note that the erroneous transaction was committed on disk.
9. Set the database offline, rename the log file back to **CorruptDB_log.ldf** to simulate accessing it from a mirror copy, and then set the database online again.
10. Put the database in single user mode and run DBCC CHECKDB with the REPAIR_ALLOW_DATA_LOSS option.
11. Put the database back into multiuser mode and check that the data is restored to its original state.
12. Close the solution without saving changes, but leave SQL Server Management Studio open for the next exercise.

Results: After this exercise, you should have used DBCC CHECKDB to repair a corrupt database.

Exercise 2: Rebuild Indexes

Scenario

You have identified fragmentation in the **Sales.SalesOrderHeader** table in the **AdventureWorks** database and you are sure that performance is decreasing as the amount of fragmentation increases. You will rebuild the indexes for the table and confirm that the fragmentation level decreases.

The main tasks for this exercise are as follows:

1. Monitor and Maintain Indexes

► Task 1: Monitor and Maintain Indexes

1. Using SQL Server Management Studio, query the **sys.dm_db_index_physical_stats** function to retrieve information about the index fragmentation for indexes on the **Sales.SalesOrderHeader** table in the **AdventureWorks** database.
2. Note the **avg_page_space_used_in_percent** and **avg_fragmentation_in_percent** values for each index level.
3. Run the query in the **Ex2-InsertRows.sql** file in the **D:\Labfiles\Lab05\Starter\Ex2** folder to simulate OLTP activity.
4. Query the **sys.dm_db_index_physical_stats** function again, noting that the values for the same columns have changed due to the activity.
5. Rebuild all the indexes on the **Sales.SalesOrderHeader** table.
6. Query the **sys.dm_db_index_physical_stats** function again, confirming that the index fragmentation has decreased.
7. Close the solution without saving changes, but leave SQL Server Management Studio open for the next exercise.

Results: After this exercise, you should have rebuilt indexes on the **Sales.SalesOrderHeader** table, resulting in better performance.

Exercise 3: Create Database Maintenance Plans

Scenario

You want to ensure that there is an early detection of any integrity issues in the **AdventureWorks** database and that the database is regularly backed up to minimize recovery times. To do this, you will create database maintenance plans to schedule core operations on a weekly, daily and hourly basis.

The main tasks for this exercise are as follows:

1. Create a Database Backup Maintenance Plan
2. Create a Database Integrity Check Maintenance Plan
3. Run a Maintenance Plan

► Task 1: Create a Database Backup Maintenance Plan

- Create a database maintenance plan for the **AdventureWorks** database. The maintenance plan should perform the following operations:
 - A full database backup on a weekly basis.
 - A differential backup on a daily basis.
 - Transaction log backups on an hourly basis.

► Task 2: Create a Database Integrity Check Maintenance Plan

- Create a database maintenance plan for the **AdventureWorks** database. The maintenance plan should perform the following operation:
 - Check the integrity of the database using a short-term lock rather than an internal database snapshot.

► Task 3: Run a Maintenance Plan

1. Run the maintenance plan you created in the previous task.
2. View the history of the maintenance plan in the Log File Viewer in SQL Server Management Studio.
3. Close SQL Server Management Studio without saving changes.

Results: After this exercise, you should have created the required database maintenance plans.

Question: What is the difference between an OLTP database and an OLAP database in terms of recoverability and the probability that you will have to use Emergency mode?

Question: Is fragmentation always a bad thing in a database?

Module Review and Takeaways

In this module, you learned how to use the DBCC CHECKDB command to detect and repair database integrity issues. You then learned how to observe and repair index fragmentation. Finally, you learned how to use a maintenance plan to automate core maintenance tasks.



Best Practice: When planning ongoing database maintenance, consider the following best practices:

- Run DBCC CHECKDB regularly.
- Synchronize DBCC CHECKDB with your backup strategy.
- If corruption occurs, consider restoring the database from a backup, and only repair the database as a last resort.
- Defragment your indexes when necessary, but if they get beyond about 30 percent fragmentation, consider rebuilding instead.
- Update statistics on a schedule if you don't want it to occur during normal operations.
- Use maintenance plans to implement regular tasks.

Real-world Issues and Scenarios

Where possible, it is always a good idea to separate the data from the log files within the operating system to ensure that, if one goes down, the other is still available.

It is also preferable to have multiple files and filegroups for the data and system—and to change the default from the primary system filegroup to one of the business focused ones. This will ensure that you can perform partial recoveries rather than having to do a full database recovery, in the case where damage only occurs to part of the database structures.

Module 6

Database Storage Options

Contents:

Module Overview	6-1
Lesson 1: SQL Server Storage Performance	6-2
Lesson 2: SMB Fileshare	6-12
Lesson 3: SQL Server Storage in Microsoft Azure	6-15
Lesson 4: Stretch Database	6-19
Lab: Implementing Stretch Database	6-23
Module Review and Takeaways	6-25

Module Overview

One of the most important roles for database administrators working with Microsoft® SQL Server® is the management of databases and storage. It is important to know how data is stored in databases, how to create databases, how to manage database files, and how to move them. Other tasks related to storage include managing the **tempdb** database and using fast storage devices to extend the SQL Server buffer pool cache.

Objectives

After completing this module, you will be able to:

- Understand how I/O performance affects SQL Server.
- Implement SQL Server storage on SMB Fileshare.
- Store SQL Server data files in Microsoft Azure®.
- Implement a Stretch Database.

Lesson 1

SQL Server Storage Performance

In this lesson, you will learn about the different options available for SQL Server storage and consider how this might affect performance.

Lesson Objectives

After completing this lesson, you will be able to:

- Understand how I/O performance impacts SQL Server.
- Understand the relationship between the number of spindles (disks) and I/O performance.
- Decide on appropriate RAID levels for SQL Server.
- Understand Windows® storage spaces and how they can be used with SQL Server.
- Understand the requirements for the storage of data files and transaction log files.
- Work with **tempdb** data files.
- Evaluate other storage options for SQL Server.

I/O Performance

I/O performance has a direct impact on the overall performance of SQL Server, making it critical that the storage layer is designed and implemented in a way that delivers the best performance for the server's anticipated workload.

When SQL Server writes data to a database, the write takes place in memory before being committed to the transaction log file on disk. Only when the data is written to the transaction log file is the write considered complete. Any delay in writing data to the transaction log can result in undesirable performance within SQL Server, unacceptable performance of any application using the SQL Server databases, and a poor user experience.

When designing storage for your SQL Server estate, it is important to consider the likely workload. A read-heavy database will have different requirements to a write-heavy database; a data warehouse will have different requirements to a transactional system. Having a good idea of the anticipated read/write split will mean you can develop a storage solution that performs well.

For a new instance of SQL Server, finding the likely workload can be challenging and will involve a certain amount of guesswork. However, for an existing SQL Server instance, you can use built-in dynamic management views, such as **sys.dm_io_virtual_file_stats**, to gain an insight into the SQL Server workload.

- Critical to SQL Server performance
- Design storage solutions to meet workload
- Use DMVs to assess workload on a running instance
- **sys.dm_io_virtual_file_stats**

Reviewing Wait Statistics

SQL Server is a multithreaded application that is designed to make optimal use of CPU, memory, and disk resources. In any multithreaded application, a single thread might have to wait for access to a resource. This is because threads share resources with other threads from the same application, threads from other applications on the same computer, and threads from the operating system, which may have higher priorities for access. One way to assess the performance of your system is to analyze wait statistics for threads and SQL Server provides many Database Management Views (DMVs) for this purpose. Some of these DMVs relate to wait times for access to disk storage. Therefore, you should review these DMVs to see if your disk configuration could be improved and reduce average wait times.

- Use DMVs to review wait statistics:
 - sys.dm_os_waiting_tasks
 - sys.dm_os_wait_stats
 - sys.dm_exec_session_wait_stats
- Wait types that relate to the storage system:
 - Page IO Latch waits
 - WRITELOG waits
 - LOGBUFFER waits
 - ASYNC_IO_COMPLETION waits
 - IO_COMPLETION waits

The analysis of wait statistics and queues is complex and a full description of the statistics available is beyond the scope of this course. For more information about this subject, see:

 **SQL Server Best Practices Article – SQL Server 2005 Waits and Queues**

<http://aka.ms/nzz69y>

DMVs you can use to access wait statistics include:

- **sys.dm_os_waiting_tasks.** This DMV displays the list of tasks that are currently waiting together with the wait type, the wait duration, the resource that the task is waiting for, and other details.
- **sys.dm_os_wait_stats.** This DMV displays the total wait time for each wait type.
- **sys.dm_exec_session_wait_stats.** This DMV displays the same data as the previous DMV but filtered for a single active user session.

When you are investigating the performance of your storage system, the following wait types are often relevant:

- **Page IO Latch Waits.** These wait types have names that start with "PAGEIOLATCH". They indicate that a task is waiting to access a data page that is not in the buffer pool, so it must be read from disk to memory. A high wait time for PAGEIOLATCH wait types might indicate a disk I/O bottleneck. It might further indicate that the I/O subsystem is slow to return the pages to SQL Server.
- **WRITELOG.** The WRITELOG wait type indicates that a task is waiting for a transaction log block buffer to be flushed to disk. If the disks that store the transaction logs perform poorly, such waits may multiply.
- **LOGBUFFER.** The LOGBUFFER wait type indicates that a task is waiting for a log buffer to become available when it is flushing the log content.
- **ASYNC_IO_COMPLETION.** The ASYNC_IO_COMPLETION wait type indicates that a task is waiting for an I/O operation that does not relate to a data file to complete—for example, initializing a transaction log file or writing to backup media.
- **IO_COMPLETION.** The IO_COMPLETION wait type indicates that a task is waiting for a synchronous I/O operation that does not relate to a data file to complete; for example, reading a transaction log for a rollback or for transactional replication.

Number of Spindles

All hard disks have physical limitations that affect their performance. They are mechanical devices with platters that can only spin at a certain speed, and heads that can only move at a certain speed, reading a single piece of data at a time. These performance-limiting characteristics can be an issue for SQL Server, which in a production environment, might need to deal with hundreds or even thousands of requests per minute.

Adding more spindles (physical disks) can significantly improve storage performance. Therefore, to ensure that physical disk limitations do not impact on the performance of SQL Server, database administrators should design the SQL Server storage layer to have as many spindles as necessary to meet the I/O and latency demands expected on the server.

The key point to remember when designing SQL Server Storage is that many small capacity high-speed disks will outperform a single large capacity high-speed disk.

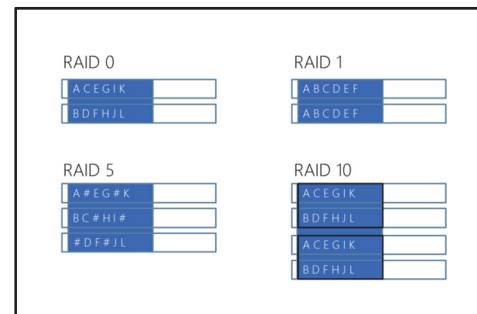
- Hard disk drives have physical limitations:
 - Platter spin speed
 - Seek time (head movement)
 - Serialized access
- Overcoming limitations:
 - Use many spindles (physical disks)

RAID Disks

Many storage solutions use RAID hardware to provide fault tolerance through data redundancy, and in some cases, to improve performance. You can also use the Windows Server® operating system to implement software-controlled RAID 0, RAID 1, and RAID 5; other levels might be supported by third-party SANs.

Commonly used types of RAID include:

- **RAID 0, disk striping.** A stripe set consists of space from two or more disks that is combined into a single volume. The data is distributed evenly across all of the disks, which improves I/O performance, particularly when each disk device has its own hardware controller. RAID 0 offers no redundancy, and if a single disk fails, the volume becomes inaccessible.
- **RAID 1, disk mirroring.** A mirror set is a logical storage volume that is based on space from two disks, with one disk storing a redundant copy of the data on the other. Mirroring can provide good read performance, but write performance can suffer. RAID 1 is expensive for storage because 50 percent of the available disk space is used to store redundant data.
- **RAID 5, disk striping with parity.** RAID 5 offers fault tolerance using parity data that is written across all the disks in a striped volume that is comprised of space from three or more disks. RAID 5 typically performs better than RAID 1. However, if a disk in the set fails, performance degrades. In terms of disk space, RAID 5 is less costly than RAID 1 because parity data only requires the equivalent of one disk in the set to store it. For example, in an array of five disks, four would be available for data storage, which represents 80 percent of the total disk space.



- **RAID 10, mirroring with striping.** In RAID 10, a nonfault tolerant RAID 0 stripe set is mirrored. This arrangement delivers the excellent read/write performance of RAID 0, combined with the fault tolerance of RAID 1. However, RAID 10 can be expensive to implement because, like RAID 1, 50 percent of the total space is used to store redundant data.

Consider the following points when planning files on RAID hardware:

- Generally, RAID 10 offers the best combination of read/write performance and fault tolerance, but is the most costly solution.
- Write operations on RAID 5 can sometimes be relatively slow compared to RAID 1 because you need to calculate parity data (RAID 5). If you have a high proportion of write activity, therefore, RAID 5 might not be the best candidate.
- Consider the cost per GB. For example, implementing a 500 GB database on a RAID 1 mirror set would require (at least) two 500 GB disks. Implementing the same database on a RAID 5 array would require substantially less storage space.
- Many databases use a SAN, and the performance characteristics can vary between SAN vendors and architectures. For this reason, if you use a SAN, you should consult with your vendors to identify the optimal solution for your requirements. When considering SAN technology for SQL Server, always look beyond the headline IO figures quoted and consider other characteristics, such as latency, which can be considerably higher on a SAN, compared to DAS.
- Windows storage spaces help you create extensible RAID storage solutions that use commodity disks. This solution offers many of the benefits of a specialist SAN hardware solution, at a significantly lower cost.

Storage Spaces

Storage Spaces is a Windows Server 2012 technology that enables administrators to create highly flexible and recoverable configurations on hard disks. You use storage spaces to virtualize storage by grouping industry-standard disks into storage pools. You can then create virtual disks, called storage spaces, from the available capacity of the storage pool.

You create storage pools from hard disks and solid-state disks of either IDE, SATA, SAS, or USB formats. If, after creating a storage pool, you run low on disk space, you can add more disks to the pool to increase the available storage capacity, without needing to copy or move data. You can also implement storage tiers within storage pools; you can also move frequently accessed data to faster disks within the pool.

Resiliency is built into storage spaces, with three different levels available, depending on your storage needs.

- **Mirroring:** writes multiple copies of data across multiple disks in a similar way to a RAID 1 disk set. In the event of disk failure, mirroring offers maximum protection for your data, giving good read and write performance, though disk capacity is reduced.

- Storage spaces
- Allow pooling of disks
 - Including internal and external IDE, SATA, SAS, USB
 - Provide resiliency
 - Flexible
 - Can be expanded easily

- **Parity:** writes a single copy of data striped across the disks, along with a parity bit to assist with data recovery. Parity gives good capacity and read performance, but write performance is generally slower, due to the need to calculate parity bits. Parity is similar to a RAID 5 disk set.
- **Simple:** stripes data across all disks as a single copy with no parity and is technically similar to a RAID 0 disk set. Simple maximizes storage capacity, giving high performance but offering no resiliency. Losing a disk will mean data is lost.

For more information on Storage Spaces, see:



Storage Spaces Overview

<http://aka.ms/dyg4dk>

Configuring Storage Pools and Storage Spaces

To use storage spaces on a single Windows server, you must start by creating one or more storage pools, then create one or more storage spaces (also known as virtual disks) within those pools. Finally, you must create one or more volumes within those storage spaces, which an application such as SQL Server sees as partitions on a hard disk.

A storage pool is a group of multiple physical disks. To create a storage pool:

1. In the Server Management application, in the navigation pane, click **File and Storage Services** and then click **Storage Pools**.
2. Click the **TASKS** list, and then click **New Storage Pool**.
3. In the New Storage Pool wizard, click **Next**.
4. Enter a name and a description for the new storage pool.
5. Select the group of physical disks to add the new storage pool, and then click **Next**.
6. Select the individual disks that you want to include in the storage pool, and then click **Next**.
7. Click **Create**.
8. On the **View Results** page, click **Close**.

To create a storage space:

1. In Server Manager, on the Storage Pools page, select the storage pool that you already created.
2. Under **VIRTUAL DISKS**, click the **TASKS** list, and then click **New Virtual Disk**.
3. In the New Virtual Disk wizard, click **Next**.
4. Select a storage pool, and then click **Next**.
5. Enter a name and a description for the new storage space, and then click **Next**.
6. Select a storage resiliency layout, and then click **Next**.
7. Select a provisioning type, and then click **Next**.
8. Specify the size of the storage space and then click **Next**.

1. Create a storage pool
2. Create a storage space
3. Create a volume

9. Click **Create**.
10. When the process is complete, click **Close**.

To create a volume:

1. In Server Manager, on the Storage Pools page, under **VIRTUAL DISKS**, right-click the storage space you just created, and then click **New Volume**.
2. In the New Volume wizard, click **Next**.
3. Select a server and a storage space on which you want to create a volume, and then click **Next**.
4. Specify the size of the volume, and then click **Next**.
5. Assign a drive letter or folder to the new volume, and then click **Next**.
6. Select a file system, allocation unit size, and volume label, and then click **Next**.
7. Click **Create**.
8. When the process is complete, click **Close**.

For more information about storage spaces, see:



Deploy Storage Spaces on a Stand-Alone Server

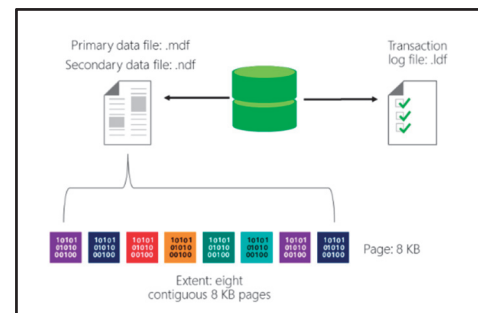
<https://aka.ms/vkbeuy>

Data and Log Storage

SQL Server uses two types of database file: data files and transaction log files.

Data Files

Every database has at least one data file. The first data file usually has the filename extension **.mdf**; in addition to holding data pages, the file holds pointers to the other files used by the database. Additional data files usually have the file extension **.ndf**, and are useful for spreading data across multiple physical disks to improve performance.



Transaction Log Files

Transaction log files hold details of all transactions that have occurred on a database. The information in a transaction log file is useful if you need to recover a database to a point in time between backups. There must be at least one transaction log file for each database. All transactions are written to the log file using the write-ahead logging (WAL) mechanism to ensure the integrity of the database in case of a failure, and to support rollbacks of transactions. The recommended extension for log files is **.ldf**.

When data pages need to be changed, they are fetched into memory and changed there. The changes are then written to the transaction log in a synchronous manner. During a background process known as a "checkpoint", the changes are written to the database files. For this reason, the pages contained in the transaction log are critical to the ability of SQL Server to recover the database to a known committed state. Write performance of transaction log files is critical to database performance; the files are usually placed on fast resilient disks, such as a RAID 1 or RAID 10 disk set. You should avoid putting transaction log files onto RAID 5 disk sets because the time taken to calculate parity bits can impact SQL Server performance.



Note: SQL Server also uses the transaction log file for other features, such as transactional replication, database mirroring, and change data capture. These topics are beyond the scope of this course.

Pages and Extents

Data files store data on pages, which are grouped into extents.

Data File Pages

SQL Server numbers the pages in a data file sequentially, starting with zero for the first page. Each file in a database has a unique file ID number. To identify a page in a database uniquely, both the file ID and the page number are required. Each page is 8 KB in size. After allowing for header information that is needed on each page, there is a region of 8,096 bytes remaining for holding data. Data rows can hold fixed length and variable length column values. All fixed length columns of a data row need to fit on a single page, within an 8,060-byte limit. Data pages only hold data from a single database object, such as a table, or an index.

Extents

Groups of eight contiguous pages are referred to as an extent. SQL Server uses extents to simplify the management of data pages. There are two types of extents:

- **Uniform extents.** All pages in the extent contain data from only one object.
- **Mixed extents.** The pages of the extent can hold data from different objects.

The first allocation for an object is at the page level, and always comes from a mixed extent. If they are free, other pages from the same mixed extent will be allocated to the object as needed. After the object has grown bigger than its first extent, then all future allocations are from uniform extents.

In all data files, a small number of pages are allocated to track the usage of extents within the file.

tempdb Storage

The performance of the **tempdb** database is critical to the overall performance of most SQL Server installations. The **tempdb** database consists of the following objects:

- **Internal objects**

Internal objects are used by SQL Server for its own operations. They include work tables for cursor or spool operations, temporary large object storage, work files for hash join or hash aggregate operations, and intermediate sort results.



Note: Working with internal objects is an advanced concept beyond the scope of this course.

- **tempdb:**

- Contains temporary data for internal objects, row versioning, and user objects
- Is truncated or rebuilt with every restart of the instance
- Occupies varying amounts of space
- Should be tested with real-life workloads

- Place **tempdb** on a fast and separate I/O subsystem to ensure good performance
- Split **tempdb** into data files of equal size per core

- **Row versions**

Transactions that are associated with snapshot-related transaction isolation levels can cause alternate versions of rows to be briefly maintained in a special row version store in **tempdb**. Other features can also produce row versions, such as online index rebuilds, Multiple Active Result Sets (MARS), and triggers.

- **User objects**

Most objects that reside in the **tempdb** database are user-generated and consist of temporary tables, table variables, result sets of multistatement table-valued functions, and other temporary row sets.

Planning tempdb Location and Size

By default, the data and log files for **tempdb** are stored in the same location as the files for all other system databases. If your SQL Server instance must support database workloads that make extensive use of temporary objects, you should consider moving **tempdb** to a dedicated volume—to avoid fragmentation of data files—and set its initial size based on how much it is likely to be used. You can leave autogrowth enabled, but set the growth increment to be quite large, to ensure that performance is not interrupted by frequent growth of **tempdb**. You can choose the location of **tempdb** files during installation, and move them later if required.

Because **tempdb** is used for so many purposes, it is difficult to predict its required size in advance. You should carefully test and monitor the size of your **tempdb** database in real-life scenarios for new installations. Running out of disk space in the **tempdb** database can cause significant disruptions in the SQL Server production environment, and prevent applications that are running from completing their operations. You can use the **sys.dm_db_file_space_usage** dynamic management view to monitor the disk space that the files are using. Additionally, to monitor the page allocation or deallocation activity in **tempdb** at the session or task level, you can use the **sys.dm_db_session_space_usage** and **sys.dm_db_task_space_usage** dynamic management views.

By default, the **tempdb** database automatically grows as space is required, because the MAXSIZE of the files is set to UNLIMITED. Therefore, **tempdb** can continue growing until it fills all the space on the disk.

Using Multiple Files

Increasing the number of files in **tempdb** can overcome I/O restrictions and avoid latch contention during page free space (PFS) scans, when temporary objects are created and dropped, resulting in improved overall performance. You should not create too many files, because this can degrade the performance. As a general rule, it is advised to have 0.25-1 file per processing core, with the ratio lower as the number of cores on the system increases. However, you can only identify the optimal configuration for your system by doing real live tests.

Demonstration: Moving tempdb Files

In this demonstration, you will see how to move **tempdb** files.

Demonstration Steps

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are running, and log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Demofiles\Mod06** folder, run **Setup.cmd** as Administrator. Click **Yes** when prompted.
3. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.

4. In Object Explorer, expand **Databases**, expand **System Databases**, right-click **tempdb**, and then click **Properties**.
5. In the **Database Properties** dialog box, on the **Files** page, note the current files and their location. Then click **Cancel**.
6. Open the **MovingTempdb.sql** script file in the **D:\Demofiles\Mod06** folder.
7. View the code in the script, and then click **Execute**. Note the message that is displayed after the code has run.
8. View the contents of **D:** and note that no files have been created in that location, because the SQL Server service has not yet been restarted.
9. In Object Explorer, right-click **MIA-SQL**, and then click **Restart**. When prompted, click **Yes**.
10. In the **Microsoft SQL Server Management Studio** dialog boxes, when prompted to allow changes, to restart the service, and to stop the dependent SQL Server Agent service, click **Yes**.
11. View the contents of **D:** and note that the **tempdb MDF** and **LDF** files have been moved to this location.
12. Keep SQL Server Management Studio open for the next demonstration.

Other Storage Options

Other storage options available to SQL Server include the following:

Network Attached Storage (NAS) is generally low cost and easy to administer. While it is possible to use NAS as storage for SQL Server, it is not usually recommended because NAS can rarely offer the performance that SQL Server requires. NAS I/O is limited by network connectivity and network bandwidth is usually shared with other traffic. Some NAS devices use RAID disk sets internally.

Direct Attached Storage (DAS) is a more traditional storage solution where disks are attached directly to the physical server. DAS, when combined with appropriate RAID configuration, generally offers excellent performance and low latency. DAS is not shared, and therefore can only be accessed by or via the server to which it is attached. DAS can be difficult to expand, although using DAS combined with Windows storage spaces addresses this issue to a certain degree.

Storage Area Network (SAN) generally comprises a separate network used specifically for attaching dedicated storage devices. SANs are fast, reliable, and scalable, but can be expensive to implement. SANs generally come in two formats, Fiber Channel and iSCSI, with the latter being cheaper and slower. SAN implementations vary by vendor and you should check their documentation carefully before choosing a SAN technology. A SAN will generally give performance that is almost as good as DAS, although latency is usually higher. SAN administration can be difficult and it is not unusual for large organizations to employ specialists just to look after a SAN.

Other Storage Options

- NAS: Network Attached Storage
- DAS: Direct Attached Storage
- SAN: Storage Area Network

Question: You are implementing a SQL Server instance and decide to use RAID disks. Which RAID levels might you choose for storing the following types of SQL Server files?

Transaction log files

Data files

tempdb

Lesson 2

SMB Fileshare

In this lesson, you will learn how about Server Message Block (SMB) Fileshare storage and the benefits of using it with SQL Server.

Lesson Objectives

After completing this lesson, you will be able to:

- Explain what SMB Fileshare is.
- Understand the benefits of using SMB Fileshare with SQL Server.
- Know how to implement SMB Fileshare for SQL Server.

What Is SMB Fileshare?

SMB is a network protocol developed by Microsoft and built into the Microsoft Windows® operating system. SMB has traditionally been used for file sharing and printer access. However, improvements made to recent versions of SMB and advances in networking technologies have dramatically improved performance and reliability. It is a viable option for storage in areas where it might not have been previously been considered, such as SQL Server data storage.

SMB file locations are addressed using standard UNC paths.

For clients and servers running Windows 8, Windows 2012 server, or later, an SMB file cluster can be built, making cost effective, highly available storage solutions.

SMB: Server Message Block

- Used for file and printer sharing
- Considerably improved in recent versions
- Shares addressed using UNC paths, for example:
 \\servername\share-name
- Can be configured in Windows Failover Clusters to provide highly available storage

Benefits of SMB Fileshare

Due to improvements in the SMB protocol and advances in networking technologies, SMB can rival the performance of fiber channel storage solutions, at a fraction of the cost.

Using SMB storage for SQL Server offers considerable management benefits when compared to DAS and SAN solutions. Moving a database to a new SQL Server instance becomes a simple process of detaching the database from one instance of SQL Server then attaching it to the other. There is no need for file copies, mapping and remapping of LUNs, or the involvement of storage administrators.

Benefits of SMB Storage for SQL Server

- Cost
- Manageability

Implementing SMB Fileshare

Before SQL Server can use an SMB share for storage, the SQL Server and SQL Server agent service accounts must be granted **Full Control** NTFS and share permissions.

After an SMB fileshare is configured with correct NTFS and share permissions, using the storage in SQL Server is straightforward. You create databases in the normal way, the only difference being how you reference the storage location of data and/or log files. Instead of using a drive letter and path as you would for other storage types, you use the full UNC path in the format \\servername\share-name.

SMB Share Permissions:

- Full Control – SQL Server Service Account
- Full Control – SQL Server Agent Service Account

Use UNC Path in Create Database statement

Share which cannot be used:

- Administrative Shares
- Loopback Shares
- Mapped Network Drives

Filestream on SMB is not supported

The example below shows a CREATE DATABASE statement using an SMB share for storage:

CREATE DATABASE statement

```
CREATE DATABASE [MySMBDatabase]
ON PRIMARY
( NAME = N'MySMBDatabase', FILENAME = N'\\Servername\share-name\MySMBDatabase.mdf' )
LOG ON
( NAME = N'MySMBDatabase', FILENAME = N'\\Servername\share-name\MySMBDatabaseLog.ldf' )
GO
```

Some SMB shares cannot be used for SQL Server storage. They are:

- Administrative shares (for example, \\servername\c\$).
- Loopback shares (for example, \\127.0.0.1\sharename).
- Mapped network drives.

You cannot currently place filestream data on an SMB share.

Demonstration: Storing a Database on an SMB Fileshare

In this demonstration, you will see how to store SQL Server data files on an SMB share.

Demonstration Steps

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are running, and log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Open File Explorer and navigate to the **D:** drive, right-click the **smbshare** folder, and then click **Properties**.
3. In the **smbshare Properties** dialog box, on the **Sharing** tab, in the **Network File and Folder Sharing** section, note that this folder is shared with the network path **\\MIA-SQL\smbshare**, and then click **Cancel**.
4. In SQL Server Management Studio, open the file **SMBDemo.sql** located in the **D:\Demofiles\Mod06** folder and execute the code it contains.
5. In File Explorer, navigate to the **D:\smbshare** folder and note the database files have been created.

6. Close SQL Server Management Studio without saving any changes.

Check Your Knowledge

Question	
You are creating a database on an SMB fileshare. Which of the following statements is a valid CREATE DATABASE statement?	
Select the correct answer.	
	<pre>CREATE DATABASE [Sales] ON PRIMARY (NAME = N'Sales_Data', FILENAME = N'\\SMBServer\d\$\SMBShare\Sales_data.mdf') LOG ON (NAME = N'Sales_Log', FILENAME = N'\\SMBServer\d\$\SMBShare\Sales_Log.ldf') GO</pre>
	<pre>CREATE DATABASE [Sales] ON PRIMARY (NAME = N'Sales_Data', FILENAME = N'\\127.0.0.1\SMBShare\Sales_data.mdf') LOG ON (NAME = N'Sales_Log', FILENAME = N'\\127.0.0.1\SMBShare\Sales_Log.ldf') GO</pre>
	<pre>CREATE DATABASE [Sales] ON PRIMARY (NAME = N'Sales_Data', FILENAME = N'\\SMBServer\SMBShare\Sales_data.mdf') LOG ON (NAME = N'Sales_Log', FILENAME = N'\\SMBServer\SMBShare\Sales_Log.ldf') GO</pre>

Lesson 3

SQL Server Storage in Microsoft Azure

You can use Microsoft Azure to store SQL Server data files in the cloud. In this lesson, you will learn about the benefits of storing SQL Server data files in the cloud and how to implement this technology.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe SQL Server data files in Azure.
- Explain the benefits of storing SQL Server data files in Azure.
- Implement SQL Server data files in Microsoft Azure.

What are SQL Server Data Files in Azure?

With Microsoft SQL Server 2016, you can create databases that store data files in Microsoft Azure storage as Microsoft Azure page blobs. On-premises and Microsoft Azure virtual machines both support this storage format, natively providing an entirely independent and dedicated storage location for your SQL Server data.

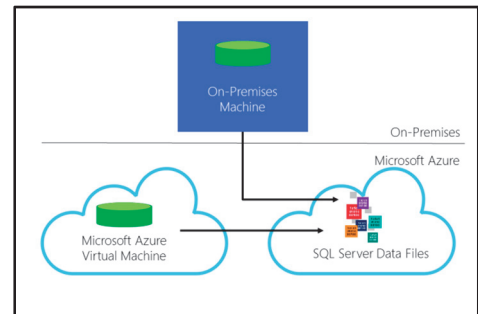
This technology helps the development of hybrid cloud/on-premises solutions, simplifies the provisioning of database storage, and facilitates the easy migration of databases between machines.

To use SQL Server data files in Microsoft Azure, you create a storage account, along with a container. You then create a SQL Server credential containing policies and the shared access signature essential to access the container. You can store page blobs inside the container, with each having a maximum size of 1 TB. There is no limit to the number of containers that a storage account can have, but the total size of all containers must be under 500 TB.

For more information on storage limits, see:

 **Azure Subscription and Service Limits, Quotas, and Constraints**

<http://aka.ms/n4eein>



Benefits of SQL Server Data Files in Microsoft Azure

The main benefits of storing SQL Server data files in Microsoft Azure are:

1. **Simpler Migration:** The migration of databases between on-premises and the cloud is greatly simplified. You can move data to the cloud without changing your applications.
2. **Separate Compute and Storage Nodes:** Using on-premises compute nodes, along with SQL Server data files in Azure, separates compute nodes from storage nodes. If you lose a compute node, you can quickly start up another without any data movement. Simply set up your node and attach the data files stored in Microsoft Azure.
3. **Simpler Disaster Recovery:** When storing SQL Server data files in Microsoft Azure, you can greatly simplify disaster recovery. If a physical or virtual SQL Server instance fails, you can quickly start up a new instance and attach the SQL Server data files stored in Microsoft Azure.
4. **Security:** By separating the compute node from the storage node, you can have an encrypted database where decryption only occurs on the compute node. All the data in the cloud is encrypted using Transparent Data Encryption (TDE) certificates stored in the master database in the on-premises instance. If your cloud data storage credentials are compromised, your data remains secure because the TDE certificates needed to decrypt the data are kept in a physically separate location to the data.
5. **Backup Strategy:** With SQL Server data files in Azure, you can use Azure snapshots for almost instantaneous backups of your data.

Benefits of SQL Server data files in Microsoft Azure

- Simpler migration
- Separation of compute and storage nodes
- Simpler database recovery
- Security
- Backup strategy

Choosing a Storage Account and Performance Tier

To use the Azure Storage service, you must first create a storage account. Storage accounts are of two types:

- **General-purpose storage accounts.** Within a general-purpose storage account, you can create various types of containers:
 - **Blob containers.** These contain Binary Large Objects (Blobs) such as files, images, videos, and virtual hard disks. Blob containers can store all types of blob, including page blobs.
 - **Tables.** These contain data in a semi-structured table.
 - **Queues.** These contain messages in queues that can be picked up and processed in order by other components in Azure.
 - **File Shares.** These are shared folders that virtual machines and on-premises computers can access through Server Message Block (SMB) like traditional file and print servers.

Storage Account Types:

- General-Purpose Storage Accounts
- Blob Storage Accounts

Storage Account Performance Tiers:

- Standard
- Premium

- For SQL Server data files in Azure, create general-purpose storage accounts and use the standard tier

- **Virtual Machine Disks.** These are specialized blobs that store virtual hard disk (VHD) files for virtual machines in Azure.
- **Blob storage accounts.** Within a blob storage account, you can create only block blobs and append blobs.

In order to place SQL Server data files in Azure, you must create them as page blobs. For this reason, you must create a general-purpose storage account, with a blob container. Note that a blob storage account does not support SQL Server data files, because it does not support page blobs.

When you create a general-purpose storage account, you must choose one of two performance tiers: Standard and Premium. Premium storage accounts currently only support virtual machine disks. Therefore, to place SQL Server data files in Azure you must choose the Standard tier.

For more information about Azure storage accounts, see:



About Azure storage accounts

<https://aka.ms/jxjrtr>

Implementing SQL Server Data Files in Azure

Implementing SQL Server data files in Microsoft Azure requires the following steps:

1. Create an Azure storage account.
2. Add a container to the account and apply an access policy.
3. Generate an SAS key for accessing the data.
4. Create a credential on the required SQL Server instance with a name that matches the Azure storage account container. See the following example code:

Implementing SQL Server Data Files in Azure

1. Create an Azure storage account
2. Add a container
3. Generate SAS key
4. Create a credential on the SQL Server instance
5. Create a database using the container credential

```
CREATE CREDENTIAL [https://myStorage.blob.core.windows.net/data]
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',
SECRET = 'CONTAINER SAS KEY'
```

5. Create your database with the data and log files in the Microsoft Azure container. See the following CREATE DATABASE statement:

```
CREATE DATABASE myDB on
( NAME = myDB_data,
FILENAME = 'https://myStorage.blob.core.windows.net/data/myDB_data.mdf' )
LOG ON
( NAME = myDB_log,
FILENAME = 'https://myStorage.blob.core.windows.net/data/myDB_log.ldf' )
```

For more in-depth information on implementing SQL Server data files in Azure, see:



Tutorial: Using the Microsoft Azure Blob storage service with SQL Server 2016 databases

<http://aka.ms/jgwswk>

Check Your Knowledge

Question	
One of your organization's databases has grown to a size that makes it difficult to back up during the available backup window. How might storing the SQL Server data files in Microsoft Azure help with this?	
Select the correct answer.	
☐	An Azure blob can be read faster than a local MDF file.
☐	SQL Server data files in Azure do not need to be backed up.
☐	With SQL Server data files in Azure, you can use Azure snapshots, providing almost instantaneous backups for your data.
☐	Putting data files in Azure will not help because SQL Server data files in Azure take longer to back up because the data is sourced from the Internet.

Lesson 4

Stretch Database

In this lesson, you will learn about Stretch Database, a feature you can use for the migration of historical, or cold, data to the cloud.

Lesson Objectives

After completing this lesson, you will be able to:

- Explain what a Stretch Database is.
- Understand the Stretch Database architecture.
- Understand Stretch Database security.
- Run the Stretch Database Advisor.
- Implement a Stretch Database.

What Is Stretch Database?

Stretch Database is a feature of SQL Server 2016 where data can be split between on-premises storage and cloud storage. With Stretch Database, cold, historical data is kept in the cloud and active data is kept on-premises for maximum performance. Stretch Database requires no changes to client applications or existing Transact-SQL queries, so you can implement it seamlessly for existing applications.

Stretch Database can reduce on-premises storage requirements both for data and associated backups. Backups of on-premises data are smaller and therefore run quicker than standard backups. Data in the cloud is backed up automatically.

With Stretch Database, cold historic data remains available for users to query, although there might be a small amount of additional latency associated with queries.

What is Stretch Database?

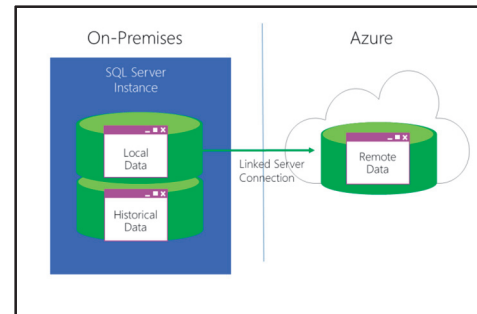
- Historical data – cloud storage
- Current data – on-premises

Benefits

- Quicker backups
- Reduced on-premises storage requirements
- Seamless to users

Stretch Database Architecture

With Stretch Database, the storage and query processing for cold data is offloaded to the Microsoft Azure cloud. When you enable Stretch Database for a database, SQL Server creates a secure linked server that uses the remote Microsoft Azure instance as the endpoint. Enabling Stretch Database for a table will provide remote resources, and if data migration is enabled, the migration of data will begin.



When you execute a query against a database with Stretch Database, it results in the query being run against both the on-premises and the secure linked server. Microsoft Azure rewrites the query, which is then shown in the local execution plan as a remote query operator.

Users still send their queries to the local SQL Server instance in the same way they have always done. There is no need for a user to have access to, or directly query, the remote linked server.



Note: A user will not normally know where the data they are querying resides. However, they might notice an increase in latency when querying data hosted in Azure.

Stretch Database Security

Before you can use Stretch Database, you must enable the instance level remote data archive configuration option. You can do this by using the **sp_configure** stored procedure with either **sysadmin** or **serveradmin** privileges.

```
exec sp_configure N'remote data archive',
N'1'
```

Any user with the control database permission can enable a database for Stretch Database, although they also need to provide administrator credentials for the Azure remote endpoint.

You can only configure a table for Stretch Database if it belongs to a Stretch Database enabled database. Any user with ALTER privilege for the table can configure Stretch Database for it.

There are some limitations on tables that can use Stretch Database. For more details on limitations, see:



Surface area limitations and blocking issues for Stretch Database

<http://aka.ms/prl0lw>

Data Security

Only server processes can access Stretch Database linked server definition. It is not possible for a user login to query the Stretch Database linked server definition directly.

Enable Stretch Database at the instance level Requires

- SysAdmin or ServerAdmin

Enable a database for Stretch Database requires

- Control Database
- Administrator Permissions for Remote Endpoint

Data security

- Unaffected by Stretch Database

Enabling a database for Stretch Database does not change the existing database or table permissions. The local database continues to check access permissions and manage security because all queries are still routed through the local server. A user's permissions remain the same, regardless of where the data is physically stored.

Stretch Database Advisor

The Stretch Database Advisor helps you identify databases and tables that are candidates for Stretch Database. It will also identify any issues preventing a database or table from being eligible for Stretch Database.

The Stretch Database Advisor is part of the Microsoft SQL Server Upgrade Advisor. Analyzing a database for compatibility involves the following steps:

1. Start the **Microsoft SQL Server Upgrade Advisor** application.
2. On the **Scenarios** tab, click **Run Stretch Database Advisor**.
3. Click **Select Databases to Analyze** and enter the appropriate server and authentication details.
4. Click **Connect**, select the databases you wish to analyze, and then click **Select**.
5. Click **Run** to start the analysis.

After the Stretch Database Advisor has completed its analysis, the results are displayed showing database size, along with the number of tables analyzed for each database. You can drill into these results to obtain details of any tables that are not compatible and the reasons for that.

For more details on tables that might not be compatible with Stretch Database, see:



Surface area limitations and blocking issues for Stretch Database

<http://aka.ms/prl0lw>



Note: While tables with Primary and/or unique keys are compatible with Stretch Database, a stretch enabled table will not enforce uniqueness for these constraints.

Stretch Database Advisor

- Part of SQL Server 2016 Upgrade Advisor
- Analyzes Databases for Stretch Database compatibility
- Highlights issues and blockers

Implement a Stretch Database

You can implement Stretch Database entirely within SQL Server Management Studio; you do not need to pre-configure servers or storage within Microsoft Azure.

Implementing Stretch Database involves the following steps:

1. Start Microsoft SQL Server Management Studio and connect to the instance of SQL Server.
2. In Object Explorer, expand **Databases**.
3. Right-click the database, point to **Tasks**, point to **Stretch**, and then click **Enable**.
4. Complete the steps in the **Enable Database for Stretch** wizard to create a Database Master Key; identify the appropriate tables and configure the Microsoft Azure deployment.

After implementing Stretch Database, you can monitor it from SQL Server Management Studio. In Object Explorer, expand **Databases**, right-click the stretch-enabled database, point to **Tasks**, point to **Stretch**, and then click **Monitor** to open the **Stretch Database Monitor**. This monitor shows information about both the local and Azure SQL instances, along with data migration status.



Note: If, after implementing Stretch Database, you later decide to disable Stretch Database for a table, you must migrate the data to the local instance before disabling—otherwise you will lose any data in the stretch table.

You can also implement Stretch Database by using Transact-SQL statements. For more information on using Transact-SQL to enable Stretch Database, see:



Enable Stretch Database for a database

<http://aka.ms/p5lb1z>

For more information on Stretch Database, see:



Stretch Database

<http://aka.ms/tgy2wd>

Entirely in Management Studio

- No need to preconfigure Azure storage or servers

Two Ways to Implement

- Wizard
- Transact-SQL

Question: Where might Stretch Database prove useful in your organization?

Lab: Implementing Stretch Database

Scenario

As the database administrator for Adventure Works, you have been asked to reduce the on-premises storage requirements for the production database without losing access to the data.

Objectives

After completing this lab, you will be able to:

- Run the Stretch Database Advisor.
- Implement Stretch Database.

Estimated Time: 45 minutes

Virtual machine: **20765B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Run Stretch Database Advisor

Scenario

As a database administrator for Adventure Works, you have been asked to identify data that could potentially be moved to the cloud to reduce load on the on-premises SQL Instance and shorten backup times.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Run Stretch Database Advisor

► Task 1: Prepare the Lab Environment

1. Ensure that the 20765B-MIA-DC, 20765B-MIA-SQL and MSL-TMG1 virtual machines are running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab06\Starter** folder as Administrator.

► Task 2: Run Stretch Database Advisor

1. Start SQL Server Upgrade Advisor.
2. Run Stretch Database Advisor on the **localhost** database instance and analyze only the **AdventureWorks** database.
3. Examine the results, noting that the **Sales.OrderTracking** table is marked as **Ready for Stretch**.

Results: After this exercise, you will know which tables within the Adventure Works database are eligible for Stretch Database.

Exercise 2: Implement Stretch Database

Scenario

You have identified the OrderTracking table as a good starting point for migrating data to the cloud and must now implement Stretch Database for this table.

The main tasks for this exercise are as follows:

1. Enable Stretch Database for the SQL Server Instance
2. Enable Stretch Database
3. Monitor Stretch Database

► Task 1: Enable Stretch Database for the SQL Server Instance

1. Start SQL Server Management Studio and connect to the **MIA-SQL** SQL Server instance.
2. In SQL Server Management Studio, enable Stretch Database for the MIA-SQL SQL Server instance.

► Task 2: Enable Stretch Database

- Enable Stretch Database for the **Sales.OrderTracking** table in the **AdventureWorks** database. Use **Pa\$\$w0rd** for the Master Key Password, **Student** for the Server Admin Login username, and **Pa\$\$w0rd** for the Admin login password.

► Task 3: Monitor Stretch Database

1. Open the **Stretch Database Monitor** and review the information provided.
2. Close SQL Server Management Studio without saving changes.

Results: After this exercise, you will have Stretch Database implemented for the OrderTracking table.

Module Review and Takeaways

In this module, you have learned how to:

- Understand how I/O performance affects SQL Server.
- Implement SQL Server storage on SMB Fileshare.
- Store SQL Server data files in Microsoft Azure.
- Implement a Stretch Database.

Review Question(s)

Question: What are the advantages of SMB Fileshare storage over SAN storage?

Module 7

Planning to Deploy SQL Server on Microsoft Azure

Contents:

Module Overview	7-1
Lesson 1: SQL Server on Virtual Machines and Azure SQL Database	7-2
Lesson 2: Azure Storage	7-11
Lesson 3: Azure SQL Server Authentication	7-15
Lesson 4: Deploying Databases in Azure SQL Database	7-20
Lab: Migrating SQL Server by Using Azure	7-23
Module Review and Takeaways	7-26

Module Overview

Microsoft® SQL Server® is integrated into the Microsoft cloud platform, Microsoft Azure®. It is possible to use SQL Server in Azure in two ways:

- SQL Server running on Azure virtual machines.
- Azure SQL Database run as a cloud service.

You will learn about the differences between these options for using SQL Server in Azure, and the considerations for choosing between them. You will also gain some experience of the migration process.

Objectives

At the end of this module, you will be able to:

- Describe the options that are available to run SQL Server on Azure.
- Assess the suitability of an existing database for migration to Azure.
- Migrate an on-premises database to Azure SQL Database.

Lesson 1

SQL Server on Virtual Machines and Azure SQL Database

In this lesson, you will explore the differences between the options for using SQL Server on Azure and learn about the strengths of each model.

Lesson Objectives

At the end of this lesson, you will be able to:

- Describe the options when you implement SQL Server on Azure virtual machines.
- Explain the architecture of Azure SQL Database.
- Describe the differences between, and the benefits of, virtual machines that host databases in Azure, and Azure SQL Database.
- Choose whether to implement Azure virtual machines in virtual networks.
- Choose whether to integrate SQL Server and Azure SQL Database with Azure Active Directory.
- Choose a performance tier for a deployment of Azure SQL Database.
- Describe how database transaction units measure database performance.
- Provision a virtual machine in Azure.

SQL Server on Azure Virtual Machines

When you run SQL Server on an Azure virtual machine, Microsoft provides compute and storage resources; you configure and administer Windows® and SQL Server. SQL Server on Azure virtual machines falls into the infrastructure as a service (IaaS) model of cloud services, and is conceptually similar to on-premises hardware virtualization.

Apart from the infrastructure on which it is hosted, running SQL Server on Azure virtual machines is almost identical to running SQL Server on-premises; you must manage all aspects of administration, such as backups, upgrades, and high availability.

Two different licensing models are available to run SQL Server on Azure virtual machines:

- **Platform-provided SQL Server image.** In this model, you use a virtual machine image that Azure provides, which includes a licensed copy of SQL Server. Licensed virtual machine images are available with SQL Server versions 2008 R2, 2012, 2014, and 2016 in Web Edition, Standard Edition, or Enterprise Edition. The billing rate of the image incorporates license costs for SQL Server and varies, based on the specification of the virtual machine and the edition of SQL Server. The billing rate does not vary by SQL Server version.
- **Bring-your-own SQL Server license.** In this model, you use a SQL Server license that you already own to install SQL Server on an Azure virtual machine. You may install the 64-bit edition of SQL Server 2008, 2008 R2, 2012, 2014, or 2016. This license model is only available to Microsoft Volume Licensing customers who have purchased Software Assurance.

- Infrastructure as a service model
- Azure provides compute and storage resources
- Customer administers Windows and SQL Server
- Licensed either through a prebuilt virtual machine image or with an existing license
- Web, Standard, and Enterprise Editions for SQL 2008 R2 and later are available as prebuilt images
- Azure virtual machines have an availability SLA of 99.95 percent
- Resource groups

Regardless of the license model that you select, and in addition to any SQL Server licensing costs, you will be billed for:

- Usage of Windows Server®.
- Azure storage costs for the virtual machine disks.
- Outbound Internet traffic (from Azure data centers).

Other than usage costs, and the limitations that are imposed by the version and edition of SQL Server that you select, there are no restrictions on SQL Server functionality when you use SQL Server on Azure virtual machines.

Azure virtual machines have a service level agreement (SLA) of 99.95 percent availability. Note that this SLA does not extend to services (such as SQL Server) that run on Azure virtual machines.

For more information about licensing requirements for the bring-your-own SQL Server license model, see *License Mobility through Software Assurance on Azure* in the Azure documentation:



License Mobility through Software Assurance on Azure

<http://aka.ms/mb002x>

For current information about pricing for Azure virtual machines and licensing costs for platform-provided SQL Server images, see *Linux Virtual Machines Pricing* in the Azure documentation:



Linux Virtual Machines Pricing

<http://aka.ms/x7cmej>

By default, when you create resources such as virtual machines and storage accounts in Azure, they are organized into logical containers called resource groups. You should place all of the Azure resources that are required to run an application into a single resource group, so that you can easily monitor them, control access to them, and manage costs. For example, if you want to run a database application on Azure virtual machines, you should place those virtual machines in a single resource group, along with the storage accounts where those virtual machines maintain virtual hard disks, and along with other infrastructure resources such as virtual networks (VNets) and load balancers.

Resources that are created in resource groups are deployed by using the Azure Resource Manager. You can create templates that use Azure Resource Manager to make it easy to deploy in a single operation all of the resources that an application needs. This is helpful, for example, when you want to deploy a single application multiple times, such as in different physical locations or for staging and production.

Azure SQL Database

Azure SQL Database aims to minimize administration costs for using SQL Server. The operating system, SQL Server instance, and most aspects of database administration—such as upgrades, backups, and high availability—are managed by Microsoft. You are only responsible for the data that is held in the database. Azure SQL Database falls into the software as a service (SaaS) and platform as a service (PaaS) models of cloud services.

- Platform as a service and Software as a service model
- Hardware, software, and almost all administration provided by Azure
- No license considerations or additional storage costs
- Some concepts and Transact-SQL commands not supported
- Limits on database size
- Azure SQL Database has an availability SLA of 99.95 percent

The billing cost of Azure SQL Database is based on the performance tier that is selected for the database. Performance tiers will be covered in more detail later in this lesson.

In addition to the performance tier cost, you will be billed for outbound Internet traffic.

Azure SQL Database does not have a version or edition that is directly comparable to SQL Server installations that you manage yourself. New features may be added to Azure SQL Database at any time. A small subset of SQL Server features is not available to (or not applicable for) Azure SQL Database, including some elements of Transact-SQL.

Azure SQL Database places limits on maximum database size. This size limit varies by performance tier, with the highest service tier supporting databases that have a maximum size of 1 terabyte at the time of writing.

The tools that are available to assist in the migration of databases to Azure SQL Database will be covered later in this module.

Azure SQL Database has an SLA of 99.95 percent availability.

For a list of general restrictions for Azure SQL Database, see *Azure SQL Database features* in the Azure documentation:



Azure SQL Database features

<http://aka.ms/fuyplz>

For details of unsupported Transact-SQL features in Azure SQL Database, see *Azure SQL Database Transact-SQL differences* in the Azure documentation:



Azure SQL Database Transact-SQL differences

<http://aka.ms/ybpqh8>

For current information about pricing for Azure SQL Database, see *SQL Database Pricing* in the Azure documentation:



SQL Database Pricing

<http://aka.ms/tskc9i>

Virtual Machine or Azure SQL Database?

Azure SQL Database and SQL Server on Azure virtual machines each meet a different group of requirements. When you choose between them, you must consider how your organizational requirements and resources match with the strengths of each service.

SQL Server on Azure virtual machines is recommended when:

- You want to migrate existing databases to the cloud with minimal application changes.
- You have a requirement for dedicated servers running SQL Server for development and testing, but do not want to buy on-premises hardware and software.
- You have existing IT resources to provide support and maintenance.
- You want full administrative control of SQL Server and Windows.
- You want to work with databases that are larger than 1 terabyte.
- You want to run hybrid applications with some components in the cloud and some components on-premises.

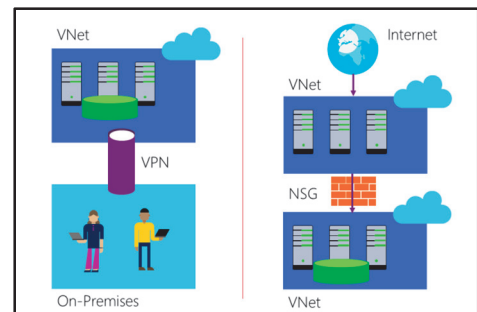
Azure SQL Database is recommended when:

- You are building new cloud-based applications.
- You are building applications that use a scale-out pattern.
- You want to minimize administration and support costs.
- You do not have suitable IT resources to provide support and maintenance.
- You do not need full administrative control of SQL Server.
- You want to work with databases smaller than 1 terabyte.

- Is this a new or existing application?
- Will databases of greater than 1 terabyte be needed?
- Are IT resources available for support and administration?
- Is full administrative control required?
- Will the application be a cloud/on-premises hybrid?

Azure Virtual Networks

On-premises computer networks are divided into subnets with routers that enable computers on separate subnets to communicate. Some subnets, such as perimeter networks (also known as DMZs, demilitarized zones, and screened subnets), provide extra security because a firewall filters traffic at subnet boundaries.



When you create virtual machines in Azure, you can build an analogous arrangement called a virtual network. Virtual machines and other resources in the same VNet can communicate by using private IP addresses. To enable computers on separate VNets to communicate, you must create routes between those VNets by using network gateways at either end.

You can also connect an Azure VNet to your on-premises network by using a virtual private network (VPN). A VPN is an encrypted point-to-point connection. After the VPN has been established, on-premises clients can access virtual machines in the VNet in exactly the same way as if they were present in the on-premises network.

Using VNets to Host SQL Server Virtual Machines

Commonly, network administrators want to migrate on-premises database servers into Azure to eliminate the need to maintain an on-premises data center or to reduce maintenance costs. By creating a VNet that has multiple virtual machines to host SQL Server 2016, you can migrate databases into Azure with no modification to the database schema and little change to the client applications (frequently, only a new connection string is required in the client). By using multiple virtual machines, you increase the availability and resilience of the database system and ensure that the 99.95 percent availability guarantee in the SLA applies. By placing those virtual machines in the same VNet, you enable them to communicate with each other without extra configuration. Finally, a VPN enables on-premises client computers to connect to the database virtual machines through a secure private connection.

Using VNets to Isolate SQL Server Virtual Machines

In on-premises data networks, SQL Server is often used to host databases that support websites. For example, the website may use a database to store a product catalog. It is common to place web servers in a perimeter network, where any Internet user can access the website by using TCP port 80 and the HTTP protocol through an Internet-facing firewall. The supporting servers that are running SQL Server are placed behind a second firewall, which is open only to the Internet servers on specific and private port numbers. This arrangement protects the servers that are running SQL Server from any direct access by malicious users on the Internet.

A similar arrangement can be created in Azure by separating web server virtual machines and SQL Server virtual machines into two VNets. Instead of firewalls, you use Azure network security groups (NSGs) to filter network traffic and determine which hosts can communicate with the virtual machines on each VNet.

For more information about Azure VNets, see *Virtual networks* in the Azure documentation:



Virtual networks

<https://aka.ms/vuzuv3>

Azure Active Directory

Windows Active Directory

Using on-premises SQL Server, when you choose mixed authentication, user account credentials are stored in SQL Server to enable users to choose SQL Server Authentication to connect. SQL Server Authentication grants access only to the databases on that server. However, users can also access a SQL Server database by using the same account that they use to log on to Windows, by choosing Windows Authentication. In this case, the user account credentials are usually stored in Active Directory®. Active Directory is a distributed directory system that can replicate security principles such as user accounts, security groups, and computer accounts to multiple directory servers based in multiple physical locations.

- Use user accounts in Azure Active Directory to access SQL Server
- Use the same accounts to access all other Azure resources and applications
- Synchronize user accounts in Azure Active Directory with on-premises Active Directory

Azure Active Directory

When you move a database into Azure by migrating it onto an Azure virtual machine, or onto databases in Azure SQL Database, you can continue using either SQL Server Authentication or Windows Authentication. To use Windows Authentication, you must use Azure Active Directory to store user accounts and credentials. Azure Active Directory enables users to connect to many different Azure resources and applications by using the same account that has a single set of credentials.

Furthermore, you can synchronize an on-premises Active Directory forest with Azure Active Directory. In this case, each user need only remember a single user name and password to access all resources in the cloud and on your local network.

SQL Database Performance Tiers

There are two ways to purchase Azure SQL Database services: single database and elastic database pool.

Single Database

Single database is the original performance configuration option that is offered for Azure SQL Database. In it, each database has its own service tier.

- Single database—each database has its own service tier
- Elastic database pool—a group of databases share a pool of resources, allocated dynamically
- Service tiers
 - Basic—small databases that have low concurrency
 - Standard—suitable for most concurrent databases
 - Premium—highest levels of performance and availability
- Switch tiers at any time

Elastic Database Pool

Elastic database pool enables several databases to share a fixed pool of resources, whereby resources are directed to the busiest databases as demands change. This is designed to address the scenario in which several databases that run on Azure SQL Database have performance demands that vary widely over time—perhaps on a fixed daily or weekly schedule—where it would be time-consuming to switch single database service tiers in response to demand. The service tier is set at the level of the pool.

For both single database and elastic database pool, Azure SQL Database service tiers divide into three categories, as the following table shows.

Service tier	Suitable for
Basic	Simple, low-usage databases that have no concurrent users.
Standard	Databases that have concurrent users, but are not mission-critical.
Premium	Databases that have large numbers of concurrent users and a requirement for high levels of business continuity.

The key areas where service tiers differ include:

- **Point-in-time restore.** The premium tier offers the longest backup retention; the basic tier offers the shortest.
- **Disaster recovery strategy.** The premium tier offers multiple online replicas, the standard tier offers a single offline replica, and the basic tier offers database restore.
- **In-memory tables.** Only the premium tier supports in-memory tables.
- **Maximum concurrent requests, maximum concurrent logins and maximum concurrent sessions.** These all increase from basic to premium tiers.

Each tier contains one or more performance levels that offer different numbers of database transaction units (DTUs). A database, or elastic database pool, can be moved between service tiers and performance levels at any time.

The details of Azure SQL Database performance tiers and levels change over time. For current information about service tiers and performance levels, see *SQL Database options and performance: Understand what's available in each service tier* in the Azure documentation:



SQL Database options and performance: Understand what's available in each service tier

<http://aka.ms/dr66jn>

Database Transaction Units

DTUs are a measure of database performance that Microsoft uses to establish performance metrics for Azure SQL Database in a single database service tier. The measure is based on the number of transactions that a given hardware configuration can complete in a second—one DTU is equivalent to one transaction per second.

The higher the DTU value for an Azure SQL Database service tier, the more transactions per second it will support.

The performance of elastic database pools is given in elastic database transaction units (eDTUs). As with a DTU, one eDTU is equivalent to one transaction per second. A distinction is made between DTUs and eDTUs because eDTUs are only allocated as needed.

- Single database performance measured in database transaction units (DTUs)
 - 1 DTU = 1 transaction per second
- Elastic database pool performance measured in elastic database transaction units (eDTUs)
 - 1 eDTU = 1 transaction per second
 - eDTUs are only allocated to databases in the pool as they are required

Elastic database pools have a pool of available eDTUs and a limit of eDTUs per database, configured by the service tier. Overall performance of the pool will not degrade until all of the eDTUs that are assigned to it are actively consumed.

For complete details about the benchmark tests that are used to calculate DTUs, see *Azure SQL Database benchmark overview* in the Azure documentation:



Azure SQL Database benchmark overview

<http://aka.ms/smlp9r>

Demonstration: Provisioning an Azure Virtual Machine

In this demonstration, you will see how to provision an Azure virtual machine with SQL Server.

Demonstration Steps

1. Ensure that the MSL-TMG1, 20765B-MIA-DC, and 20765B-MIA-SQL VMs are running and log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Open **Internet Explorer** and go to **https://portal.azure.com/**.
3. Sign in to the Azure portal with your Azure Pass or Microsoft Account credentials.
4. Click **New**, click **See all**, click **Databases**, and then click **SQL Server**.
5. In the **SQL Server** pane, click **SQL Server 2016 SP1 Enterprise on Windows Server 2016**.
6. In the **SQL Server 2016 SP1 Enterprise on Windows Server 2016** pane, in the **Select a deployment model** box, click **Resource Manager**, and then click **Create**.
7. On the **Basics** blade, in the **Name** box, type a name for your server. This must be unique throughout the whole Azure service, so cannot be specified here. A suggested format is **sql2016vm-*<your initials>**<one or more digits>***. For example, **sql2016vm-js123**. Keep a note of the name you have chosen.
8. In the **User name** box, type **demoAdmin**.
9. In the **Password** box, type **Pa\$\$w0rd1234**.
10. In the **Resource group** box, type **resource1**.
11. Change the value of the **Location** box to a region near your current geographical location, and click **OK**.
12. In the **Choose a size** blade, click **View all** then click **D11 Standard**, and then click **Select**.
13. On the **Settings** blade, click **OK**.
14. On the **SQL Server settings** blade, click **OK**.
15. On the **Summary** blade, click **OK**. Deployment may take some time to complete.
16. When deployment is complete, click **Virtual machines**, then click the name of the machine you created in step 7.
17. In the server name blade, click **Connect**, then click **Open**.
18. In the **Remote Desktop Connection** dialog box, click **Connect**.
19. In the **Windows Security** dialog box, in the **User name** box, type **\demoAdmin**, in the **Password** box, type **Pa\$\$w0rd1234**, and then click **OK**.

20. In the **Remote Desktop Connection** dialog box, click **Yes**.
21. Inside the remote desktop session, start **SQL Server Management Studio** (SSMS) and connect to the VM instance of SQL Server. Demonstrate that it's a standard SQL Server installation.
22. Close the Remote Desktop connection, and then delete the Azure Virtual Machine.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false: Azure SQL Database supports exactly the same set of functionality as an on-premises SQL Server installation.	

Lesson 2

Azure Storage

In this lesson, you will learn about Azure virtual machines, including the sizes available, virtual disks that you can add, database page compression, and instant file initialization.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe the different sizes of Azure virtual machines that are available.
- Describe the different types of virtual disks that are available for Azure virtual machines.
- Describe what database page compression is.
- Describe the benefits of instant file initialization.

Sizes of Azure Virtual Machines

When you create an Azure virtual machine, you must choose a size. Larger sizes have more CPU and memory resources and can sustain high rates of disk input and output and higher network bandwidth. However, larger sizes incur larger costs. Choose a size that satisfies your expected demand, but minimizes expense.

Definition Parameters for Virtual Machine Sizes

Each virtual machine size is defined by six values:

- **Number of CPU cores.** Virtual machines that have more CPU cores execute processes more quickly.
- **Memory.** As with real computers, virtual machines that have more memory usually perform well because processes can store all of the information that they need in fast-access random access memory (RAM).
- **Size of the temporary hard disk.** All virtual machines include a temporary hard disk that is present on the physical machine that hosts the virtual machine. Operating system disks and data disks are stored separately in an Azure storage account. It is possible to move virtual machines at any time to a different physical server, so you may lose the information in this drive. You should consider it only suitable for temporary information. The temporary drive is created in a solid-state drive (SSD) for some sizes of virtual machine.
- **Maximum number of data disks.** Data disks can be created as binary large objects (BLOBs) in Azure storage accounts. Any given virtual machine size includes a maximum number of data disks.
- **Maximum data disk throughput.** This is measured in input/output operations per second.
- **Maximum network interface cards or network bandwidth.** Using more network cards can enable you to increase the network bandwidth that the virtual machine can utilize.

- Azure virtual machines are defined by:
 - CPU cores
 - Memory
 - Temporary hard disk
 - Maximum number of data disks
 - Maximum data throughput
 - Maximum network interface cards/network bandwidth
- The Azure compute unit
- Virtual machine sizes are divided into 'series':
 - D-Series—fast
 - F-Series—best value
 - G-Series—more memory

Azure Compute Units

The Azure compute unit (ACU) does not have a published definition; rather, it is used to understand the relative performance differences between Azure virtual machines. The ACU is calibrated against a Small (Standard A1) virtual machine being 100 ACUs. Other virtual machines are measured against this benchmark, and their ACU published accordingly.



Note: The ACU should only be used as a guideline. It has been designed as a shortcut to understanding the relative computing power of virtual machines, to help people make decisions about which size of virtual machine to use.

Size Series

Azure virtual machines are divided into different size series. This is designed to help you choose the best machine for your needs. Some examples are described below:

D-Series virtual machines have:

- Faster processors.
- A high memory-to-core ratio.
- An SSD temporary disk.

Dv2-Series virtual machines have:

- Intel Xeon E5-2673 v3 (Haswell) processors

F-Series virtual machines have:

- The best value to price performance.

G-Series virtual machines have:

- Intel Xeon E5 V3 processors.
- The most memory.

Other size series are available, and virtual machine sizes are being retired. For more information about the sizing of Azure virtual machines, see *Sizes for Windows virtual machines in Azure* in the Azure documentation:



Sizes for Windows virtual machines in Azure

<https://aka.ms/xecbxs>

Virtual Disk Types

Azure virtual machines are created with at least two disks: the operating system disk and a temporary disk. Optionally, you can add more disks.

Operating System Disk

A dedicated operating system disk is created when you create an Azure virtual machine. This is a 1,023-gigabyte (GB) SATA drive. By default, it is labeled Drive C.

Temporary Disk

This is not used for storing data; rather, it stores `pagefile.sys`. The temporary disk is created automatically and labeled Drive D. The size of the virtual machine that you have chosen determines the size and type of the temporary disk. You should never store data on the temporary disk unless you are happy to lose the data. For example, in the event of your virtual machine being received on a new host, perhaps because of hardware failure, a new temporary disk will be created and any data will not be moved.

Any storage space that is used on the temporary disk is not chargeable.

Data Disks

You can add data disks to your virtual machine, and the number of disks that you can add depends on the size of your virtual machine. You can choose the size of the data disk that you want to add, up to a maximum of 1,023 GB. Data disks can be either premium or standard. Standard storage uses hard disk drives (HDDs), and premium storage uses SSDs.

- Azure virtual machine disk types:
 - Operating system disk
 - Stores the operating system
 - 1,023 GB SATA drive
 - Labelled Drive C
 - Temporary disk
 - Labelled Drive D
 - Used for temporary data
 - No data is persisted in the event of failure
 - Do not use to store data
 - Data disks
 - The number that you can add depends on the size of the virtual machine
 - They determine the size of disk, up to a maximum of 1,023 GB

Data Compression

SQL Server supports compressed data for both rowstore and columnstore tables and indexes. Compression not only reduces the disk space required to store data, but can also improve performance. Because compressed data is stored in fewer pages, the time required to fetch data from the disk is reduced, thereby improving performance. Compression does, however, require CPU processing to compress and decompress the data. Over the years, CPU power has increased substantially, whilst disk seek and transfer times have improved at a slower rate, so compression makes good use of CPU power.

Rowstore tables and indexes

SQL Server and Azure SQL Database support both row and page compression. Using page compression can improve performance, particularly for databases that have high I/O workloads.

- Azure SQL Database supports row and page compression
- For rowstore tables and indexes with high I/O workloads, enable compression
- Columnstore compression is always used for columnstore indexes
- Further columnstore compression can be applied using columnstore archive compression
 - This reduces the space required, but also reduces performance
 - Use for data that is access infrequently, and where you want to minimize storage space



Note: The number of rows stored remains the same, despite data being compressed. However, more rows can be stored on a page with compression.

For rowstore tables and indexes, using compression both reduces the size of the database and for high I/O workloads, also improves performance. This is a trade-off, however, because compression means that fewer pages are read, but CPU resources are used to compress and decompress data.

You can compress tables without an index (a heap), tables with a clustered index, nonclustered indexes, and indexed views. Partitioned tables and indexes can also be compressed, with individual partitions using row, page, or no compression, as required.

Compressing data will change the way queries perform, as the query plan for compressed data will be different to the same query accessing uncompressed data. This is because compressed data is stored in fewer pages.



Note: You cannot compressed tables that include sparse columns.

There are other restrictions and considerations when compressing data, for more information see the MSDN article, *Data Compression*.

Columnstore tables and indexes

Columnstore indexes are always compressed. The use of compression with columnstore indexes reduces the amount of space that is needed, providing an automatic improvement in performance. You cannot turn off compression with columnstore indexes.

In addition to the automatic compression with columnstore indexes, you can also apply archival compression. Although this further reduces the space used by the index, it reduces performance as the CPU is further loaded to compressed and decompress data. Archival compression, as the name suggests, is most suited to archive data that is used relatively infrequently, and you want to minimize the space used to store it.

For an overview of compression in Azure SQL Database, see *Data Compression* on MSDN:



Data Compression

<https://aka.ms/mn5fcw>

Instant File Initialization

What Is Instant File Initialization?

Instant file initialization enables files to be created, or grow, very quickly. This is achieved by not filling the disk space with zeros. For security reasons, instant file initialization is off by default. When instant file initialization is disabled, SQL Server initializes by filling files with zeros, thereby overwriting any existing data that might have been left from deleted files. This happens when you create a database, add files to an existing database, or increase the size of a file, for example, by using autogrow. It is also done when you restore a database or filegroup.

- Instant file initialization
 - Faster initialization of disk space
 - Does not overwrite disks with zeros before using
- Off by default
- Use with on-premises SQL Server or SQL Server running in an Azure virtual machine

Instant file initialization skips the step of zeroing the disk space, and instantly uses the disk space without first writing zeros. Instant file initialization can only be used for data files, not log files.

When to Use Instant File Initialization

You can specify instant file initialization for Azure virtual machines, or SQL Server running on-premises. Azure SQL Database does not use instant file initialization.

Check Your Knowledge

Question	
Of the following statements, which is not true about Azure virtual machines?	
Select the correct answer.	
☐	You can add extra virtual disks.
☐	You can remove the temporary disk.
☐	You can choose the type of virtual machine based on CPU, memory, and storage capacity.
☐	Virtual machine data disks can be either HDD or SSD.
☐	You can delete an Azure virtual machine when you no longer need it.

Lesson 3

Azure SQL Server Authentication

In this lesson, you will learn about how security is managed in Azure SQL Database.

Lesson Objectives

After completing this lesson, you will:

- Understand the layered security model that Azure SQL Database uses.
- Understand how security is managed at the point of:
 - Connection.
 - Authenticating users.
 - Authorizing users to access securables within the database.
- Understand how to set up auditing in Azure SQL Database.

Security Overview of Azure SQL Database

Azure SQL Database protects data by using a layered security model. When Azure SQL Database V12 was released, major new security features were supported, including Always Encrypted, row-level security, and transparent data encryption (TDE). Since then, dynamic data masking has been added to the security options. Always Encrypted and dynamic data masking are discussed in a later module of this course.

- Azure SQL Database uses a layered security model
- Azure SQL Database V12 now supports:
 - Always Encrypted
 - Transparent data encryption
 - Row-level security
 - Dynamic data masking
- Data encryption
 - Making data unreadable to attackers
- Controlling access
- Controlling permissions to database objects
- Auditing and monitoring

Encrypting Data

Data is vulnerable to attack at various points, including when it is stored in the database and as it is communicated to users. Data encryption provides the highest level of protection, providing that the keys are held securely. Azure SQL Database provides options to encrypt data at all points:

- **During communication, using Transport Layer Security (TLS).** Azure SQL Database manages the encryption keys, including rotation.
- **While it is stored, using TDE.** TDE can be enabled or disabled through the Azure portal or by using Windows PowerShell®.
- **Data in use, using Always Encrypted.** This is set at the column level, either through the Azure portal or by using Windows PowerShell.

Who Has Access?

Controlling access to a database in Azure SQL Database happens at different points:

- **Firewall rules.** These allow or disallow access to the Azure server (or Azure SQL Database) based on IP addresses. Tight firewall rules are a first-line defense against hacking.
- **Authentication.** This occurs either through SQL Server Authentication or through Azure Active Directory.

Controlling What Users Can Do

After a user has access to a database in Azure SQL Database, you must then define what he or she can do. This is controlled by granting permissions to roles or users.



Best Practice: Grant users the least permissions necessary. Security is often a trade-off between convenience and protecting data. Grant the least permissions necessary for users to complete their work, and grant additional permissions temporarily for exceptional tasks.

Auditing and Monitoring

No system is 100 percent secure, added to which, security is almost impossible to measure. This means that auditing and monitoring are an essential part of data security. A surprisingly high number of data breaches come from within organizations, so it is important to monitor data access to ensure that it is within expected parameters.

Connection Security

Connection security is managed by using firewall rules. This limits access to the database server only to those computers that have been identified by using IP addresses. Many companies use a range of fixed IP addresses, and these are used to create firewall rules. Any computer that accesses a server running Azure SQL Database, or an Azure virtual machine, must be using an IP address within one of the firewall rules.

It is possible to set firewall rules at either the server level or the database level.

If you create firewall rules by using the Azure portal, you will be prompted to add the current IP address, or add new firewall rules. Alternatively, you can create firewall rules by using Windows PowerShell.

- Connection security is enforced by using firewall rules
- Firewall rules allow only certain IP addresses to connect
- Firewall rules can be set at the server or database level
- You can add or amend firewalls either through the Azure portal or using Windows PowerShell

Authentication

Like the on-premises version of SQL Server, Azure SQL Database now supports both SQL Server Authentication and Windows Authentication. You can now connect to Azure SQL Database by using Azure Active Directory Authentication. This means that each time users connect to Azure SQL Database, they can be authenticated by using Active Directory credentials.



Note: Windows Authentication is only supported in Azure SQL Database V12.

- Azure SQL Database supports both SQL Server Authentication and Windows Authentication
- Use Azure Active Directory for Windows Authentication

When you provision a new server running Azure SQL Database, several steps are completed:

- A server running Azure SQL Database is created, unless you have specified an existing server.
- A master database is created.
- A server-level login is created with the credentials that you specified. This is similar to the **sa** login that is created for on-premises SQL Server instances. You can then create additional logins as required.

For more information about Azure Active Directory Authentication, see *Use Azure Active Directory Authentication for authentication with SQL Database or SQL Data Warehouse* in the Azure documentation:



Use Azure Active Directory Authentication for authentication with SQL Database or SQL Data Warehouse

<https://aka.ms/cz224r>

Authorization

SQL Server grants permissions to users to view or work with a securable. Securables are things such as tables, views, and stored procedures. Users must have the appropriate permissions to work with various securables.

Access is allowed or denied to logins and users. Transact-SQL commands include:

- GRANT
- REVOKE
- DENY
- Sp_addrolemember
- Sp_droprolemember

- Authorization is required to work with SQL Server securables
- Securables are things such as tables, views, and stored procedures.
- Authorization is granted to principals: logins or database users

Auditing and Compliance

Azure SQL Database auditing enables you to gain access to security logs for auditing and compliance purposes. Logs can then be analyzed by using HDInsight®, or downloaded to be analyzed by using on-premises systems. No system can ever be completely secure, so auditing and monitoring are an important part of data protection.

Azure SQL Database Auditing

Azure SQL Database auditing enables you to log database events that you can define.

- No system is 100 percent secure, so auditing forms an important part of data protection
- Enable Azure SQL Database auditing through the Azure portal
- Event logs are stored in an Azure storage account
- View logs by using Azure Storage Explorer
- View logs in the Azure portal

To enable Azure SQL Database auditing:

- In the Azure portal, use the **Dashboard** or **All Resources** to select your server running Azure SQL Database.
- In the **Settings** group, select **Auditing & Threat Detection**.
- Ensure that **Inherit settings from server** is selected, and then click **View server auditing settings**.
- In the **Auditing & Threat Detection** blade, for **Auditing**, click **ON**.
- Choose **Auditing type**—either Blob or Table. Audit logs are written to either BLOB storage or table storage. BLOB storage gives higher performance.
- Select an **Azure storage account**. This is where log files will be saved.
- Select the **Azure storage account** where the log files will be saved, and then in **Retention (Days)**, select the number of days for which the files will be retained. Log files will be deleted after the number of days that has been selected.
- You can switch **Storage key access** between **Primary** and **Secondary** to allow key rotation. Leave it at **Primary** when you are setting up auditing.
- Click **OK**.
- Leave **Email service and co-administrators** selected, and then add **Email addresses** to enable email alerts to be sent.
- Click **Save**. Events will be logged for all databases on the server.

Analyzing Logs

You can use Azure Storage Explorer to view the logs in your storage account. This is discussed in a later module in this course.

If you have chosen BLOB storage, log files are saved in a container called **sqldbauditlogs**.

You can also view logs through the Azure portal:

- Select the relevant Azure SQL Database logical server.
- In the **Settings** group, select **Auditing & Threat Detection**.
- In the **Auditing & Threat Detection** blade, select **View audit logs**.

For a good explanation about Azure logs, together with links to other articles of interest, see *Get started with SQL database auditing* in the Azure documentation:



Get started with SQL database auditing

<https://aka.ms/m5gt1b>

Question: To what degree is your organization concerned with security when it considers moving data to the cloud? Do you think the security measures that are available in Azure SQL Database adequately answer those concerns? If not, what concerns do you feel have not been addressed?

Lesson 4

Deploying Databases in Azure SQL Database

Before you use Azure SQL Database, it is important to understand the performance and pricing model that is used for the service.

Azure SQL Database is offered on a SaaS model, so the performance and, correspondingly, the price of the service is measured in DTUs. Different performance levels are offered by different service tiers; the higher the performance of a service tier, the greater the number of DTUs it supports.

For current information about pricing for Azure SQL Database, see *SQL Database Pricing* in the Azure documentation:



SQL Database Pricing

<http://aka.ms/tskc9i>

Lesson Objectives

At the end of this lesson, you will be able to:

- Describe the performance tiers that are available for Azure SQL Database.
- Explain how tier performance is measured by using DTUs.
- Provision and connect to a database in Azure SQL Database.

Demonstration: Provisioning a Database in Azure SQL Database

In this demonstration, you will see how to provision a database in Azure SQL Database by using Azure PowerShell.

Demonstration Steps

1. On MIA-SQL, start **Windows PowerShell**.
2. Link your Azure account to PowerShell by running the following cmdlet:

```
Add-AzureRmAccount
```

When prompted press **y**, and then press Enter. When the sign-in screen appears, use the same email and password you use to sign in to the Azure portal. If you have already linked your Azure account to PowerShell on this VM, use the command:

```
Login-AzureRMAccount
```

3. Use the subscription Id returned in the output of the previous step and run the following cmdlet:

```
Select-AzureRmSubscription -SubscriptionId <your subscription id>
```

(replace <your subscription id> with the GUID value returned by the previous step.)

4. Run the following cmdlet to return the list of Azure data center locations supporting SQL Database:

```
(Get-AzureRmResourceProvider -ListAvailable | Where-Object {$_.ProviderNamespace -eq 'Microsoft.Sql'}).Locations
```


- Run the following cmdlet to create a resource group. Substitute a location near your current geographical location from the result returned by the previous step for <location>:

```
New-AzureRmResourceGroup -Name "resourcegroupPSTest" -Location "<location>"
```

- Run the following cmdlet to create a server in the new resource group. Substitute the location used in the previous step for <location>. Substitute a unique server name for <your server name>; this must be unique throughout the whole Azure service, so cannot be specified here. A suggested format is **sql2016ps-<your initials> <one or more digits>**. For example, **sql2016ps-js123**.

```
New-AzureRmSqlServer -ResourceGroupName "resourcegroupPSTest" -ServerName "<your server name>" -Location "<location>" -ServerVersion "12.0"
```

In the credential request dialog box, type the User name **psUser** and the password **Pa\$\$w0rd**. This step may take a few minutes to complete.

- Run the following cmdlets separately to create a firewall rule to allow your current client to connect to the server. Substitute the server name created in the previous step for <your server name>. Substitute your current external IP address for <your external ip>. You can get your current external IP address from the Azure Portal (see the value returned by the "Add Client IP" button on the firewall for an existing server), or from third party services such as Google (search for "what is my ip") or www.whatismyip.com:

```
$currentIP = "<your external ip>"
New-AzureRmSqlServerFirewallRule -ResourceGroupName "resourcegroupPSTest" -ServerName "<your server name>" -FirewallRuleName "clientFirewallRule1" -StartIpAddress $currentIP -EndIpAddress $currentIP
```

- Run the following cmdlet to create a database on the new server. Substitute the server name created in a previous step for <your server name>.

```
New-AzureRmSqlDatabase -ResourceGroupName "resourcegroupPSTest" -ServerName "<your server name>" -DatabaseName "TestPSDB" -Edition Standard -RequestedServiceObjectiveName "S1"
```

- Close Windows PowerShell ISE without saving any changes.

Demonstration: Connecting to a Database in Azure SQL Database

In this demonstration, you will see how to connect to a database in Azure SQL Database.

Demonstration Steps

- Open **Internet Explorer** and go to <https://portal.azure.com/>.
- Sign in to the Azure portal with your Azure Pass or Microsoft Account credentials.
- Click **SQL Databases**, then click **TestPSDB**. Note the value of **Server name**, and then click **Show database connection strings**.
- Open **SQL Server Management Studio**.
- In the **Connect to Server** dialog box, type the server name noted in the previous step (it will take the form <your server name>.database.windows.net).
- Set **Authentication** to **SQL Server Authentication**. In the **Login** box, type **psUser**, in the **Password:** box, type **Pa\$\$w0rd**, and then click **Connect**.

7. In **Object Explorer**, expand the **Databases** node to show the **TestPSDB** database.
8. On the **File** menu, point to **Open**, and click **File**.
9. In the **Open File** dialog box, open **D:\Demofiles\Mod07\query Azure.sql**.
10. On the **Available Databases** drop-down menu, click **TestPSDB**.
11. Select the script below **2. execute the following query**, and click **Execute**.
12. Open **Windows PowerShell**, and then type the command below **3. Open Windows PowerShell and type the following command** into the Windows PowerShell window, replacing <your server name> in the command with the server name used on step 3.
13. At the **Password** prompt, type **Pa\$\$w0rd**, and then press Enter.
14. Close PowerShell, close SSMS without saving any changes, and then close Internet Explorer.

Check Your Knowledge

Question	
Which of the following best defines a database transaction unit?	
Select the correct answer.	
☐	A database transaction unit is a measure of CPU performance.
☐	A database transaction unit is a measure of storage I/O performance.
☐	A database transaction unit is a measure of memory performance.
☐	A database transaction unit is a measure of overall system performance.

Lab: Migrating SQL Server by Using Azure

Scenario

As a pilot project for Adventure Works Cycles, you have been tasked with the migration of a SQL Server database, which is used by a sales application, to Microsoft Azure.

Objectives

After completing this lab, you will be able to:

- Use tools to assess the suitability of a database to be moved to Azure.
- Migrate a database to Azure.

Estimated Time: 45 minutes

Virtual machine: **20765B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Check Database Compatibility

Scenario

You are preparing to migrate the **salesapp1** database to Azure for Adventure Works Cycles. Before you carry out the migration, you want to confirm that the database is compatible with Azure.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Run the Export Data-Tier Application Wizard to Check Database Compatibility
3. Resolve the Failure by Removing a Stored Procedure
4. Rerun the Export Data-Tier Application Wizard to Check Database Compatibility

► Task 1: Prepare the Lab Environment

1. Ensure that the MSL-TMG1, 20765B-MIA-DC, and 20765B-MIA-SQL VMs are both running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab07\Starter** folder as Administrator.

► Task 2: Run the Export Data-Tier Application Wizard to Check Database Compatibility

1. Using SQL Server Management Studio, run a verification test on the **salesapp1** database to determine whether it is suitable for migration to Azure SQL Database. Don't attempt to export any data at this stage.
2. What is the name of the stored procedure that caused the verification test to fail?

► Task 3: Resolve the Failure by Removing a Stored Procedure

- Drop the **dbo.up_CrossDatabaseQuery** stored procedure from the **salesapp1** database.

► Task 4: Rerun the Export Data-Tier Application Wizard to Check Database Compatibility

1. Use SQL Server Management Studio to run a verification test on the **salesapp1** database to determine whether it is suitable for migration to Azure SQL Database. Do not attempt to export any data at this stage.
2. Review the results of the test when it has completed.

Results: After this exercise, you should have run the tools to check database compatibility with Azure from SQL Server Management Studio.

Exercise 2: Migrate a Database to Azure

Scenario

After a successful compatibility test in the previous exercise, you will now migrate the **salesapp1** database to Azure SQL Database.

The main tasks for this exercise are as follows:

1. Create an Instance of Azure SQL Server
2. Configure the Azure Firewall
3. Migrate the salesapp1 Database to Azure
4. Connect to the Migrated Database

► Task 1: Create an Instance of Azure SQL Server

1. Open the Azure portal at <https://portal.azure.com/> by using your Azure pass or Microsoft account.
2. Create a new empty database in Azure SQL Database. Add the database to a new logical server by using a name of your choosing.
3. The new logical server should use the admin login name **salesappadmin** with the password **Pa\$\$w0rd1**.

► Task 2: Configure the Azure Firewall

- Amend the firewall configuration for the logical server that was created in the previous task so that you can allow connections from your client IP address.

► Task 3: Migrate the salesapp1 Database to Azure

1. Use SQL Server Management Studio to connect to the MIA-SQL database, and then use the **Deploy Database to Microsoft Azure SQL Database** wizard to deploy the **salesapp1** database to Azure.
2. Deploy **salesapp1** to the server that you created in the first task of this exercise:
 - To connect to the server, use the fully qualified name of the server that you created in the first task in this exercise (ending .database.windows.net).
3. Review the outcome of the migration wizard.

► Task 4: Connect to the Migrated Database

1. Verify that the **salesapp1** database has been created in the Azure portal.
2. Use SQL Server Management Studio to connect to the Azure server that is now hosting the **salesapp1** database. Execute a sample query to verify that the data has been transferred successfully, such as the following example.

```
SELECT TOP(10) * FROM Sales.Customers;
```

Results: After this exercise, you will have created an empty database in Azure SQL Database, migrated an on-premises database to Azure SQL Database, and be able to connect to, and query, the migrated database.

Module Review and Takeaways

In this module, you have learned about the options for using SQL Server on the Azure cloud service, seen the reasons that you may select them, and gained some experience of the process of creating new databases and migrating existing databases to Azure.

Real-world Issues and Scenarios

You may have to consider compliance with your local privacy and data protection legislation before you move sensitive data or personal identity data into cloud services such as Azure.

Review Question(s)

Question: Are Azure database services suitable for your organization?

Is Azure SQL Database or SQL Server on Azure virtual machines more suitable for you?

Module 8

Migrating Databases to Azure SQL Database

Contents:

Module Overview	8-1
Lesson 1: Database Migration Testing Tools	8-2
Lesson 2: Database Migration Compatibility Issues	8-8
Lesson 3: Migrating an On-Premises Database to an Azure SQL Database	8-14
Lab: Migrating SQL Server with Azure	8-21
Module Review and Takeaways	8-23

Module Overview

SQL Server® and Azure SQL Database are closely related products and continue to get closer in terms of features and supported Transact-SQL keywords. However, because they are designed to work in different environments, they are not identical. Therefore, when you migrate a database from a SQL Server to Azure SQL Database, you might find some incompatibilities. In this module, you will learn how to detect and resolve such issues and how to migrate a compatible database into the cloud.

Objectives

At the end of this module, you will be able to:

- Choose a tool to use to test a database for compatibility with Azure SQL Database.
- Describe reasons why a database hosted in SQL Server might not be compatible with Azure SQL Database.
- Migrate an on-premises database to Azure SQL Database.

Lesson 1

Database Migration Testing Tools

Azure SQL Database does not support all the features and Transact-SQL statements that are supported by SQL Server 2016. This is because the two products are designed for different environments. The differences can cause problems when you migrate a database into the cloud from an on-premises server. The first stage of any migration project is, therefore, to investigate the features of your database that are incompatible with Azure SQL Database. In this lesson, you will see four different tools that you can use for this investigation.

Lesson Objectives

At the end of this lesson, you will be able to:

- List the main stages in a project to migrate a database from an on-premises SQL Server to Azure SQL Database.
- Use SQL Server Data Tools for Visual Studio® to investigate a database's compatibility with Azure SQL Database.
- Describe how to list incompatibility issues by using SqlPackage.
- Use the Export Data-tier Application Wizard in SQL Server Management Studio to investigate a database's compatibility with Azure SQL Database.
- Use the SQL Azure Migration Wizard to investigate a database's compatibility with Azure SQL Database.
- Use SqlPackage to investigate a database's compatibility with Azure SQL Database.

Overview of Database Migration

Azure SQL Database is a cloud platform as a service (PaaS) solution that hosts databases in a multitenant data center. It replicates most of features you know in Microsoft SQL Server 2016, although there are some unsupported features and differences in Transact-SQL support. Often, you can migrate a database from SQL Server into Azure SQL Database with few changes. However, it is important to identify those changes before you execute the migration, so that users are not affected by fire-fighting operations in a production environment.

Databases hosted on SQL Server 2005 or any later versions may be migrated to Azure SQL Database. However, you are likely to find and have to fix more issues when the database is hosted on earlier versions of SQL Server.

To perform database migrations smoothly, the following project stages are recommended.

1. Testing for Compatibility
 - SQL Server Data Tools for Visual Studio
 - SqlPackage
 - The Export Data-tier Application Wizard
 - SQL Azure Migration Wizard
2. Fixing Detected Issues
 - Using SQL Server Management Studio
 - Using SQL Server Data Tools for Visual Studio
 - Using SQL Azure Migration Wizard
3. Performing the Migration
 - Using SQL Azure Migration Wizard
 - Using BACPAC Files

Testing for Compatibility

Your database might have a large and complex schema that has evolved over years and supports complex business requirements. Features that are incompatible with Azure SQL Database might not be immediately obvious even to a Database Administrator (DBA) who is intimately familiar with your database. Microsoft has expedited this issue by providing several testing tools that systematically examine databases and identify problems. These include:

- SQL Server Database Tools (SSDT) for Visual Studio.
- SqlPackage.
- The Export Data-tier application Wizard.
- SQL Azure Migration Wizard (SAMW).

You will learn about each of these tools in detail later in this lesson.

Fixing Detected Issues

You can choose to fix any detected issues manually by using Transact-SQL commands in SQL Server Management Studio (SSMS). This approach gives you maximum control over the process and presents an opportunity to optimize the database design for your current business requirements. However, it might be complex, require a great deal of expertise, and take time to implement.

If you have used SSDT to identify issues, you can also use that tool to resolve those issues. In this method, you still fix issues manually by altering the Transact-SQL Data Definition Language (DDL) statements that create the database in Azure SQL Database. However, the tool provides more help for each issue.

If you have used SAMW to analyze compatibility, you can continue with it to generate a script for creating the new database in Azure SQL Database. In this case, SAMW automatically creates a script in which compatibility issues have been fixed—but you have no control over each fix.

Performing the Migration

The final stage of the project is to perform the migration that creates the database in Azure SQL Database, configures the schema and, optionally, imports data into tables.

In this step, you can continue with SAMW if you want the greatest level of automation and have a simple database with good bandwidth to the Azure datacenter. Alternatively, you can create a BACPAC file to import into Azure.

You will learn more about this stage of the project in Lesson 3.

SQL Server Data Tools for Visual Studio

SQL Server Data Tools (SSDT) is an add-on to Visual Studio that enables developers to define and populate databases. You can also use it to migrate database schema and content from an on-premises SQL Server to Azure SQL Database.

- Testing a database for compatibility in SSDT:
 1. Ensure you have the latest version of SSDT.
 2. In Visual Studio, add the on-premises server to SSDT.
 3. Create a new project for the database and import it into SSDT.
 4. Set the project target to **Microsoft Azure SQL Database V12**.
 5. Build the project. Compatibility issues are listed in the **Error List**.

To use SSDT to test a database for compatibility with Azure SQL Database, take the following steps:

1. Check that you are using the latest version of SSDT. If not, obtain the latest version and install it. By using the latest version of SSDT, you ensure that the tool checks for all known compatibility issues.
2. Open Visual Studio and then open the **SQL Server Object Explorer**.
3. Click the **Add SQL Server** button and connect to the on-premises database server that hosts the database you want to test.
4. Right-click the database, and then click **Create New Project**.
5. In the Create New Project dialog, under Import Settings, select Import application-scoped objects only.
6. Clear **Import referenced logins**, **Import permissions**, and **Import database settings**, and then click **Start**. SSDT imports the database and creates a script file for each object in the database.
7. When the import is complete, click **Finish**.
8. In the Solution Explorer, right-click the database project, and then click **Properties**.
9. On the Project Settings page, in the Target Platform drop-down list, select Microsoft Azure SQL Database V12.
10. In the Solution Explorer, right-click the project, and then click **Build**. If SSDT detects any incompatibilities, they are listed in the **Error List** window with a description.

To download the latest version of SSDT and to find more information, see:



Download SQL Server Data Tools (SSDT)

<https://aka.ms/uydxwk>

SqlPackage

SqlPackage is a command-line utility that you can use for exporting and importing operations in both on-premises SQL Server databases and in cloud databases. SqlPackage supports the following operations:

- **Extract.** Creates a database snapshot DACPAC file from a SQL Server database or from Azure SQL Database.
- **Publish.** Updates the schema in a live database to match the schema in a DACPAC file. If the database does not exist on the destination server, the publish operation creates it.
- **Export.** Exports both schema and data from a SQL Server database or from Azure SQL Database into a BACPAC file.
- **Import.** Imports the schema and data from a BACPAC into a new database.
- **DeployReport.** Creates an XML report that describes the changes that would be made by a publish operation.

- SqlPackage operations
- Obtaining the latest version of SqlPackage
- Using SqlPackage to generate a list of compatibility issues:

```
sqlpackage.exe /Action:Export
/ssn:localhost\SQLEXPRESS
/sdn:AdventureWorks
/tf:ExportedDatabase.bacpac
/p:TableData=dbo.errorlog
> ExportReport.txt 2>&1
```

- **DriftReport.** Creates an XML report of the changes that have been made to a registered database.
- **Script.** Creates a Transact-SQL script that you can use to update the schema of a target database to match the schema of a source database.

Use the **/Action:** or **/a** parameter to specify which action to execute. For more information about the SqlPackage utility, see:



SqlPackage.exe

<https://aka.ms/tacja7>

Before you use SqlPackage to check compatibility with Azure SQL Database, you should ensure that you have the latest version. You can download the latest version of SqlPackage as part of the Microsoft SQL Server Data-Tier Application Framework:



Microsoft SQL Server Data-Tier Application Framework

<https://aka.ms/yn9lpz>

To check for Azure SQL Database compatibility, export a single table from the source database and create a report file. You then check the report file, which contains information about any errors, including incompatibilities.

The following SqlPackage command exports a table from the AdventureWorks database on the local SQL Server. It creates a report file called ExportReport.txt. The "2>&1" string ensures that the report file contains both standard output and standard errors.

A SqlPackage Example

```
sqlpackage.exe /Action:Export /ssn:localhost\SQLEXPRESS /sdn:AdventureWorks
/tf:ExportedDatabase.bacpac /p:TableData=dbo.errorlog > ExportReport.txt 2>&1
```

The Export Data-tier Application Wizard

The Export Data-tier Application Wizard is a tool within SQL Server Management Studio (SSMS). Like SqlPackage, you can use it to export databases into BACPAC files that contain both the schema and content of the source database. However, unlike SqlPackage, the wizard guides you through the process with a series of pages. By selecting the correct options, you obtain a list of compatibility errors without exporting any data.

Before you use the wizard to check for compatibility, you should ensure you have the latest version of SSMS. The latest version is available from:



Download SQL Server Management Studio (SSMS)

<https://aka.ms/phaq68>

In SQL Server Management Studio:

1. Start the Export Data-tier Application Wizard for the database you want to migrate.
2. Configure a destination for the BACPAC file.
3. Configure the wizard to export no data.
4. When you finish the wizard, any compatibility issues are displayed in the Results list.

To use the Export Data-tier Application Wizard to list Azure SQL Database compatibility issues:

1. Open SSMS and connect to the SQL Server instance that hosts the database you want to check.
2. In Object Explorer, expand **Databases**.
3. Right-click the database you want to test, click **Tasks**, and then click **Export Data-tier Application**.
4. In the export wizard, on the **Introduction** page, click **Next**.
5. On the **Export Settings** page, on the **Settings** tab, configure the export to save the BACPAC file in any location on the local disk or in Azure.
6. Click the **Advanced** tab, clear the **Select All** check box to avoid exporting any data, and then click **Next**.
7. Click **Finish**. If there are any compatibility errors, they are listed on the results page. You can obtain more information by clicking the link in the **Results** column. If there are no errors, the wizard generates a BACPAC file and you can continue with the migration.

The Export Data-tier Application Wizard detects and displays compatibility issues but cannot fix them. You must use a different tool to resolve the issues before you migrate the database.

SQL Azure Migration Wizard

The SAMW tool is available from CodePlex and you can use it to generate a Transact-SQL script from a database that is incompatible with Azure SQL Database. In many cases, the wizard can transform the generated script to make it compatible with Azure SQL Database and connect to Azure to create the new database.



Note: The wizard can fix many—but not all—types of incompatibility issues by using built-in transformations. Issues that the wizard cannot fix are reported as errors and as comments in the generated script. Use SSDT or another tool to fix these issues before you import the database into Azure SQL Database.

1. Install the SAMW.
2. Analyze the on-premises database.
3. Modify the exported script to remove compatibility issues.
4. Log on to Windows Azure.
5. Create a new database in Azure SQL Database.
6. Execute the modified Transact-SQL script.
7. Validate that the new database design is correct.

You can download the SAMW from CodePlex. You can also find full documentation on the CodePlex site:



SQL Database Migration Wizard

<https://aka.ms/okys2p>

The SAMW developers recommend the following seven-step process to migrate your database to Azure:

1. Install the SAMW then choose the correct version of the SAMW to match the version of SQL that hosts your database.
2. Analyze the on-premises database. You can configure the SAMW to analyze the database only, or to export the schema and data to a Transact-SQL script.
3. Modify the exported script to remove compatibility issues.

4. Log on to Windows® Azure. If you do not already have an Azure account, you must create one.
5. Create a new database in Azure SQL Database. SAMW can perform this action as part of its process.
6. Execute the Transact-SQL script modified in step 3.
7. Validate that the new database design is correct.

Demonstration: Test Compatibility of a SQL Server Database with Azure SQL Database

In this demonstration, you will see how to run an Azure SQL Database compatibility test for an on-premises database using SqlPackage.

Demonstration Steps

1. Run **Setup.cmd** in the **D:\Demofiles\Mod08** folder as Administrator.
2. In the **User Account Control** dialog box, click **Yes**.
3. Wait for the script to complete and then press Enter.
4. Start a command prompt. Type the following command, and then press Enter:

```
cd C:\Program Files (x86)\Microsoft SQL Server\130\DAC\bin
```

5. Type the following command, and then press Enter:

```
sqlpackage.exe /Action:Export /ssn:MIA-SQL /sdn:TestPSDB  
/tf:D:\Demofiles\Mod08\TSQL.compatibility.bacpac /p:TableData=Stats.Tests >  
D:\Demofiles\Mod08\ExportReport.txt 2>&1
```

6. Type the following command, and then press Enter:

```
Notepad D:\Demofiles\Mod08\ExportReport.txt
```

7. Examine the contents of the text file, and then close Notepad.
8. Close the command prompt.

Question: You have installed the latest version of SSDT for Visual Studio. You have imported a database as a new project from an on-premises SQL Server, which you want to migrate to Azure SQL Database. When you build the project, no compatibility issues are displayed in the error list. What should you do next?

Lesson 2

Database Migration Compatibility Issues

The tools described in the previous lesson can identify issues that prevent you from migrating a database to Azure SQL Database from an on-premises SQL Server. However, it is also helpful for you to be aware of the issues that might arise and how you can fix them. In this lesson, you will see a range of features and Transact-SQL operations that are not supported in Azure SQL Database, so that you can ensure migrations execute as smoothly as possible.

Lesson Objectives

At the end of this lesson, you will be able to:

- Describe how the design and operating environment differs in SQL Server on-premises and Azure SQL Database in the cloud.
- Identify the features of SQL Server that are not supported in Azure SQL Database and select alternatives.
- Describe Transact-SQL statements that are not supported in Azure SQL Database.
- Describe the features of Microsoft SQL Server 2016, 2014, and 2012 that are discontinued.

Comparing SQL Server and Azure SQL Database

Azure SQL Database is designed to be similar to SQL Server and to support most of the same features and language. For this reason, database administrators often find that a database can be migrated to Azure SQL Database without modification. However, because the two systems run in different environments, some features you use in a SQL Server database may not be supported in Azure. To perform a migration as smoothly as possible, it's important to understand those differences.

- SQL Server 2016
 - Executes on-premises
 - You must maintain operating systems, hardware, and system databases
- Azure SQL Database
 - Executes in the cloud
 - Platform as a service
 - You configure and maintain only the user databases

Design Differences

When you run a database on-premises in SQL Server, you are responsible for the entire infrastructure, including the operating system, hardware, and server configuration. Within SQL Server, you must operate several system databases, such as the master database and tempdb. As a PaaS offering, Azure SQL Database is designed to free administrators from concerns about operating systems, hardware, and system databases, so that they can focus on user databases that directly support business requirements.

For this reason, many server-level features, and the Transact-SQL statements that configure them, are unavailable in Azure SQL Database.

Always On and Geo-Replication

In SQL Server 2016, you can increase the availability and resilience of your system by configuring Always On availability groups. These are groups of servers that host replicas of databases. Users usually connect to the primary replica of the database, although they can connect to a secondary replica for read-only access. In the event of a failure on the primary replica, users automatically fail over to a secondary replica, which becomes the new primary replica. In this way, users can continue to work, unaffected by whatever

problem caused the original failure. Always On is an excellent feature for business-critical databases but requires significant effort in terms of planning, configuration, and maintenance. Several Transact-SQL statements, such as CREATE AVAILABILITY GROUP, exist solely to configure Always On.

Azure SQL Database does not support Always On because the server hardware and infrastructure is maintained by Microsoft and not the tenant's database administrators. Instead, you can choose to use active geo-replication for any database. In this case, the database is automatically replicated to a data center in another physical location for the purposes of resilience and availability. However, this feature is a simple on/off option. No complex configuration procedure or Transact-SQL script is required to configure it.

Features Not Supported in Azure SQL Database

The following table shows many of the features of SQL Server 2016 that, at the time of writing, are not supported in Azure SQL Database:

Always On	Attaching Databases	Change Data Capture
BACKUP and RESTORE	CLR	Database Mail
Database Mirroring	Database Snapshots	Extended Stored Procedures
FILESTREAM	Log Shipping	Minimal Logging
Modifying System Data	Policy-Based Management	Semantic Search
Service Broker	SQL Agent	SSAS

Feature	Description	Comments
Always On	Provides fault-tolerance through replication.	Use active geo-replication instead.
Attaching and detaching databases	You can use this feature to copy, move, or upgrade a SQL Server database.	Use other methods, such as BACPAC files, to move databases.
Change Data Capture	You can use this feature to capture insert, update, and delete operations.	No current equivalent.
BACKUP and RESTORE statements	Backs up and restore a database in SQL Server.	Azure SQL Database automatically performs backups to geo-redundant storage.
Common Language Runtime (CLR)	Use classes in the .NET Framework to create database objects.	Use Transact-SQL instead.
Database Mail	Send emails by using system stored procedures.	No current equivalent.
Database Mirroring	Deprecated feature for increasing availability.	Use active geo-replication.

Feature	Description	Comments
Database Snapshots	Use to create a read-only static view of a database.	No current equivalent.
Extended Stored Procedures	Deprecated feature for created stored procedures in C and C++.	Use Transact-SQL instead.
FILESTREAM	Store Binary Large Objects (BLOBs) on the file system instead of the database.	Store BLOBs within the Azure SQL Database.
Log Shipping	Synchronize a primary database with secondaries by automatically sending transaction logs.	Use active geo-replication.
Minimal Logging in Bulk Imports	Ensures that transaction logs are not flooding during large data imports.	Transaction logs are maintained automatically.
Modifying System Data	Update system objects by using the SQL-SMO API or system stored procedures.	Not permitted because the system configuration is maintained by Microsoft.
Policy-Based Management	Create policies to govern database configuration.	No current equivalent.
Semantic Search	Search documents in a database to match the meaning of a phrase.	No current equivalent.
Service Broker	Native support for messaging and queuing.	Use Azure Service Bus queues and messaging instead.
SQL Server Agent	Execute scheduled administrative tasks.	Use elastic jobs instead.
SQL Server Analysis Services	Create multidimensional or tabular data models for pivot-table analysis.	Use Azure Analysis services instead.
Windows Authentication	Use an account from a Windows computer or on-premises Active Directory to authenticate with SQL Server.	Use Azure Active Directory instead.

Azure SQL Database continues to evolve. For an up-to-date and complete comparison of SQL Server and Azure SQL Database features, see:



Azure SQL Database Features

<https://aka.ms/w4accz>

Transact-SQL Differences in Azure SQL Database

Transact-SQL is an extensive language that has evolved over many generations of SQL Server. The vast majority of statements and commands are supported in Azure SQL Database. Examples of unsupported statements and techniques include:

- The collation of system objects.
- Cross-database queries using three- or four-part names. You can execute read-only queries against multiple databases by using elastic database queries.
- Cross database ownership chaining and the TRUSTWORTHY setting.
- DATABASEPROPERTY. Use DATABASEPROPERTYEX instead.
- EXECUTE AS LOGIN. Use EXECUTE AS USER instead.
- Events, event notifications and query notifications.
- Any Transact-SQL statements that relate to database files. These are automatically managed in Azure SQL Database.
- Any statements that relate to global temporary tables.
- HAS_DBACCESS.
- KILL STATS JOB.
- OPENQUERY, OPENROWSET, OPENDATASOURCE, and BULK INSERT.
- Many server-level items. For example, you cannot use GRANT, REVOKE, or DENY permissions on server-level objects.
- SET REMOTE_PROC_TRANSACTIONS.
- SHUTDOWN.
- sp_helpuser stored procedure.
- Trace flags.
- Transact-SQL debugging.
- Server-scoped triggers and logon triggers.
- USE statements. To change database, you must disconnect, and then reconnect to the new database.

Collation of System Objects	Cross-Database Ownership Chains	DATABASEPROPERTY
EXECUTE AS LOGIN	Events and Notifications	Database File Management
HAS_DBACCESS	KILL STATS JOB	OPENQUERY, OPENROWSET, OPENDATASOURCE
FILESTREAM	Log Shipping	Minimal Logging
SET REMOTE_PROC_TRANSACTIONS	SHUTDOWN	sp_helpuser
Trace Flags	Transact-SQL Debugging	USE

These differences are likely to change as Azure SQL Database evolves. For the latest information, see:



Azure SQL Database Transact-SQL differences

<https://aka.ms/yguqzt>



Note: New Azure SQL databases have the READ_COMMITTED_SNAPSHOT option set to ON. This setting can significantly alter the function of your database code if you use the default READ COMMITTED transaction isolation level and rely on locking to manage concurrent activity. For more information about the effects of the READ_COMMITTED_SNAPSHOT setting, see the topic SET TRANSACTION ISOLATION LEVEL (Transact-SQL) in the SQL Server 2016 Technical Documentation:



SET TRANSACTION ISOLATION LEVEL (Transact-SQL)

<http://aka.ms/faim9a>

Discontinued Database Engine Functionality

In each new version of SQL Server, some features are new and some are discontinued. Features that are discontinued in SQL Server are not present in Azure SQL Database, so it is helpful to check these lists before you migrate a database.

Features Discontinued in SQL Server 2016

In SQL Server 2016, the following features were discontinued:

- 32-bit installations. SQL Server only supports 64-bit operating systems.
- The ActiveX Subsystem. You can replace any custom ActiveX code with command-line or PowerShell™ scripts.

- Features discontinued in SQL Server 2016
 - 32-bit installations
 - ActiveX subsystem
- Features discontinued in SQL 2014 and 2012
 - Compatibility levels 90 and 80
 - VIA connection protocol
 - WITH APPEND clause on triggers
 - sp_dboption stored procedure
 - SQL Mail
 - COMPUTE and COMPUTE BY statements
 - *= and =* in Transact-SQL expressions

Features Discontinued in SQL Server 2014 and 2012

In SQL Server 2014, compatibility Level 90 was the only discontinued feature. Use the ALTER DATABASE command to set the compatibility level to 100 or higher.

In SQL Server 2012, discontinued features included:

- Compatibility Level 80. Use the ALTER DATABASE command to set the compatibility level to 100 or higher.
- Support for the VIA connection protocol. Use TCP instead.
- The WITH APPEND clause on triggers. You must recreate the trigger from scratch instead of appending code.
- The sp_dboption stored procedure. Use ALTER DATABASE instead.
- SQL Mail. For later versions of SQL Server, use database mail. For Azure SQL Database, there is no support for email generation within the database system.
- Transact-SQL COMPUTE and COMPUTE BY statements. Use ROLLUP instead.
- Use of *= and =* in Transact-SQL expressions. Use joins instead.

Question: You are migrating a database from SQL Server 2012 to Azure SQL Database. The database is hosted on two database servers and is synchronized by using database mirroring. What feature should you use to replace database mirroring in Azure SQL Database?

Lesson 3

Migrating an On-Premises Database to an Azure SQL Database

When you have identified and fixed any features of your database that are not compatible with Azure SQL Database, you can proceed to migrate the database. The method you choose for this operation will depend on your requirements. For example, if you cannot support an interruption in service to users, you must use transactional replication. In this lesson, you will see several different tools and methods for migrating database schema and contents.

Lesson Objectives

At the end of this lesson, you will be able to:

- Choose an appropriate method to migrate a database from SQL Server to Azure SQL Database.
- Describe the contents and use of DACPAC and BACPAC files.
- Use the bcp.exe tool to migrate data to Azure SQL Database.
- Use transactional replication to migrate a database to Azure SQL Database.
- Use SqlPackage to migrate a database to Azure SQL Database.

Migration Options

When you have identified and resolved any incompatibilities with Azure SQL Database, you can proceed to execute the migration. There are many methods and tools you can use for this process.

First, you must consider how much downtime you can accept for the database during the migration process.

- Migration methods that include downtime
 - SqlPackage
 - SSMS Wizards
 - Data Tools for Visual Studio
 - BACPAC files with bcp.exe
 - SQL Azure Migration Wizard
 - SQL Server Transactional Replication

Migration Methods That Include Downtime

It is simpler to migrate a database if you can accept a period of time when the database will be inaccessible to end users. For databases that are nonbusiness critical or support less frequent activities, this might be no problem. For example, a small HR department might be happy to avoid access to its database for a few hours during the migration to Azure.

Larger and business-critical databases can also be migrated this way, providing there is an off-peak period for demand. For example, a company that operates only in one time zone might not require a database to be accessible overnight or at the weekend. Note, however, that e-commerce and general global activities have made such scenarios less common.

If you can support downtime during the migration, choose from the following migration methods:

- **SqlPackage.** In lesson 1, you saw how to use this command-line tool to check database compatibility. By using different options, you can publish a database to Azure SQL Database instead.
- **SSMS.** You can use the Export Data-tier Application Wizard and the Import Data-tier Application Wizard in SSMS to create a BACPAC file and import it into Azure SQL Database.

- **Data Tools for Visual Studio.** You can choose Azure SQL Database as a location to publish a database project from Visual Studio.
- **BACPAC files with bcp.exe.** The bcp.exe command-line tools can import bulk database into an Azure SQL Database.
- **The SQL Azure Migration Wizard.** In lesson 1, you saw how to use this tool from CodePlex to check database compatibility. The later stages of the wizard create and execute a Transact-SQL script against Azure SQL Database to complete the migration.

SQL Server Transactional Replication

If you have a large or business-critical database, downtime might be unacceptable during the migration process. In such cases, you must use SQL Server transaction replication as a migration method.

In this method, you configure an Azure SQL Database as a subscriber to the database you wish to migrate in SQL Server on-premises. The transaction replication distributor automatically synchronizes any changes to the schema and content of the database to Azure. You can then reconfigure clients to use the Azure database and disable transaction replication when all transactions have reached Azure.

For the latest information about Azure SQL Database migration, see:



SQL Server database migration to SQL Database in the cloud

<https://aka.ms/mypsao>

BACPAC and DACPAC Files

In SQL Server, a Data-tier Application (DAC) is a logical entity that includes all of the management objects required for a database. DACs include objects from the database itself, such as tables, views, and stored procedures, in addition to objects from the SQL Server instance, such as logins. After you have registered a database as a DAC, you can create a portable file, called a DACPAC, which you can then use to recreate the database design on another database instance.

DACPAC files do not include the contents of a database. If you want to encapsulate the data along with the management objects, you can create a BACPAC file instead.

File Structure

A DACPAC file is a compressed file with a .dacpac extension. It contains multiple XML sections that describe the original server instance, the management objects, and other characteristics. You can use the DacUnpack.exe utility to examine the contents of a DACPAC file.

A BACPAC file is also a compressed file but has a .bacpac extension. The XML schema is identical to that for DACPAC files. The difference is that table data from the original database is included in JavaScript Object Notation (JSON) format.

DACPACs are used to deploy and upgrade the design of a database. For example, if you have a database in a development environment that contains test data, you can use a DACPAC to deploy to a production environment without deploying the test data.

- File Structure
- Creating DACPACs and BACPACs

```
sqlpackage / Action:Export
/ssn:LOCALHOST\SQLEXPRESS /sdn:AdventureWorks
/tf:C:\data\AdventureWorks.bacpac
```

- Deploying DACPACs and BACPACs to Azure SQL Database

```
sqlpackage / Action:Import
/tcs:<Your Connection String>
/sf:C:\data\AdventureWorks.bacpac
/p:DatabaseEdition=Premium
/p:DatabaseServiceObject=P11
```

In most cases, when you migrate a database from SQL Server to Azure SQL Database, you are migrating from one production environment to another—so you want to migrate the data, in addition to the database design. For these operations, use a BACPAC.

Creating DACPACs and BACPACs

Both the SqlPackage tool and the SSMS data-tier application wizards use DACPAC and BACKPAC files to migrate data and database schemas.

To create a BACPAC from a SQL Server database called AdventureWorks on the local computer's SQLExpress instance, use the following command:

Using SqlPackage to Create a BACPAC File

```
sqlpackage /Action:Export /ssn:LOCALHOST\SQLEXPRESS /sdn:AdventureWorks  
/tf:C:\data\AdventureWorks.bacpac
```

To create a DACPAC file instead, use the /Action:Extract operation.

To create a BACPAC file by using SSMS, use the Export Data-tier Application Wizard:

1. Open SSMS and connect to the SQL Server instance that hosts the database you want to migrate.
2. In Object Explorer, expand **Databases**.
3. Right-click the database you want to migrate, click **Tasks**, and then click **Export Data-tier Application**.
4. In the export wizard, on the **Introduction** page, click **Next**.
5. On the **Export Settings** page, on the **Settings** tab, configure the export to save the BACPAC file in any location on the local disk, or in Azure, and then click **Next**.
6. Click **Finish**. The wizard generates a BACPAC file.

To create a DACPAC file by using SSMS, use the Extract Data-tier Application Wizard with similar options.

Deploying DACPACs and BACPACs to Azure SQL Database

To use a DACPAC or BACPAC file to create a database in Azure, you must have already created a logical server in Azure and have the connection string for it.

Use the following command to deploy a BACPAC file by using SqlPackage:

Using SqlPackage to Deploy a BACPAC File

```
sqlpackage /Action:Import /tcs:<Your Connection String> /sf:  
C:\data\AdventureWorks.bacpac /p:DatabaseEdition=Premium /p:DatabaseServiceObject=P11
```

Alternatively, if you connect SSMS to the Azure SQL Database logical server, you can use the Import Data-tier Application Wizard to create the database in Azure.

Using BCP with Azure SQL Database

Microsoft SQL Server 2016 includes the `bcpx.exe` command-line tool. Use this tool to copy large amounts of data from one database table to another. The `bcpx.exe` tool is compatible with Azure SQL Database so it can be used to perform a bulk copy operation of data into Azure as part of a migration.

Note that `bcpx.exe` can be used to migrate data only, not the schema of a database table. Use DACPAC files or other methods to migrate the database design.

You can use `bcpx.exe` with no knowledge of Transact-SQL, unless you want to use the **queryout** option to export the results of a query.

Export Files and Format Files

When `bcpx.exe` exports data from a table, it creates a comma-separated text file with no header row. The text file contains no information about the format of the data in the source table. For example, a string in the text file might represent a CHAR, VARCHAR, or TEXT column.

Such bulk copy files do not present problems when you know that the column format and column order are the same in the source and destination tables. This is the case, for example, when you have exported a DACPAC from the source database and imported it, with no modifications, into the destination database. In such cases, you can export data from the source database and import it into the destination database by using two `bcpx.exe` commands.

If you do not know that the source and destination database schemas are identical, you should create and use a format file. A format file stores information about the schema of the source database tables, including the field names, data types, and column orders. During the import operation, `bcpx.exe` can use this format file to match columns in the text file with the destination database columns.

Using bcpx.exe to Migrate Data into Azure

In the following examples, a format file is created followed by a bulk export from an on-premises database table. The bulk export and format files are then used to populate an Azure SQL Database table.

To create a format file in XML format, use the following command:

Creating a Format File for a Bulk Import

```
bcpx.exe AdventureWorks.HumanResources.Employees format nul -f
C:\dataformats\Employees.xml -x -n -T
```

To export the data in the Employees table, use the following command. The `-T` option specifies that you will authenticate against SQL Server using your current Windows account:

Exporting a Database Table

```
bcpx.exe AdventureWorks.HumanResources.Employees out C:\dataexports\Employees.bcp -f
C:\dataformats\Employees.xml -n -T
```

- Export Files and Format Files
- Using `bcpx.exe` to migrate data into Azure


```
bcpx.exe AdventureWorks.HumanResources.Employees
format nul -f C:\dataformats\Employees.xml -x -n -T

bcpx.exe AdventureWorks.HumanResources.Employees
out C:\dataexports\Employees.bcp
-f C:\dataformats\Employees.xml -n -T

bcpx.exe AdventureWorks.HumanResources.Employees
in C:\dataexports\Employees.bcp
-S myserver.database.windows.net
-U username@myserver -P password
-f C:\dataformats\Employees.xml -n
```
- Using `bcpx.exe` with BACPAC Files

To import the data into Azure SQL Database, use the following command where *myserver* is the name of your Azure SQL Database logical server, *username* is the name of your Azure user account and *password* is your password:

Importing a Table into Azure SQL Database

```
bcp.exe AdventureWorks.HumanResources.Employees in C:\dataexports\Employees.bcp -S
myserver.database.windows.net -U username@myserver -P password -f
C:\dataformats\Employees.xml -n
```

Note that the above commands migrate only the Employees table into Azure SQL Database. To migrate an entire database, you can create a batch file of multiple bcp.exe commands.

For full information about the bcp.exe utility, see:



bcp Utility

<https://aka.ms/atro1f>

Using bcp.exe with BACPAC Files

Like DACPAC files, BACPAC files include the schema and management objects required for a database. Unlike DACPAC files, BACPAC files also include the exported data from the source database. This is included in the compressed .bacpac file, in the form of bcp.exe text files. Therefore, you can extract data from a BACPAC file and use bcp.exe to import that data into Azure SQL Database tables by using the following steps:

1. Rename the .bacpac file with a .zip extension.
2. Use the Windows Explorer zip utility to open the .zip file and extract all .bcp files from it.
3. Use bcp.exe commands to import the .bcp files for each table into Azure SQL Database tables.

For more information about using bcp.exe with Azure SQL Database, see:



BCP and SQL Azure

<https://aka.ms/ap5qjo>

Using Transactional Replication to Migrate a Database

Transactional replication is a technology for synchronizing data and database objects between instances of SQL Server. After replication is set up, synchronization continues to maintain consistency. Users of the two synchronized databases see approximately the same data, although some latency might be experienced on new or changed data.

Azure SQL Database is compatible with transactional replication. This means that transactional replication can be used to migrate the schema and contents of an on-premises database into Azure SQL Database. This method is more complex to manage than other migration methods, but it has the advantage that it requires no interruption in service to users. Use this method if you want to migrate a database that must remain online throughout the process.

- Use transactional replication when you must migrate a database with no downtime
- Server roles:
 - Distributors: must be a SQL Server
 - Publishers: hosts source database
 - Subscribers: Azure SQL Database logical server
- Project stages:
 - Set up replication
 - Move users to Azure
 - Wait until all remaining changes are replicated
 - Uninstall transactional replication

This method focuses on transactional replication for migration to Azure. For more information about replication, see:



SQL Server Replication

<https://aka.ms/nufkrz>

Transactional replication is implemented by setting up SQL Server Agent to execute publishing and distribution executables. SQL Servers take three roles in the architecture:

- **Distributors.** A distributor is a SQL Server that creates a distribution database and a log file. Distributors propagate publications from publisher servers to subscriber servers.
- **Publishers.** A publisher is a SQL Server that publishes a database or data objects for replication. A publication consists of database snapshots that will be replicated to subscribed servers.
- **Subscribers.** A subscriber is a SQL Server that subscribes to a publication.

When you use replication for migration to Azure SQL Database, the SQL Server that hosts the unmigrated database is the publisher. The Azure SQL Database logical server is the subscriber. You can choose any SQL Server as a distributor. Often, the publisher and distributor are the same SQL Server.

For replication to Azure SQL Database, only one-way transactional replication is supported. You cannot use peer-to-peer replication or merge replication. This is because Azure SQL Database cannot host SQL Agent or run the replication executables. The distributor cannot be an Azure SQL Database logical server. The Azure SQL Database logical server must be configured as a push subscriber.

Your migration project should take the following approach:

1. **Setup replication.** Configure distribution and publication on SQL Server servers. Configure the Azure SQL Database logical server as a push subscriber to the publication.
2. **Move users to Azure.** Reconfigure connection strings in client applications to connect to the Azure SQL Database.
3. **Wait until all changes are replicated.** When all clients are using Azure, the remaining changes from on-premises databases will gradually replicate to the cloud database.
4. **Uninstall transactional replication.** When all clients have been moved to the cloud database and all transactions have reached Azure, you can uninstall transactional replication. Azure SQL Database now hosts your production database.

For more information about using transactional replication to migrate a database to Azure, see:



SQL Server database migration to SQL Database in the cloud

<https://aka.ms/jx2m7i>

Demonstration: Migrate a SQL Server Database to Azure SQL Database with BACPAC

In this demonstration, you will see how to migrate an on-premises database to Azure SQL Database using SqlPackage.

Demonstration Steps

1. Start a command prompt. Type the following to generate an export BACPAC file for the TestPSDB database:

```
cd C:\Program Files (x86)\Microsoft SQL Server\130\DAC\bin
sqlpackage.exe /Action:Export /ssn:MIA-SQL /sdn:TestPSDB
/tf:D:\Demofiles\Mod08\TSQL.export.bacpac
```

2. Verify that the export BACPAC file exists at **D:\Demofiles\Mod08\TSQL.export.bacpac**.
3. Type the following command to import the database to Azure SQL Database. Substitute <your server name> with the name of the Azure server hosting the target database:

```
sqlpackage.exe /Action:Import /tsn:<your server name> /tdn:TestPSDB /tu:Student
/tp:Pa^$^$w0rd /sf:D:\Demofiles\Mod08\TSQL.export.bacpac
```



Note: This step may take several minutes to complete.

4. Verify that the import has completed successfully by connecting to the Azure SQL Database **TestPSDB** using SSMS.
5. Close SSMS, and then close the command prompt.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? Azure SQL Database includes SQL Server Agent.	

Lab: Migrating SQL Server with Azure

Scenario

As a pilot project for Adventure Works Cycles, you have been tasked with the migration of a SQL Server database, which is used by a sales application, to Microsoft Azure.

Objectives

After completing this lab, you will be able to:

- Use tools to assess the suitability of a database to be moved to Azure.
- Migrate a database to Azure.

Estimated Time: 45 minutes

Virtual machine: **20765B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Check Database Compatibility

Scenario

You are preparing to migrate the **salesapp1** database to Azure for Adventure Works Cycles. Before you carry out the migration, you want to confirm that the database is compatible with Azure.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Run the Export Data-tier Application Wizard to Check Database Compatibility
3. Resolve the Failure by Removing a Stored Procedure
4. Rerun the Export Data-tier Application Wizard to Check Database Compatibility

► Task 1: Prepare the Lab Environment

1. Ensure that the MSL-TMG1, 20765B-MIA-DC, and 20765B-MIA-SQL VMs are both running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab08\Starter** folder as Administrator.

► Task 2: Run the Export Data-tier Application Wizard to Check Database Compatibility

1. Using SQL Server Management Studio, run a verification test on the **salesapp1** database to determine whether it is suitable for migration to Azure SQL Database. Don't attempt to export any data at this stage.
2. What is the name of the stored procedure that caused the verification test to fail?

► Task 3: Resolve the Failure by Removing a Stored Procedure

- Drop the **dbo.up_CrossDatabaseQuery** stored procedure from the **salesapp1** database.

► Task 4: Rerun the Export Data-tier Application Wizard to Check Database Compatibility

1. Use SSMS to run a verification test on the **salesapp1** database to determine whether it's suitable for migration to the Azure SQL Database. Do not attempt to export any data at this stage.
2. Review the results of the test when it has completed.

Results: After this exercise, you should have run the tools to check database compatibility with Azure from SSMS.

Exercise 2: Migrate a Database to Azure

Scenario

After a successful compatibility test in the previous exercise, you will now migrate the **salesapp1** database to Azure SQL Database.

The main tasks for this exercise are as follows:

1. Create an Azure SQL Server
2. Configure the Azure Firewall
3. Migrate the salesapp1 Database to Azure
4. Connect to the Migrated Database

► Task 1: Create an Azure SQL Server

1. Open the Azure portal <https://portal.azure.com/> with your Azure Pass or Microsoft Account.
2. Create a new blank SQL Database, and then add the database to a new server with a name of your choosing.
3. The new server should use the admin login name **salesappadmin** with the password **Pa\$\$w0rd1**.

► Task 2: Configure the Azure Firewall

- Amend the firewall configuration for the server created in the previous task to allow connections from your client IP address.

► Task 3: Migrate the salesapp1 Database to Azure

1. Use SSMS to connect to the MIA-SQL database and use the **Deploy Database to Microsoft Azure SQL Database** wizard to deploy the **salesapp1** database to Azure.
2. Deploy **salesapp1** to the server you created in the first task of this exercise.
 - To connect to the server, use the fully qualified name of the server you created in the first task in this exercise (ending .database.windows.net).
3. Review the outcome of the migration wizard.

► Task 4: Connect to the Migrated Database

1. Verify that the **salesapp1** database has been created in the Azure portal.
2. Use SSMS to connect to the Azure server now hosting the **salesapp1** database. Execute a sample query to verify that the data has been transferred successfully. For example:

```
SELECT TOP(10) * FROM Sales.Customers;
```

Results: After this task, you will have created an Azure SQL Database instance, migrated an on-premises database to Azure SQL Database, and be able to connect to, and query, the migrated database.

Module Review and Takeaways

In this module, you have learned about the options for using SQL Server on the Azure cloud service and seen the reasons why you might select them. You have gained some experience of the process of creating new databases and migrating existing databases to Azure.

Real-world Issues and Scenarios

You might have to consider compliance with your local privacy and data protection legislation before you move sensitive data or personal identity data into cloud services such as Azure.

Review Question(s)

Question: Are Azure database services suitable for your organization?

Is Azure SQL Database or SQL Server on Azure VMs more suitable for you?

Module 9

Deploying SQL Server on a Microsoft Azure Virtual Machine

Contents:

Module Overview	9-1
Lesson 1: Deploying SQL Server on Azure Virtual Machines	9-2
Lesson 2: Migrating a Database to a Microsoft Azure Virtual Machine	9-10
Lab: Deploying SQL Server on an Azure Virtual Machine	9-16
Module Review and Takeaways	9-20

Module Overview

Microsoft® Azure SQL Database, which you saw in the previous module, is a platform as a service (PaaS) product onto which you can deploy a database without taking responsibility for the underlying database software and operating systems. However, you may need to make changes to a database to deploy it to Azure SQL Database because some features of Microsoft SQL Server® are incompatible with it. If you want to migrate a database to the cloud with the minimum number of changes, consider deploying to an instance of SQL Server running on a Microsoft Azure® virtual machine. This environment is much more closely analogous to the on-premises environment than Azure SQL Database. However, because it is an infrastructure as a service (IaaS) product, you must take responsibility for the software and operating systems. In this module, you will see how to set up SQL Server on Azure virtual machines and migrate databases to those virtual machines.

Objectives

At the end of this module, you will be able to:

- Describe how to license, patch, and protect SQL Server when it runs on one or more Azure virtual machines.
- Migrate a database from an on-premises server that runs SQL Server to a virtual machine in Azure.

Lesson 1

Deploying SQL Server on Azure Virtual Machines

Azure virtual machines can provide an environment that is very closely analogous to servers that are running Windows Server® in an on-premises data center. The principal advantage of virtual machines is that they free you from building and maintaining infrastructure such as server racks, uninterruptible power supplies, network components, and so on. You can move SQL Server instances to Azure virtual machines with little or no modification to the databases that they host. In this module, you will learn how to replicate your on-premises SQL Server architecture in Azure by using virtual machines.

Lesson Objectives

At the end of this lesson, you will be able to:

- Understand how to create and connect to SQL Server when it is hosted on a virtual machine in Azure.
- Provision a SQL Server 2016 instance as a virtual machine in Azure.
- License a SQL Server 2016 instance to execute on a virtual machine in Azure.
- Choose a highly available architecture for SQL Server hosted on virtual machines in Azure.
- Describe how to protect a SQL Server virtual machine installation from disasters.
- Use Automated Patching to maintain SQL Server 2016 on a virtual machine in Azure.

Creating a SQL Server Architecture by Using Azure Virtual Machines

Azure provides many versatile IaaS features, in which Microsoft runs the physical infrastructure such as data centers, hardware, and networking components. You can install virtual machines on this infrastructure. A virtual machine is a virtual computer that runs an operating system and server software that you install and configure. In IaaS, you as an Azure subscriber are responsible for installing, securing, updating, and configuring the operating system and software for the virtual machine.

- Running SQL Server on Azure virtual machines
- Connecting on-premises clients to SQL Server on Azure virtual machines
- Replicating on-premises architectures in Azure
- Hybrid Architectures

Running SQL Server on Azure Virtual Machines

A virtual machine that is running Windows Server behaves just like a physical server running Windows Server on your premises. You can install and run SQL Server 2016 in exactly the same way as on an on-premises server. For this reason, it is possible to migrate databases to a virtual machine with little or no modification. Such a migration project requires no redesign of the database or modification of client code. If you want to move to a cloud infrastructure as simply and easily as possible, choose to run SQL Server on a virtual machine because it is likely to be a simpler migration than choosing Azure SQL Database.

The disadvantage of migrating a database to a virtual machine is that you remain responsible for the operating system and SQL Server. You must apply software updates, maintain security, configure resilience, ensure that backups are taken, and so on. When you choose Azure SQL Database, the customer does not perform these functions.

The SQL Server Agent Extension for SQL Server Virtual Machines

Virtual machine extensions are small software packages that you can install on virtual machines to support extra functionality. You can install the SQL Server Agent Extension for SQL Server virtual machines on a virtual machine that runs Windows Server 2012 and SQL Server 2012, 2014, or 2016. It is included in SQL Server images in the Azure Virtual Machine Gallery. This extension supports three features:

- **SQL Server Automated Backup.** You can use this feature to back up databases automatically on the default instance of SQL Server on a virtual machine.
- **SQL Server Automated Patching.** You can use this feature to schedule times when updates can be installed.
- **Azure Key Vault Integration.** You can use this feature to install and configure Azure Key Vault automatically on your SQL Server virtual machine. Azure Key Vault can centrally store and manage encryption keys that are used in SQL Server functions.

If you created a SQL Server virtual machine that does not have the agent installed, you can use the following Azure PowerShell command to install it. Substitute your own values for the *ResourceGroupName*, *VMName*, and *Location* parameters.

Installing the SQL Server Agent Extension for SQL Server virtual machines

```
Set-AzureRmVMSQLServerExtension -ResourceGroupName "myresourcegroup" -VMName "MyVM" -Name "SQLIaaSExtension" -Version "1.2" -Location "Northern Europe"
```

Connecting On-Premises Clients to SQL Server on Azure Virtual Machines

SQL Server clients connect to on-premises servers directly through TCP port 1433 or a custom port. When you migrate to Azure virtual machines, you must consider how to enable clients to connect while maintaining security. There are two possible approaches:

- **Connect to SQL Server through the Internet.** In this approach, port 1433 is opened on the Windows® firewall. Clients are reconfigured with a connection string that includes the fully qualified domain name (FQDN) of the server that runs SQL Server.
- **Connect to SQL Server through a virtual private network (VPN).** In this approach, virtual machines that are hosting SQL Server are placed in a virtual network within Azure. You can then establish a VPN that encrypts all traffic between the virtual network and your on-premises network.

For more information about client connection options, see *Connect to a SQL Server Virtual Machine on Azure (Resource Manager)* in the Azure documentation:



Connect to a SQL Server Virtual Machine on Azure (Resource Manager)

<https://aka.ms/bcbawy>

Replicating On-Premises Architectures in Azure

Virtual machines are closely analogous to servers in a traditional on-premises architecture. You can also model other aspects of a physical architecture in Azure. For example, you can use virtual networks to model Ethernet networks and subnets.

For on-premises architectures, it is common to optimize the resilience and availability of databases by hosting them on multiserver architectures. By using multiple virtual machines in virtual networks, you can implement the following availability technologies in Azure:

- Always On Availability Groups
- Always On Failover Cluster Instances

- Log shipping
- Database mirroring

You will learn more about highly available architectures that use Azure virtual machines later in this lesson.

Hybrid Architectures

A hybrid architecture, in the context of Azure, is one in which some servers are hosted on-premises and some in the cloud. Using SQL Server, you can host a database both on-premises and in the cloud in several different ways. For example:

- **Always On Availability Groups.** Virtual machines in Azure can act as secondary replicas to a primary replica that is on-premises. In this case, you must create a Windows Server Failover Clustering (WSFC) cluster that spans your local network and the Azure virtual network. You must also connect the networks by using a VPN.
- **Database mirroring.** You can configure virtual machines in Azure to mirror a principal database that is hosted on-premises. No VPN is required for this configuration unless you need the database servers to run in the same Active Directory® domain.
- **Log shipping.** Virtual machines in Azure can host databases that are log shipping secondaries to primaries that are hosted on-premises. A VPN connection is required.

For more information and examples of fault-tolerant and hybrid architectures that can be created by using SQL Server on Azure virtual machines, see *High availability and disaster recovery for SQL Server in Azure Virtual Machines* in the Azure documentation:



High availability and disaster recovery for SQL Server in Azure Virtual Machines

<https://aka.ms/eiter5>

Demonstration: Provisioning an Azure Virtual Machine

In this demonstration, you will see how to provision an Azure virtual machine with SQL Server.

Demonstration Steps

1. Ensure that the MSL-TMG1, 20765B-MIA-DC, and 20765B-MIA-SQL VMs are running and log on to 20765B-MIA-SQL as **AdventureWorks\Student** with the password **Pa\$\$w0rd**.
2. Open **Internet Explorer** and go to <https://portal.azure.com/>.
3. Sign in to the Azure portal with your Azure Pass or Microsoft Account credentials.
4. Click **New**, click **Compute**, click **SQL Server 2016 SP1 Enterprise on Windows Server 2016**.
5. In the **SQL Server 2016 SP1 Enterprise on Windows Server 2016** blade, in the **Select a deployment model** list, click **Resource Manager**, and then click **Create**.
6. On the **Basics** blade, in the **Name** box, type a name for your server. This must be unique throughout the whole Azure service, so cannot be specified here. A suggested format is **sql2016vm<your initials><one or more digits>**. For example, **sql2016vmjs123**. Keep a note of the name you have chosen.
7. In the **VM disk type** list, ensure that **SSD** is selected.
8. In the **User name** box, type **demoAdmin**.
9. In the **Password**, type **Pa\$\$w0rd1234**.

10. In the **Confirm password** box, type **Pa\$\$w0rd1234**.
11. Under **Resource group**, click **Create new** and then type **resource1**.
12. In the **Location** list, select a location near you, and then click **OK**.
13. In the **Choose a size** blade, click **View all**, click **DS11_V2 Standard**, and then click **Select**.
14. On the **Settings** blade, click **OK**.
15. On the **SQL Server settings** blade, click **OK**.
16. On the **Summary** blade, click **OK**. Deployment may take some time to complete.
17. When deployment is complete, Azure starts the VM and displays the overview blade. Click **Connect**, and then click **Open**.
18. In the **Remote Desktop Connection** dialog box, click **Connect**.
19. In the **Windows Security** dialog box, in the **User name** box, type **demoAdmin**, in the **Password** box, type **Pa\$\$w0rd1234**, and then click **OK**.
20. In the **Remote Desktop Connection** dialog box, click **Yes**.
21. When the remote desktop session has started, start **SQL Server Management Studio** and connect to the local SQL Server instance. Demonstrate the structure of the installation.
22. If the **Networks** pane appears, click **No**.

Licensing SQL Server on Azure Virtual Machines

It is important to understand licensing requirements for SQL Server before you deploy a database in any architecture. This is both to ensure that you are operating legally and also to understand whether the total cost of ownership will be increased or decreased by a proposed migration. Even if costs will increase, the advantages of a new design may lead you to proceed.

For a cloud solution that is based on SQL Server that is hosted in Azure virtual machines, it can be difficult to adjust to new requirements when you are used to on-premises licensing. There are three possibilities:

- You could use an image from the Azure Virtual Machine Gallery that includes SQL Server. In this case, the per-minute rate that is shown in the gallery for the image and server size that you select includes the SQL Server license fee.
- If you are a Microsoft customer who has Software Assurance, you can install or upload your own image that includes SQL Server by using the license mobility benefits that the Software Assurance scheme includes.
- If you are a Microsoft service provider, you can install or upload your own image that includes a SQL Server Subscriber Access License (SAL) reported through your Services Provider Licensing Agreement (SPLA).

- SQL Server images from the Azure Virtual Machine Gallery include a license to run SQL Server in the per-minute rate
- Microsoft customers can use license mobility to add a SQL Server license to their own images
- Microsoft partners can install or upload an image that includes an SAL



Note: SQL Server is licensed on a per-CPU core basis. This is equivalent to virtual cores for virtual machines in Azure.

Some SQL Server images in the Azure Virtual Machine Gallery are marked as bring your own license (BYOL) images. These do not include the SQL Server license fee in the per-minute rate. You must add your own license by using Software Assurance or an SPLA.

The license to run Windows Server is also included in the per-minute cost of a Windows Server virtual machine. A separate Windows Server client access license (CAL) is not required to connect to a virtual server that is running Windows Server.

For more information about licensing software in Azure virtual machines, see *Virtual Machines Licensing FAQ* on the Azure website:



Virtual Machines Licensing FAQ

<https://aka.ms/syd965>

High Availability in Azure Virtual Machines

Azure virtual machines constitute an IaaS offering. That is to say, Microsoft provides the underlying physical and logical infrastructure that is required to host virtual machines. This includes data centers that have robust power supplies and network hardware, a high-bandwidth Internet connection, and so on. When you create virtual machines that run on this infrastructure, you remain responsible for the operating systems and software that they run, including SQL Server.

Although the provided infrastructure is robust, you should not consider your virtual machines to be infallible. Problems from many sources, including poorly tested custom code and poorly implemented security, can cause virtual machines to fail. Therefore, just as on-premises SQL Server systems should have a fault-tolerant architecture, you should ensure that Azure SQL Server installation can continue if one or more virtual machines stops responding to user requests.

Many of the fault-tolerant technologies that you use to protect SQL Server on-premises are supported in Azure. The following paragraphs describe how to configure these technologies in a cloud scenario.

High-Availability Solutions

Highly available solutions use multiple servers to provide fault-tolerance to mitigate the impact of failures. In the event of a server failure, users can connect to a different server and continue to execute queries. If you want to implement SQL Server entirely on Azure virtual machines, you can consider the following highly available architectures:

- **Always On Availability Groups.** In an Always On Availability Group, synchronous commit operations are used to copy database changes from a primary to a secondary replica that is hosted on different SQL Server instances. In Azure, those instances can be hosted on separate Azure virtual machines with a third virtual machine acting as a file share witness in the same WSFC cluster. WSFC requires an Active Directory domain, so you also need an Azure virtual machine to act as a domain controller.

- Highly available SQL Server architectures in Azure:
 - Always On Availability Groups
 - Always On Failover Cluster Instances
- Availability Groups reduce downtime by placing virtual machines in different:
 - Fault domains
 - Update domains

- **Always On Failover Cluster Instances.** In Always On Failover Cluster Instances on-premises, multiple servers that run SQL Server share disks on which database files are stored. This disk sharing is achieved by using storage area networks (SANs), iSCSI, or other technologies. In Azure, you can use Windows Server 2016 Storage Spaces Direct (S2D) to create a virtual SAN or third-party clustering software.

For more information about configuring highly available SQL Server architectures in Azure, see *High availability and disaster recovery for SQL Server in Azure Virtual Machines* in the Azure documentation:

 **High availability and disaster recovery for SQL Server in Azure Virtual Machines**

<https://aka.ms/eiter5>

Availability Sets

In an Azure data center, physical and virtual resources are organized to permit two types of virtual machine grouping:

- **Fault domains.** The virtual machines in a single fault domain share a common power source and network switch. This makes it more likely that a single hardware failure could affect those virtual machines.
- **Update domains.** Microsoft plans maintenance tasks for the hardware and software that runs virtual machines to preserve the highest level of security and stability. The virtual machines in a single update domain may be taken offline for maintenance at the same time.

It is therefore essential that you ensure that the virtual machines in a highly available SQL Server architecture are spread across different fault domains and different update domains. You can do this by using availability sets. When you place virtual machines into a single availability set, Azure automatically spreads those virtual machines across five different update domains and three different fault domains by default. Resource Manager deployments can increase these numbers for a single availability set.

For more information about availability sets and virtual machines, see *Manage the availability of virtual machines* in the Azure documentation:

 **Manage the availability of virtual machines**

<https://aka.ms/vgha5a>

Disaster Recovery in Azure Virtual Machines

Disaster recovery solutions enable administrators to recover quickly and completely from a catastrophic failure. This includes disasters that destroy all of the servers or virtual machines in a data center. Some highly available solutions should not be considered disaster recovery solutions because there is the potential for some data loss during a disaster and because a site-wide disaster could cause a service outage or result in data loss.

- Disaster recovery solutions:
 - Always On Availability Groups across multiple Azure regions
 - Database mirroring
 - Backup and Restore by using the Azure Blob storage service
 - Using the SQL Server IaaS Agent Extension to automate backups

If you want to implement SQL Server entirely on Azure virtual machines, you can consider the following disaster recovery solutions:

- **Always On Availability Groups.** If you place the secondary replica on a virtual machine in a different Azure region from the primary, you ensure that your solution is recoverable even in the event of a complete site outage. Note that this configuration requires an Azure virtual machine to act as a domain controller in each region. Virtual machines in each region should be placed in a virtual network (VNet) and you must configure VNet-to-VNet connectivity.
- **Database mirroring.** By creating a database mirror on a virtual machine in a different region, you can create a disaster recovery solution that automatically creates and updates a copy of a database in another site.
- **Backup and Restore by using the Azure Blob storage service.** SQL Server 2014 and SQL Server 2016 support Azure Blob storage as a backup destination, whether those servers are running on-premises or in the cloud. By configuring a virtual server that runs SQL Server in one region to back up to an Azure storage account in a different region, you can protect against a complete site outage.

For more information about disaster recovery planning for SQL Server in Azure virtual machines, see *High availability and disaster recovery for SQL Server in Azure Virtual Machines* in the Azure documentation:



High availability and disaster recovery for SQL Server in Azure Virtual Machines

<https://aka.ms/eiter5>

Using the SQL Server IaaS Agent Extension to Automate Backups

If you have chosen to use backups to the Azure Blob storage service as your disaster recovery system, you can use the SQL Server IaaS Agent Extension to automate and configure those backups. This system has the following requirements:

- A virtual machine that is running Windows Server 2012 R2 or Windows Server 2016.
- SQL Server 2016.
- A database that uses the full recovery model.
- A virtual machine that uses the Resource Manager deployment model.



Note: These are the requirements for Automated Backup V2. If you have a virtual machine that is running Windows Server 2012 and/or SQL Server 2014, you can use Automated Backup V1. This latter version is similar, but has fewer configuration options, such as those relating to backup frequency and start times.

You can configure Automated Backup in the Azure portal. For example, to configure a patching schedule for an existing virtual machine:

1. Double-click the virtual machine.
2. On the virtual machine's details blade, click **All Settings**.
3. On the **Settings** blade, click **SQL Server Configuration**.
4. Under **Automated backup**, click **Edit**.
5. Configure the properties and schedule for backups, and then click **OK**.

For more information about Automated Backups for SQL Server in Azure virtual machines, see *Automated Backup v2 for SQL Server 2016 Azure Virtual Machines (Resource Manager)* in the Azure documentation:



Automated Backup v2 for SQL Server 2016 Azure Virtual Machines (Resource Manager)

<https://aka.ms/dc0msi>

Automated Patching for SQL Server in Azure Virtual Machines

Windows Update can automatically install updates for Windows and SQL Server. Some updates require a system restart and others may place a significant load on server resources as they are installed. To ensure that this has the minimum impact on users, you can use Automated Patching to establish a schedule of off-peak hours during which updates can be applied.

Automated Patching requires the SQL Server IaaS Agent Extension.

You can configure Automated Patching in the Azure portal. For example, to configure a patching schedule for an existing virtual machine:

1. Double-click the virtual machine.
2. On the virtual machine's details blade, click **All Settings**.
3. On the **Settings** blade, click **SQL Server Configuration**.
4. Under **Automated patching**, click **Edit**.
5. Create the schedule, and then click **OK**.

To configure a patching schedule for an existing virtual machine by using Windows PowerShell commands, execute the following code, substituting your own values and names:

Configuring a patching schedule in Windows PowerShell

```
$vmname = "MyVM"
$resourcegroupname = "MyResourceGroup"

$Schedule = AzureRM.Compute\New-AzureVMSqlServerAutoPatchingConfig -Enable -DayOfWeek
"Sunday" -MaintenanceWindowStartingHour 3 -MaintenanceWindowDuration 120 -PatchCategory
"Important"

Set-AzureRmVMSqlServerExtension -AutoPatchingSettings $Schedule -VMName $vmname -
ResourceGroupName $resourcegroupname
```

- Automated Patching creates a schedule of times during which updates may be installed
- It requires the SQL Server IaaS Agent Extension
- Configure the schedule in the portal or in Azure PowerShell:

```
$vmname = "MyVM"
$resourcegroupname = "MyResourceGroup"
$Schedule =
    AzureRM.Compute\New-AzureVMSqlServerAutoPatchingConfig
    -Enable -DayOfWeek "Sunday"
    -MaintenanceWindowStartingHour 3
    -MaintenanceWindowDuration 120 -PatchCategory "Important"
Set-AzureRmVMSqlServerExtension -AutoPatchingSettings $Schedule
-VMName $vmname -ResourceGroupName $resourcegroupname
```

Question: You want to implement an instance of SQL Server 2016 on a virtual machine in Azure to host a business-critical database. You want to ensure that you license the server properly, but you do not have Software Assurance. How can you license SQL Server to run on the virtual machine?

Lesson 2

Migrating a Database to a Microsoft Azure Virtual Machine

There are many different methods that you can use to move an existing database, including both its schema and content, from an on-premises database server to an instance of SQL Server that runs on an Azure virtual machine. The method that you choose will depend on your circumstances. For example, if you can sustain a period of downtime, you may choose to use backups to migrate the database. In this lesson, you will learn about the available migration methods.

Lesson Objectives

At the end of this lesson, you will be able to:

- Choose the most suitable method to migrate a database to an Azure virtual machine.
- Use backup and restore operations to migrate a database to an Azure virtual machine.
- Describe how the Deploy Database to a Microsoft Azure VM Wizard migrates a database.
- Migrate a database to an Azure virtual machine.

Choosing a Migration Method

Administrators migrate databases from an on-premises SQL Server instance to a cloud-based Azure virtual machine for several reasons. Often, an organization is moving to the cloud to reduce administration costs and to reduce the need for resilient IT infrastructure at its physical sites. In this case, production databases that include both database design and complete database contents must be migrated and downtime must be reduced or eliminated. In other cases, a database has been developed, tested, and staged on-premises and administrators want to deploy it to a production environment in the cloud.

- Migration methods that include downtime:
 - Backup and restore with compression
 - Backup to an Azure Blob storage account
 - Detach and copy the database files to Azure Blob storage
 - Convert the server to a VHD and deploy it as an Azure virtual machine
 - Ship a hard disk to an Azure physical location
- Migration methods that minimize downtime:
 - Add an Always On replica on an Azure virtual machine
 - Use SQL Server transactional replication

Migration Methods That Require Downtime

In many cases, you can find a period of time in the evening or at weekends when you can sustain an interruption in service to users without serious consequences. If this is possible, you can choose from several different methods to migrate your database. These include:

- Back up the database by using compression, manually copy the backup file to the Azure virtual machine, and then restore the database there. This method is simple and well tested. The use of compression minimizes the time that is required. Use this method when you can support the downtime that is required for backup, upload, and restore.
- Perform a backup to an Azure Blob storage account, and then restore the database from that backup. This method removes the necessity to manually copy the backup file. It is only supported when the source database server runs SQL Server 2012 SP1 or greater.

- Detach the database, copy the database files to an Azure Blob storage account, and then attach them to the SQL Server instance in the virtual machine. Use this method if you plan to store database files in Azure Blob storage permanently instead of on a virtual hard disk in the virtual machine.
- Convert an on-premises server to a virtual hard disk (VHD) image by using Microsoft Virtual Machine Converter. Upload the VHD to Azure storage, and then create a new Azure virtual machine by using the upload VHD. Use this method when you want to bring your own SQL Server license to use in Azure from your Software Assurance scheme.
- Ship a hard drive to an Azure physical location by using the Azure Import/Export Service. The Azure Import/Export Service enables you to send a physical hard disk to an Azure data center. Microsoft can then add the data on this disk to your Azure Blob storage account. Use this method with large databases when network bandwidth makes upload operations too slow.

Migration Methods That Do Not Require Downtime

Many business-critical databases in global organizations must remain online at all times. In such cases, you must ensure that it is possible to make changes to a primary copy of the database at all times. You can do this by using replication to copy the database design and content to the Azure virtual machine while users continue to make changes to the primary copy of the database. After the replication has completed, you can move users to the virtual machine instance of the database and make it the primary copy. There are two migration methods that use this approach:

- **Use the Add Azure Replica Wizard.** If you have an Always On deployment in your on-premises SQL Server system, you can use the Add Azure Replica Wizard to create a new secondary replica on the Azure virtual machine. After replication has completed, you can fail over to make the Azure virtual machine the primary replica.
- **Use SQL Server transactional replication to replicate the database to the Azure virtual machine.** Use this method when you need to minimize downtime, but do not have an Always On deployment in your on-premises SQL Server system.

For more information about migration methods, see *Migrate a SQL Server database to SQL Server in an Azure virtual machine* in the Azure documentation:



Migrate a SQL Server database to SQL Server in an Azure virtual machine

<https://aka.ms/wx5z5z>

Using Backup and Restore for Migration

Database backups are principally used for disaster protection and recovery. By creating a backup file that contains the data and schema from your database, you ensure that the database can be recovered even if a catastrophic event occurs. A full database backup operation creates a file that contains a complete copy of your database up to the moment when the backup was taken, which can be restored on a different instance of SQL Server if necessary. Therefore, you can use such a backup file to migrate a database from one instance of SQL Server to another, including to an instance of SQL Server that is hosted on an Azure virtual machine.

• Backing up to an on-premises location:

1. Back up the database by using compression:

```
USE AdventureWorks;
GO
BACKUP DATABASE AdventureWorks
TO DISK = 'D:\Backups\AdventureWorks.Bak'
WITH COMPRESSION, FORMAT,
MEDIANAME = 'MigrationBackups',
NAME = 'Full Backup of AdventureWorks';
GO
```

2. Copy the database to the Azure virtual machine
3. Restore the database on the Azure virtual machine

• Backing up to an Azure Blob storage account

Backing Up to an On-Premises Location

A simple and well-tested migration method consists of backing up a database to a file in an on-premises location, such as a local folder on the server running SQL Server, or a file share. You must then manually copy the file to an Azure virtual machine that runs SQL Server and restore it there.

Consider the following requirements when you migrate by using backups:

- This method can migrate any database from an original server that runs SQL Server 2005 or later.
- If you choose to compress the backup file during the backup operation, you will be able to upload it to Azure faster.
- It is easiest to perform this kind of migration by taking a full backup. Other types of backup may require you to restore multiple backup files on the destination virtual machine.

You can use a wizard in SQL Server Management Studio to create backups or use a Transact-SQL script such as the following one.

Backing up a SQL Server database by using compression

```
USE AdventureWorks;
GO
BACKUP DATABASE AdventureWorks
TO DISK = 'D:\Backups\AdventureWorks.Bak'
WITH COMPRESSION, FORMAT,
     MEDIANAME = 'MigrationBackups',
     NAME = 'Full Backup of AdventureWorks';
GO
```

After the backup operation is complete, manually copy the database file to the Azure virtual machine. You can use a Remote Desktop connection, Windows Explorer, or the copy command from the command prompt.

To restore the database from the backup file, log on to the SQL Server virtual machine in Azure, and then either use the restore wizard in SQL Server Management Studio or use a Transact-SQL RESTORE DATABASE command.

The following Transact-SQL command restores a database from a full backup file and moves data and transaction logs to appropriate locations for the new virtual machine.

Restoring a database on a SQL Server virtual machine

```
RESTORE DATABASE [AdventureWorks] FROM DISK = 'D:\Migration\AdventureWorks.bak'
WITH REPLACE,
MOVE 'AdventureWorks_Data' TO 'E:\Databases\AdventureWorks_Data.mdf',
MOVE 'AdventureWorks_Log' TO 'E:\Logs\AdventureWorks_log.ldf';
GO
```

Backing Up to a URL

In SQL Server 2016, when you perform a database backup, you can select a URL as a backup destination instead of a local folder or file share on the local network. You can use this tool to back up the database to an Azure Blob storage account and then restore from that account on the destination SQL Server virtual machine in Azure. This approach avoids the need to manually copy the database backup file to Azure. As for the manual copy approach, you can use backup compression to reduce the size of the backup and optimize the time that is required for the upload.

For complete information about backing up to an Azure storage account, see *SQL Server Backup to URL* on MSDN:



SQL Server Backup to URL

<https://aka.ms/p4m0pt>

The Deploy Database to a Microsoft Azure VM Wizard

The Deploy Database to a Microsoft Azure VM Wizard is a tool that forms part of SQL Server Management Studio. The wizard is designed to make it simple to migrate a database from an onsite server to an instance of SQL Server running on an Azure virtual machine. You can use this tool to deploy a database to an existing virtual machine that you have already created or to a new virtual machine that is created from an image by the wizard itself.

- A wizard in SQL Server Management Studio that automatically:
 1. Creates a compressed backup file in an on-premises location
 2. Uploads the backup file to Azure
 3. Restores the backup file to a new database in an Azure virtual machine
- Create a new virtual machine or use an existing virtual machine with SQL Server

The wizard works by creating a compressed backup file of the on-premises database in an on-premises folder or network share. The wizard uploads this compressed backup to Azure, and then restores it to a new database. Both the database schema and all of the database contents are migrated. The wizard uses a compressed file and provides guidance at every step, so this is a fast, simple method that you can use to deploy an existing database to an Azure virtual machine.



Note: The Deploy Database to a Microsoft Azure VM Wizard can only work with virtual machines, cloud services, and storage accounts in the Classic deployment model in Azure. You must use other migration methods to move databases to virtual machines in the Resource Manager deployment model. The latter is the recommended model for new deployments.

You can instruct the wizard to create a new virtual machine and cloud service during the migration. Alternatively, you can choose to migrate the database to an existing virtual machine that has SQL Server already installed. If you choose the second approach, you must:

- Configure the instance of SQL Server on the destination virtual machine to listen on a specific port number.
- Install and configure the Cloud Adapter for SQL Server on the destination virtual machine.
- Configure an open endpoint for the Cloud Adapter for SQL Server on the destination virtual machine by using a private port number 11435.

If the database that you want to deploy uses FILESTREAM to store binary large objects (BLOBs) outside the database, you cannot use this wizard to deploy the database to a new virtual machine and cloud service. Instead, you must create the virtual machine and cloud service manually before deployment.

The destination virtual machine, cloud service, and data disk storage location that the wizard uses must all be in the same Azure region.

For more information about the Deploy Database to a Microsoft Azure VM Wizard, see *Deploy a SQL Server Database to a Microsoft Azure Virtual Machine* on MSDN:



Deploy a SQL Server Database to a Microsoft Azure Virtual Machine

<https://aka.ms/k9fjy3>

Demonstration: Migrating a Database to an Azure Virtual Machine

In this demonstration, you will see how to:

- Back up a database in an on-premises instance of SQL Server.
- Copy the backup file to an Azure virtual machine.
- Restore the backup on the Azure virtual machine to complete the migration.

Demonstration Steps

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are running and log on to 20765B-MIA-SQL as **AdventureWorks\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the **D:\Demofiles\Mod09** folder as Administrator.
3. When the script has completed, press any key to close the window.
4. Start **SQL Server Management Studio**.
5. In the **Connect to Server** dialog box, click **Connect**.
6. In Object Explorer, expand **Databases**, expand **ExampleDB**, and then expand **Tables**.
7. Right-click **HR.Employees**, and then click **Select Top 1000 Rows**. Show the students the results of the query against the local database. You will compare results against the cloud-hosted database after the migration.
8. On the **File** menu, point to **New**, and then click **Query with current connection**.
9. Type the following Transact-SQL script, and then click **Execute**:

```
USE ExampleDB;
GO
BACKUP DATABASE ExampleDB
TO DISK = 'D:\Demofiles\Mod09\ExampleDB.bak'
WITH COMPRESSION, FORMAT,
    MEDIANAME = 'MigrationBackups',
    NAME = 'Full Backup of ExampleDB';
GO
```

10. In File Explorer, browse to **D:\Demofiles\Mod09**, right-click **ExampleDB.bak**, and then click **Copy**.
11. Open Internet Explorer, and browse to **http://portal.azure.com**.
12. Sign in to the Azure portal with your Azure Pass or Microsoft Account credentials.
13. In the navigation on the left, click **Virtual machines**.
14. In the list of virtual machines, click the one you created in the first demonstration in this module.
15. On the virtual machine blade, click **Connect**, and then click **Open**.
16. In the **Remote Desktop Connection** dialog box, click **Connect**.
17. In the **Windows Security** dialog box, click **Use another account**.

18. In the **User name** box, type **demoAdmin**.
19. In the **Password** box, type **Pa\$\$w0rd1234**, and then click **OK**.
20. In the **Remote Desktop Connection** dialog box, click **Yes**.
21. When the remote desktop session has started, open File Explorer, and browse to **C:**.
22. On the **Home** menu, click **Paste**. Explorer pastes the backup file into the VM.
23. Start **SQL Server Management Studio**.
24. In the **Connect to Server** dialog box, click **Connect**.
25. In Object Explorer, expand **Databases** and show that there are no user databases.
26. On the **File** menu, point to **New**, and then click **Query with Current Connection**.
27. Type the following Transact-SQL script, and then click **Execute**:

```
RESTORE DATABASE [ExampleDB] FROM DISK = 'C:\ExampleDB.bak'
WITH REPLACE,
MOVE 'TSQL' TO 'F:\Data\ExampleDB.mdf',
MOVE 'TSQL_log' TO 'F:\Log\ExampleDB_log.ldf';
GO
```
28. When the query completes, in Object Explorer, right-click **Databases**, and then click **Refresh**.
29. Expand **ExampleDB**, and then expand **Tables**.
30. Right-click **HR.Employees**, and then click **Select Top 1000 Rows**. The results should be the same as they were on the original database.
31. Close the Remote Desktop connection, close Internet Explorer, and then close SSMS without saving any changes.

Question: You want to migrate a business-critical production database to an Azure virtual machine. Users must be able to make changes to the database throughout the migration process. An Always On Availability Group protects the on-premises system. What method should you use for the migration?

Lab: Deploying SQL Server on an Azure Virtual Machine

Scenario

You have an on-premises database that you want to migrate into the cloud. To minimize the number of compatibility issues that will need correcting, you have decided to host the migrated database on a Microsoft® Azure® virtual machine that runs Microsoft SQL Server® 2016. You have been asked to create the virtual machine and migrate the database by using compressed backups. After the database has been migrated, you will connect to it and run a query.

Objectives

At the end of this lab, you will be able to:

- Create an Azure virtual machine that runs SQL Server 2016.
- Use compressed backups to migrate a database to an Azure virtual machine.
- Connect to a SQL Server instance that is running on an Azure virtual machine.

Estimated Time: 45 minutes

Virtual machine: **20765B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Provision an Azure Virtual Machine

Scenario

You are preparing to migrate a database from an on-premises instance of SQL Server into the cloud to be hosted on an Azure virtual machine. You have been asked to create a virtual machine that runs SQL Server 2016.

The main tasks for this exercise are as follows:

1. Prepare the Lab Environment
2. Create a New Azure Virtual Machine
3. Connect to the Virtual Machine by Using the Microsoft Remote Desktop Connection Client

► Task 1: Prepare the Lab Environment

1. Ensure that the MSL-TMG1, 20765B-MIA-DC, and 20765B-MIA-SQL virtual machines are all running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab09\Starter** folder as Administrator.

► Task 2: Create a New Azure Virtual Machine

1. Use Microsoft Internet Explorer® to sign in to the Azure portal by using your Azure pass or Microsoft account.
2. Create a new virtual machine. Use the following information:
 - **Template:** SQL Server 2016 SP1 Enterprise on Windows Server 2016
 - **Deployment model:** Resource Manager
 - **Name:** Choose a unique name for the virtual machine

- **User name:** SQLAdmin
- **Password:** Pa\$\$w0rd1234
- **Resource Group:** SQLResourceGroup
- **Location:** Choose a location near you
- **Size:** DS 11 V2 Standard

► Task 3: Connect to the Virtual Machine by Using the Microsoft Remote Desktop Connection Client

1. After the virtual machine has been created and started, connect to it by using the Remote Desktop Connection client. Use the following information:
 - User name: **SQLAdmin**
 - Password: **Pa\$\$w0rd1234**
2. In the Remote Desktop Connection client, open SQL Server Management Studio, and then use it to examine the databases that are present by default in the new SQL Server 2016 instance.
3. Close SQL Server Management Studio, and then log off from the Remote Desktop session.

Results: After this exercise, you will have created a virtual machine in your Azure subscription that runs SQL Server 2016.

Exercise 2: Migrate a Database to a Microsoft Azure Virtual Machine

Scenario

Now that you have a SQL Server 2016 instance running on an Azure virtual machine, you want to migrate a database to it from the on-premises server that runs SQL Server. To perform this migration, you will back up the on-premises database and restore it on the Azure virtual machine.

The main tasks for this exercise are as follows:

1. Back Up an On-Premises Database
2. Copy the Backup File to the Azure Virtual Machine
3. Restore the Database to the Azure Virtual Machine

► Task 1: Back Up an On-Premises Database

- In the 20765B-MIA-SQL image, use SQL Server Management Studio to backup the **ExampleDB** database. Use the following information:
 - Destination: **D:\Labfiles\Lab09\Backups.bak**
 - Back up to a new media set named **MigrationBackups**
 - Backup Set name: **ExampleDBMigration**
 - Compress the backup

► Task 2: Copy the Backup File to the Azure Virtual Machine

1. In the 20765B-MIA-SQL image, copy the backup file that you created in the last task to the clipboard.
2. Use the Remote Desktop client to connect to the SQL Server virtual machine in Azure.

3. Paste the backup file to the root folder of drive F.

► **Task 3: Restore the Database to the Azure Virtual Machine**

1. In the Remote Desktop connection to the Azure virtual machine, use SQL Server Management Studio to restore the **ExampleDB** database. Use the following information:
 - Source file: **F:\Backups.bak**
 - Relocate all files to the default locations
2. Close SQL Server Management Studio, and then log off from the Remote Desktop session.

Results: At the end of this exercise, you will have migrated a database from the on-premises server that runs SQL Server to the SQL Server instance on the new virtual machine.

Exercise 3: Configure SQL Server Connections

Scenario

You want to enable clients to connect to the migrated database so that they can query and update data. You have been asked to configure this connection and check that it is functional.

The main tasks for this exercise are as follows:

1. Configure SQL Server Connections for the Virtual Machine
2. Connect SQL Server Management Studio to the Azure Virtual Machine

► **Task 1: Configure SQL Server Connections for the Virtual Machine**

- In the Azure portal, edit the SQL Server configuration for the virtual machine that you created in Exercise 1. Use the following information:
 - **SQL connectivity:** Public (Internet)
 - **SQL Authentication:** Enabled
 - **SQL Authentication:** SQL Admin
 - **Password:** Pa\$\$w0rd1234

► **Task 2: Connect SQL Server Management Studio to the Azure Virtual Machine**

1. Make a note of the public IP address of the virtual machine on the **Overview** blade.
2. In SQL Server Management Studio, connect to that IP address by using the following information:
 - **Authentication:** SQL Server Authentication
 - **Login:** SQLAdmin
 - **Password:** Pa\$\$w0rd1234
3. In the connection to the virtual machine, execute a query against a table in the **ExampleDB** database.
4. Close SSMS, and then close Internet Explorer.

Results: When you have completed this exercise, you will have connected an instance of SQL Server Management Studio to the Azure instance of SQL Server 2016.

Question: After you created the backup file, the on-premises server remained available for clients to connect to and modify data. What would happen to these modifications after you had moved clients to the new database?

Module Review and Takeaways

By installing SQL Server 2016 on virtual machines that are hosted in Azure, you can construct an environment that very closely approximates an on-premises environment. However, by using the cloud-based IaaS features of Azure, you free yourself from the responsibility of building and maintaining server hardware and networking infrastructure. This is an easy way to realize many of the benefits of moving databases to the cloud without major database redesign.

Review Question(s)

Question: You have created an Always On Availability Group that includes three virtual machines in Azure. You want to ensure that the three virtual machines are in different fault domains and different update domains. What should you do?

Module 10

Managing Databases in the Cloud

Contents:

Module Overview	10-1
Lesson 1: Managing Security in Azure SQL Database	10-4
Lesson 2: Configuring Azure Storage	10-18
Lesson 3: Azure Automation	10-25
Lab: Managing Databases in the Cloud	10-33
Module Review and Takeaways	10-38

Module Overview

*** IMPORTANT ***

There are two things that you need to prepare before you start this module. These take some time to complete, so start them well before the training starts:

1. Creating an **AdventureWorksLT** database in Microsoft® Azure® for use with the "Encrypting Sensitive Data" demonstration.
2. Creating a virtual machine in Azure for use with the "Creating a Storage Pool" demonstration.

You will need your Azure login credentials, and the range of public-facing IP addresses that your training center uses.

Create an AdventureWorksLT database

1. Sign in to the Azure Portal: <https://portal.azure.com> with your Azure credentials.
2. Click **New > Databases > SQL Database**. Enter **AdventureWorksLT** for the database name.
3. Under **Subscription**, there is no need to change the subscription unless you have more than one subscription.
4. Under Resource Group, click **Create new** and type **20765B + YourInitials**. The Resource Group name must be unique; if the name is valid, a green tick appears.
5. Under **Select source**, select **Sample (AdventureWorksLT)**.
6. Under **Server**, click the right arrow to **Configure required settings**.
7. Under server name, type **20765B+YourInitials**. A green tick appears if the server name is valid.
8. Type **Student** for the server admin login, and **Pa\$\$w0rd** for the password. Confirm the password.
9. Select a nearby location.
10. Select **Yes** for **Create V12 server**.
11. Do not change the setting for **Allow azure services to access server**.

12. Click **Select**.
 13. For **Want to use SQL elastic pool**, select **Not now**.
 14. Under **Pricing tier**, click the right arrow for **Configure required settings**. Select **S1 Standard 20 DTUs**.
 15. Do not change the **Collation** setting.
 16. Select **Pin to Dashboard**.
 17. Select **Create** to create the SQL Server® instance, including the AdventureWorksLT database. This will take a few minutes.
 18. To configure the firewall, click **All resources**, and then click the name of the server you created. The SQL Server properties blade is displayed.
 19. In the Settings section, click **Firewall**, and click **Add Client IP**. Click Save before closing.
- The Azure SQL Database is now ready for demonstration later in the module.

Create an Azure VM with Data Disks

1. Start the MSL-TMG1, 20765B-MIA-DC and 20765B-MIA-SQL virtual machines, and log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Demofiles\Mod10** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and wait for the script to finish.
4. On the taskbar, right-click **PowerShell ISE**, and then click **Run ISE as Administrator**.
5. In the **User Account Control** dialog box, click **Yes**.
6. Open the Windows PowerShell script **UpdateAzureRM.ps1** from **D:\Demofiles\Mod10\Demo** folder.
7. Use **Run Selection** to run the script block under **#Install latest AzureRM module**.
8. In the **NuGet provider is required to continue** dialog box, click **Yes**.
9. In the **Untrusted repository** dialog box, click **Yes to All**.
10. Open the PowerShell script **StorageSpacesDemo.ps1** from **D:\Demofiles\Mod10\Demo** folder.
11. In the **# Initialize** block, under **# Initialize variables**, amend the **\$subscriptionName** and **\$TenantID** to those relating to your subscription. You will find these in the Azure portal. The **TenantID** is the same as **DirectoryID** in **Azure Active Directory, Properties**. You can copy the string and paste it into the PowerShell script.
12. Use **Run Selection** to run the script block under **# Initialize**.
13. Use **Run Selection** to run the script block under **#Sign in to Azure and select subscription**.
14. When prompted, sign in with your Azure credentials.
15. Use **Run Selection** to run the script block under **# Prompt for credentials for VM and SQL Server**.
16. When prompted, enter the password **Pa\$\$w0rd**, and then click **OK**.
17. Use **Run Selection** to run the script block under **# Create resource group**.
18. Use **Run Selection** to run the script block under **# Create a virtual network**. Note the warning message.
19. Use **Run Selection** to run the script block under **# Create public IP address and network interface**. Note the error message. This will take a minute or two to run.

20. Use **Run Selection** to run the script block under **# Create virtual machine**. This will take several minutes to complete.
21. Use **Run Selection** to run the script block under **# Add data disks to the VM**. Note the warning message. This will take a minute or so to complete.
22. Close the Windows PowerShell ISE, without saving any changes.

A new virtual machine that has additional data disks has been created. This is now ready for the “Creating a Storage Pool” demonstration.

Objectives

After completing this module, you will be able to:

- Describe how to manage SQL Server security on Azure.
- Describe how to automate deployment of Azure SQL Database.

Lesson 1

Managing Security in Azure SQL Database

This lesson considers several security features that are available in Azure SQL Database. As cybercrime increases, and more data moves to the cloud, it is more important than ever to ensure that databases are secure. This lesson considers some of the security features that are built into SQL Server and Azure SQL Database.

Lesson Objectives

In this lesson, you will learn:

- What Always Encrypted is, and how to configure it in Azure SQL Database.
- How to configure column-level encryption in Azure SQL Database.
- How to add dynamic data masking to a database in Azure SQL Database.
- How to configure transparent data encryption in Azure SQL Database.
- How to configure firewall rules for an Azure server.

Always Encrypted for Azure SQL Database

What Is Always Encrypted?

Always Encrypted protects sensitive data by encrypting specific columns, such as those containing credit card numbers, date of birth, and so on. It is not designed to encrypt entire tables or a database, but instead, to encrypt specific columns that contain data that needs to be encrypted.

As with all types of encryption, there is a small performance overhead when you implement Always Encrypted, depending on the type of data that is being encrypted.

Always Encrypted has two innovative features:

1. **Data is encrypted at rest and in motion.** This means that data is encrypted when it is stored on disk; when it moves between the client and the server; while it is being processed; and when the results are displayed.
2. **Data owners can store encryption keys on-premises to prevent Microsoft cloud administrators seeing sensitive data within their databases.** Encryption keys are stored on the client side, so an administrator cannot decrypt the data. The server does not have access to plaintext keys.



Note: When both the client application and the data are hosted in Azure, such as for web-based applications, the encryption keys must be stored in a cloud key store such as Azure Key Vault. In this situation, Always Encrypted does not provide complete protection against a rogue administrator, but it does significantly reduce the attack area.

- Always Encrypted encrypts specific columns
 - “At rest” and “in flight”
 - Keys are stored on the client
- Deterministic or randomized encryption
- Two keys:
 - Column Master Key—encrypts the Column Encryption Key
 - Column Encryption Key—encrypts the data
- Prerequisites
 - .NET Framework 4.6 or higher, and a supported data access driver
 - SQL Server Management Studio version 13.0.700.242 or higher

Encryption Types

There are two ways of encrypting data by using Always Encrypted:

1. **Randomized encryption.** This type of encryption produces a different result every time a value is encrypted.
2. **Deterministic encryption.** This type of encryption produces the same result every time a value is encrypted.

Randomized encryption produces different cipher text each time plaintext is entered. This prevents an eavesdropper from learning anything about the encryption process. Although it is secure, randomized encryption does not allow operations on data, nor can you index these columns.

Deterministic encryption always produces the same cipher text each time a plaintext value is encrypted. Columns that are encrypted by using the deterministic option support equality joins and the GROUP BY option, which means that you can join tables and filter the results of a WHERE clause. Deterministic encryption also supports indexed columns. Do not use deterministic encryption for columns that only have a few values, such as gender or regions. Using deterministic encryption on highly repetitive values is theoretically vulnerable to a hacker deciphering patterns.

Always Encryption uses a secure 256-bit encryption key. The algorithm that is used to encrypt data is AEAD_AES_256_CBC_HMAC_SHA_256, which uses cipher block chaining (CBC) to conceal patterns. Two variations of the algorithm are used so that both deterministic and randomized encryption can be supported.

If you are interested in the cryptographic algorithm that Always Encrypted uses, see *Always Encrypted Cryptography* on MSDN:



Always Encrypted Cryptography

<https://aka.ms/rz9e59>

Encryption Keys

Always Encrypted uses encryption keys that are stored on the client side, not on the server. Two kinds of keys are required:

- **Column Master Key.** This key is used to encrypt column encryption keys.
- **Column Encryption Key.** This key is used to encrypt data.

Master encryption keys are stored in a secure key store such as the Windows® certificate store or Azure Key Vault. The **Column Master Key** object in the database stores information about the location of the key.

You can create the keys by using SQL Server Management Studio. First, navigate to **Security**, and then click **Always Encrypted Keys**. Create the **Column Master Key** object first because this will be required when you create the **Column Encryption Key** object:

1. Right-click **Column Master Keys** in the tree, and then click **New Column Master Key ...** to display the **New Column Master Key** dialog box.
2. Type a name for the key, and then select a location where the key will be stored: either the Windows certificate store or Azure Key Vault. You will be prompted to sign in to Azure Key Vault.
3. Click **Generate Certificate** so that the key is stored in an appropriate certificate. Alternatively, highlight an existing certificate.
4. Click **OK** to create the key and certificate.

Then, create the **Column Encryption Key** object:

1. Right-click **Column Encryption Keys**, and then click **New Column Encryption Key ...** to display the **Column Encryption Keys** dialog box.
2. Type a name for the key, and then select a **Column Master Key** object. This protects your **Column Encryption Key** object.
3. Click **OK**.



Reference Links: Click the **Script** option, and then select **New Query Window** to see the syntax for both **Column Master Key** and **Column Encryption Key**. You can also right-click existing keys, and then select **Script ...** to view the script in a new query window.



Best Practice: You can create Windows PowerShell scripts to create encryption keys. You can also create Windows PowerShell scripts to encrypt data.

When you have created both keys, you are ready to encrypt columns by using Always Encrypted.

Prerequisites for Always Encrypted

There is just one prerequisite—you must install Microsoft .NET Framework 4.6 or later and be running SQL Server Management Studio version 13.0.700.242 or later. Always Encrypted uses the data access driver to encrypt and decrypt data. Currently, three data access drivers are supported:

1. The .NET Framework Data Provider for SQL Server.
2. Microsoft ODBC Driver for SQL Server 13.1 or higher.
3. Microsoft JDBC Driver 6.0 or higher.

You can now freely download and upgrade SQL Server Management Studio independently of the SQL Server database engine. You can also install it side by side with previous versions of SQL Server Management Studio.

For more information and to download the latest version, see *Download SQL Server Management Studio (SSMS)* on MSDN:



Download SQL Server Management Studio (SSMS)

<https://aka.ms/q57mha>

Encrypting Data

To encrypt the relevant columns:

1. Using Object Explorer, expand the **Tables** node. Select the table and column that you want to encrypt.
2. Right-click the column name, and then click **Encrypt Column** on the context-sensitive menu to display the Always Encrypted Wizard.
3. An introductory page is displayed. Click **Next**, and then optionally, click **Do not show this page again**.
4. Select the name of the column that you want to encrypt.
5. Select **Randomized** or **Deterministic** encryption.
6. Select the Column Encryption Key that you created earlier. Optionally, the wizard will create a new Column Encryption Key if you have not already done so.

7. Select either **Proceed to finish now** or **Generate a PowerShell script to run later**. If you choose to generate a Windows PowerShell script, provide a location for the script. Click **Next**.
8. The wizard displays the summary screen. Check that everything is correct, and then click **Next**. Either the column is encrypted, or a Windows PowerShell script is created in the specified location.

To check that the data is encrypted, execute a simple **SELECT** query to display the column data. Always Encrypted columns will appear with obfuscated text.

To remove Always Encrypted encryption from a column, right-click the column name, and then click **Encrypt Column** to start the Always Encrypted Wizard. Choose **plaintext** instead of either randomized or deterministic encryption, and use the same encryption key. Click **Next** to see a summary, and then click **Next** again to decrypt the column.



Note: For Always Encrypted encryption to work, columns must have one of the binary2 collations. If the column does not have a binary2 collation, a message appears to say that the collation will be changed.

There are other restrictions, such as columns that have the following characteristics:

- FILESTREAM columns.
- Columns that use the following data types: XML, timestamp/rowversion, image, ntext, text, sql_variant, hierarchyid, geography, geometry, alias, and user-defined data types.
- The ROWGUIDCOL property.
- Columns that are referenced by CHECK constraints.
- Columns that are masked with dynamic data masking.

There are more restrictions. For full details, see the Always Encrypted documentation on MSDN:



Always Encrypted (Database Engine)

<https://aka.ms/p3zyx0>

Column-Level Encryption

Column-level encryption is also known as cell-level encryption because it applies encryption to individual columns in a table. Column-level encryption uses symmetric key encryption, and certificates to store the keys. You can then use Transact-SQL to encrypt columns of data.

Use column-level encryption with either on-premises versions of SQL Server or Azure SQL Database. Column-level encryption functionality was included in Azure SQL Database to improve the compatibility between the on-premises and Azure versions of SQL Server. The two work in a very similar way, although with a couple of differences:

1. In on-premises SQL Server, the key encryption hierarchy uses the Service Master Key, which is specific to the SQL Server instance. In Azure SQL Database, the root encryption uses a certificate that is managed by Azure SQL Database. This simplifies key management significantly.

- Column-level encryption is also known as cell-level encryption
- It is available in Azure SQL Database
- Use certificates to store the symmetric key
- Ensure that exported data can be decrypted
- ENCRYPTBYKEY to encrypt a data column

2. Azure SQL Database syntax does not support references to files, including backing up the Master Key and certificates, restoring backups, or importing certificates or asymmetric keys from files. This restriction can result in encrypted data that is exported from an Azure SQL Database being unable to be decrypted. You should therefore use certificates that can be extracted and then re-created.

Implementing Column-Level Encryption

To encrypt data in Azure SQL Database by using column-level encryption:

1. Create a database master key (or use the existing database master key).
2. Create a certificate to hold the symmetric key by using **CREATE SYMMETRIC KEY** with the **KEY_SOURCE** and **IDENTITY_VALUE** fields set so that the key can be re-created elsewhere if needed.
3. Use the symmetric key to encrypt a column of data.



Note: Symmetric encryption uses the same cryptographic key both to encrypt and decrypt plaintext. Symmetric keys are, therefore, a shared secret between the party that is encrypting the data, and one or more parties that are decrypting the data. Symmetric key encryption gives better database performance compared to asymmetric encryption.

In the following example, a new column called **CardNumber_Encrypted** is added to the **SalesLT.Customer** table in the **AdventureWorksLT** database in Azure SQL Database. By using a symmetric key that is stored in a certificate, the **CardNumber_Encrypted** column is encrypted. A card number is added to the table, and a **SELECT** statement is used so that we can see that the value has been encrypted.

EncryptByKey

```
--If there is no master key, create one now.
IF NOT EXISTS
    (SELECT * FROM sys.symmetric_keys WHERE symmetric_key_id = 101)
    CREATE MASTER KEY ENCRYPTION BY
        PASSWORD = '23987hxJKL93QYV43j9#ghf0%lekjg5k3fd117r$$#1946kcj$n44ncjidld'
GO

CREATE CERTIFICATE CreditCard
    WITH SUBJECT = 'Credit Card Numbers';
GO

CREATE SYMMETRIC KEY CreditCards_Key1
    WITH ALGORITHM = AES_256
    ENCRYPTION BY CERTIFICATE CreditCard;
GO

-- Create a column in which to store the encrypted data.
ALTER TABLE SalesLT.Customer
    ADD CardNumber_Encrypted varbinary(128);
GO

-- Open the symmetric key with which to encrypt the data.
OPEN SYMMETRIC KEY CreditCards_Key1
    DECRYPTION BY CERTIFICATE CreditCard;

-- Encrypt the value in column CardNumber using the
-- symmetric key CreditCards_Key1.
-- Save the result in column CardNumber_Encrypted.
UPDATE SalesLT.Customer
    SET CardNumber_Encrypted = EncryptByKey(Key_GUID('CreditCards_Key1'),
        CardNumber_Encrypted );
GO
```

```

UPDATE SalesLT.Customer
SET CardNumber_Encrypted = 12345678
WHERE CustomerID = 1

SELECT * FROM SalesLT.Customer

```

For more information about using cell-level encryption in Azure SQL Database, see *Recommendations for using Cell Level Encryption in Azure SQL Database* on MSDN:



Recommendations for using Cell Level Encryption in Azure SQL Database

<https://aka.ms/gsexle>

Dynamic Data Masking

Dynamic data masking limits the exposure of sensitive, personal information by obscuring all or part of the data from nonauthorized users. Azure SQL Database V12 supports dynamic data masking, as do on-premises editions of SQL Server 2016. If you have Azure SQL Database admin permissions, service admin, or security office roles, you can configure dynamic data masking on your Azure SQL Database.

- Add dynamic data masking in the Azure portal, or use Windows PowerShell cmdlets
- Dynamic data masking obfuscates sensitive data
- Use for masking sensitive data such as social security numbers, credit card numbers, and so on.
- The data masks that are available:
 - Default
 - Email
 - Customer
 - Random
 - Credit card
- Users who have admin permissions see unmasked data



Note: SQL Server administrators can always see unmasked data; they will see the full data in plaintext.

With dynamic data masking, the underlying data is not encrypted; it is just masked or obfuscated. This means that when the results are returned to anyone who does not have the necessary permissions, either all or part of the data is masked. Dynamic data masking has minimal impact on the application layer because it is configured on the database. The data does not change—only the way in which it is presented changes. Data masking is defined at the column level—for example, credit card numbers, date of birth, and so on. There are different masking rules that either completely obscure the data, or just partially obscure it.

Use dynamic data masking together with other security features as part of your overall security strategy. For a good overview of dynamic data masking, see *Dynamic Data Masking* on MSDN:



Dynamic Data Masking

<http://aka.ms/cn6gg3>

Dynamic Data Mask Types

Define a data mask at the column level when you use CREATE TABLE or ALTER TABLE statements. Typically, these are columns that contain sensitive or personal data such as a credit card number.

The following table lists the various data masks that are available for you to choose from.

Mask type	Masking logic	Example
Default	The complete value is masked with a character that is appropriate to the data type. For example, characters are shown as xxx, and numeric with 0.	abcdefgh would be masked as xxxx. 1234 would be masked as 0000. Shorter fields would be masked with fewer than four characters.
Email	Displays the first letter of the email address, masking the remainder. All suffixes are replaced with .com.	jennyk@adventure.co.uk would be masked as jxxxxx@xxxxx.com.
Custom string	Partially masks the value, exposing several characters at the start and end with custom padding. Syntax is (FUNCTION = 'partial(prefix, [padding], suffix)').	
Random number	For numeric data types. Replaces the actual value with a random value in a given range.	Depends on the data type.
Credit card value	Only the last four digits of a credit card number are displayed; the rest of the number is masked.	xxxx-xxxx-xxxx-9876

Add Dynamic Data Masking to Columns

To add dynamic data masking to one or more fields in a database in Azure SQL Database in the Azure portal:

1. In the Azure portal, select the database.
2. In the **Settings** group, select **Dynamic Data Masking**.
3. The schema, table, and column names of **recommended fields to mask** are displayed.
4. Select the field that you want to mask, and then click **Add Mask**.
5. The field is then displayed in the **Masking Rules** list, together with a **Mask Function**.
6. Amend the **Mask Function** by clicking the mask rule. The **Edit Masking Rule** blade appears.
7. Under **Select how to mask**, click the drop-down arrow to view the different masking field formats.
8. Select the correct **Masking field format**, and then click **Update** to save it.
9. Click **X** to close the blade.
10. In the box labeled **SQL users excluded from masking (administrators are always excluded)**, type a list of users, separated by a semicolon. These are users who will be able to see unmasked data.

You can also use Windows PowerShell to set up dynamic data masking.

Which Columns Are Suitable for Dynamic Data Masking?

You can add a dynamic data mask to most columns. Exceptions include:

- Columns that are encrypted by using Always Encrypted.
- FILESTREAM columns.
- Column set columns.

What Permissions Do You Need?

Adding dynamic data masking to columns while creating a new table requires the normal CREATE TABLE and ALTER permissions. However, to alter existing tables, you need the ALTER ANY MASK permission. Using this permission, you can add, replace, or remove a mask on a column.

For those viewing the data, by default, users will see masked data, where a mask has been set up. To view plaintext data, users must be added to the **SQL users excluded from masking (administrators are always excluded)** box, or have admin permissions.

Transparent Data Encryption

Azure SQL Database supports transparent data encryption, often abbreviated to TDE, to encrypt the whole database, including log files, and backups while it is stored on disk. It was introduced with SQL Server 2008.

No changes are required to the application because data is encrypted and decrypted on the disk. This is encryption “at rest,” meaning that when data is stored on the disk, it is encrypted; when it is read from disk, it is decrypted. This means that no modification is needed to any applications that use the database. The whole database is encrypted, with no option to have certain tables or columns left in plaintext.

TDE protects the complete database from the theft of physical data storage media, but provides no protection for data as it is processed and communicated. For example, TDE permits a database administrator, or an intruder, to see plaintext in SQL Server Management Studio query results. If you want to ensure that database administrators do not see sensitive data, use Always Encrypted.

Always Encrypted and TDE are complementary encryption technologies that provide protection in different ways, which means that you can encrypt the database by using TDE and encrypt sensitive columns by using Always Encrypted.

- Transparent Data Encryption encrypts the whole database “at rest”
- Enable TDE through the Azure portal by using Transact-SQL, Windows PowerShell, or the Rest API
- Automatic key creation and key rotation
- No support for integration with Azure Key Vault
- Protects against theft of physical media
- No protection against unauthorized access to data in SQL Server Management Studio or in an application
- Use in combination with Always Encrypted

How to Configure Transparent Data Encryption

You can configure TDE through the Azure portal:

1. On the Azure dashboard, select the database that you want to encrypt.
2. In the **Settings** group, select **transparent data encryption**.
3. Change the **Data Encryption** status from **OFF** to **ON**.
4. Click **Save**.

When the database has been encrypted, the **Encryption status** icon will change first to **Encryption is in progress**, and then to **Encrypted**.

Alternatively, you can configure TDE by using Windows PowerShell or the REST application programming interface (API). You must connect as the SQL Server Manager, Azure Owner, or Contributor.

You can also use Transact-SQL to encrypt a database in Azure SQL Database. Using SQL Server Management Studio, attach to the relevant database in Azure SQL Database, and then use the ALTER DATABASE command.

Use the ALTER DATABASE command to encrypt a database in Azure SQL Database. Use SET ENCRYPTION OFF to decrypt the database.

ALTER DATABASE

```
ALTER DATABASE [AdventureWorksLT] SET ENCRYPTION ON;  
GO
```

Encryption Keys

The database is encrypted by using the database encryption key, which is a symmetric key that is itself protected by a server certificate. The encryption keys are automatically created when the encryption option is selected. Plus, the encryption keys are changed approximately every 90 days for increased security. This is a real benefit because secure key management is not trivial.

In contrast to the on-premises versions of SQL Server, including SQL Server running in an Azure virtual machine, Azure SQL Database does not support integration with Azure Key Vault. However, for most organizations, this is unlikely to be an issue because Microsoft manages encryption keys when TDE is selected.

When to Use TDE

TDE is designed to protect databases that are stored on-premises. It protects against physical media being stolen, such as a hard disk or backup media. The database, backups, log files, and tempdb are all encrypted, so it provides good protection in the event of physical media being stolen, and assuming that encryption keys are stored securely and separately. Azure SQL Database offers encryption without the overhead of having to manage encryption keys, so it is a sensible precaution in the event of disks or backups being stolen at a Microsoft data center.

TDE does not provide any security against anyone who has access to the database, such as an administrator or user. Data that is selected from the database is presented in plaintext, both in SQL Server Management Studio and in applications that are attached to the database. For this reason, it is recommended that TDE is used in combination with Always Encrypted, which reduces the attack area considerably and presents data in cipher text on the server.

For more information about using TDE with Azure SQL Database, see *Transparent Data Encryption with Azure SQL Database* on MSDN:



Transparent Data Encryption with Azure SQL Database

<https://aka.ms/tvoddw>

Configuring Firewall Rules

Firewall rules enable administrators to limit access to Azure servers or databases to specific IP addresses, or ranges of IP addresses. You can configure firewalls at either the server level or the database level. If the IP address that is trying to gain access is not allowed at either the server level or the database level, access will be denied.

This is not an optional setting—you must configure one or more IP addresses, or ranges of IP addresses. The narrower the range you set, the more secure your Azure servers or databases will be. However, you must make sure that the firewall is configured so that everyone who needs access is allowed access.

- Azure firewall rules limit access based on IP addresses
- Set server-level or database-level firewall rules
- Server-level firewall rules
 - Apply to many databases on the same server
 - All need the same or similar access requirements
 - Are set using the Azure portal or Windows PowerShell scripts
- Database-level firewall rules
 - Are recommended by Microsoft
 - Improve security and portability
 - Are set by using `sp_set_databases_firewall_rule`

Server-Level Firewall Rules

For servers that host several different databases that all have the same access requirements, it is better to configure firewall rules at the server level. This means that you don't have to configure firewalls for individual databases. However, it also means that the range of IP addresses is likely to be wider. So, although this is more convenient, it potentially means that you have to open up a wider range of IP addresses, not all of which may be necessary.

To create a firewall rule at the server level:

1. In the Azure portal, select the server that you want to configure. The server blade appears.
2. In the **Settings** group, select **Firewall**. The firewall blade is displayed.
3. The Client IP address that you are currently using is displayed. You can add this IP address by clicking **Add Client IP**.
4. Create firewall rules by providing values for **Rule Name**, **Start IP**, and **End IP**, and then click **Save**.
5. Alternatively, to remove a rule that you no longer want, select the rule, and then click **Discard**.



Note: Save each firewall rule as you create it. You cannot save more than one firewall rule at a time.



Best Practice: Before you add the current Client IP address to the firewall rules, check that the correct IP address is displayed.

For more information about configuring server-level firewall rules, see the Azure documentation:



Create and manage Azure SQL Database server-level firewall rules using the Azure portal

<https://aka.ms/kwgkny>

Database-Level Firewall Rules

In addition to setting server-level firewall rules, you can also apply firewall rules at the database level. Database-level firewall rules enable you to allow access specifically to certain IP addresses, thereby creating a more secure environment. Configuring database-level firewall rules also makes a database more portable.

Unlike server-level firewall rules, which are set by using the Azure portal or a Windows PowerShell script, database-level firewall rules are set by using Transact-SQL. The **sp_set_database_firewall_rule** stored procedure is used with either the master database or user databases. Note that the name of the firewall rule must be unique for each database.

Set database-level firewall rules by using Transact-SQL.

Sp_set_database_firewall_rule

```
EXEC sp_set_database_firewall_rule @name = N'AdventureWorks HR',
    @start_ip_address = 'nnn.nnn.nnn.nnn', @end_ip_address = 'nnn.nnn.nnn.nnn';
```



Note: If you attempt to connect to a database from an IP address that is not allowed access, a dialog box appears that enables you to add your client IP address. You must, however, first sign in to your Azure account.

To delete a database-level firewall rule, use **sp_delete_database_firewall_rule**.

For more information about configuring database-level firewall rules, see the Azure documentation:



Overview of Azure SQL Database firewall rules

<https://aka.ms/h3os15>

Demonstration: Encrypting Sensitive Data

You are a data security consultant working for the Adventure Works Bicycle Company. The IT director is worried about data security after a competitor's website was hacked. She has asked you to show IT staff how Always Encrypted works, and obtain feedback on how the company might use Always Encrypted to protect sensitive data.

Demonstration Steps

1. Start the MSL-TMG1, 20765B-MIA-DC and 20765B-MIA-SQL virtual machines, and log on to **20765B-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Open SQL Server Management Studio and connect to the server you created earlier, for example **20765BCE.database.windows.net**, using SQL Server Authentication.
3. In the **Login** box, type **Student**, and in the **Password** box, type **Pa\$\$w0rd**, and then click **Connect**.

Create the encryption keys

1. Click **New Query** to open a new query window.
2. In Available Databases, select **AdventureWorksLT**.
3. In Object Explorer, expand **Databases**, expand **AdventureWorksLT**, expand **Security**, and then expand **Always Encrypted Keys**.
4. Right-click **Column Master Keys** and click **New Column Master Key**.
5. In the **New Column Master Key** dialog box, in the **Name** box, type **CMK1**.
6. In the **Key store** list, select **Windows Certificate Store - Current User**.
7. Click **Generate Certificate**. The certificate appears in the list, and then click **OK**.
8. In Object Explorer, right-click **Column Encryption Keys** and click **New Column Encryption Key**.

9. In the **New Column Encryption Key** dialog box, in the **Name** box, type **CEK1**.
10. In the **Column master key** list, click **CMK1**, and then click **OK**.

Encrypt the Data

1. In Object Explorer, under **AdventureWorksLT**, expand **Tables**, expand **SalesLT.Address**, and then expand **Columns** to display the list of columns.
2. In the query window, type:

```
SELECT * FROM [SalesLT].[Address]
```

3. Click **Execute** to show that all the columns are displayed in plaintext.
4. Under **Columns**, right-click **City**, and then click **Encrypt Column**.
5. In the **Always Encrypted** wizard, on the **Introduction** page, click **Next**.
6. On the **Column Selection** page, select **City**, and in the **Encryption Type** column, select **Deterministic**, and in the **Encryption Key** column, select **CEK1**. Note the collation message, and then click **Next**.
7. On the **Master Key Configuration** page, click **Next**.
8. On the **Run Settings** page, ensure **Proceed to finish now** is selected, and then click **Next**.
9. On the **Summary** page, click **Finish**.
10. When it has completed, click **Close**.
11. In the query window, type:

```
SELECT * FROM [SalesLT].[Address]
```

12. Click **Execute** to run the query to show the column appears with obfuscated text. NOTE: Values are repeated because deterministic encryption has been used, and each City appears more than once.
13. In the query window, type:

```
SELECT CountryRegion, PostalCode, count(City) as CityCount
FROM salesLT.Address
GROUP BY CountryRegion, PostalCode, City
```

14. Highlight the query and click **Execute** to show the number of Cities in each postal code. This is possible because deterministic encryption was selected.
15. In Object Explorer, under **SalesLT.Address**, expand **Indexes**, right-click **IX_Address_AddressLine1_AddressLine2_City_StateProvince_PostalCode**, point to **Script Index as**, point to **DROP And CREATE To**, and then click **New Query Editor Window** in order to drop the index.
16. The **PostalCode** column cannot be encrypted while it is included in an index. Run the first part of the script to drop the index.
17. In Object Explorer, right-click **PostalCode**, and then click **Encrypt Column**.
18. In the **Always Encrypted** wizard, on the **Introduction** page, click **Next**.
19. On the **Column Selection** page, select **PostalCode**, and in the **Encryption Type** column, click **Randomized**, and in the **Encryption Key** column, click **CEK1**. Note the collation message, and then click **Next**.
20. On the **Master Key Configuration** page, click **Next**.

21. On the **Run Settings** page, ensure **Proceed to finish now** is selected, and then click **Next**.
22. On the **Summary** page, click **Finish**.
23. When it has completed, click **Close**.
24. Click **New Query** to open a new query window.
25. In the query window, type:

```
SELECT * FROM [SalesLT].[Address]
```

26. Click **Execute** to show that the columns appear with obfuscated text.
27. In the query window, type:

```
SELECT COUNT(*) FROM SalesLT.Address  
GROUP BY PostalCode
```

28. Highlight the query and click **Execute** to show that the GROUP BY operation fails with randomized encryption. The error message explains that Deterministic encryption is required for the statement to succeed.

Remove Encryption from a Column

1. In Object Explorer, right-click **PostalCode** and click **Encrypt Column**.
2. In the **Always Encrypted** wizard, on the **Introduction** page, click **Next**.
3. On the **Column Selection** page, select **PostalCode**, and in the **Encryption Type** column, select **Plaintext**, and in the **Encryption Key** column, select **CEK1**, and then click **Next**. Note that the correct key is required to decrypt the data.
4. On the **Master Key Configuration** page, click **Next**.
5. On the **Run Settings** page, ensure **Proceed to finish now** is selected, and then click **Next**.
6. On the **Summary** page, click **Finish**.
7. When it has completed, click **Close**.
8. In the query window, type:

```
SELECT * FROM [SalesLT].[Address]
```

9. Highlight the query and click **Execute** to show that the **PostalCode** column has been decrypted.
10. In Object Explorer, right-click **SalesLT.Address**, point to **Script Table as**, point to **CREATE To**, and then click **New Query Editor Window**. Point out the encrypted column, and the encryption algorithm used.
11. Close SQL Server Management Studio, without saving any changes.

Check Your Knowledge

Question	
When might you use Always Encrypted?	
Select the correct answer.	
☐	To encrypt your database backup in the event of theft.
☐	To protect sensitive data by encrypting specific columns.
☐	As an alternative to transparent data encryption.
☐	To protect data against hard disk failure.
☐	To prevent data from being migrated to the cloud.

Lesson 2

Configuring Azure Storage

In this lesson, you will learn how to configure Azure storage and manage storage pools.

Lesson Objectives

In this lesson, you will:

- Understand the different types of Azure storage.
- Understand what Microsoft Azure Storage Explorer is, and how to use it.
- Understand how to manage storage pools.

Azure Storage Accounts

Azure storage offers a secure and flexible storage solution that includes benefits such as geo-replication; paying only for what you need, rather than investing in on-premises hardware; and pricing that is tailored to your needs.

There are four different types of Azure storage:

- **Blob.** Binary large object (BLOB) storage is for unstructured data.
- **Table.** This type of storage is suitable for NoSQL storage.
- **Queue.** You can use this type of storage for asynchronous messaging.
- **File.** This type of storage is for file storage.

In addition, you can create solid-state drives (SSDs) or hard disks for use with Azure virtual machines.

To use Azure storage, you must first create an Azure storage account. An Azure storage account holds different types of Azure storage.

Create an Azure Storage Account

You need to create an Azure storage account to hold your data. Everything in your Azure storage account is billed together. You can create Azure storage by using the Azure portal:

1. In the Azure portal, click **New**.
2. In the **New** blade, click **Storage**. A list of the different storage options is displayed.
3. Click **Storage account**.
4. In the **Name** box, type a unique name in lowercase characters and numbers. A green tick appears when an acceptable name is entered.
5. In **Deployment model**, select either **Resource manager** or **Classic**. Select **Resource manager** for all new applications.

- Four types of Azure storage:
 - **Blob.** For unstructured data
 - **File.** For file shares
 - **Queue.** For asynchronous message storage
 - **Table.** For noSQL storage
- Create a storage account in the Azure portal:
 - General—holds blobs, file, queue and tables
 - Blob—only holds BLOBs. Can specify:
 - Hot access tier—frequently accessed data
 - Cool access tier—data accessed less often (less expensive)

6. In **Account kind**, select either **General** or **Blob**. General storage accounts enable you to store different types of storage in one account including blobs, tables, queues, and files. Blob storage accounts are optimized for storing BLOB data, and additionally enable you to specify how often you will access the data.
7. In **Performance**, select either **Standard** or **Premium**. Premium uses SSDs whereas Standard uses magnetic disks. You cannot change the Performance setting after the storage account has been created.
8. Select **Replication**, and then choose from:
 - a. Read-access geo-redundant storage (RA-GRS)
 - b. Zone-redundant storage (ZRS)
 - c. Locally redundant storage (LRS)
 - d. Geo-redundant storage (GRS)
9. Select **Storage service encryption**, and then select either **Disabled** or **Enabled**. The data is encrypted when it is stored on the disk, and is decrypted as it is retrieved from the database. Keys are managed automatically, and AES 256-bit encryption is used. At the time of writing, this feature is in preview for file storage. Note that encryption is only available if you select **Resource manager** as the **Deployment model**.
10. In **Location**, select the data center where you want your storage account to be created.
11. Click **Create** to create your storage account. A message is displayed when the storage account has been successfully created.



Note: Although it is possible to change some options after the storage account has been created, **Performance** and **Replication** cannot be changed.

Blobs Storage

Blob storage is used to store data as unstructured BLOBs. You can choose between either a **Hot access tier**, for data that you need to access often, or a **Cool access tier** for data that is accessed infrequently and is less expensive. If you will only store Blobs, select **Blob** in **Account Kind** when you create your Azure storage account.

Azure Storage Explorer

Azure Storage Explorer is a cross-platform tool that enables you to work with Azure storage through a friendly GUI. It is possible to download it for free and it is available for macOS, Linux, and Windows. At the time of writing, Azure Storage Explorer is in preview.

You can download Azure Storage Explorer from the Microsoft Azure website:



Microsoft Azure Storage Explorer

<https://aka.ms/rpasrv>

- Azure Storage Explorer is a stand-alone tool
 - Used to manage Azure storage
 - Free to download
 - Currently in preview
- Connect to Azure by using:
 - An Azure account
 - A Shared Access Signature
 - A Storage account name and access key
- Provides a friendly user interface to manage Azure storage
 - Copy or import data into storage
 - Download or export data

Using Azure Storage Explorer

To use Azure Storage Explorer, you must first have created an Azure storage account, as discussed in the previous lesson. You can view and manage Azure storage, including attaching by using a shared access signature (SAS) key.

You can connect to Azure storage in one of three ways:

1. By providing your Azure account credentials.
2. By using a shared access signature.
3. By using a storage account name together with an access key.

When you are connected, you can:

- View the storage resources that you have created.
- Create new storage assets.
- Import or copy to storage in Azure.
- Export and download from storage in Azure.
- Delete BLOB containers, tables, queues, and files.

Azure Storage Explorer provides a friendly interface to manage Azure storage.

For more information about Azure Storage Explorer, and specifically about managing BLOB storage by using Azure Storage Explorer, see the Azure documentation:



Manage Azure Blob Storage resources with Storage Explorer (Preview)

<https://aka.ms/rx38oe>

Managing Storage Pools

Storage spaces and storage pools are a feature of the Windows Server® operating system. You can pool physical hard disks together to create virtual disks, known as storage spaces. The same functionality is available in the Windows operating system when you configure an Azure virtual machine.

The disks that are used for Azure virtual machines use Azure Blob storage, which can have performance implications. Configuring storage spaces can help with performance. In addition, storage pools enable you to configure storage tiers, which are described in the next topic.

You can configure storage spaces and storage pools as part of an availability group, or on a single Azure virtual machine. Use Server Manager to configure storage pools, or Windows PowerShell.

Configure Storage Pools by Using Server Manager

1. Connect to your Azure virtual machine by using **Remote Desktop**.
2. Open **Server Manager**, navigate to **File and Storage Services**, navigate to **Volumes**, and then navigate to **Storage Pools**.

- Storage spaces and storage pools are a feature of Windows Server
- A number of physical disks are configured to create a virtual disk
- Storage spaces can be used with Azure virtual machines
- You can configure them as part of an availability group, or on a single virtual machine
- You can create a new storage pool by using the New Storage Pool Wizard

3. In **Storage Pools**, select **Primordial**. This represents the unassigned data disks.
4. Next to **Storage Pools**, select **Tasks**, and then choose **New Storage Pool**.
5. The New Storage Pool Wizard starts and displays the **Before you begin** screen. Click **Next**.
6. On the **Specify a storage pool name and subsystem** page, type a name, and then click **Next**.
7. On the **Select physical disks for the storage pool** page, select the disks that you want to include.
8. On the **Confirm Selection** page, click **Create**. When it has completed, click **Close**.
9. The new storage pool is displayed.

Storage Tiers

Storage tiers are a feature of storage spaces, and they enable you to optimize performance, for the most economical price. To use storage tiers, you must create a virtual hard disk that has a mix of hard disk drive (HDD) storage, and SSD storage. Two tiers are created: the SSD tier is used for data that is accessed frequently and the HDD tier is used for data that is accessed less frequently.

Storage tiers enable you to pay for less expensive HDD storage for data that is not being used, while getting the performance benefits of SSD storage for data that is current and accessed frequently.

- Storage tiers enable data to be silently moved between an SSD tier and an HDD tier
- Data is written to the HDD tier, and then moved between the tiers based on how frequently data is accessed
- Storage tiers provide performance improvements, whilst making use of inexpensive HDD storage
- Use Windows PowerShell or Server Manager

How Do Storage Tiers Work?

Data is normally written to HDD storage first, and then moved to the SSD tier if required. Data is moved silently between the tiers to optimize performance based on frequency of access.

How to Configure Storage Tiers

You can configure storage tiers either through Server Manager or by using Windows PowerShell. You must first have created a storage pool that uses both HDD storage and SSD storage. You can then create a virtual disk that has a storage tier for SSD storage, and then a storage tier for HDD storage.



Note: To configure storage tiers correctly, the operating system must recognize disks as either HDD or SSD. This sometimes requires running a Windows PowerShell script to ensure that they are categorized correctly. Storage tiers will not work if they are not correctly identified.

Use Windows PowerShell to create an SSD tier and an HDD tier.

New=StorageTier

```
New-StorageTier -StoragePoolFriendlyName TieredPool -FriendlyName MySSDTier -MediaType SSD
```

```
New-StorageTier -StoragePoolFriendlyName TieredPool -FriendlyName MyHDDTier -MediaType HDD
```

Demonstration: Creating a Storage Pool

In this demonstration, you will see how to:

- Create a storage pool.
- Change the media type of virtual disks.
- Create a virtual disk that has tiered storage.

Demonstration Steps

Create a Storage Pool

1. Start the MSL-TMG1, 20765B-MIA-DC and 20765B-MIA-SQL virtual machines, and log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Demofiles\Mod10** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and wait for the script to finish.
4. Open **Internet Explorer** and navigate to the **Azure portal** at **www.azure.portal.com**. Sign in with your Azure pass credentials.
5. Click **Resource Groups**, and then click **StorageSpacesDemo**.
6. From the list, click **VM1**.
7. From the **Essentials** group, copy the **Public IP address** onto the clipboard. If prompted, click **Allow Access**.
8. Change to the **Start** screen, type **Remote Desktop** and then click **Remote Desktop Connection**.
9. Connect to the Azure VM by pasting the IP address into the **Computer** box (delete everything after the IP address), and then click **Connect**.
10. When prompted, type the password **Pa\$\$w0rd**, and then click **OK**.
11. In the **Remote Desktop Connection** dialog box, click **Yes**. The **Server Manager** dashboard is displayed.
12. If the **Networks** pane appears, click **No**.
13. Click **File and Storage Services**, **Volumes**, and then **Storage Pools**.
14. In Storage Pools, click **Primordial**.
15. Next to Storage Pools, click **Tasks**, and then click **New Storage Pool**.
16. In the **New Storage Pool Wizard** dialog box, on the **Before you begin** page, click **Next**.
17. On the **Specify a storage pool name and subsystem** page, in the **Name** box, type **MyStoragePool**, and click **Next**.
18. On the **Select physical disks for the storage pool**, select all four disks, and click **Next**.
19. On the **Confirm Selections** page, click **Create**.
20. When it has completed, click **Close**.
21. Click **MyStoragePool** to see that it is made up of the four disks you selected.

Set the Media Type

1. The four disks added to the Storage Pool all show the media type **Unknown**. However, there are three HDDs and one SSD. We will now use a PowerShell script to assign the correct media type.
2. Minimize Server Manager.

3. On the Start menu, right-click **Windows PowerShell ISE**, point to **More**, and then click **Run as administrator**.
4. Click **OK** when prompted at the **User Account**.
5. From **D:\Demofiles\Mod10\Demo** copy the PowerShell script called **ChangeMediaType.ps1** to the Azure VM **C:** drive.
6. In Windows PowerShell ISE, on the **File** menu, click **Open**.
7. In the **Open** dialog box, browse to **C:**, click **ChangeMediaType**, and then click **Open**.
8. Use **Run Selection** to run the script under **# List the physical disks**. Note that the media type is unspecified.
9. Copy the **UniqueID** of the disk identified as **LUN 1** to the **\$uniqueIdPremium1** variable value (between "").
10. Copy the **UniqueID** of the disk identified as **LUN 2** to the **\$uniqueIdStandard1** variable value.
11. Copy the **UniqueID** of the disk identified as **LUN 3** to the **\$uniqueIdStandard2** variable value.
12. Copy the **UniqueID** of the disk identified as **LUN 4** to the **\$uniqueIdStandard3** variable value.
13. Once all the variables have a value, use **Run Selection** to run the script under **# Initialize variables**.
14. Use **Run Selection** to run the script under **# Set media type**.
15. To check the media types have been assigned correctly, run the script under **# List the physical disks**. You will see that the media types are now set.
16. Close Windows PowerShell ISE, without saving any changes.

Create a Virtual Disk with Tiered Storage

1. In Server Manager, in the top right corner, click **Tasks**, and then click **Refresh**. The Media type of the disks should now be shown correctly.
2. Next to Virtual Disks, click **Tasks**, and click **New Virtual Disk**.
3. In the **Select the storage pool** dialog box, click **MyStoragePool**, and then click **OK**.
4. In the **New Virtual Disk Wizard**, on the **Before you begin** page, click **Next**.
5. On the **Specify the virtual disk name**, in the **Name** box, type **MyVirtualDisk**.
6. Select the **Create Storage tiers on this virtual disk** check box, and then click **Next**.
7. On the **Specify enclosure resiliency** page, click **Next**.
8. On the **Select the storage layout** page, click **Simple**, and then click **Next**.
9. On the **Specify the provisioning type** page, click **Fixed**, and then click **Next**.
10. On the **Specify the size of the virtual disk** page, in the **Faster Tier** box, type **85**, and in the **Standard Tier** box, type **180**, and then click **Next**.
11. On the **Confirm selections** page, click **Create**. The tiered virtual disk will be created.
12. On the **View results** page, click **Close** to finish. You can then create a drive letter.
13. In the **New Volume Wizard**, on the **Before you begin** page, click **Next**.
14. On the **Select the server and disk** page, click **MyVirtualDisk**, and then click **Next**.
15. On the **Specify the size of the volume** page, click **Next**.
16. On the **Assign to a drive letter or folder** page, click **Next**.

17. On the **Select file system settings** page, click **Next**.
18. On the **Confirm selections** page, click **Create**.
19. On the **Completion** page, click **Close**. You can now see the new virtual disk you created.
20. Close Server Manager.
21. Disconnect from the Azure VM.

Check Your Knowledge

Question	
What types of storage does Azure not support?	
Select the correct answer.	
☐	NoSQL table data
☐	Queue storage
☐	Hierarchical storage
☐	File storage
☐	BLOB storage

Lesson 3

Azure Automation

In this lesson, you will be introduced to Azure features that enable you to automate tasks. This lesson includes an introduction to Windows PowerShell and the Azure SQL Database PowerShell cmdlets. You will be introduced to Azure Automation, including creating an Azure Automation account and using runbooks to automate tasks.

Lesson Objectives

After completing this lesson, you will be able to:

- Understand what Azure Automation is, and how you can use it.
- Set up an Azure Automation account.
- Use Windows PowerShell to set up Azure resources.
- Use Azure Automation to manage Azure resources.

What Is Azure Automation?

Azure Automation enables you to automate tasks in Azure by using Windows PowerShell. This saves time and reduces errors. You can test scripts before deployment and you can correct any problems before scripts are used in a production environment. Azure Automation uses Windows PowerShell scripts. These are either existing scripts, or new ones that have been created to set up workflows in Windows PowerShell Workflow. By automating tasks, people have more time to focus on value-added, nonrepetitive work. Azure Automation enables you to integrate different systems within Azure, such as monitoring systems and deployment. You can use Azure Automation with either on-premises systems or Azure resources.

- Azure Automation enables you to automate tasks by using Windows PowerShell scripts
- Azure Automation is supported by:
 - Azure SQL Database
 - Azure storage
 - Azure virtual machines
 - Azure cloud services
 - Azure websites
 - Azure backups
- Create an Azure Automation account
- An Azure Automation Runbooks is a Windows PowerShell workflow
- There are sample runbooks in the Runbook Gallery

What Should Be Automated?

Any task that is repeated, whether once or many times, or is likely to be repeated, is a candidate for Azure Automation. Complex tasks that may not necessarily be repeated, but need to be executed without error are also candidates. Azure Automation is supported by:

- Azure SQL Database
- Azure storage
- Azure virtual machines
- Azure cloud services
- Azure websites
- Azure backups

Azure Automation Runbooks

An Azure Automation runbook is a Windows PowerShell workflow. A Windows PowerShell workflow consists of several steps that constitute a useful task. Workflows make use of Windows PowerShell functions and parameters to create scripts. There are three types of Azure Automation runbooks:

- **Windows PowerShell.** This type of runbook consists of Windows PowerShell text-based scripts.
- **Windows PowerShell Workflow.** This type of runbook consists of text-based workflows in Windows PowerShell Workflow.
- **Graphical.** This type of runbook is based on Windows PowerShell, but edited by using a graphical editor.

In the Azure Automation Runbook Gallery, each type of runbook has an appropriate icon. It is necessary to edit graphical runbooks in the Azure portal, and provide a visual representation of the tasks. Windows PowerShell and Windows PowerShell Workflow runbooks are both text-based, and can be created and edited either in the Azure portal, or by using a Windows PowerShell editor and then importing the runbook.

Desired State Configuration

Desired state configuration (DSC) enables you to automatically monitor and update the configuration of Azure or on-premises computing resources. DSC makes configuration management easier and less error-prone, particularly when you have several machines to manage. DSC uses Windows PowerShell Desired State Configuration, which includes cmdlets and resources that you can use to specify declaratively an environment configuration.

For example, you can use DSC to set or maintain:

- Registry settings.
- Starting or stopping processes, and services.
- Running other Windows PowerShell scripts.
- Changing configurations that do not match the desired configuration.

You import configuration files into an Azure Automation account.

Create an Azure Automation Account

You can add an Azure Automation account in the Azure portal. This is the umbrella account for all of your Azure Automation objects:

1. On the Dashboard, click **New** (or **Add**).
2. On the **New** blade, click **Monitoring + Management**.
3. On the **Monitoring + Management** blade, click **Automation**.
4. On the **Add Automation Account** blade, in the **Name** box, type a unique name that uses lowercase characters and numbers. A green tick appears when an acceptable name has been entered.
5. In the **Resource group** box, either create a new resource group or select an existing one from the drop-down box.
6. In the **Location** box, select a suitable Microsoft data center.
7. Select whether you want to create an Azure Run As account. You must be a member of the admin role to create Azure Run As accounts. Azure Run As accounts authenticate runbooks in Azure Resource Manager or Azure Service Management.
8. Click **Create**. The Azure Automation account takes a short while to be created.

When you have created an Azure Automation account, you can view the Runbook Gallery and amend or create runbooks and jobs.

Azure SQL Database PowerShell Cmdlets

Windows PowerShell is a powerful scripting language that you use to perform a wide variety of Windows system administration tasks, including working with Azure. Windows PowerShell is object-oriented and built on the .NET Framework. It consists of cmdlets that are organized into modules.

The latest version, Windows PowerShell 5.0, is available to download with the Windows Management Framework. Windows PowerShell 5.0 is installed by default on Windows Server 2016 and Windows 10. It has several new features, including keywords that help you to develop using classes and a new structured information stream to pass structured data from a script to its caller or hosting environment. There are many new cmdlets and modules. In addition, there are bug fixes and performance improvements.

For information about downloading Windows Management Framework 5.0, go to the Microsoft Download Center:



Windows Management Framework 5.0

<https://aka.ms/m5y8it>

- Windows PowerShell is a scripting language
 - It uses cmdlets and aliases
 - It may have one or more parameters
 - Cmdlets are in modules
- Create an Azure server and Azure SQL Database
 - Add-AzureRMAccount
 - Login-AzureRmAccount
 - New-AzureRmResourceGroup
 - Get-AzureRmResourceProvider
 - New-AzureRmSqlServer
 - New-AzureRmSqlServerFirewallRule
 - New-AzureRmSqlDatabase

Cmdlets and Aliases

Windows PowerShell uses cmdlets that have a consistent format in their verb-noun syntax. For example, to call up help, type **Get-Help**. The exception to the verb-noun syntax is the alias. An alias can be thought of as a shortcut to a cmdlet. For example, you can use **dir** as a shortcut for **Get-Children** and **cd** as a shortcut for **Set-Location**. These aliases are commands that are familiar to users of different systems, such as MS-DOS or Unix, but have been incorporated into Windows PowerShell to make life simpler.

To find out more information about an alias, type **Get-Help** followed by the alias—for example, **Get-Help cls**.

Parameters

Cmdlets may have one or more parameters, to affect the way in which the cmdlet behaves. The parameter may be a direct input, or piped from another cmdlet.

There are different types of parameters:

- **Named or positional.** A named parameter is where you type the parameter after the cmdlet—for example, **Get-Command -All**—to list all of the installed cmdlets. A positional parameter is defined by its position. The default parameter type for cmdlets is the named parameter.
- **Mandatory or optional.** By default, Windows PowerShell cmdlets take optional parameters. You can also write your own Windows PowerShell cmdlets and, if necessary, define a parameter as mandatory.
- **Switch parameters.** Switch parameters are either on or off—that is, if you do not provide a parameter, the value is automatically set to false. An example of a switch parameter is **Get-Help -Full**.

Azure SQL Database Cmdlets

Several cmdlets are designed to help automate tasks that are associated with Azure SQL Databases. There are cmdlets to create new items such as servers and databases. The cmdlets are divided into groups with familiar Windows PowerShell verbs:

- **Get-**
- **New-**
- **Remove-**
- **Set-**
- **Start-**
- **Stop-**
- **Suspend-**
- **Use-**

For a list of Azure SQL Database cmdlets, see *Azure SQL Database Cmdlets* on MSDN:



Azure SQL Database Cmdlets

<http://aka.ms/Uwm0dm>

Creating a Server in Azure

As an example, let's consider how to create a new server in Azure by using Windows PowerShell. You must have already set up an Azure subscription. Open the Windows PowerShell ISE as administrator.

Add your Azure account and sign in.

Add account and sign in.

```
Add-AzureRMAccount
Login-AzureRMAccount
```

To create a new server, use the **New-AzureRmSqlServer** cmdlet. When you create a new server on Azure, you must specify the data center and the resource group that you want to use. The resource group is a way of keeping resources together. You must create the resource group in the same location in which you want to host your database in Azure SQL Database.

When you create a server, you must also provide a valid data center name. You can use the **Get-AzureRmResourceProvider** cmdlet to list the available data centers. To return the information that you need, you must limit the results to only data centers that host Azure SQL Database. This is done by using a pipe to return only **Microsoft.Sql** locations.

List the Azure data centers that support Azure SQL Database.

Get-AzureRmResourceProvider

```
(Get-AzureRmResourceProvider -ListAvailable | Where-Object {$_.ProviderNamespace -eq
'Microsoft.Sql'}).Locations
```

You must also specify a specific subscription by using the **Select-AzureRmSubscription** cmdlet. To create a new resource group, use the **New-AzureRmResourceGroup** cmdlet.

Specify the correct subscription and create a new resource group.

Select-AzureRmSubscription and New-AzureRmResourceGroup

```
Select-AzureRmSubscription -SubscriptionID 3bcd6ecd-e7d0-4156-8f40-dfc2c95c4cbd
New-AzureRmResourceGroup -Name "PowerShellTest" -Location "West Europe"
```

After you have created the resource group, you can create the server by using the **New-AzureRmServer** cmdlet. The server name must be in lowercase characters and unique.

Create a new Azure server.

New-AzureRmSqlServer

```
New-AzureRmSqlServer -ResourceGroupName "PowerShellTest" -ServerName "mytest2017ce" -
Location "West Europe" -ServerVersion "12.0"
```

You will be prompted to enter your user name and password.

To access the server, you need to create firewall rules. This limits the IP addresses that can access your Azure server. You can either specify a range of IP addresses or make the **StartIpAddress** value the same as the **EndIpAddress** value. Use the **New-AzureRmSqlServerFirewallRule** cmdlet.

Create a named firewall rule that has a range of IP addresses.

New-AzureRmSqlServerFirewallRule

```
New-AzureRmSqlServerFirewallRule -ResourceGroupName "PowerShellTest" -ServerName
"mytest2017ce" -FirewallRuleName "myFirewallRule" -StartIpAddress "nnn.nnn.nnn.nnn" -
EndIpAddress "nnn.nnn.nnn.nnn"
```

Finally, we can create an Azure SQL Database on the Azure server. Use the **New-AzureRmSqlDatabase** cmdlet.

Create a new Azure SQL Database.

New-AzureRmSqlDatabase

```
New-AzureRmSqlDatabase -ResourceGroupName "PowerShellTest" -ServerName "mytest2017ce" -
DatabaseName "mytestdb" -Edition Standard -RequestedServiceObjectiveName "S1"
```

This example has shown you the Windows PowerShell cmdlets that are used to create a new Azure server and a new Azure SQL Database. There are many more cmdlets in the Azure SQL Database PowerShell module.

The Azure Automation Runbook Gallery

When you create an Azure Automation account, you get access to the Azure Automation Runbook Gallery.

Viewing the Runbook Gallery

On the **Automation Account** blade, under **Process Automation**, click **Runbooks Gallery**. This displays various runbooks that Microsoft and third-party organizations have created. You can filter by different criteria:

- After you have created an Azure Automation account, you can view the Runbook Gallery
- Microsoft scripts are reviewed and are good examples of how to automate tasks
- Community contributed runbooks share knowledge of how to automate a task
 - They are not necessarily maintained
- Click **Import** to import the runbook to your account
- Edit it to suit your needs

- **Gallery source.** Select the Script Center or the Windows PowerShell gallery.
- **Type.** Select Windows PowerShell scripts, graphical runbooks, and Windows PowerShell workflows.
- **Publisher.** Select Microsoft and/or community.

Click **OK** after you have selected the filters that you require.

In the Runbook Gallery, you will find sample runbooks that perform various tasks such as starting Azure V2 or classic virtual machines, stopping Azure V2 or classic virtual machines, and connecting to an Azure virtual machine, in addition to several educational runbooks that you can use to familiarize yourself with Azure Automation. The runbooks that Microsoft has created have gone through a review process with the Azure Automation product team, so you can be sure that they are good examples of automation.



Note: Runbooks that have been contributed by the community are not necessarily supported, and there is no requirement for them to be maintained to use the latest cmdlets. They do, however, share how people have successfully automated tasks.

To see what a runbook does, click the title to see a description and a process flow diagram. Each runbook is rated and a count of the number of downloads is displayed, together with when the runbook was last updated. There are three different types of runbooks:

- Windows PowerShell script
- Graphical runbook
- Windows PowerShell workflow

How to Use a Runbook from the Gallery

When you find a runbook that you want to use, click the **Import** tab, and then on the **Import** blade, click **OK**. The runbook is imported into your Azure Automation account. To view the imported runbook:

- Close the **Runbook Gallery** blades.
- On the **Automation Account** blade, under **Process Automation**, click **Runbooks**. A list of all of the runbooks is displayed.
- Alternatively, on the Azure Automation account dashboard, under **Resources**, click **Runbooks**.
- Click the runbook that you imported, and then click **Edit**. The **Edit** blade appears.
- Click to edit it.

Demonstration: Introducing Azure Automation

In this demonstration, you will see how to:

- Create an Azure Automation account.
- Browse the Runbook Gallery.
- Import and edit a simple runbook.

Demonstration Steps

1. Start the MSL-TMG1, 20765B-MIA-DC and 20765B-MIA-SQL virtual machines, and log on to **20765B-MIA-SQL** as **AdventureWorks\Student** with the password **Pa\$\$w0rd**.
2. Using Internet Explorer, navigate to **https://portal.azure.com**.
3. Login using your Azure Pass credentials.
4. At the dashboard, click **New**.
5. On the **New** blade, click **Monitoring + Management**.
6. On the **Monitoring + Management** blade, click **Automation**.
7. On the **Add Automation Account** blade, in the **Name** box, type **automate + your initials**. The name must be in lowercase characters and numbers. A green tick appears when the name is acceptable. If necessary, add additional numbers to create a unique name.
8. In the **Resource group** box, either create a new resource group, or select the resource group created in the first demo.
9. In the **Location** box, select a Microsoft data center close to you.
10. For **Create Azure Run As account**, click **Yes**.
11. Click **Create**. The Automation Account takes a short while to be created. A message is displayed when the account has been created.
12. Click **All resources** to see that the new Automation account has been created.
13. Click on the new Automation account name as named in step 7. The **Automation Account** blade is displayed.
14. Click **Runbooks Gallery**.
15. Point out the options to filter the Runbook Gallery, including **Gallery Source**, **Type**, and **Publisher**, and then click **OK**.
16. Point out the icon for **Graphical** runbooks.
17. Point out the icon for **PowerShell** runbooks.
18. Point out the icon for **PowerShell Workflow** runbooks.
19. Click the **Hello World for Azure Automation** runbook (PowerShell Workflow Runbook).
20. Examine the script, and then click **Import**.
21. On the **Import** blade, click **OK**. The runbook is imported into your account.
22. Click **Edit** to edit the code.
23. On line 33, overwrite **World** with your name, and then click **Save**.
24. Click **Publish**, and when prompted, click **Yes**.
25. Click **Start**.

26. In the **Start Runbook** blade, click **OK**, the workflow runs in a test window.
27. Click **Output** to see the results.
28. Close each blade until you return to the dashboard.
29. Close Internet Explorer.

Sequencing Activity

Put the following steps in order by numbering each to indicate the correct order.

	Steps
	Create an Azure Automation account
	Browse the Runbook Gallery
	Select a runbook
	Import the runbook
	Edit the runbook
	Publish the runbook
	Test the runbook
	Schedule the runbook
	Check whether the task has been completed

Lab: Managing Databases in the Cloud

Scenario

As the database administrator for the Adventure Works Bicycle Company, you are investigating how to improve data protection. You decide to try both dynamic data masking and testing a Microsoft® Azure Automation runbook.

Objectives

After completing this lab, you will be able to:

- Apply different data masks to obfuscate data.
- Test an Azure Automation runbook.

Estimated Time: 45 minutes

Virtual machine: **20765B-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa\$\$w0rd**

Exercise 1: Add Data Masking

Scenario

As part of a drive to improve security at the Adventure Works Bicycle Company, you have been asked to add data masks to several columns that contain personal information.

The main tasks for this exercise are as follows:

1. Create an AdventureWorksLT SQL Database in Azure
2. Examine the AdventureWorks Database
3. Create the Data Masks
4. Test Data Masking

► Task 1: Create an AdventureWorksLT SQL Database in Azure

1. Open the Azure portal at **<https://portal.azure.com/>** by using your Azure pass or Microsoft account.
2. Create a new SQL Database.
3. Name the new database **AdventureWorksLT**.
4. Create the database by using the **Sample AdventureWorksLT** database.
5. Create a new resource group and server by using unique names. Make a note of the server name. Alternatively, use the resource group and server that you created in previous lab exercises.
6. Create a server admin login with the name **Student** and a password of **Pa\$\$w0rd**.
7. Do not create a **SQL elastic pool**.
8. Do not change the details in **Pricing tier** or **Collation**.
9. Click **Create**.
10. Add the current client IP address to the server firewall.
11. Leave the Azure portal open for the next lab exercise.

► **Task 2: Examine the AdventureWorks Database**

1. Open SQL Server Management Studio, and connect to the database you created in Azure in the last exercise. Refer to your note of the server name. For example, **20765Bab.database.windows.net**, where **ab** are your initials. Alternatively,
2. Login with the credentials **Student** and **Pa\$\$w0rd**.
3. Change the current database to **AdventureWorksLT**.
4. Open the sql script **D:\Labfiles\Lab10\Starter\Data Masking.sql**.
5. Run each section of the script in turn. Note that all the columns are displayed unmasked.
6. Leave SQL Server Management Studio open for the next lab exercise.

► **Task 3: Create the Data Masks**

1. If necessary, sign in to the Azure portal.
2. Select the **AdventureWorksLT** database that you created in the last task.
3. Select **Dynamic Data Masking**.
4. Set a data mask on the following fields:
 - a. **SalesLT.Customer.LastName**—custom mask showing the first two characters.
 - b. **SalesLT.Customer.EmailAddress**—email mask
5. Save the settings.

► **Task 4: Test Data Masking**

1. Open SQL Server Management Studio, and then open the script called **Check Masking.sql** in the **D:\Labfiles\Lab10\Starter** folder.
2. Check that the **AdventureWorksLT** database is selected.
3. Run the portion of the script marked **Execute as Test User**.
4. Run the portion of the script marked **Test Dynamic Data Masking**. Check that the two columns have been masked correctly.
5. Run the portion of the script marked **Revert**.
6. Run the portion of the script marked **Test as Admin**. Note that the columns appear unmasked.
7. Close the script without saving.
8. Close SQL Server Management Studio and the Azure portal window.

Results: After this exercise, you will have added data masks to columns that contain sensitive data and verified that the data is masked to unauthorized users.

Exercise 2: Use Azure Automation to Stop Virtual Machines

Scenario

The Adventure Works Bicycle Company has an Azure virtual machine. As part of a project to improve security, you want to implement Azure Automation to manage this virtual machine.

The main tasks for this exercise are as follows:

1. Create a Virtual Machine by Using Windows PowerShell
2. Create an Azure Automation Account
3. Stop the Virtual Machine by Using Azure Automation
4. Start the Virtual Machine by Using a Schedule in Azure Automation
5. List Resources by Using Azure Automation

► Task 1: Create a Virtual Machine by Using Windows PowerShell

1. Start the MSL-TMG1, 20765B-MIA-DC and 20765B-MIA-SQL virtual machines, and log on to **20765B-MIA-SQL** as **AdventureWorks\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Labfiles\Lab10\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and wait for the script to finish.
4. Open the **Azure portal** using your Azure pass credentials.
5. Start **Windows PowerShell ISE** as **Administrator**.
6. Click **Yes** to the User Control dialog box.
7. In the **PowerShell ISE**, open the file **D:\Labfiles\Mod10\Starter\CreateResources.ps1**.
8. In the Initialize section of the script, amend the **\$subscriptionName** and **\$tenantID** variables to those associated with your subscription. (**TenantID** is the same as the **Azure Active Directory DirectoryID**).
9. Run the script titled **# Initialize**.
10. Sign in to Azure by running the script under **# Sign in to Azure**.
11. Run the script **# Check name**. If the storage account name is not available, change the name in the **# Initialize** section and rerun both scripts.
12. Run the script **# Prompt for credentials** and when prompted, enter **Pa\$\$w0rd** in both dialog boxes.
13. Run the script under **# Create resource groups**.
14. Run the script under **# Create storage account**.
15. Run the script under **# Create virtual network**.
16. Run the script under **# Create public IP**.
17. Run the script under **# Create virtual machine**.
18. Run the script under **# Create and configure SQL Server**.
19. Run the script under **# Create SQL database**.
20. Close the **PowerShell ISE**.
21. Leave the **Azure portal** open for the next lab exercise.

► Task 2: Create an Azure Automation Account

1. In the Azure portal, if necessary, sign in by using your Azure pass credentials.
2. On the dashboard, create a new Azure Automation account named **automate + your initials**.
3. Check that the Azure Automation account has been created.
4. Leave the Azure portal open.

► Task 3: Stop the Virtual Machine by Using Azure Automation

1. If necessary, sign in to the Azure portal.
2. Open the Azure Automation account that you have just created.
3. Locate, and then click the **Stop Azure V2 VMs** runbook.
4. Import the **Stop Azure V2 VMs** runbook into your Azure Automation account.
5. Publish the **Stop Azure V2 VMs** runbook.
6. Start the **Stop Azure V2 VMs** runbook.
7. Select **Output** to view the results. Note the erroneous error message.
8. Verify that the virtual machine has been stopped by viewing the properties of **VM1**.
9. Note that the status of **VM1** is **Stopped (deallocated)**.

► Task 4: Start the Virtual Machine by Using a Schedule in Azure Automation

1. If necessary, sign in to the Azure portal.
2. Locate, and then click the **Start Azure V2 VMs** runbook.
3. Import the **Start Azure V2 VMs** runbook into your Azure Automation account.
4. Publish the **Start Azure V2 VMs** runbook.
5. On the **Start Azure V2 VMs** blade, create a schedule for the runbook to run six minutes from now.
6. Configure parameters and run settings with the correct resource group and virtual machine name.
7. Confirm that your schedule has been created.
8. At the appropriate time, view the job.
9. Note that the status of **VM1** is **Running**.

► Task 5: List Resources by Using Azure Automation

1. If necessary, sign in to the Azure portal.
2. Select the Azure Automation account that you created in the previous lab exercise.
3. Select the **AzureAutomationTutorial**, and then examine the contents.
4. Publish it, and then start the **AzureAutomationTutorial**.
5. Click **Output** to see the list of resources.
6. Close the Azure portal when you have finished.

Results: After this exercise, you will have understood how a Windows PowerShell® script is used to create Azure resources, created an Azure Automation account, and used Azure Automation to stop a virtual machine.

Question: What are the different types of dynamic data masking? When might you use each one?

Question: What are the benefits of using Azure Automation?

Module Review and Takeaways

In this module, you have learned about three aspects of managing databases in the cloud:

- Security in Azure SQL Database
- Azure storage
- Azure Automation



Best Practice: Whether you hold your data on-premises or in Azure, carry out a threat analysis. This will help you to identify where data protection is weakest, and which features might help you to mitigate the risks that you have identified.

Azure offers significant benefits for managing data, including security, geo-replication, anywhere access, automation, and unlimited storage. This module has introduced three important aspects of moving data to Azure.

Review Question(s)

Question: What are the main data security concerns in your organization? Which features of Azure SQL Database are most appropriate to mitigate those concerns?

Course Evaluation

Course Evaluation

- Your evaluation of this course will help Microsoft understand the quality of your learning experience.
- Please work with your training provider to access the course evaluation form.
- Microsoft will keep your answers to this survey private and confidential and will use your responses to improve your future learning experience. Your open and honest feedback is valuable and appreciated.

Your evaluation of this course will help Microsoft understand the quality of your learning experience.

Please work with your training provider to access the course evaluation form.

Microsoft will keep your answers to this survey private and confidential and will use your responses to improve your future learning experience. Your open and honest feedback is valuable and appreciated.

Module 2: Installing SQL Server 2016

Lab: Installing SQL Server 2016

Exercise 1: Preparing to Install SQL Server

► Task 1: Prepare the Lab Environment

1. Ensure that the MSL-TMG1, 20765B-MIA-DC, and 20765B-MIA-SQL virtual machines are running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Labfiles\Lab02\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.

► Task 2: View Hardware and Software Requirements

1. In the **X:** folder, double-click **setup.exe**.
2. In the **User Account Control** dialog box, click **Yes**.
3. In the SQL Server Installation Center, on the **Planning** page, click **Hardware and Software Requirements**.
4. In Internet Explorer, note that the documentation provides detailed information about hardware and software requirements for SQL Server 2016. Close Internet Explorer.

► Task 3: Run the System Configuration Checker

1. In the SQL Server Installation Center, on the **Tools** page, click **System Configuration Checker**, and wait for the tool to start.
2. When the tool has run, review the checks that were performed. (If the checks are not visible, click **Show details**.)
3. Click **OK** to close SQL Server 2016 Setup.
4. Keep the SQL Server Installation Center window open. You will use it again in a later exercise.

Results: After this exercise, you should have run the SQL Server setup program and used the tools in the SQL Server Installation Center to assess the computer's readiness for SQL Server installation.

Exercise 2: Installing SQL Server

► Task 1: Install the SQL Server Instance

1. In the SQL Server Installation Center window, on the **Installation** page, click **New SQL Server stand-alone installation or add features to an existing installation** and wait for SQL Server setup to start.
2. If the **Microsoft Updates** or **Product Updates** pages are displayed, clear any check boxes and click **Next**.
3. On the **Install Rules** page, note that the list of rules has been checked. If the list of checks is not shown, click **Show Details**. If a warning about Windows Firewall is displayed, you can continue.

4. On the **Install Rules** page, click **Next**.
5. On the **Installation Type** page, ensure that **Perform a new installation of SQL Server 2016** is selected, and then click **Next**.
6. On the **Product Key** page, in the **Specify a free edition** box, select **Evaluation**, and then click **Next**.
7. On the **License Terms** page, note the Microsoft Software License Terms, select **I accept the license terms**, and then click **Next**.
8. On the **Feature Selection** page, select **Database Engine Services**, and then click **Next**.
9. On the **Instance Configuration** page, ensure that **Named instance** is selected, in the **Named instance** box, type **SQLTEST**, and then click **Next**.
10. On the **Server Configuration** page, on the **SQL Server Agent** and **SQL Server Database Engine** rows, enter the following values:
 - **Account Name:** ADVENTUREWORKS\ServiceAcct
 - **Password:** Pa\$\$w0rd
 - **Startup Type:** Manual
11. On the **Collation** tab, ensure that **SQL_Latin1_General_CP1_CI_AS** is selected and click **Next**.
12. On the **Database Engine Configuration** page, on the **Server Configuration** tab, in the **Authentication Mode** section, select **Mixed Mode (SQL Server authentication and Windows authentication)**. Enter and confirm the password, **Pa\$\$w0rd**.
13. Click **Add Current User**; this will add the user **ADVENTUREWORKS\Student (Student)** to the list of Administrators.
14. On the **Data Directories** tab, change the **User database directory** to **M:\SQLTEST\Data**.
15. Change the **User database log directory** to **L:\SQLTEST\Logs**.
16. On the **TempDB** tab, review the default values that have been selected for **tempdb** data files.
17. On the **FILESTREAM** tab, ensure that **Enable FILESTREAM for Transact-SQL access** is not selected, and then click **Next**.
18. On the **Ready to Install** page, review the summary, then click **Install** and wait for the installation to complete.
19. On the **Complete** page, click **Close**.
20. Close the SQL Server Installation Center window.

Results: After this exercise, you should have installed an instance of SQL Server.

Exercise 3: Perform Post-Installation Checks

► Task 1: Start the SQL Server Service

1. On the Start screen, type **SQL Server 2016 Configuration Manager**, and then click **SQL Server 2016 Configuration Manager**.
2. In the **User Account Control** dialog box, click **Yes**.
3. In the left-hand pane of the SQL Server Configuration Manager window, click **SQL Server Services**.
4. In the right-hand pane, double-click **SQL Server (SQLTEST)**.
5. In the **SQL Server (SQLTEST) Properties** dialog box, verify that the service is configured to log on as **ADVENTUREWORKS\ServiceAcct**, and then click **Start**. When the service has started, click **OK**.

► Task 2: Configure Network Protocols and Aliases

1. In SQL Server Configuration Manager, expand **SQL Server Network Configuration**, click **Protocols for SQLTEST**, and verify that the **TCP/IP** protocol is **Enabled** for this instance of SQL Server.
2. In SQL Server Configuration Manager, expand **SQL Native Client 11.0 Configuration (32bit)**, click **Client Protocols**, and verify that the **TCP/IP** protocol is **Enabled** for 32-bit client applications.
3. Click **Aliases**, and note that there are currently no aliases defined for 32-bit clients.
4. Right-click **Aliases** and click **New Alias**.
5. In the Alias - New window, in the **Alias Name** text box, type **Test**.
6. In the **Protocol** drop-down list box, click **TCP/IP**.
7. In the **Server** text box, type **MIA-SQL\SQLTEST** and click **OK**.
8. In SQL Server Configuration Manager, expand **SQL Native Client 11.0 Configuration**, click **Client Protocols**, and verify that the **TCP/IP** protocol is enabled for 64-bit client applications.
9. Click **Aliases**, and note that there are currently no aliases defined for 64-bit clients.
10. Right-click **Aliases** and click **New Alias**.
11. In the Alias - New window, in the **Alias Name** text box, type **Test**.
12. In the **Protocol** drop-down list box, click **TCP/IP**.
13. In the **Server** text box, type **MIA-SQL\SQLTEST** and click **OK**.
14. Close SQL Server Configuration Manager.

► Task 3: Verify Connectivity to SQL Server

1. Right-click the **Start** button and click **Command Prompt**.
2. At the command prompt, enter the following command to connect to the MIA-SQL\SQLTEST instance of SQL Server:

```
sqlcmd -S MIA-SQL\SQLTEST -E
```

3. At the sqlcmd prompt, enter the following command to display the SQL Server instance name, and then press ENTER:

```
SELECT @@ServerName;  
GO
```

4. Close the command prompt window.

5. Start SQL Server Management Studio, and when prompted, connect to the database engine named **Test** using Windows Authentication.
6. In Object Explorer, right-click **Test**, and then click **Properties**.
7. Verify that the value of the **Name** property is **MIA-SQL\SQLTEST** and click **Cancel**.
8. In Object Explorer, right-click **Test** and click **Stop**.
9. In the **User Account Control** dialog box, click **Yes**.
10. When prompted to confirm that you want to stop the MSSQL\$SQLTEST service, click **Yes**.
11. When the service has stopped, close SQL Server Management Studio.

Results: After this exercise, you should have started the SQL Server service and connected using SSMS.

Exercise 4: Automating Installation

► Task 1: Review an Unattended Installation File

1. On the taskbar, click the **File Explorer** shortcut.
2. In File Explorer, browse to **D:\Labfiles\Lab02\Starter**.
3. Double-click **ConfigurationFile.ini**. The file will open in Notepad.
4. Review the content in conjunction with the *Install SQL Server 2016 From the Command Prompt* topic in the SQL Server 2016 Technical Documentation. In particular, note the values of the following properties:
 - a. INSTANCEID
 - b. INSTANCENAME
 - c. ACTION
 - d. FEATURES
 - e. TCPENABLED
 - f. SQLUSERDBDIR
 - g. SQLUSERDBLOGDIR
 - h. SQLTEMPDBFILECOUNT
5. Leave the file open in Notepad.

► Task 2: Update an Unattended Installation File

1. Return to the **ConfigurationFile.ini** file open in Notepad.
2. Locate the **INSTANCENAME** parameter in the file. Edit so that its value is **SQLDEV**. The line should look like this:

```
INSTANCENAME="SQLDEV"
```

3. Locate the **INSTANCEID** parameter in the file. Edit so that its value is **SQLDEV**. The line should look like this:

```
INSTANCEID="SQLDEV"
```

4. Locate the **TCPENABLED** parameter in the file. Edit so that its value is **0**. The line should look like this:

```
TCPENABLED="0"
```

5. Locate the **SQLUSERDBDIR** parameter in the file. Edit so that its value is **C:\devdb**. The line should look like this:

```
SQLUSERDBDIR="C:\devdb"
```

6. Locate the **SQLUSERDBLOGDIR** parameter in the file. Edit so that its value is **C:\devdb**. The line should look like this:

```
SQLUSERDBLOGDIR="C:\devdb"
```

7. Locate the **SQLTEMPDBFILECOUNT** parameter in the file. Edit so that its value is **2**. The line should look like this:

```
SQLTEMPDBFILECOUNT="2"
```

8. Save the file and then close Notepad.

Results: After this exercise, you will have reviewed and edited an unattended installation configuration file.

Module 3: Upgrading SQL Server to SQL Server 2016

Lab: Upgrading SQL Server

Exercise 1: Create the Application Logins

► Task 1: Prepare the Lab Environment

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are both running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Labfiles\Lab03\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, wait for the script to finish, and then press any key.

► Task 2: Create the appuser Login

1. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
2. In Object Explorer, under the **MIA-SQL** server node, expand **Security**.
3. Right-click **Logins**, then click **New Login**.
4. In the Login - New window, in the **Login name** box, type **appuser**.
5. Click **SQL Server authentication**, then in the **Password** and **Confirm password** boxes, type **Pa\$\$w0rd1**.
6. Clear **User must change password at next login**, then click **OK**.

► Task 3: Create the reportuser Login Using CREATE USER

1. In SQL Server Management Studio, on the **File** menu, point to **Open**, and click **Project/Solution**.
2. In the **Open Project** dialog box, browse to **D:\Labfiles\Lab03\Starter\Project**, and then double-click **Project.ssmssl**.
3. In Solution Explorer, expand **Queries**, and then double-click the query **Lab Exercise 01 - create login.sql**.
4. In the query window, highlight the statement **USE master**, and click **Execute**.
5. In the query pane, after the **Task 2** description, type the following query:

```
CREATE LOGIN [reportuser]
WITH      PASSWORD=<password_hash_value> HASHED,
          SID=<sid_value> ;
```

6. From the task description, copy and paste the password hash value (starting **0x02...**) over **<password_hash_value>**.
7. From the task description, copy and paste the SID value (starting **0x44...**) over **<sid_value>**.
8. Highlight the query and click **Execute**.

Results: After this exercise, you should be able to create a login using SSMS and the CREATE USER command.

Exercise 2: Restore the Backups of the TSQL Database

► Task 1: Restore the Full Backup

1. In SQL Server Management Studio, in Object Explorer, right-click **Databases**, and then click **Restore Database**.
2. On the **General** page, in the **Source** section, click **Device**, and then click the ellipsis (...) button.
3. In the **Select backup devices** dialog box, click **Add**.
4. In the **Locate Backup File - MIA-SQL** dialog box, browse to the **D:\Labfiles\Lab03\Starter** folder, click **TSQL1.bak**, and then click **OK**.
5. In the **Select backup devices** dialog box, click **OK**.
6. On the **Files** page, select **Relocate all files to folder**, in the **Data file folder** box, type **D:\Labfiles\Lab03**, and then in the **Log file folder** box, type **D:\Labfiles\Lab03**.
7. On the **Options** page, in the **Recovery state** list, click **RESTORE WITH NORECOVERY**, and then click **OK**.
8. In the **Microsoft SQL Server Management Studio** dialog box, click **OK**.

► Task 2: Restore the Transaction Log Backup

1. In SQL Server Management Studio, in Object Explorer, right-click **Databases**, and then click **Restore Files and Filegroups**.
2. On the **General** page, in the **To database** box, type **TSQL**.
3. In the **Source for restore** section, click **From device**, and then click the ellipsis (...) button.
4. In the **Select backup devices** dialog box, click **Add**.
5. In the **Locate Backup File - MIA-SQL** dialog box, browse to the **D:\Labfiles\Lab03\Starter** folder, click **TSQL1_trn1.trn**, and then click **OK**.
6. In the **Select backup devices** dialog box, click **OK**.
7. In the **Select the backup sets to restore** section, select the check box in the **Restore** column, and then click **OK**.
8. In the **Microsoft SQL Server Management Studio** dialog box, click **OK**.

► Task 3: Update Statistics

1. In SQL Server Management Studio, in Solution Explorer, double-click the query **Lab Exercise 02 - restore database.sql**.
2. In the query window, after the **Task 3** heading, type the following:

```
EXEC TSQL.sys.sp_updatestats;
```

3. Highlight the text you have typed and click **Execute**.

Results: At the end of this exercise, you should be able to:

Prepare for a migration by creating application logins.

Restore a database backup taken from one SQL Server instance and restore it to another.

Detect and repair orphaned users.

Exercise 3: Orphaned Users and Database Compatibility Level

► Task 1: Detect Orphaned Users

1. In SQL Server Management Studio, in Solution Explorer, double-click the query **Lab Exercise 03 - orphaned users.sql**.
2. In the query window, highlight the statement **USE TSQL**; and then click **Execute**.
3. After the **Task 1** description, type the following:

```
EXEC sp_change_users_login @Action = 'Report'
```

4. Highlight the query and click **Execute**.

► Task 2: Repair Orphaned Users

1. In the query pane, after the **Task 2** description, type the following query:

```
EXEC sp_change_users_login @Action = 'Update_One', @UserNamePattern = 'appuser1',  
@LoginName = 'appuser';
```

2. Highlight the query and click **Execute**.

► Task 3: Change Database Compatibility Level

1. In SQL Server Management Studio, in Object Explorer, expand the **Databases**, right-click **TSQL**, and click **Properties**.
2. In the Databases Properties - TSQL window, on the **Options** page, change the value of the **Compatibility level** box to **SQL Server 2016 (130)**, and then click **OK**.
3. Close SQL Server Management Studio, without saving any changes.

Results: After this lab, you should be able to:

Identify orphaned database users.

Repair orphaned database users.

Update the database compatibility level.

Module 4: Working with Databases

Lab: Managing Database Storage

Exercise 1: Configuring tempdb Storage

► Task 1: Prepare the Lab Environment

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are both running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab04\Starter** folder as **Administrator**.
3. In the **User Account Control** dialog box, click **Yes**.
4. Wait for the script to complete.

► Task 2: Configure tempdb Files

1. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
2. In Object Explorer, expand **Databases**, expand **System Databases**, right-click **tempdb**, and click **Properties**.
3. On the **Files** page, view the current file settings, and then click **Cancel**.
4. On the toolbar, click **New Query**.
5. Enter the following statements and click **Execute**:

```
USE master;
GO
ALTER DATABASE tempdb
MODIFY FILE (NAME = tempdev, SIZE = 10MB, FILEGROWTH = 5MB, FILENAME =
'T:\tempdb.mdf');
ALTER DATABASE tempdb
MODIFY FILE (NAME = templog, SIZE=5MB, FILEGROWTH = 1MB, FILENAME =
'T:\templog.ldf');
GO
```

6. In Object Explorer, right-click **MIA-SQL** and click **Restart**.
7. In the **User Account Control** dialog box, click **Yes**.
8. When prompted to allow changes, to restart the service, and to stop any dependent services, click **Yes**.
9. View the contents of **T:** and note that the **tempdb.mdf** and **templog.ldf** files have been moved to this location.
10. In SQL Server Management Studio, in Object Explorer, right-click **tempdb**, and click **Properties**.
11. On the **Files** page, verify that the file settings have been modified, and then click **Cancel**.
12. Save the script file as **Configure TempDB.sql** in the **D:\Labfiles\Lab04\Starter** folder.
13. Keep SQL Server Management Studio open for the next exercise.

Results: After this exercise, you should have inspected and configured the **tempdb** database.

Exercise 2: Creating Databases

► Task 1: Create the HumanResources Database

1. In SQL Server Management Studio, click **New Query**.
2. Enter the following statements and click **Execute**:

```
CREATE DATABASE HumanResources
ON PRIMARY
(NAME = 'HumanResources', FILENAME = 'M:\Data\HumanResources.mdf', SIZE = 50MB,
FILEGROWTH = 5MB)
LOG ON
(NAME = 'HumanResources_log', FILENAME = 'L:\Logs\HumanResources.ldf', SIZE = 5MB,
FILEGROWTH = 1MB);
GO
```

3. In Object Explorer, right-click the **Databases** folder, and then click **Refresh** to confirm that the **HumanResources** database has been created.
4. Save the script file as **Create HumanResources.sql** in the **D:\Labfiles\Lab04\Starter** folder.
5. Keep SQL Server Management Studio open for the next exercise.

► Task 2: Create the InternetSales Database

1. In SQL Server Management Studio, click **New Query**.
2. Enter the following statements and click **Execute**:

```
CREATE DATABASE InternetSales
ON PRIMARY
(NAME = 'InternetSales', FILENAME = 'M:\Data\InternetSales.mdf', SIZE = 5MB,
FILEGROWTH = 1MB),
FILEGROUP SalesData
(NAME = 'InternetSales_data1', FILENAME = 'M:\Data\InternetSales_data1.ndf', SIZE =
100MB, FILEGROWTH = 10MB ),
(NAME = 'InternetSales_data2', FILENAME = 'N:\Data\InternetSales_data2.ndf', SIZE =
100MB, FILEGROWTH = 10MB )
LOG ON
(NAME = 'InternetSales_log', FILENAME = 'L:\Logs\InternetSales.ldf', SIZE = 2MB,
FILEGROWTH = 10%);
GO
```

3. Under the existing code, enter the following statements, select the statements you have just added, and then click **Execute**:

```
ALTER DATABASE InternetSales
MODIFY FILEGROUP SalesData DEFAULT;
```

4. Save the script file as **Create InternetSales.sql** in the **D:\Labfiles\Lab04\Starter** folder.
5. Keep SQL Server Management Studio open for the next exercise.

► Task 3: View Data File Information

1. In SQL Server Management Studio, open the **ViewFileInfo.sql** script file in the **D:\Labfiles\Lab04\Starter** folder.
2. Select the code under the comment **View page usage** and click **Execute**. This query retrieves data about the files in the **InternetSales** database.
3. Note the **UsedPages** and **TotalPages** values for the **SalesData** filegroup.

4. Select the code under the comment **Create a table on the SalesData filegroup** and click **Execute**.
5. Select the code under the comment **Insert 10,000 rows** and click **Execute**.
6. Select the code under the comment **View page usage again** and click **Execute**.
7. Note the **UsedPages** value for the **SalesData** filegroup, and verify that the data in the table is spread across the files in the filegroup.
8. Keep SQL Server Management Studio open for the next exercise.

Results: After this exercise, you should have created a new **HumanResources** database and an **InternetSales** database that includes multiple filegroups.

Exercise 3: Attaching a Database

► Task 1: Attach the AWDataWarehouse Database

1. Using File Explorer, move **AWDataWarehouse.ldf** from the **D:\Labfiles\Lab04\Starter** folder to the **L:\Logs** folder.
2. Using File Explorer, move the following files from the **D:\Labfiles\Lab04\Starter** folder to the **M:\Data** folder:
 - **AWDataWarehouse.mdf**
 - **AWDataWarehouse_archive.ndf**
 - **AWDataWarehouse_current.ndf**
3. In SQL Server Management Studio, in Object Explorer, right-click **Databases** and click **Attach**.
4. In the **Attach Databases** dialog box, click **Add**.
5. In the **Locate Database Files - MIA-SQL** dialog box, in the **M:\Data** folder, select the **AWDataWarehouse.mdf** database file, and click **OK**.
6. In the **Attach Databases** dialog box, after you have added the **master** databases file, note that all of the database files are listed, and then click **OK**.
7. In Object Explorer, under **Databases**, verify that **AWDataWarehouse** is now listed.

► Task 2: Configure Filegroups

1. In Object Explorer, right-click the **AWDataWarehouse** database and click **Properties**.
2. On the **Filegroups** page, view the filegroups used by the database.
3. Select the **Read-Only** check box for the **Archive** filegroup and click **OK**.
4. In Object Explorer, expand **AWDataWarehouse**, expand **Tables**, right-click the **dbo.FactInternetSales** table and click **Properties**.
5. On the **Storage** page, verify that the **dbo.FactInternetSales** table is stored in the **Current** filegroup. Then click **Cancel**.
6. Right-click the **dbo.FactInternetSalesArchive** table and click **Properties**.
7. On the **Storage** page, verify that the **dbo.FactInternetSalesArchive** table is stored in the **Archive** filegroup, and then click **Cancel**.

8. In Object Explorer, right-click the **dbo.FactInternetSales** table and click **Edit Top 200 Rows**.
9. In the results output, change the **SalesAmount** value for the first record to **2500**, press Enter to update the record, and then close the **dbo.FactInternetSales** results.
10. In Object Explorer, right-click the **dbo.FactInternetSalesArchive** table and click **Edit Top 200 Rows**.
11. In the results output, change the **SalesAmount** value for the first record to **3500** and press Enter to update the record.
12. View the error message that is displayed, click **OK**, and then press Esc to cancel the update and close the **dbo.FactInternetSalesArchive** results.
13. Close SQL Server Management Studio without saving any changes.

Results: After this exercise, you should have attached the **AWDataWarehouse** database to MIA-SQL.

Module 5: Performing Database Maintenance

Lab: Performing Ongoing Database Maintenance

Exercise 1: Use DBCC CHECK to Verify Data Integrity

► Task 1: Prepare the Lab Environment

1. Ensure that the 20765B-MIA-DC and 20765B-MIA-SQL virtual machines are both running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Labfiles\Lab05\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes** to confirm that you want to run the command file, and wait for the script to finish.

► Task 2: Manage Database Integrity

1. Start SQL Server Management Studio, and connect to the **MIA-SQL** database engine using Windows authentication.
2. On the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, navigate to the **D:\Labfiles\Lab05\Starter\Ex1** folder, click **Ex1.ssmssln**, and then click **Open**.
4. In Solution Explorer, expand **Queries**, and double-click **Ex1-DBCC.sql**.
5. In the query pane, review the code under the comment -- **Update the Discontinued column for the Chai product**, select the code, and then click **Execute**.
6. In the query pane, review the code under the comment -- **Commit the transaction**, select the code, and then click **Execute**.
7. In the query pane, review the code under the comment -- **Simulate an automatic checkpoint**, select the code, and then click **Execute**.
8. In Solution Explorer, double-click **Ex1-Shutdown.sql**.
9. In the query pane, review the code under the comment -- **Open up a different connection and run this**, select the code, and then click **Execute**.
10. Close the **Ex1-Shutdown.sql** query pane without saving changes.
11. In Windows Explorer, navigate to the **D:\Labfiles\Lab05\Starter\Data** folder, right-click **CorruptDB_log**, and click **Rename**.
12. Type **CorruptDB_log_renamed** and press ENTER. This simulates losing the transaction log file in a disk failure.
13. Click **Start**, type **SQL Server Configuration**, and then click **SQL Server 2016 Configuration Manager**.
14. In the **User Account Control** dialog box, click **Yes**.
15. In SQL Server Configuration Manager, under **SQL Server Configuration Manager (Local)**, click **SQL Server Services**.
16. In the right-hand pane, right-click **SQL Server (MSSQLSERVER)**, and then click **Start**.

17. Right-click **SQL Server Agent (MSSQLSERVER)**, and then click **Start**.
18. Close SQL Server Configuration Manager.
19. In SQL Server Management Studio, in the **Ex1-DBCC.sql** query pane, review the code under the comment -- **Try to access the CorruptDB database**, select the code, and then click **Execute**. Note that the query causes an error.
20. Repeat Step 19, and note that the query causes an error.
21. In the query pane, review the code under the comment -- **Check the status of the database**, select the code, and then click **Execute**.
22. In the query pane, review the code under the comment -- **Confirm that the database is not online**, select the code, and then click **Execute**.
23. In Object Explorer, right-click **MIA-SQL**, click **Refresh**, and then expand **Databases**. Note that **CorruptDB** shows as **Recovery Pending**.
24. In the query pane, review the code under the comment -- **Use Emergency mode to review the data in the database**, select the code, and then click **Execute**.
25. In the query pane, review the code under the comment -- **Review the state of the discontinued products data**, select the code, and then click **Execute**. Note that the query shows zero products in stock, because the erroneous transaction was committed on disk and the transaction log file has been lost.
26. In the query pane, review the code under the comment -- **Set CorruptDB offline**, select the code, and then click **Execute**.
27. In Windows Explorer, navigate to the **D:\Labfiles\Lab05\Starter\Data** folder, right-click **CorruptDB_log_renamed**, and click **Rename**.
28. Type **CorruptDB_log** and press ENTER. This simulates replacing the lost transaction log file with a mirror copy.
29. In SQL Server Management Studio, in the query pane, review the code under the comment -- **After replacing the transaction log file, set CorruptDB back online**, select the code, and then click **Execute**.
30. In the query pane, review the code under the comment -- **Set the database in single user mode and use DBCC CHECKDB to repair the database**, select the code, and then click **Execute**.
31. In the query pane, review the code under the comment -- **Switch the database back into multi-user mode**, select the code, and then click **Execute**.
32. In the query pane, review the code under the comment -- **Check the data has returned to the pre-failure state**, select the code, and then click **Execute**. Note that the **Discontinued** column now shows that 77 products are in stock—the state that the data was in before the erroneous transaction was run.
33. Close the solution without saving changes, but leave SQL Server Management Studio open for the next exercise.

Results: After this exercise, you should have used DBCC CHECKDB to repair a corrupt database.

Exercise 2: Rebuild Indexes

► Task 1: Monitor and Maintain Indexes

1. In SQL Server Management Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, navigate to the **D:\Labfiles\Lab05\Starter\Ex2** folder, click **Ex2.ssmssl**, and then click **Open**.
3. In Solution Explorer, expand **Queries**, and then double-click **Ex2-Indexes.sql**.
4. In the query pane, review the code under the comment -- **View the statistics for index fragmentation on the Sales tables in the AdventureWorks database**, select the code, and then click **Execute**.
5. In the query pane, review the code under the comment -- **View the statistics for index fragmentation on the SalesOrderHeader table**, select the code, and then click **Execute**.
6. Note the results in the **Fragmentation Percent** column.
7. In the query pane, review the code under the comment -- **Insert an additional 10000 rows**, select the code, and then click **Execute**. Wait for the query to complete.
8. In the query pane, review the code under the comment -- **View the statistics for index fragmentation following the data insertion**, select the code, and then click **Execute**.
9. Note the results in the **Fragmentation Percent** column.
10. In the query pane, review the code under the comment -- **Rebuild the indexes**, select the code, and then click **Execute**.
11. In the query pane, review the code under the comment -- **View the statistics for index fragmentation following the index rebuild**, select the code, and then click **Execute**.
12. Note the results in the **Fragmentation Percent** column.
13. Close the solution without saving changes, but leave SQL Server Management Studio open for the next exercise.

Results: After this exercise, you should have rebuilt indexes on the **Sales.SalesOrderHeader** table, resulting in better performance.

Exercise 3: Create Database Maintenance Plans

► Task 1: Create a Database Backup Maintenance Plan

1. In SQL Server Management Studio, in Object Explorer, expand **Management**, right-click **Maintenance Plans**, and then click **Maintenance Plan Wizard**.
2. In the SQL Server Maintenance Plan Wizard, click **Next**.
3. On the **Select Plan Properties** page, in the **Name** box, type **Maintenance Plan for Backup of AdventureWorks Database**, select **Separate schedules for each task**, and then click **Next**.
4. On the **Select Maintenance Tasks** page, select the following tasks, and then click **Next**:
 - Back Up Database (Full)
 - Back Up Database (Differential)
 - Back Up Database (Transaction Log)

5. On the **Select Maintenance Task Order** page, click **Next**.
6. On the **Define Back Up Database (Full) Task** page, in the **Databases(s)** list, click **AdventureWorks**, and then click **OK** to close the drop-down list box.
7. Review the options on the **Destination** and **Options** tabs to see further changes possible.
8. On the **General** tab, in the **Schedule** section, click **Change**. Review the default schedule, and then click **OK**.
9. On the **Define Back Up Database (Full) Task** page, click **Next**.
10. On the **Define Back Up Database (Differential) Task** page, in the **Database(s)** list, select **AdventureWorks**, and then click **OK** to close the drop-down list box.
11. In the **Schedule** section, click **Change**.
12. In the **New Job Schedule** dialog box, in the **Frequency** section, in the **Occurs** drop-down list box, click **Daily**, and then click **OK**.
13. On the **Define Back Up Database (Differential) Task** page, click **Next**.
14. On the **Define Back Up Database (Transaction Log) Task** page, in the **Database(s)** list, select **AdventureWorks**, and then click **OK** to close the drop-down list box.
15. In the **Schedule** section, click **Change**.
16. In the **New Job Schedule** dialog box, in the **Frequency** section, in the **Occurs** drop-down list box, click **Daily**, in the **Daily frequency** section, select **Occurs every**, and then click **OK**.
17. On the **Define Back Up Database (Transaction Log) Task** page, click **Next**.
18. On the **Select Report Options** page, accept the default options, and then click **Next**.
19. On the **Complete the Wizard** page, click **Finish**. Wait for the operation to complete, and then click **Close**.

► **Task 2: Create a Database Integrity Check Maintenance Plan**

1. In Object Explorer, right-click **Maintenance Plans**, and then click **Maintenance Plan Wizard**.
2. In the **SQL Server Maintenance Plan Wizard**, click **Next**.
3. On the **Select Plan Properties** page, in the **Name** box, type **Maintenance Plan for Checking Integrity of the AdventureWorks Database**, and then click **Next**.
4. On the **Select Maintenance Tasks** page, select **Check Database Integrity** and click **Next**.
5. On the **Select Maintenance Task Order** page, click **Next**.
6. On the **Define Database Check Integrity Task** page, in the **Database(s)** list, click **AdventureWorks**, and then click **OK** to close the drop-down list box.
7. On the **Define Database Check Integrity Task** page, select the **Tablock** check box to minimize resource usage and maximize performance of the operations, and then click **Next**.
8. On the **Select Report Options** page, click **Next**.
9. On the **Complete the Wizard** page, click **Finish** to create the Maintenance Plan. Wait for the operation to complete, and then click **Close**.

► **Task 3: Run a Maintenance Plan**

1. In Object Explorer, expand **Maintenance Plans**, right-click **Maintenance Plan for Checking Integrity of the AdventureWorks Database**, and then click **Execute**.
2. In the **Execute Maintenance Plan** dialog box, wait until the maintenance plan succeeds, and then click **Close**.
3. Right-click **Maintenance Plan for Checking Integrity of the AdventureWorks Database**, and then click **View History**.
4. In the **Log File Viewer - MIA-SQL** dialog box, expand the **Date** value for the **Daily Maintenance** plan to see the individual task.
5. Review the data in the **Log file summary** section, and then click **Close**.
6. Close SQL Server Management Studio without saving changes.

Results: After this exercise, you should have created the required database maintenance plans.

Module 6: Database Storage Options

Lab: Implementing Stretch Database

Exercise 1: Run Stretch Database Advisor

► Task 1: Prepare the Lab Environment

1. Ensure that the 20765B-MIA-DC, 20765B-MIA-SQL and MSL-TMG1 virtual machines are running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Labfiles\Lab06\Starter** folder, right-click **Setup.cmd** and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and wait for the script to finish.

► Task 2: Run Stretch Database Advisor

1. On the **Start** page, type **SQL Server Upgrade**, and then click **Microsoft SQL Server 2016 Upgrade Advisor**.
2. On the **Scenarios** tab, click **Run Stretch Database Advisor**.
3. Click **Select Databases to Analyze** then in the **Server Name** box, type **localhost**, select the **Use Windows Authentication?** check box, and then click **Connect**.
4. In the list of databases, click **AdventureWorks**, and then click **Select**.
5. Click **Run** and wait for the analysis to complete.
6. In the **Analyzed databases** list, click **AdventureWorks**.
7. Examine the results, noting that the **Sales.OrderTracking** table is marked as **Ready for Stretch**.
8. Close SQL Server 2016 Upgrade Advisor without saving the results.

Results: After this exercise, you will know which tables within the Adventure Works database are eligible for Stretch Database.

Exercise 2: Implement Stretch Database

► Task 1: Enable Stretch Database for the SQL Server Instance

1. Start SQL Server Management Studio, and connect to the **MIA-SQL** SQL Server instance using **Windows Authentication**.
2. In SQL Server Management Studio, click **New Query**.
3. In the new query pane, type the following Transact-SQL, and then click **Execute** to enable Stretch Database for the SQL Server instance.

```
exec sp_configure N'remote data archive', N'1';  
reconfigure with override;
```

► Task 2: Enable Stretch Database

1. In Object Explorer, expand **Databases**, right-click **AdventureWorks**, point to **Tasks**, point to **Stretch**, and then click **Enable**.
2. In the **Enable Database for Stretch** wizard, on the **Introduction** page, click **Next**.
3. On the **Select tables** page, in the tables list, select the check box next to the table **OrderTracking**, and then click **Next**.
4. On the **Configure Azure** page, click **Sign In**.
5. On the **Sign in to your account** page, enter your Azure pass credentials, and then click **Sign in**.
6. On the **Configure Azure** page, in the **Microsoft Azure Sign In** section, in the **Select a subscription to use** box, select **Azure Pass**, in the **Select Azure region** choose an appropriate region.
7. In the **Select Azure server** section, ensure the **Create new server** option is selected.
8. In the **Server admin login** box, type **Student**, in the **Password** and **Confirm Password** boxes, type **Pa\$\$w0rd**, and then click **Next**.
9. On the **Secure credentials** page, in the **New Password** and **Confirm Password** boxes, type **Pa\$\$w0rd**, and then click **Next**.
10. On the **Select IP address** page, in the **From** and **To** box, type the IP addresses as provided by your instructor, and then click **Next**.
11. On the **Summary** page, review the details shown, and then click **Finish**. SQL Server will now configure Stretch Database for the OrderTracking table. This process may take several minutes.
12. Click **Close** to exit the Stretch Database wizard.

► Task 3: Monitor Stretch Database

1. In Object Explorer, right-click **AdventureWorks**, point to **Tasks**, point to **Stretch**, and then click **Monitor**.
2. In the **Stretch Database Monitor**, review the information.
3. Close SQL Server Management Studio without saving changes.

Results: After this exercise, you will have Stretch Database implemented for the OrderTracking table.

Module 7: Planning to Deploy SQL Server on Microsoft Azure

Lab: Migrating SQL Server by Using Azure

Exercise 1: Check Database Compatibility

► Task 1: Prepare the Lab Environment

1. Ensure that the MSL-TMG1, 20765B-MIA-DC, and 20765B-MIA-SQL VMs are both running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Labfiles\Lab07\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish, and then press any key to continue.

► Task 2: Run the Export Data-Tier Application Wizard to Check Database Compatibility

1. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
2. In the Object Explorer pane, expand the **Databases** node.
3. Right-click **salesapp1**, point to **Tasks**, then click **Export Data-tier Application**.
4. In the **Export Data-tier Application 'salesapp1' dialog box**, on the **Introduction** page, click **Next**.
5. On the **Export Settings** page, on the **Settings** tab, ensure that **Save to local disk** is selected. Enter the value **D:\Labfiles\Lab07\salesapp1.bacpac**.
6. On the **Advanced** tab, clear the **Select All** check box, and then click **Next**.
7. On the **Summary** page, verify the options you have selected, then click **Finish**. Wait for the test to complete.
8. On the **Results** page, examine the output of the process. Notice that the database has failed verification. Click any of the instances of **Error** in the **Result** column of the output to see details of the error.

The verification test reports a failure because of the syntax of the T-SQL statement in the **dbo.up_CrossDatabaseQuery** stored procedure, click **OK**.

9. On the **Results** page, click **Close**.

► Task 3: Resolve the Failure by Removing a Stored Procedure

1. In SSMS, click **New Query**.
2. Type the following, and then click **Execute**:

```
USE salesapp1;
GO
DROP PROCEDURE dbo.up_CrossDatabaseQuery;
GO
```

► Task 4: Rerun the Export Data-Tier Application Wizard to Check Database Compatibility

1. In Object Explorer, in the **Databases** node, right-click **salesapp1**, point to **Tasks**, then click **Export Data-tier Application**.
2. In the **Export Data-tier Application 'salesapp1'** dialog box, on the **Introduction** page, click **Next**.
3. On the **Export Settings** page, on the **Settings** tab, ensure that **Save to local disk** is selected. Enter the value **D:\Labfiles\Lab07\salesapp1.bacpac**.
4. On the **Advanced** tab, clear the **Select All** check box, and then click **Next**.
5. On the **Summary** page, verify the options you have selected, then click **Finish**. Wait for the test to complete.
6. On the **Results** page, examine the output of the process. Notice that the database has passed verification, and then click **Close**.

Results: After this exercise, you should have run the tools to check database compatibility with Azure from SQL Server Management Studio.

Exercise 2: Migrate a Database to Azure

► Task 1: Create an Instance of Azure SQL Server

1. Open **Internet Explorer** and go to **https://portal.azure.com/**.
2. Sign in to the Azure portal with your Azure Pass or Microsoft Account credentials.
3. Click **New**, then click **Databases**, and then click **SQL Database**.
4. On the **SQL Database** blade, in the **Name** box, type **empty**.
5. In the **Resource group** box, type a name for your resource group—this must be unique, so cannot be specified here. A suggested format is **sql2016-*<your initials>* *<one or more digits>***. For example, sql2016-js123. Keep a note of the name you have chosen.
6. Verify that **Select source** has the value **Blank database**.
7. Click **Server**.
8. On the **Server** blade, click **Create a new server**.
9. On the **New server** blade, in the **Server name** box type a name for your server—this must be unique throughout the whole Azure service, so cannot be specified here. A suggested format is **sql2016-*<your initials>* *<one or more digits>***. For example, sql2016-js123. Keep a note of the name you have chosen.
10. In the **Server admin login** box, type **salesappadmin**.
11. In the **Password** and **Confirm password** boxes, type **Pa\$\$w0rd**.
12. Under **Location**, select a region nearest your current geographical location, and then click **Select**.
13. On the **SQL Database** blade, click **Create**.
14. It will take some time for the new server and database to be created. The Azure portal will notify you when this step is finished.
15. Leave Internet Explorer open for the next task.

► Task 2: Configure the Azure Firewall

1. In the Azure portal, click **All resources**.
2. In the **All resources** pane, click the server name you created in the previous task.
3. In the server blade, click **Show firewall settings**.
4. In the **Firewall settings** blade, click **Add client IP**, and then click **Save**.
5. When the firewall changes are complete, click **OK**.
6. Leave Internet Explorer open.

► Task 3: Migrate the salesapp1 Database to Azure

1. In **SQL Server Management Studio**, in Object Explorer, expand **Databases**, right-click **salesapp1**, point to **Tasks**, and then click **Deploy Database to Microsoft Azure SQL Database**.
2. In the **Deploy Database 'salesapp1'** dialog box, on the **Introduction** page, click **Next**.
3. On the **Deployment Settings** page, click **Connect**.
4. In the **Connect to Server** dialog box, in the **Server name** box, type the fully qualified server name you created in the first task of this exercise. The server name must end with the suffix **.database.windows.net**.
5. In the **Authentication** list box, click **SQL Server Authentication**.
6. In the **Login** box, type **salesappadmin**, in the **Password** box, type **Pa\$\$w0rd**, then click **Connect**.
7. On the **Deployment Settings** page, click **Next**.
8. On the **Summary** page, verify the settings, and then click **Finish**. The migration process will take some time to complete.
9. On the **Results** page, review the outcome of the wizard, and then click **Close**.

► Task 4: Connect to the Migrated Database

1. In **Internet Explorer**, return to the Azure portal.
2. Click **SQL databases** and verify that **salesapp1** appears in the list of databases.
3. Close Internet Explorer.
4. In **SQL Server Management Studio**, click **New Query**.
5. On the **Query** menu, point to **Connection**, then click **Change Connection**.
6. In the **Connect to Database Engine** dialog box, in the **Server name** box, type the fully qualified server name you created in the first task of this exercise. The server name must end with the suffix **.database.windows.net**.
7. In the **Authentication** list, click **SQL Server Authentication**.
8. In the **Login** box, type **salesappadmin**, in the **Password** box, type **Pa\$\$w0rd**, and then click **Connect**.
9. In the **Available databases** list, click **salesapp1**.
10. Type the following query:

```
SELECT TOP(10) * FROM Sales.Customers;
```

11. Click **Execute**. Observe that 10 rows are returned from the Azure database.

Results: After this exercise, you will have created an empty database in Azure SQL Database, migrated an on-premises database to Azure SQL Database, and be able to connect to, and query, the migrated database.

Module 8: Migrating Databases to Azure SQL Database

Lab: Migrating SQL Server with Azure

Exercise 1: Check Database Compatibility

► Task 1: Prepare the Lab Environment

1. Ensure that the MSL-TMG1, 20765B-MIA-DC, and 20765B-MIA-SQL VMs are both running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the D:\Labfiles\Lab08\Starter folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**.
4. Wait for the script to finish, and then press any key to continue.

► Task 2: Run the Export Data-tier Application Wizard to Check Database Compatibility

1. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
2. In Object Explorer, expand **Databases**, right-click **salesapp1**, point to **Tasks**, and then click **Export Data-tier Application**.
3. In the **Export Data-tier Application 'salesapp1'** dialog box, on the **Introduction** page, click **Next**.
4. On the **Export Settings** page, on the **Settings** tab, ensure that **Save to local disk** is selected, and then in the text box, type **D:\Labfiles\Lab08\salesapp1.bacpac**.
5. On the **Advanced** tab, clear the **Select All** check box, and then click **Next**.
6. On the **Summary** page, verify the options you have selected, and then click **Finish**. Wait for the test to complete.
7. On the **Results** page, examine the output of the process. Notice that the database has failed verification. Click any of the instances of **Error** in the **Results** column of the output to see details of the error.

The verification test reports a failure because of the syntax of the T-SQL statement in the **dbo.up_CrossDatabaseQuery** stored procedure.

8. On the **Results** page, click **Close**.

► Task 3: Resolve the Failure by Removing a Stored Procedure

1. In SSMS, click **New Query**.
2. Type the following, and then click **Execute**:

```
USE salesapp1;  
GO  
DROP PROCEDURE dbo.up_CrossDatabaseQuery;  
GO
```

► Task 4: Rerun the Export Data-tier Application Wizard to Check Database Compatibility

1. In Object Explorer, in the **Databases** node, right-click **salesapp1**, point to **Tasks**, and then click **Export Data-tier Application**.
2. In the **Export Data-tier Application 'salesapp1'** dialog box, on the **Introduction** page, click **Next**.

3. On the **Export Settings** page, on the **Settings** tab, ensure that **Save to local disk** is selected, and then in the text box, type **D:\Labfiles\Lab08\salesapp1.bacpac**.
4. On the **Advanced** tab, clear the **Select All** check box, and then click **Next**.
5. On the **Summary** page, verify the options you have selected, and then click **Finish**. Wait for the test to complete.
6. On the **Results** page, examine the output of the process. Notice that the database has passed verification, and then click **Close**.

Results: After this exercise, you should have run the tools to check database compatibility with Azure from SSMS.

Exercise 2: Migrate a Database to Azure

► Task 1: Create an Azure SQL Server

1. Open **Internet Explorer®** and go to **https://portal.azure.com/**.
2. Sign in to the Azure portal with your Azure Pass or Microsoft Account credentials.
3. Click **New**, click **Databases**, and then click **SQL Database**.
4. In the **Name** box on the **SQL Database** blade, type **empty**.
5. Click **Server**, and then click **Create a new server**.
6. On the **New server** blade, in the **Server name** box, type a name for your server—this must be unique throughout the whole Azure service, so cannot be specified here. A suggested format is **sql2016-
<your initials> <one or more digits>**. For example, sql2016-js123. Keep a note of the name you have chosen.
7. In the **Server admin login** box, type **salesappadmin**.
8. In the **Password** and **Confirm password** boxes, type **Pa\$\$w0rd1**.
9. Under **Location**, select a region nearest your current geographical location, and then click **Select**.
10. On the **SQL Database** blade, verify that **Select source** has the value **Blank database**, and then click **Create**.
11. It will take some time for the new server and database to be created. The Azure portal will notify you when this step is finished.
12. Leave Internet Explorer open for the next task.

► Task 2: Configure the Azure Firewall

1. In the Azure portal, click **All Resources**.
2. In the **All resources** pane, click the server name you created in the previous task (if you followed the suggested naming convention, the server name will start **sql2016**).
3. In the server blade, click **Show firewall settings**.
4. In the **Firewall settings** blade, click **Add client IP**, and then click **Save**.
5. When the firewall changes are complete, click **OK**.

6. Leave Internet Explorer open.

► **Task 3: Migrate the salesapp1 Database to Azure**

1. If it is not already connected, in **SQL Server Management Studio**, connect to the **MIA-SQL** database engine using Windows authentication.
2. In the Object Explorer pane, expand **Databases**, right-click **salesapp1**, point to **Tasks**, and then click **Deploy Database to Microsoft Azure SQL Database**.
3. In the Deploy Database '**salesapp1**' dialog box, on the **Introduction** page, click **Next**.
4. On the **Deployment Settings** page, click **Connect**.
5. In the **Connect to Server dialog box**, in the **Server name** box, type the fully qualified server name you created in the first task of this exercise (if you followed the suggested naming convention, the server name will start **sql2016**). The server name must end with the suffix **.database.windows.net**.
6. In the **Authentication** list, click **SQL Server Authentication**.
7. In the **Login** box, type **salesappadmin**, in the **Password** box, type **Pa\$\$w0rd1**, and then click **Connect**.
8. On the **Deployment Settings** page, click **Next**.
9. On the **Summary** page, verify the settings, and then click **Finish**. The migration process will take some time to complete.
10. On the **Results** page, review the outcome of the wizard, and then click **Close**.

► **Task 4: Connect to the Migrated Database**

1. Return to the Azure portal in **Internet Explorer**.
2. Click **SQL databases** and verify that **salesapp1** appears in the list of databases.
3. Close Internet Explorer.
4. In **SQL Server Management Studio**, click **New Query**.
5. On the **Query** menu, point to **Connection**, and then click **Change Connection**.
6. In the **Connect to Database Engine dialog box**, in the **Server name** box, type the fully qualified server name you created in the first task of this exercise (if you followed the suggested naming convention, the server name will start **sql2016**). The server name must end with the suffix **.database.windows.net**.
7. In the Authentication list, click **SQL Server Authentication**.
8. In the **Login** box, type **salesappadmin**, in the **Password** box, type **Pa\$\$w0rd1**, and then click **Connect**.
9. In the **Available databases** list, click **salesapp1**.
10. Type the following query:

```
SELECT TOP(10) * FROM Sales.Customers;
```

11. Click **Execute** and observe that 10 rows are returned from the Azure database.
12. Close SSMS, without saving any changes.

Results: After this task, you will have created an Azure SQL Database instance, migrated an on-premises database to Azure SQL Database, and be able to connect to, and query, the migrated database.

Module 9: Deploying SQL Server on a Microsoft Azure Virtual Machine

Lab: Deploying SQL Server on an Azure Virtual Machine

Exercise 1: Provision an Azure Virtual Machine

► Task 1: Prepare the Lab Environment

1. Ensure that the MSL-TMG1, 20765B-MIA-DC, and 20765B-MIA-SQL virtual machines are all running, and then log on to 20765B-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Labfiles\Lab09\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**.
4. Wait for the script to finish, and then press any key to continue.

► Task 2: Create a New Azure Virtual Machine

1. Start Microsoft Internet Explorer®, and then browse to **http://portal.azure.com**.
2. Sign in to the Azure portal with your Azure Pass or Microsoft account credentials.
3. In the navigation on the left, click **New**, and then click **Compute**.
4. On the **Compute** blade, click **See all**.
5. In the **Search Compute** box, type **SQL Server 2016**, and then press Enter.
6. In the list of results, click **SQL Server 2016 SP1 Enterprise on Windows Server 2016**.
7. In the **Select a deployment model** list, ensure that **Resource Manager** is selected, and then click **Create**.
8. On the **Basics** blade, in the **Name** box, type a name for your VM. This must be unique throughout the whole Azure service, so cannot be specified here. A suggested format is **sql2016vm-<your initials><one or more digits>**. For example, **sql2016vm-js123**. Keep a note of the name you have chosen.
9. In the **User name** box, type **SQLAdmin**.
10. In the **Password** box, type **Pa\$\$w0rd1234**.
11. In the **Confirm password** box, type **Pa\$\$w0rd1234**.
12. Under **Resource group**, click **Create new**, and then type **SQLResourceGroup**.
13. In the **Location** list, select a location near you, and then click **OK**.
14. On the **Size** blade, click **View all**, click **DS11_V2 Standard**, and then click **Select**.
15. On the **Settings** blade, click **OK** to accept the default values.
16. On the **SQL Server Settings** blade, click **OK** to accept the default values.
17. On the **Summary** blade, click **OK**. Azure creates the new VM. The process may take some time.

► Task 3: Connect to the Virtual Machine by Using the Microsoft Remote Desktop Connection Client

1. After the virtual machine has been created and started, the Azure portal displays the virtual machine's blade. Click **Connect**, and then click **OK**.
2. In the **Remote Desktop Connection** dialog box, click **Connect**.
3. In the **Windows Security** dialog box, click **Use another account**.
4. In the **User name** box, type **SQLAdmin**.
5. In the **Password** box, type **Pa\$\$w0rd1234**, and then click **OK**.
6. In the **Remote Desktop Connection** dialog box, click **Yes**. The Remote Desktop Connection client connects and displays the desktop for the new server. Server Manager starts.
7. If the **Networks** pane appears, click **No**.
8. Start **SQL Server Management Studio**.
9. In the **Connect to Server** dialog box, click **Connect**.
10. In Object Explorer, expand the **Databases** node. Notice that there are no user databases.
11. Close SQL Server Management Studio and log out of the remote desktop session.

Results: After this exercise, you will have created a virtual machine in your Azure subscription that runs SQL Server 2016.

Exercise 2: Migrate a Database to a Microsoft Azure Virtual Machine

► Task 1: Back Up an On-Premises Database

1. In the 20765B-MIA-SQL virtual machine, start **SQL Server Management Studio**.
2. In the **Connect to Server** dialog box, click **Connect**.
3. In Object Explorer, expand **Databases**, right-click **ExampleDB**, point to **Tasks**, and then click **Back Up**.
4. In the **Back Up Database - ExampleDB** dialog box, on the **General** page, under **Destination**, click **Remove**.
5. Click **Add**.
6. In the **Select Backup Destination** dialog box, click the ellipses (...) button.
7. In the **Locate Database Files - MIA-SQL** dialog box, in the **File name** box, type **D:\Labfiles\Lab09\Backups.bak**, and then click **OK**.
8. In the **Select Backup Destination** dialog box, click **OK**.
9. Under **Select a page** in the top left, click **Media Options**.
10. Under **Overwrite media**, click **Back up to a new media set, and erase all existing backup sets**.
11. In the **New media set name** box, type **MigrationBackups**.
12. Under **Select a page** in the top left, click **Backup Options**.

13. Under **Backup set**, in the **Name** box, type **ExampleDBMigration**.
14. In the **Set backup compression** list, click **Compress backup**, and then click **OK**.
15. In the **Microsoft SQL Server Management Studio** dialog box, click **OK**.
16. Close SQL Server Management Studio.

► **Task 2: Copy the Backup File to the Azure Virtual Machine**

1. Start File Explorer, and browse to **D:\LabFiles\Lab09**.
2. Right-click **Backups.bak**, and then click **Copy**.
3. In Internet Explorer, in the Azure Portal displaying the VM blade, click **Connect**, and then click **Open**.
4. In the **Remote Desktop Connection** dialog box, click **Connect**.
5. In the **Windows Security** dialog box, click **Use another account**.
6. In the **User name** box, type **SQLAdmin**.
7. In the **Password** box, type **Pa\$\$w0rd1234**, and then click **OK**.
8. In the **Remote Desktop Connection** dialog box, click **Yes**.
9. When the VM desktop is displayed, start File Explorer, and browse to **F:**.
10. On the **Home** menu, click **Paste**. The remote desktop protocol client copies the backup file to the VM.

► **Task 3: Restore the Database to the Azure Virtual Machine**

1. In the remote desktop connection to the Azure VM, start **SQL Server Management Studio**.
2. In the **Connect to Server** dialog box, click **Connect**.
3. In Object Explorer, right-click **Databases**, and then click **Restore Database**.
4. In the **Restore Database** dialog box, on the **General** page, under **Source**, click **Device**, and then click the ellipses (...) button.
5. In the **Select backup devices** dialog box, in the **Backup media type** list, click **File**, and then click **Add**.
6. In the **Locate Backup File** dialog box, browse to **F:**, click **Backups.bak**, and then click **OK**.
7. In the **Select backup devices** dialog box, click **OK**.
8. Under **Select a page** on the left, click **Files**.
9. Select the **Relocate all files to folder** check box, and then click **OK**. SQL Server restores the database.
10. In the **Microsoft SQL Server Management Studio** dialog box, click **OK**.
11. In the Object Explorer, expand **Databases**. If the **ExampleDB** database does not appear, right-click **Databases**, and then click **Refresh**.
12. Log out of the remote desktop session.

Results: At the end of this exercise, you will have migrated a database from the on-premises server that runs SQL Server to the SQL Server instance on the new virtual machine.

Exercise 3: Configure SQL Server Connections

► Task 1: Configure SQL Server Connections for the Virtual Machine

1. In Internet Explorer, in the Azure Portal displaying the VM blade, under **Settings**, click **SQL Server configuration**.
2. Next to **SQL connectivity level**, click **Edit**.
3. In the **SQL connectivity** list, click **Public (Internet)**.
4. Next to **SQL Authentication**, click **Enabled**.
5. In the **SQL Authentication** box, type **SQLAdmin**.
6. In the **Password** box, type **Pa\$\$w0rd1234**, and then click **OK**.
7. In the top right, click the **Notifications** button. The Notifications pane displays the status of the configuration task. This may take several minutes.

► Task 2: Connect SQL Server Management Studio to the Azure Virtual Machine

1. When the virtual machine has been successfully updated, in the Azure portal, in the virtual machine's blade, click **Overview**.
2. Make a note of the **Public IP address** of the VM.
3. In SQL Server Management Studio, in Object Explorer, click **Connect**, and then click **Database Engine**.
4. In the **Connect to Server** dialog box, in the **Server name** box, type the IP address you just noted.
5. In the **Authentication** list, click **SQL Server Authentication**.
6. In the **Login** box, type **SQLAdmin**.
7. In the **Password** box, type **Pa\$\$w0rd1234**, and then click **Connect**.
8. In Object Explorer, under the connection to the IP address, expand **Databases**, expand **ExampleDB**, and then expand **Tables**.
9. Right-click **HR.Employees**, and then click **Select Top 1000 Rows**. The complete table is displayed in the Results pane.
10. Close SSMS, and then close Internet Explorer.

Results: When you have completed this exercise, you will have connected an instance of SQL Server Management Studio to the Azure instance of SQL Server 2016.

Module 10: Managing Databases in the Cloud

Lab: Managing Databases in the Cloud

Exercise 1: Add Data Masking

► Task 1: Create an AdventureWorksLT SQL Database in Azure

1. Open Microsoft Internet Explorer®, and then go to **<https://portal.azure.com/>**.
2. Sign in to the Azure portal with your Azure Pass or Microsoft Account credentials.
3. Click **New**, click **Databases**, and from the Databases blade select **SQL Database**.
4. The SQL Database blade appears.
5. In the **Database name** box, type **AdventureWorksLT**.
6. Do not amend the Subscription name.
7. In the **Resource group** box, select **Create New** and type **20766B + your initials**. A green tick appears if the name is acceptable. Add one or more digits if necessary. Alternatively, use a resource group created in a previous lab exercise.
8. In the **Select source** box, click **Sample (AdventureWorksLT)**.
9. In server, create a **New Server**. The New server blade appears. Alternatively, select a server created in a previous lab exercise.
10. If you are creating a new server, in the Server name box, type **20765B + your initials**. This must be unique so add one or more digits if necessary. A green tick appears if the name is acceptable. Make a note of the server name.
11. In the **Server admin login** box, type **Student**.
12. In the **Password** box, type **Pa\$\$w0rd**.
13. In the **Confirm password** box, type **Pa\$\$w0rd**.
14. In the **Location** box, select the region nearest to your current geographical location.
15. Leave **Allow azure services to access server** selected, and then click **Select**.
16. In the **Want to use SQL elastic pool** box, click **Not now**.
17. Do not change the **Pricing tier**, or **Collation**.
18. Click **Create**. If you have created a new server, it will take a little time for the new server and database to be created. The Azure portal will notify you when this step is finished.
19. Select the server, and in the **Settings** group, click **Firewall**.
20. Click **Add client IP**, click **Save**, and then click **OK**.
21. Leave the Azure portal open for the next lab exercise.

► Task 2: Examine the AdventureWorks Database

1. Open SQL Server Management Studio, and then connect to the database that you created in Azure in the last exercise. Refer to your note of the server name. For example, the server name will be **20765Bab.database.windows.net**, where **ab** are your initials.
2. In the **Login** box, type **Student**, and in the **password** box, type **Pa\$\$w0rd**, and then click **Connect**.
3. On the **File** menu, point to **Open**, and then click **File**.

4. In the **Open File** dialog box, navigate to **D:\Labfiles\Lab10\Starter**, and then double-click **Data Masking.sql**.
5. In the **Available Databases** box, click **AdventureWorksLT**.
6. Run the script titled **Create Test User**.
7. Run the script titled **Execute as TestUser**.
8. Run the script titled **Test Dynamic Data Masking**. Note that all the columns are displayed unmasked.
9. Run the script titled **Revert**.
10. Leave SQL Server Management Studio open for the next lab exercise.

► Task 3: Create the Data Masks

1. If the Azure portal is not open from earlier in the exercise, open **Internet Explorer** and login to the **Azure portal** with your Azure Pass or Microsoft account.
2. On the **Dashboard**, click on the **AdventureWorksLT** database you created earlier in the exercise.
3. In the **Settings** group, click **Dynamic Data Masking**.
4. In the **Recommended fields to mask** section, click **Add Mask** next to the following:
 - a. **SalesLT.Customer.LastName**.
 - b. **SalesLT.Customer.EmailAddress**.
5. In the **Masking rules** section, click the mask function next to **SalesLT_Customer_EmailAddress**. This is currently set to **Default value**.
6. In the **Edit Masking Rule** blade, in the **Masking field format** list, click **Email (aXXX@XXXX.com)**, click **Update**, and then close the blade.
7. Click **Save**.
8. A message appears to confirm you have successfully saved Dynamic Data Masking settings, click **OK**.
9. In the **Masking rules** section, click the mask function next to **SalesLT.Customer.LastName**. This is currently set to **Default value**.
10. In the **Edit Masking Rule** blade, in the **Masking field format** list, click **Custom string (prefix [padding] suffix)**.
11. In the **Exposed Prefix** box, type **2**.
12. In the **Padding String** box, type **XXXX**.
13. In the **Exposed Suffix** box, type **0**, click **Update**, and then close the blade.
14. Click **Save**. When the settings have been saved, click **OK**.

► Task 4: Test Data Masking

1. Open SQL Server Management Studio if you closed it after the last exercise.
2. On the **File** menu, point to **Open**, and then click **File**.
3. In the **Open File** dialog box, navigate to **D:\Labfiles\Lab10\Starter**, and then double-click **Check Masking.sql**.
4. In the **Available Databases** box, click **AdventureWorksLT**.
5. Run the script under the heading **Execute as TestUser**.

6. Run the script under the heading **Test Dynamic Data Masking**. Check that the columns have been masked correctly.
7. Run the script under the heading **Revert**.
8. Run the script under the heading **Test as Admin**. Note that the columns are displayed unmasked.
9. Close the script without saving.
10. Close SQL Server Management Studio and the Azure portal window.

Results: After this exercise, you will have added data masks to columns that contain sensitive data and verified that the data is masked to unauthorized users.

Exercise 2: Use Azure Automation to Stop Virtual Machines

► Task 1: Create a Virtual Machine by Using Windows PowerShell

1. Start the MSL-TMG1, 20765B-MIA-DC and 20765B-MIA-SQL virtual machines, and log on to **20765B-MIA-SQL** as **AdventureWorks\Student** with the password **Pa\$\$w0rd**.
2. In the **D:\Labfiles\Lab10\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and wait for the script to finish.
4. Using Internet Explorer, open the **Azure portal** (<https://portal.azure.com>) using your Azure pass credentials.
5. On the taskbar, right-click **Windows PowerShell**, and then click **Run ISE as Administrator**.
6. In the **User Account Control** dialog box, click **Yes**.
7. In Windows PowerShell ISE, on the **File** menu, point to **Open**, and then click **File**.
8. In the **Open File** dialog box, navigate to **D:\Labfiles\Lab10\Starter**, and then double-click **CreateResources.ps1**.
9. In the section titled **# Initialize variable**:
10. Amend the **\$subscriptionName** variable to the name of your Azure subscription name.
11. Amend the **\$tenantID** variable to the correct TenantID. You will find this in the **Azure Active Directory** section, in **Properties**. Copy the string **Directory ID** to the **TenantID** section in the script.
12. Highlight the script block titled **# Initialize** and click **Run Selection** to execute the script.
13. Run the script under **# Sign in to Azure and select subscription**. You will be prompted to sign in.
14. Run the script under **# Check name**. If the storage name is not available, go back to the **# Initialize** section, add a number to the storage account variable, run the script to initialize the new name, and then run the **# Check name** script again.
15. Run the script under **# Prompt for credentials**. In both dialog boxes, type the password **Pa\$\$w0rd**.
16. Run the script under **# Create resource groups**.
17. Run the script under **# Create storage account**.
18. Run the script under **# Create virtual network**. Note the warning.
19. Run the script under **# Create public IP**. Note the warning.

20. Run the script under **# Create virtual machine**. This may take a couple of minutes to complete.
21. Run the script under **# Create and configure SQL Server**. This may take a couple of minutes to complete. If an error message appears the server name is not available, go back to the **# Initialize** section, add a number to the server Name variable, run the script to initialize the new name, and then run the **# Create and configure SQL Server** script again
22. Run the script under **# Create SQL database**.
23. Close Windows PowerShell ISE.
24. Leave the **Azure portal** open for the next lab exercise.

► Task 2: Create an Azure Automation Account

1. In Internet Explorer, navigate to **https://portal.azure.com**.
2. If necessary, login using your Azure Pass credentials.
3. At the dashboard, click **New**.
4. On the **New** blade, click **Monitoring + Management**.
5. On the **Monitoring + Management** blade, click **Automation**.
6. On the **Add Automation Account** blade, in the **Name** box, type **automate + your initials**. The name must be in lowercase characters and numbers. A green tick appears when the name is acceptable. If necessary, add additional numbers to create a unique name.
7. Select the correct **Subscription**.
8. In the **Resource group** box, either create a new resource group called **20765b**, or select the resource group created in the first lab exercise.
9. In the **Location** box, select a Microsoft data center close to you.
10. Ensure **Create Azure Run As account**, is set to **Yes**, and then click **Create**. The Automation Account takes a short while to be created. A message is displayed when the account has been created.
11. Click **All resources** to see that the new Automation account has been created.
12. Leave the **Azure portal** open.

► Task 3: Stop the Virtual Machine by Using Azure Automation

1. If necessary, sign in to the Azure portal.
2. Click **All resources**, and then click **automate** (the automation account you have just created).
3. Under **Process Automation**, click **Runbooks**.
4. Click **Browse gallery**.
5. In the **Browse Gallery** blade, click **Stop Azure V2 VMs**. This is a graphical runbook.
6. In the **Stop Azure V2 VMs** blade, click **Import**.
7. In the **Import** blade, check that the **StopAzureV2Vm** runbook is selected, and click **OK**.
8. In the **StopAzureV2Vm** runbook blade, click **Edit**.
9. In the **Edit Graphical Runbook** blade, click **Publish**, and then click **Yes**. A message confirms that the runbook has been published.
10. In the **StopAzureV2Vm** blade, check that the **status** is **Published**, and then click **Start**.

11. In the **Start Runbook** blade, in the **ResourceGroupName** box, type **ResGroup1**, and in the **VMName** box, type **VM1**, and then click **OK**.
12. In the **StopAzureV2Vm job** blade, click **Output**. Although an error message appears, the VM has been stopped.
13. Verify that the VM has been stopped by closing the **Output** blade, click **All resources**, and then click **VM1**. Note that the status is **Stopped (deallocated)**.

► **Task 4: Start the Virtual Machine by Using a Schedule in Azure Automation**

1. If necessary, sign in to the Azure portal.
2. In **All resources**, click **automate** (the automation account you created in the previous lab exercise).
3. Under **Process Automation**, click **Runbooks**.
4. Click **Browse gallery**.
5. In the **Browse Gallery** blade, click **Start Azure V2 VMs**. This is a graphical runbook.
6. In the **Start Azure V2 VMs** blade, click **Import**.
7. In the **Import** blade, check that the **StartAzureV2Vm** runbook is selected, and click **OK**.
8. In the **StartAzureV2Vm** runbook blade, click **Edit**.
9. In the **Edit Graphical Runbook** blade, click **Publish**, and then click **Yes**. A message confirms that the runbook has been published.
10. In the **StartAzureV2Vm** blade, check that the status is **Published**, and then click **Schedule**.
11. In the **Schedule Runbook** blade, click **Link a schedule to your runbook**.
12. In the **Schedule** blade, click **Create a new schedule**.
13. In the **Name** box, type **MySchedule**. Leave the **Description** box blank.
14. In the **Starts** box, select a time 6 minutes into the future.
15. In the **Time zone** box, select your local time zone.
16. In the **Recurrence** box, click **Once**, and then click **Create**.
17. In the **Schedule Runbook** blade, click **Configure parameters and run settings**.
18. In the **Parameters** blade, in the **ResourceGroupName** box, type **ResGroup1**.
19. In the **VMName** box, type **VM1**, and then click **OK**.
20. In the **Schedule Runbook** blade, click **OK**.
21. In the **StartAzureV2Vm** blade, under **Details**, click **Schedules** to view the schedule you created. Close the **Schedules** blade.
22. Wait 5 minutes, and when the job is due to run, click **Jobs**.
23. The job should appear as running at the scheduled time. Wait until the job has completed.
24. In **All resources**, click **VM1**. Note that the status is **Running**.

► **Task 5: List Resources by Using Azure Automation**

1. If necessary, sign in to the Azure portal.
2. In **All resources**, click **automate** (the automation account you created in the previous lab exercise).
3. Under **Process Automation**, click **Runbooks**.

4. Click **AzureAutomationTutorial**.
5. In the **AzureAutomationTutorial** runbook blade, click **Edit**. Examine the runbook.
6. In the **Edit Graphical Runbook** blade, click **Publish**, and then click **Yes**. A message confirms that the runbook has been published.
7. In the **AzureAutomationTutorial Runbook** blade, verify that the status is **Published**, then click **Start**, and then click **Yes**.
8. Wait until the job has been completed, and then click **Output**. A list of all resources is displayed.
9. Close the Azure portal when you have finished.

Results: After this exercise, you will have understood how a Windows PowerShell® script is used to create Azure resources, created an Azure Automation account, and used Azure Automation to stop a virtual machine.