Microsoft

OFFICIAL MICROSOFT LEARNING PRODUCT

# 20765A
# Provisioning SQL Databases

Product Number: 20765A

Part Number (if applicable):

Released: 04/2016

# 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 performance[1]. 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

# About This Course

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

## Course Description

**Note:** This first release ('A') MOC version of course 20761A has been developed on prerelease software (CTP3.3). Microsoft Learning will release a 'B' version of this course after the RTM version of the software is available.

This three-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**, "Deploying SQL Server on Microsoft Azure"

**Module 5**, "Working with databases"

**Module 6**, "Database storage options"

**Module 7**, "Performing database maintenance"

## 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, please go to www.microsoft.com/learning/help. To inquire about the Microsoft Certification Program, send an email to mcphelp@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 |
|---|---|
| 20765A-MIA-DC | Domain controller for the ADVENTUREWORKS domain. |
| 20765A-MIA-SQL | SQL Server 2016 and SharePoint Server |
| 20765A-MIA-DC-UPGRADE | Domain controller for the ADVENTUREWORKS domain. |
| 20765A-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 CTP3.3
- 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 RM 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

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 the use of 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 is 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.

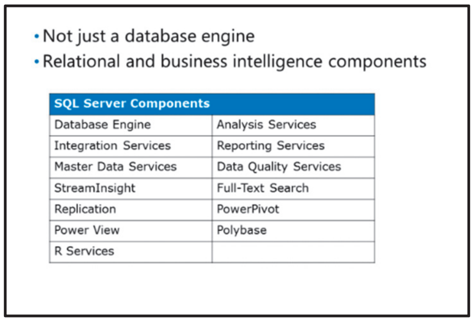
## Lesson Objectives

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:

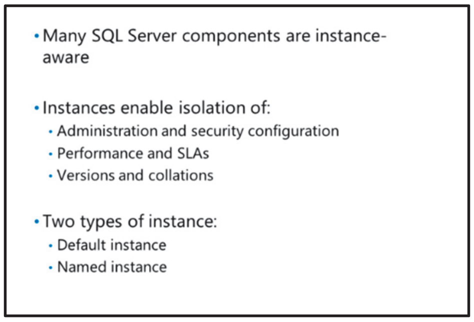


- Not just a database engine
- Relational and business intelligence components

| 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. |
| Analysis Services | SQL Server Analysis Services (SSAS) is an online analytical processing (OLAP) engine that works with analytic cubes and tables. It is 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, providing 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.<br>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.<br>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



- 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

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

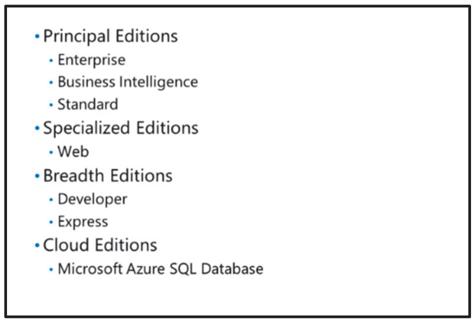
Before SQL Server 2000, just one copy of SQL Server could be installed on a server system. SQL Server was addressed by the name of the Windows server where it was hosted. To maintain backward compatibility, this mode of connection is still supported and is known as the "default instance". For internal 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.

## SQL Server Editions



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

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. |

### 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.

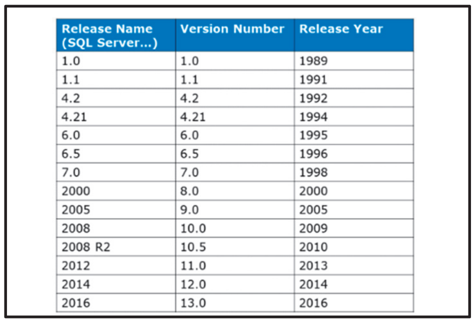
For more information, see the *Features Supported by the Editions of SQL Server 2016* topic in Books Online at:

**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.



| Release Name (SQL Server...) | 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 |

### 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.

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 non-relational 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 complied .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 multi-server 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 the 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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are running and log on to MIA-SQL-20765A-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**), 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.

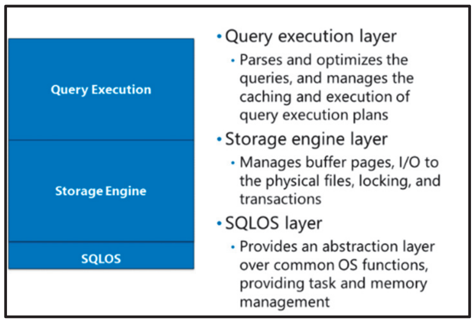
The most important resources used by SQL Server from a server platform are CPU, memory, and I/O. In this lesson, you will see how SQL Server is structured and how it uses these resources.

## 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



- Query execution layer
  - Parses and optimizes the queries, and manages the caching and execution of query execution plans
- Storage engine layer
  - Manages buffer pages, I/O to the physical files, locking, and transactions
- SQLOS layer
  - Provides an abstraction layer over common OS functions, providing task and memory management

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 sub-components 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 algebrizer 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 as, 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 sub-components of the storage engine layer are as follows:

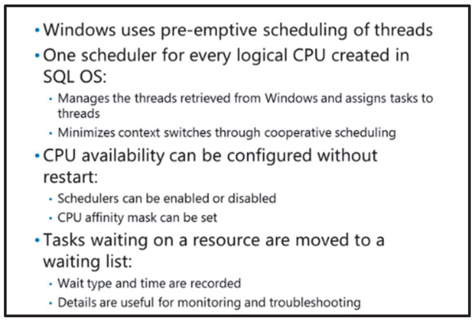
- The access methods component is used to manage how data is accessed. For example, the access methods component works with scans, seeks, and lookups.
- The page cache manages the storage of cached copies of data pages. Caching of data pages is used to minimize the time it takes to access them. The page 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



- Windows uses pre-emptive 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

All work performed in the Windows operating system is based on the execution of threads. Windows uses pre-emptive scheduling of threads to maintain control. Rather than requiring threads to give up control of the CPU voluntarily, pre-emptive 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 pre-empted.

### 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 which represents the unit of work to be done. For example, if you send a batch of Transact-SQL commands to the server, it is 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 the need for rebooting. By default, SQL Server will assume that it can use all CPUs on the host system. While it is possible to 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

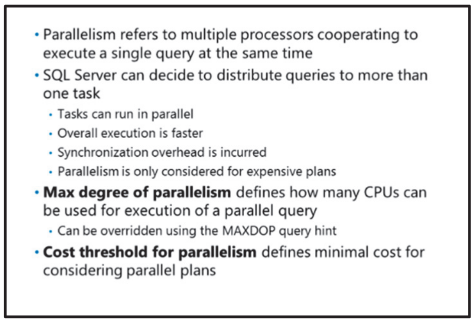
One concept that might be difficult to grasp at first is that most SQL Server tasks spend the majority of their time waiting for something external to happen. Most of the time, they are waiting for I/O or the release of locks, but this can involve other system resources.

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



- 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

All operations run against SQL Server are capable of running 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.

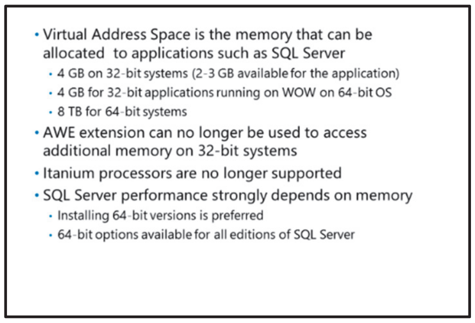
A better practice is to raise the value of **cost threshold for parallelism** so that a parallel plan is only considered for higher cost queries.

For further information, see the *Configure the cost threshold for parallelism Server Configuration Option* topic in Books Online:

**Configure the cost threshold for parallelism Server Configuration Option**

http://aka.ms/leh9eq

## 32-bit and 64-bit Servers



- 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

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. It is possible to 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

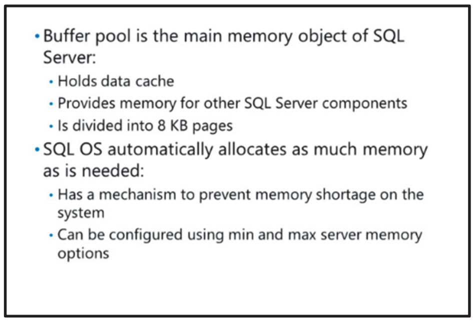
Database systems tend to work with large amounts of data and need substantial amounts of memory, so that the systems can serve significant numbers of concurrent users. SQL Server needs large amounts of memory for caching queries and data pages. A limit of 2 GB to 3 GB for VAS is a major constraint on 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 is 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.

## Overview of SQL Server Memory Management



- 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 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.

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.

As long as 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.

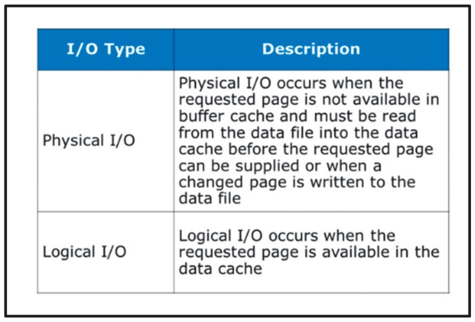
It is good practice to reduce the **max server memory** to a value where a SQL Server instance will not attempt to consume all the memory available to the host operating system.

For more information, see the *Server Memory Server Configuration Options* topic in Books Online:

**Server Memory Server Configuration Options**

http://aka.ms/afm7fd

## Physical I/O and Logical I/O



| 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 |

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.

### 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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are running and log on to 20765A-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 of 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.

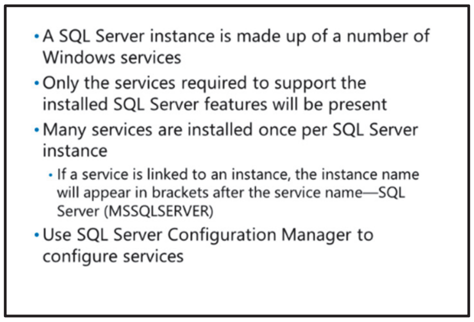
**Note:** Many topics in this lesson do not apply to an Azure SQL Database. The configuration and management of service accounts for Azure SQL Database is not exposed to Azure users. However, these topics apply to SQL Server instances running on virtual machines in the Azure cloud.

## 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



- 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 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.
- **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 Copy service (VSS). 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)**.

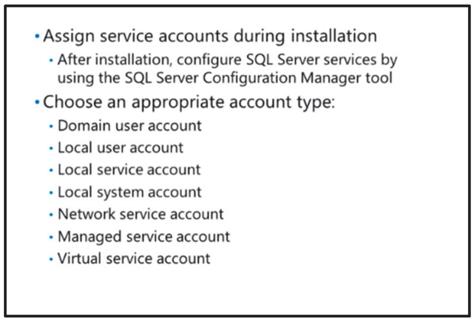
### Configuring Services

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



- 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

Each SQL Server service must run in the security context of a service account. There are several types of accounts you can use 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 to do 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.

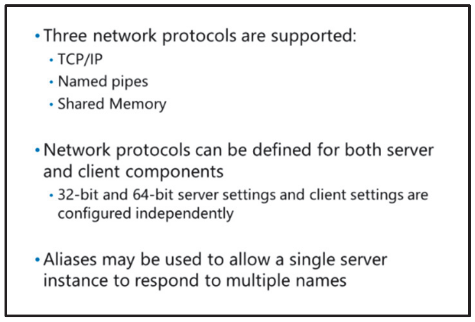
When you use the SQL Server Setup installation program (or the SQL Server Configuration Manager) to specify a service account, the account is automatically added to the relevant security group to ensure that it has the appropriate rights.

**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 non-administrator domain user account is a suitable choice for service accounts in a domain environment.
- **Local user account**. A non-administrator local user account is a secure choice for service accounts in a non-domain 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**. The local service account, a pre-defined account with restricted privileges, can be used 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 simply a virtualized instance of the built-in network service account with its own unique identifier.

## Configuring Network Protocols



- 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
- Aliases may be used to allow a single server instance to respond to multiple names

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 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 on 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

One of the key responsibilities of a database administrator (DBA) is to provision databases, servers and databases. This includes planning and performing the installation of 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

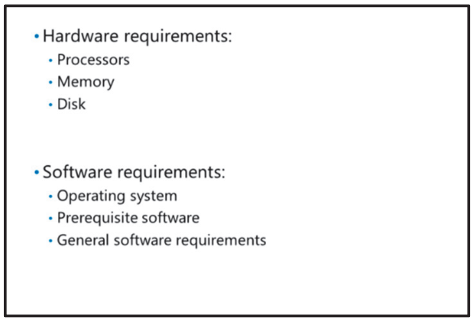
This lesson covers the hardware and software requirements for installing SQL Server 2016. You will learn about the minimum requirements specified by Microsoft®, in addition to tools and techniques to assess hardware performance before you install SQL Server.

### 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



- Hardware requirements:
  - Processors
  - Memory
  - Disk
- Software requirements:
  - Operating system
  - Prerequisite software
  - General 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 co-exist 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**. SQL Server 2016 has certain software pre-requisites that must be met by the host operating system.

For more information, see the *Planning a SQL Server Installation* topic in Books Online:

**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 as 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 and may 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<br>Minimum speed: 1.4 GHz<br>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<br>Minimum speed: 1.4 GHz<br>Recommended speed: 2.0 GHz or higher |

Processor core count, rather than processor speed, is more of a concern when planning an SQL Server installation. While you may 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. Also, 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 may 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 in order to install.

### *Operating System*

Operating system requirements vary between different editions of SQL Server. SQL Server Books Online 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 pre-install most requirements as part of the installation process. This is no longer the case—the .NET Framework (3.5 SP1) needs to be pre-installed 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 pre-install these components during any available planned downtime. Components such as the .NET Framework often require a reboot after installation, so the pre-installation of these components can further reduce downtime during installations or upgrades.

### *General Software Requirements*

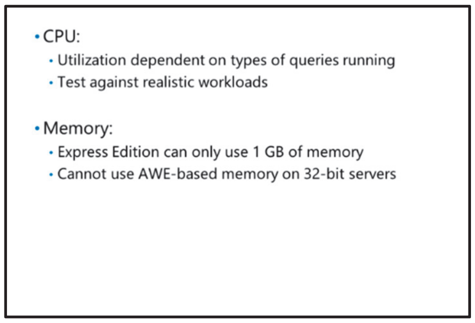
The SQL Server installer is based on the Windows Installer 4.5 technology. You should consider installing Windows Installer 4.5 before the installation of SQL Server, to minimize SQL Server installation time.

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

**Hardware and Software Requirements for Installing SQL Server 2016**

http://aka.ms/usoqbl

## Assessing CPU and Memory Requirements



- 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

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

### CPU

CPU utilization for an 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.

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 an SQL Server on the specific NUMA-based hardware.

### Memory

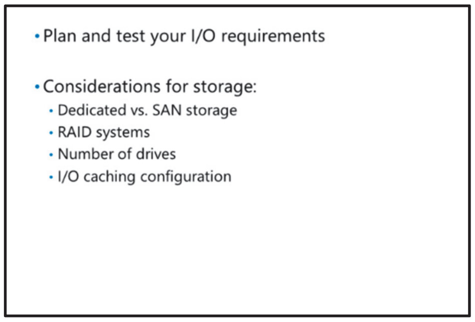
The availability of large amounts of memory for SQL Server to use is now one of the most important factors when sizing systems.

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.

64-bit operating systems have 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



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

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, the requirements of the application must be determined, 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.

### 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

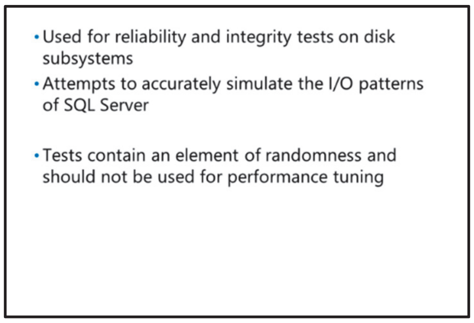
For most current systems, the number of drives (or spindles, as they are still sometimes called), matters more than the size of the disk. It is easy to find large disks that will hold substantial databases, but often a single large disk will not be able to provide sufficient I/O operations per second 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



- 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 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 pre-testing 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 no attempt should be made to compare the output of multiple executions directly, particularly in terms of timing.

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 these files to change during the test.
- **Read-ahead activity**. I/O simulating SQLOS 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 an 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

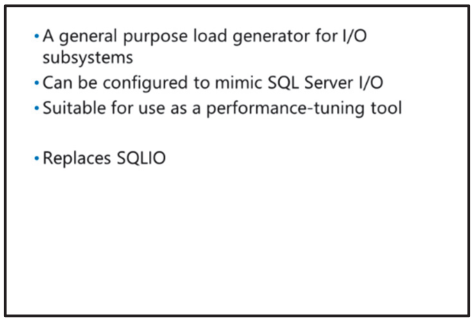
In this demonstration, you will see how to use the SQLIOSim graphical user interface.

### Demonstration Steps

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

12. Examine the results of the test.
13. In File Explorer, go to **D:\**., and open **sqliosim.log.xml** with **Office XML Handler**.
14. Review the test results in the XML file, and then close the file without saving changes.
15. Close SQLIOSim

## Assessing I/O by Using 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

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

Detailed instructions on simulating SQL Server I/O activity, as well as 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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are running and log on to 20765A-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\Diskspd-v2.0.15\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.

### 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 an SQL Server instance; it is used to hold temporary objects that users, or the database engine, create.

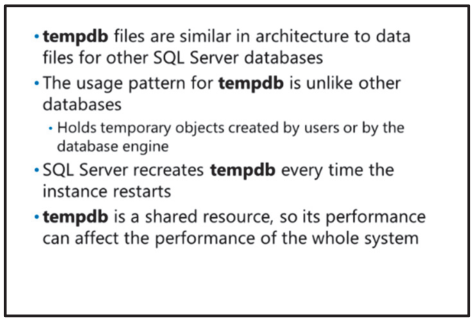
Because it is a shared object, the performance of **tempdb** can have a significant impact on the overall performance of an SQL Server instance; there are steps you can take to improve the performance of **tempdb**.

## 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?



- **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

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 **tempdb** is used by users and the database engine 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 may 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 may 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 lead to 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 pre-configured value. The **tempdb** data files and log file are configured to auto-grow 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 auto-growth increment set on the data files is small, SQL Server may need to

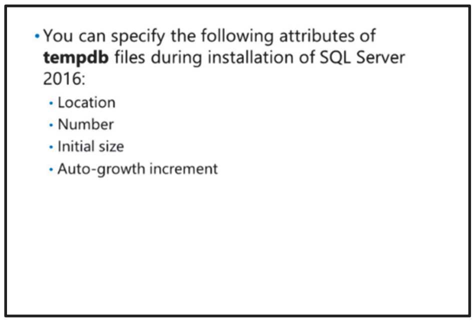
request additional disk space for **tempdb** many times before it reaches a stable size. Because a file-growth operation requires an exclusive lock on the entire **tempdb** database, and **tempdb** is central to the function of SQL Server, each file-growth operation can pause the database engine for its duration. To avoid this problem, specify file size and auto-growth settings for **tempdb** that minimize the number of auto-growth operations during normal running.

For more information, see the *tempdb Database* topic in SQL Server Books Online:

**tempdb Database**

http://aka.ms/d1jmp2

## Modifying the Number of tempdb Files During Installation



- You can specify the following attributes of **tempdb** files during installation of SQL Server 2016:
  - Location
  - Number
  - Initial size
  - Auto-growth increment

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

The setup program will set 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).
- Auto-grow increment for **tempdb** files: 64 MB.

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

# Lesson 3
# Installing SQL Server 2016

After making the decisions about your SQL Server configuration, you can go on 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.

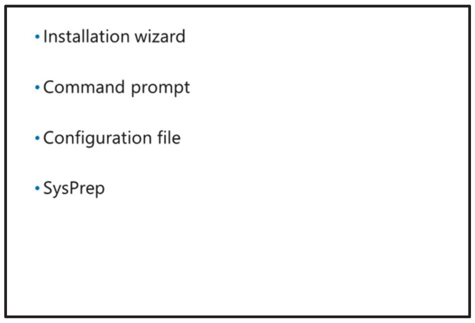
For most users, the setup program will report that all was installed as expected. For the rare situations where this does not occur, you will also learn how to carry out post-installation checks and troubleshooting.

## 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



- Installation wizard
- Command prompt
- Configuration file
- SysPrep

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

The SQL Server installation wizard provides a simple user interface for installing SQL Server. With it, you can 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 also 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 also 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

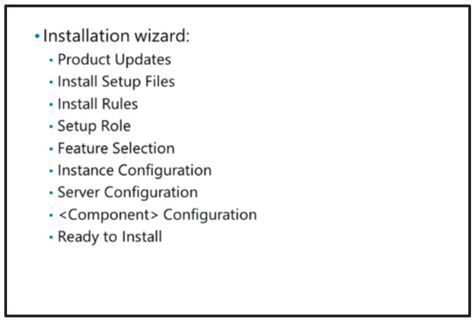
With SQL Server SysPrep, you can prepare an instance of SQL Server, and complete the installation later. This is useful when you want to prepare instances on multiple computers or multiple instances on one computer.

For more information, see the *Install SQL Server 2016* topic in Books Online:

**Install SQL Server 2016**

http://aka.ms/pzn817

## Performing an Installation of SQL Server 2016



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

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.
  - **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.

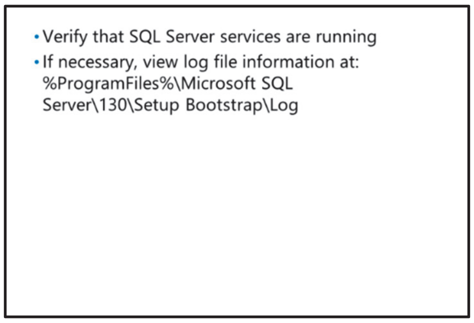
**Note:** This is the sequence of pages in the Installation Wizard when you install SQL Server 2016 on a server with no existing SQL Server 2016 instances. If SQL Server 2016 instances are already installed, most of the same pages are displayed; some pages appear at different positions in the sequence.

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

**Install SQL Server 2016 from the Installation Wizard (Setup)**

http://aka.ms/ug68a2

## Performing Post-Installation Checks



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

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.

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 |

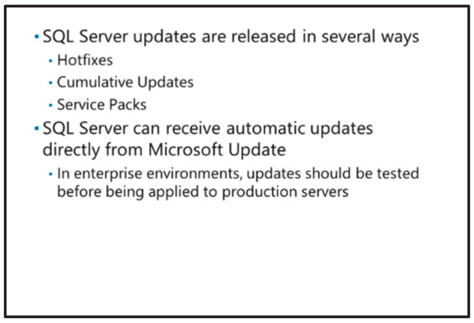
| Category 1 | | Category 2 |
|---|---|---|
| Can be used to install SQL Server 2016 | | Cannot be used to install SQL Server 2016 |
| | | |

## 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



- 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

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.

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 is likely that the updates should be applied to test or staging environments before being applied to production environments.

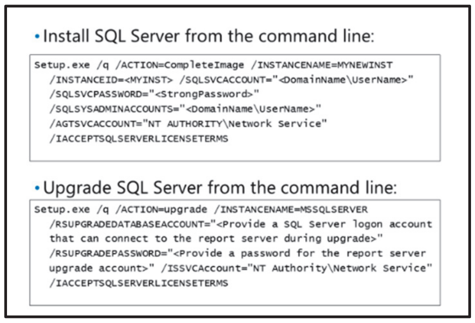
SQL Server 2016 can also have product SPs slipstreamed into the installation process to avoid having to apply them after installation.

For more information, see the *Overview of SQL Server Servicing Installation* topic in Books Online:

**Overview of SQL Server Servicing Installation**

http://aka.ms/yd8zlj

## Unattended Installation



- Install SQL Server from the command line:

```
Setup.exe /q /ACTION=CompleteImage /INSTANCENAME=MYNEWINST
   /INSTANCEID=<MYINST> /SQLSVCACCOUNT="<DomainName\UserName>"
   /SQLSVCPASSWORD="<StrongPassword>"
   /SQLSYSADMINACCOUNTS="<DomainName\UserName>"
   /AGTSVCACCOUNT="NT AUTHORITY\Network Service"
   /IACCEPTSQLSERVERLICENSETERMS
```

- Upgrade SQL Server from the command line:

```
Setup.exe /q /ACTION=upgrade /INSTANCENAME=MSSQLSERVER
   /RSUPGRADEDATABASEACCOUNT="<Provide a SQL Server logon 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"
   /IACCEPTSQLSERVERLICENSETERMS
```

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.

### /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 Books Online:

**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 Books Online:

**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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are running and log on to 20765A-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 SQL Server Books Online. In particular, note the values of the following properties:
    a. INSTANCEID
    b. ACTION
    c. FEATURES
    d. QUIET
    e. QUIETSIMPLE
    f. INSTALLSHAREDDIR
    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 an SQL Server installation.
- Install SQL Server 2016.
- Perform post-installation checks.
- Automate an SQL Server installation.

Estimated Time: 90 minutes

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

1. Ensure that the MSL-TMG1, 20765A-MIA-DC, and 20765A-MIA-SQL virtual machines are running, and then log on to 20765A-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. 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 an 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**. 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. Then 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. Then 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 would like 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; **tempdb** should be configured 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 ConfigurationFile.ini 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

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 instance:

- 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 different strategies you might use to carry it out.

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/yzzozm

### 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

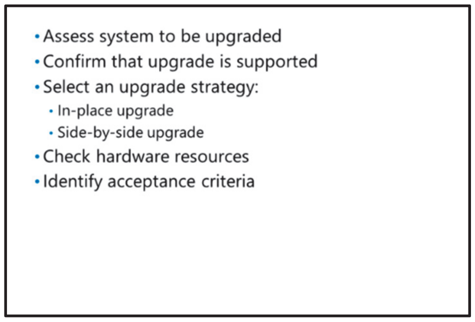
Before beginning an upgrade of an existing SQL Server to SQL Server 2016, you should draw up an upgrade plan. This lesson covers the points you will need to consider.

## 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



- 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

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?

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 edition of SQL Server 2016 you wish to upgrade to. 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 either:

- 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.

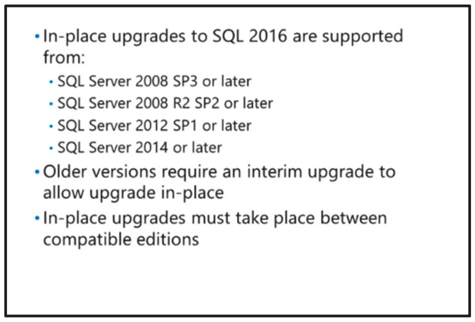
The recommended practice is to treat a SQL Server upgrade as you would any other IT project by assembling a team with the appropriate skills and developing an upgrade plan.

For more information about all aspects of upgrading to SQL Server 2016, see *Upgrade to SQL Server 2016* in Books Online:

**Upgrade to SQL Server 2016**

http://aka.ms/dem1hx

## Supported Versions and Editions



- 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

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 are only permitted between compatible editions of SQL Server. In general, you may 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 Books Online:

**Supported Version and Edition Upgrades**

http://aka.ms/ssdh5f

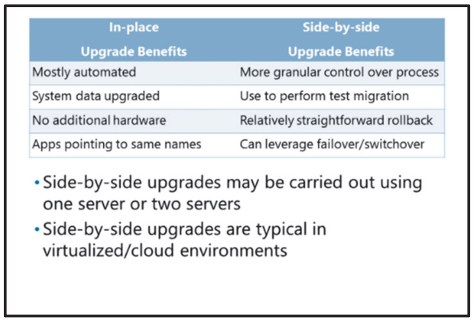
For more information about upgrading to SQL Server 2016 from SQL Server 2005, see *Are you upgrading from SQL Server 2005?* in Books Online:

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

### 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 avoid having to redirect 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.

- 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 is left 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 Books Online:

**Work with Multiple Versions and Instances of SQL Server—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.

Once user databases have been attached to a newer version of SQL Server, they cannot be reattached to an older version again, even if the database compatibility settings have not been upgraded. This is a risk that needs to be considered when using 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 which employs High Availability (HA) functionality.

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

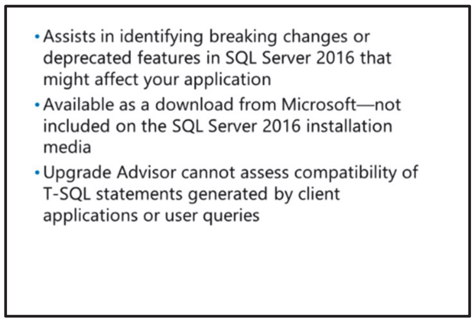
- Always-on availability groups
- Failover clustering
- Mirroring
- Log shipping
- Replication
- Reporting Services scale-out

To assist in your upgrade planning, see *Choose a Database Engine Upgrade Method* in Books Online:

**Choose a Database Engine Upgrade Method**

http://aka.ms/f2xd0w

## Upgrade Advisor



- 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

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.

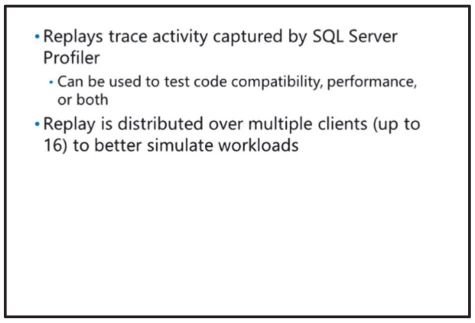
Lists of issues are generated for SQL Server 2016 and also for future versions (listing features that are still supported in SQL Server 2016 but marked for deprecation in a future version of SQL Server).

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.

**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



- 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

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.

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 20765: *Administering a SQL Database Infrastructure*.

SQL Server Profiler trace data files must be pre-processed before they can be 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 Books Online:

**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 20765A-MIA-DC-UPGRADE and 20765A-MIA-SQL-UPGRADE virtual machines are running and log on to 20765A-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**. Select the **I accept the terms in the License Agreement** check box, then click **Next**.
6. Select the **I agree to the Privacy Policy** check box, then click **Install**. Click **Yes** when prompted, and then click **Finish**.
7. On the **Start** screen, type **SQL Server 2016 Upgrade Advisor**, and then click **SQL Server 2016 Upgrade Advisor**.
8. Click **RUN DATABASE UPGRADE ANALYZER**.
9. Click **SELECT DATABASES TO ANALYZE**.
10. In the **SERVER NAME** box, type **MIA-SQL**, and then click **Connect**.
11. In the list of databases, click **TSQL** and **MDS**, click **Select**, and then click **Run**.
12. When analysis is complete, in the **Analyzed databases** blade, click **TSQL**. In the **Analysis results** blade, click **130 – SQL Server 2016**.
13. In the **Compatibility results** blade, click **SET ROWCOUNT used in context of an INSERT/UPDATE/DELETE**.
14. 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.
15. Close SQL Server 2016 Upgrade Advisor.
16. 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.

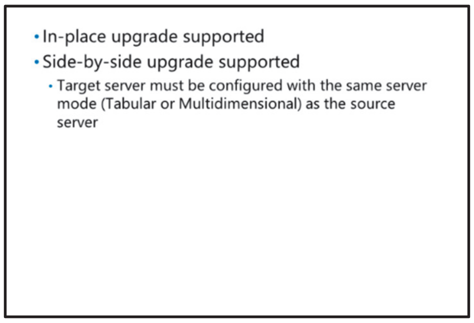
Some SQL Server features might require additional work to facilitate the upgrade process; the nature of this work might vary, depending on whether you are planning to use an in-place or side-by-side migration strategy.

## 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



- 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

- 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.

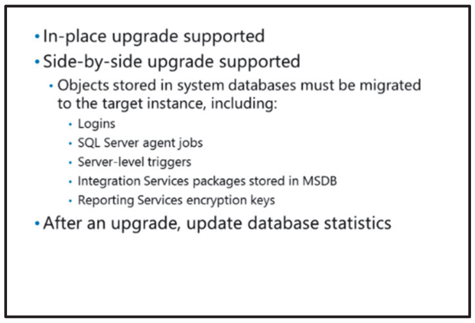
The target Analysis Services instance must use the same server mode (Tabular or Multidimensional) as the source instance.

For more information on this topic, see *Upgrade Analysis Services* in Books Online:

**Upgrade Analysis Services**

http://aka.ms/h1b1fn

## Upgrading Database Engine



- 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

- 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.

**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 20765: *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

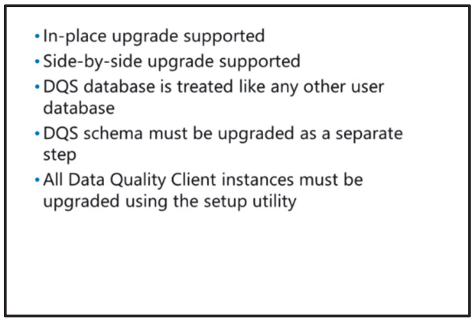
After an upgrade, you should update statistics on all user databases using **sp_updatestats**.

For more information on this topic, see *Upgrade Database Engine* in Books Online:

**Upgrade Database Engine**

http://aka.ms/ektsvf

## Upgrading Data Quality Services



- 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

- 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.

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 pf the DQS databases using the command-line utility **dsqinstaller**.

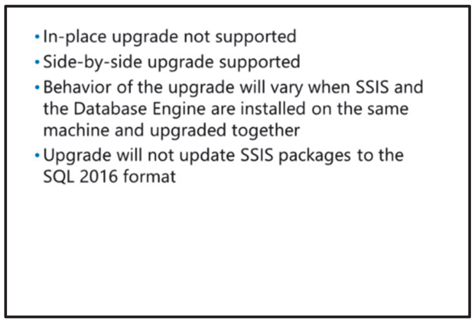
For more information on this topic, see *Upgrade Data Quality Services* in Books Online:

**Upgrade Data Quality Services**

http://aka.ms/xcfpuo

## Upgrading Integration Services

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



- 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

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.

**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 which 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 the Database Engine is upgraded alone, 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*:

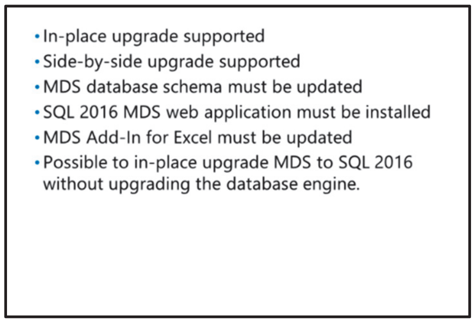
- Remove earlier versions of the Integration Services service.
- Migrate existing SSIS packages to the new package format used by SQL Server 2016.
- Move packages from file system locations other than the default location.
- Update SQL Agent job steps that call the **dtexec** utility with the file system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

For more information on this topic, see *Upgrade Integration Services* in Books Online:

**Upgrade Integration Services**

http://aka.ms/qhumpp

## Upgrading Master Data Services



- 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
- Possible to in-place upgrade MDS to SQL 2016 without upgrading the database engine.

- 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.
  - Once the schema upgrade of the MDS database is complete, any old version of the MDS web application will no long 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.
  - Once the schema upgrade of the MDS database is complete, clients using any old version of the Master Data Services Add-In for Excel will no longer be able to connect to the database.

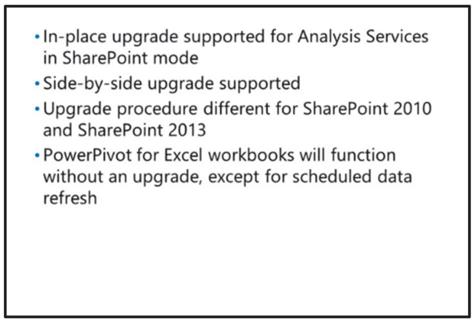
You can carry out an in-place upgrade Master Data Services to SQL Server 2016 without upgrading the database engine hosting the MDS database.

For more information on this topic, see *Upgrade Master Data Services* in Books Online:

**Upgrade Master Data Services**

http://aka.ms/e2vkak

## Upgrading Power Pivot for SharePoint



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

- 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 PowerPivot for SharePoint Add-In is always installed side-by-side with earlier installations and cannot be upgraded in-place.

Upgrading PowerPivot for SharePoint to SQL Server 2016 requires that SharePoint components are installed or upgraded on your SharePoint server or server farm.

Upgrade prerequisites and the details of the upgrade procedure for PowerPivot 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 PowerPivot for SharePoint Add-In on all servers in the SharePoint farm. In a multi-server farm, once the upgrade is completed on one server, the remaining servers in the farm become unavailable until they are upgraded.

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

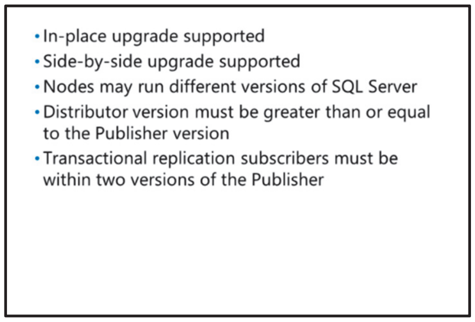
- The exception to this is workbooks using scheduled data refresh. These workbooks must be using a version of PowerPivot for Excel which matches the server version. You must manually upgrade these workbooks, or use the auto-upgrade for data refresh feature in SharePoint 2010.

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 Books Online:

**Upgrade Power Pivot for SharePoint**

http://aka.ms/uvim08

## Upgrading Replicated Databases


- 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

- 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.

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 the log reader agent. Once this process has completed, you can upgrade to SQL Server 2016.

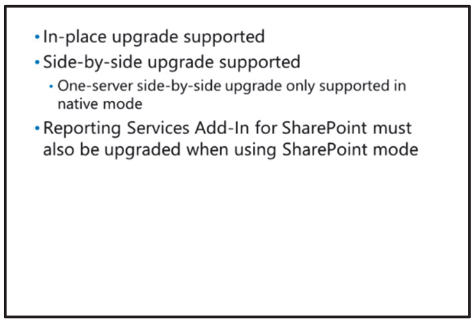
For merge replication instances, the Merge Agent and Snapshot Agent should be run for each subscription following an upgrade.

For more information, see the topic *Upgrade Replicated Databases* in Books Online:

**Upgrade Replicated Databases**

http://aka.ms/lako1d

## Upgrading and Migrating Reporting Services



- 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

- 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.

Before embarking on an upgrade:

- Back up your Reporting Services encryption keys.
- Back up customizations to Reporting Services virtual directories in 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 Books Online:

**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 Books Online:

**Install or Uninstall the Reporting Services Add-in for SharePoint**

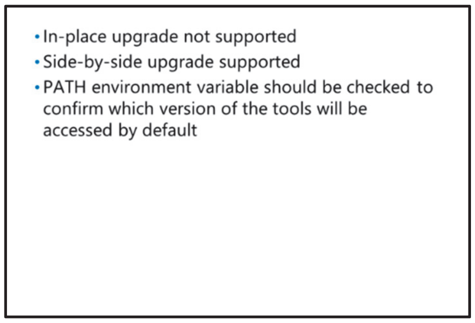
http://aka.ms/x9ehkc

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 Books Online:

**Migrate a Reporting Services Installation (Native Mode)**

http://aka.ms/c6tn20

## Upgrading SQL Server 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

- 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.

Once 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.

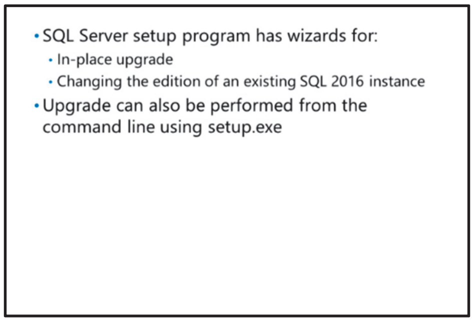
**Note:** You might need to update the PATH environment variable (or specify a full path for executable files) to ensure that you can use the new version of command-line tools and utilities.

For more information, see the topic *Upgrade SQL Server Management Tools* in Books Online:

**Upgrade SQL Server Management Tools**

http://aka.ms/hlqv5o

## Upgrading Using Setup



- 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

### 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.

For further 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 Books Online:

**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 Books Online:

**Supported Version and Edition Upgrades**

http://aka.ms/ssdh5f

For further 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 Books Online:

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

http://aka.ms/itd5ma

**Note:** It is possible to 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 20765A-MIA-DC-UPGRADE and 20765A-MIA-SQL-UPGRADE virtual machines are running, and log on to 20765A-MIA-SQL-UPGRADE as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In **File Manager**, double-click **X:\setup.exe**. Click **Yes** when prompted.
3. In **SQL Server Installation Center**, click **Installation**, then click **Upgrade from a previous version of SQL Server**.
4. On the **Product Key** page, click **Next**.
5. On the **License Terms** page, select **I accept the license terms**, then click **Next**.
6. On the **Product Updates** page, click **Next**. Any error relating to a failure to search for updates through Windows Update can be ignored.
7. On the **Upgrade Rules** page, click **Next**. A warning for **Microsoft .NET Application Security** 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**, 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 a Customer Technology Preview 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

A side-by-side upgrade is one where 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.

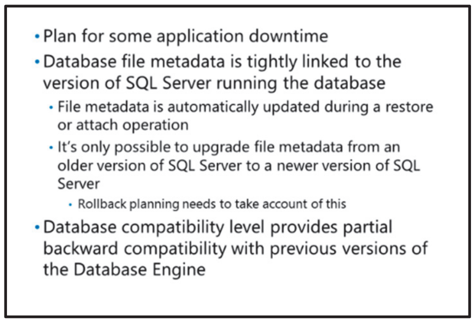
**Note:** The techniques for migrating databases discussed in this lesson are not only suitable for carrying out a side-by-side database upgrade; these same techniques can also be used to move databases between different instances of the same version of SQL Server.

### 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



- 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 of SQL Server
    - Rollback planning needs to take account of this
- Database compatibility level provides partial backward compatibility with previous versions of the Database Engine

### Application Down Time 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.

### 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.

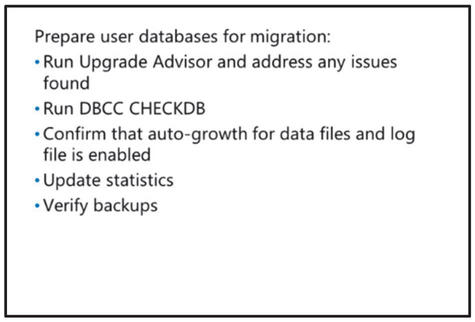
When you restore or attach a database to a new SQL Server instance, the database compatibility level remains unchanged until you manually modify it; you might do this if you want to start using new features. An administrator can modify the compatibility level of a database, either up or down, at any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

For more information on database compatibility level, see the topic *ALTER DATABASE Compatibility Level (Transact-SQL)* in Books Online:

**ALTER DATABASE Compatibility Level (Transact-SQL)**

http://aka.ms/dvkfr7

## Migrating SQL Server Databases



Prepare user databases for migration:
- Run Upgrade Advisor and address any issues found
- Run DBCC CHECKDB
- Confirm that auto-growth for data files and log file is enabled
- Update statistics
- Verify backups

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 auto-growth. This is to allow for additional space required during upgrade; if you wish, you can disable auto-growth 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.

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

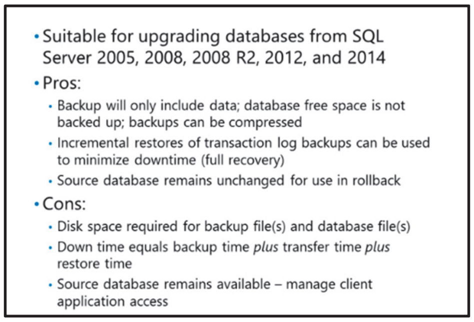
**Additional Reading:** For more information on installing SQL Server 2016, see Module 2 of this course: *Installing SQL Server 2016*.

Once 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



- 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)
  - Down time equals backup time *plus* transfer time *plus* restore time
  - Source database remains available – manage client application access

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 (once 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:

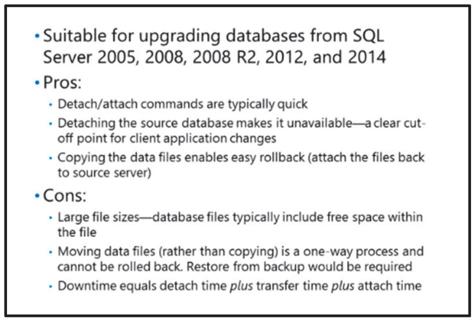
- Sufficient disk space must be available to store both the database backup and the database files, once they have been restored.
- If the database is not in full recovery mode, or you choose not to apply incremental backups, downtime for the upgrade will include time taken to run a final backup on the 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 Books Online:

**Copy Databases with Backup and Restore**

http://aka.ms/r6nsk4

## Using Detach and Attach to Migrate Databases



- 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

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 once 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.

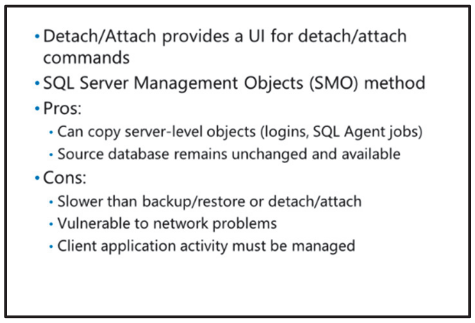
**Note:** If data files and log files for your SQL Server databases are held on a SAN volume, you might be able to save the time taken to copy files across the network by detaching or cloning the SAN volume from your old SQL Server and attaching it to your SQL Server 2016 hardware.
Cloning the SAN volume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

**Database Detach and Attach (SQL Server)**

http://aka.ms/nrxtmn

## Using the Copy Database Wizard to Migrate Databases


- 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

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 job, 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.

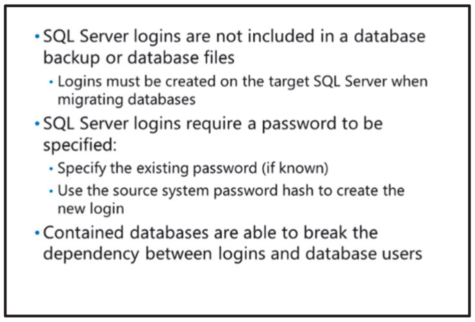
Similar advantages and disadvantages will apply for manually created steps, as for the Copy Database Wizard.

For more information, see the topic *Use the Copy Database Wizard* in Books Online:

**Use the Copy Database Wizard**

http://aka.ms/gmdzdz

## Transferring Logins and Passwords



- 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 are able to break the dependency between logins and database users

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.

**Additional Reading:** For a more detailed discussion of authentication and authorization in SQL Server, see course 20765: *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 are 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**, then right-click the login you want to script and click **Script Login as…** then click **CREATE To**, 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.

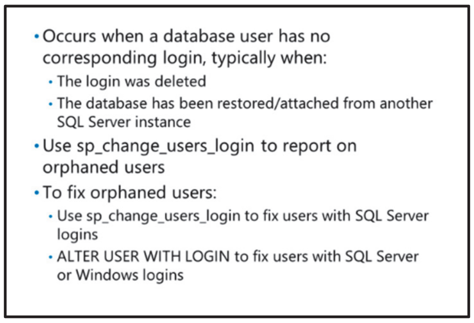
If you wish to use the same password for a SQL Server login on your source and target systems, two options are available:

- 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


- 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

If a database user is linked to a login which 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.

**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.

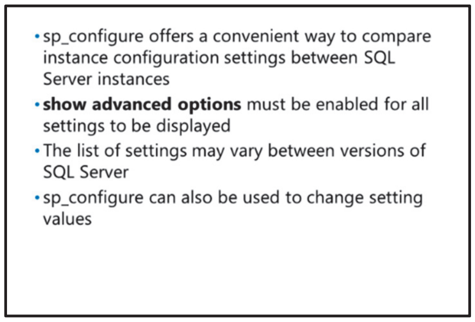
An orphaned user can also be repaired by creating a SQL Server login with a security identifier (SID) which matches the SID found in the user's definition (see the system view sys.database_principals for details). The user definition is attached to the login's SID, not the login name. The CREATE LOGIN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

For more information, see the topic *Troubleshoot Orphaned Users (SQL Server)* in Books Online:

**Troubleshoot Orphaned Users (SQL Server)**

http://aka.ms/grh3id

## Update Configuration with sp_configure



- 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 may vary between versions of SQL Server
- sp_configure can also be used to change setting values

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 will be able to 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.

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 Books Online:

**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 20765A-MIA-DC-UPGRADE and 20765A-MIA-SQL-UPGRADE virtual machines are running, and log on to 20765A-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. 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 |
|---|---|
| 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. True or false? | |

# 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: **20765A-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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are both running, and then log on to 20765A-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.ssmssln 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. Once 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. Once 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

- Raise the database compatibility level to SQL Server 2016 (compatibility level = 130).

**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

## Deploying SQL Server on Microsoft Azure

### Contents:

## Module Overview

Microsoft SQL Server® is integrated into Microsoft's cloud platform, Microsoft Azure.

SQL Server can be used in Azure in two ways:

- SQL Server running on Azure virtual machines.
- Azure SQL Database.

You will learn about the differences between these options for using SQL Server on 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 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 Virtual Machines on Azure

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.

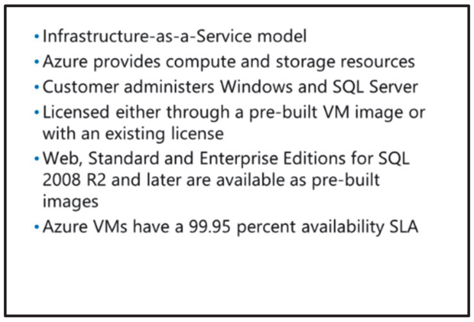
### Lesson Objectives

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

- Describe the options when you implement SQL Server on Azure VMs.
- Describe the architecture for VMs on Azure.
- Explain the architecture of Azure SQL Database.
- Describe the differences between, and the benefits of, VMs that host databases in Azure and Azure SQL Database.

### SQL Server on an Azure Virtual Machine

When you run SQL Server on an Azure VM, Microsoft provides compute and storage resources; you configure and administer Windows and SQL Server. SQL Server on Azure VMs falls into the Infrastructure-as-a-Service (IaaS) model of cloud services, and is conceptually similar to on-premises hardware virtualization.



- Infrastructure-as-a-Service model
- Azure provides compute and storage resources
- Customer administers Windows and SQL Server
- Licensed either through a pre-built VM image or with an existing license
- Web, Standard and Enterprise Editions for SQL 2008 R2 and later are available as pre-built images
- Azure VMs have a 99.95 percent availability SLA

Apart from the infrastructure on which it is hosted, running SQL Server on Azure VMs 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 VMs:

- **Platform-Provided SQL Server image**. In this model, you use a VM image provided by Azure, which includes a licensed copy of SQL Server. Licensed VM 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 VM 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 VM. 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.

Regardless of the license model you select, and in addition to any SQL Server licensing costs, you will be billed for:

- Windows Server usage
- Azure storage costs for the VM disks
- Outbound Internet traffic (from Azure data centers)

Other than usage costs, and the limitations imposed by the version and edition of SQL Server you select, there are no restrictions on SQL Server functionality when you use SQL Server on Azure VMs.

Azure VMs 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 VMs.

For more information on 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**

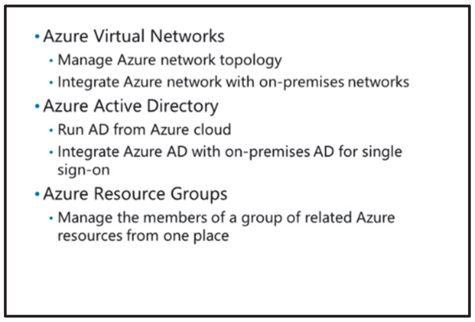
http://aka.ms/mb002x

For current information on pricing for Azure Virtual Machines and licensing costs for platform-provided SQL Server images, see *Virtual Machines Pricing* in the Azure documentation:

**Virtual Machines Pricing**

http://aka.ms/x7cmej

## Virtual Machines on the Azure Infrastructure

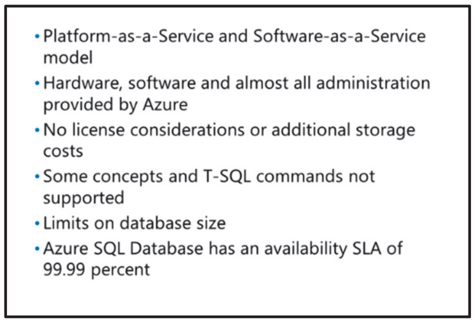

- Azure Virtual Networks
  - Manage Azure network topology
  - Integrate Azure network with on-premises networks
- Azure Active Directory
  - Run AD from Azure cloud
  - Integrate Azure AD with on-premises AD for single sign-on
- Azure Resource Groups
  - Manage the members of a group of related Azure resources from one place

A full discussion of the components of the Azure infrastructure, which you may configure when you use Azure VMs (whether they run SQL Server or any other software and services), is beyond the scope of this module. Even so, there are a number of these elements you should be aware of when planning to run SQL Server on Azure VMs, even if you decide not to make use of them, such as:

- **Azure Virtual Networks**. With these you can integrate your on-premises networks with networks in the Azure cloud and define complex network topologies between Azure VMs and services. You may consider using Azure Virtual Networks to facilitate client connectivity when you migrate existing databases to SQL Server on Azure VMs.
- **Azure Active Directory**. With this you can extend your on-premises Active Directory into the Azure cloud or host your Active Directory in Azure. You may want to consider using Azure Active Directory to have a single-sign on (SSO) for both on-premises and Azure services. Use this technology to connect to SQL Server instances on Azure VMs with Windows Authentication.
- **Azure Resource Groups**. You can use these to deploy and manage all the resources that make up an application. For example, you could use an SQL Server and web or application servers, as a group, rather than having to manage each resource individually. You might consider using Azure Resource Groups to simplify the management of applications and services where all the components have been migrated to run on Azure.

**Additional Reading:** For more information on Azure infrastructure, see Microsoft course 20533: *Implementing Microsoft Azure Infrastructure Solutions*.

## Azure SQL Database



- 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 T-SQL commands not supported
- Limits on database size
- Azure SQL Database has an availability SLA of 99.99 percent

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 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.

The billing cost of Azure SQL Database is based on the performance tier that is selected for the database. Service tiers will be covered in more detail in the next module.

In addition to the service tier cost, you will be billed for outbound Internet traffic.

Azure SQL Database does not have a version or edition directly comparable to SQL Server installations 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 (T-SQL).

Azure SQL Database places limits on maximum database size. This size limit varies by performance tier, with the highest service tier supporting databases up to a maximum size of 1 TB (one terabyte) at the time of writing.

The tools available to assist in the migration of databases to Azure SQL Database will be covered later in this module.

Azure SQL Database has a Service-Level Agreement (SLA) of 99.99 percent availability.

For a list of general restrictions for Azure SQL Database, see *Azure SQL Database General Limitations and Guidelines* in the Azure documentation:

**Azure SQL Database General Limitations and Guidelines**

http://aka.ms/fuyplz

For details of unsupported T-SQL features in Azure SQL Database, see *Azure SQL Database Transact-SQL differences* in the Azure documentation:

**Azure SQL Database Transact-SQL differences**

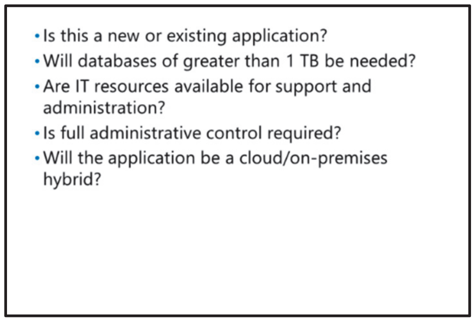
http://aka.ms/ybpqh8

For current information on 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?



- Is this a new or existing application?
- Will databases of greater than 1 TB 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 SQL Database and SQL Server on Azure VMs 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 SQL Servers 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 larger than 1 TB.
- 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 using 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 TB.

## 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, 20765A-MIA-DC, and 20765A-MIA-SQL VMs are running and log on to 20765A-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**, then click **Compute**, click **See all**, and then click **SQL Server**. In the **SQL Server** pane, click **SQL Server 2016 CTP3.3 Evaluation on Windows Server 2012 R2**.
5. In the **SQL Server 2016 CTP3.3 Evaluation on Windows Server 2012 R2** pane, in the **Select a deployment model** box, select **Resource Manager** and 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, **sql2016vm-js123**. Keep a note of the name you have chosen.
7. In the **User name** box, type **demoAdmin**.
8. In the **Password** box, type **Pa$$w0rd**.
9. In the **Resource group** box, type **resource1**.
10. Change the value of the **Location** box to a region near your current geographical location and click **OK**.
11. In the **Choose a size** blade, click **View all** then click **D11 Standard**, and then click **Select**.
12. On the **Settings** blade, click **OK**.
13. On the **SQL Server settings** blade, click **OK**.
14. On the **Summary** blade, click **OK**. Deployment may take some time to complete.
15. When deployment is complete, click **Virtual machines**, then click the name of the machine you created in step 6 above.
16. In the server name blade, click **Connect**, then click **Open**.
17. In the **Remote Desktop Connection** dialog box, click **Connect**.
18. Click **Use another account**, in the User name box, type **\demoAdmin**, then type **Pa$$w0rd** in the password box, then click **OK**, and then click **Yes**.

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.

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

| Statement | Answer |
|---|---|
| Azure SQL Database supports exactly the same set of functionality as an on-premises SQL Server installation. True or False? | |

# Lesson 2
# Deploying an Azure SQL Database

Before you use Azure SQL Database, it is important to understand the performance/pricing model used for the service.

Because Azure SQL Database is offered on a Software-as-a-Service (SaaS) model, the performance and, correspondingly, price of the service is measured in database transaction units (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 on pricing for Azure SQL Database, see *SQL Database Pricing* in the Azure documentation:

**SQL Database Pricing**

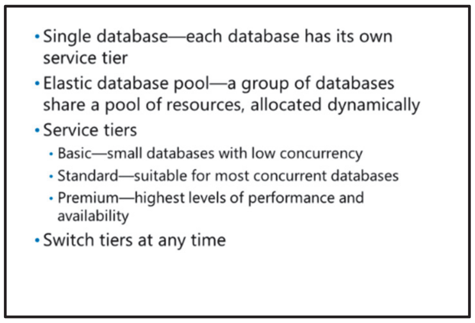
http://aka.ms/tskc9i

## Lesson Objectives

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

- Describe the performance tiers available for Azure SQL Database.
- Explain how tier performance is measured using DTUs.
- Provision and connect to an Azure SQL Database.

## SQL Database Performance Tiers



- 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 with low concurrency
  - Standard—suitable for most concurrent databases
  - Premium—highest levels of performance and availability
- Switch tiers at any time

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 offered for Azure SQL Database. In it, each database has its own service tier.

### Elastic database pool

Elastic database pool allows a number of databases to share a fixed pool of resources, with resources directed to the busiest databases as demands change. This is designed to address the scenario in which several Azure SQL Databases 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 performance tiers divide into three categories:

| Service Tier | Use-case |
|---|---|
| Basic | Simple, low usage databases with no concurrent users. |
| Standard | Databases with concurrent users that are not mission-critical. |
| Premium | Databases with 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.

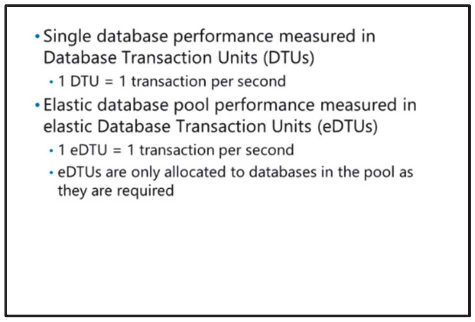
Each tier contains one or more performance levels that offer different numbers of DTUs. A database, or elastic database pool, can be moved between service tiers and performance levels at any time.

The details of Azure SQL Database performance tiers and levels changes over time. For current information on 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



- 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

DTUs are a measure of database performance used by Microsoft to establish performance metrics for Azure SQL Database in a single database service tier. The measure is based on the number of transactions 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.

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.

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 the eDTUs assigned to it are actively consumed.

For complete details of the benchmark tests 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 SQL Database

In this demonstration, you will see how to provision an Azure SQL Database by using Azure PowerShell.

### Demonstration Steps

1. On MIA-SQL, start **Windows PowerShell**. Link your Azure account to PowerShell by running the following cmdlet:

   ```
   Add-AzureRmAccount
   ```

   When prompted press **y**, 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
   ```

2. 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.)

3. 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
   ```

4. 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>"
   ```

5. 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.

6. 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
```

7. 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"
```

8. Close Windows PowerShell.

## Demonstration: Connecting to an Azure SQL Database

In this demonstration, you will see how to connect to an Azure SQL Database.

### Demonstration Steps

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 **SQL Databases**, then click **TestPSDB**. Note the value of **Server name**. Click **Show database connection strings**.
4. Open **SQL Server Management Studio**. In the **Connect to Server** dialog, type the server name noted in the previous step (it will take the form <your server name>.database.windows.net).
5. Set **Authentication** to **SQL Server Authentication**. Type **psUser** in the **Login:** box, then type **Pa$$w0rd** in the **Password:** box, and click **Connect**.
6. In **Object Explorer**, expand the **Databases** node to show the **TestPSDB** database. (If **Object Explorer** is not visible, display it by clicking **View**, then **Object Explorer**. Keyboard shortcut F8.)
7. Click **File**, then point to **Open**, and click **File**. Open **D:\Demofiles\Mod04\query Azure.sql**.
8. On the **Available Databases** dropdown menu, click **TestPSDB**.
9. Select the script below **2. Execute the following query**, and click **Execute**.
10. Open **Windows PowerShell**, and then type the command below **3. Open Windows PowerShell and type the following command** into the Windows PowerShell window, replacing <your server name> in the command with the server name used on step 4. At the **Password** prompt, type **Pa$$w0rd**, and then press Enter.

## Check Your Knowledge

| Question | |
|---|---|
| **Which of the following best describes the definition of a Database Transaction Unit?** | |
| Select the correct answer. | |
| | The Database Transaction Unit is a measure of CPU performance. |
| | The Database Transaction Unit is a measure of storage I/O performance. |
| | The Database Transaction Unit is a measure of memory performance. |
| | The Database Transaction Unit is a measure of overall system performance. |

# Lesson 3
# Migrating an On-Premises Database to an Azure SQL Database

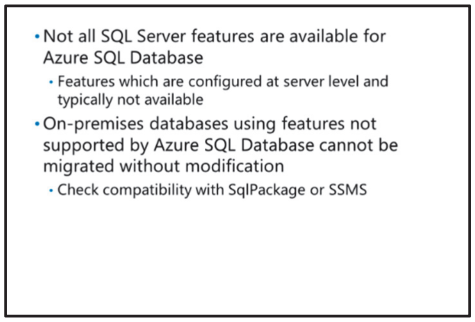
There are several tools available to assist you in migrating an on-premises database to an Azure SQL Database. Because Azure SQL Database doesn't fully support all of T-SQL functionality, the migration tools can also be used to identify whether an on-premises database is suitable for migration to Azure SQL Database without modification.

## Lesson Objectives

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

- Describe the Azure SQL Database compatibility options.
- List the migration options for moving an on-premises database to Azure SQL Database.

## Azure SQL Database Compatibility



- Not all SQL Server features are available for Azure SQL Database
  - Features which are configured at server level and typically not available
- On-premises databases using features not supported by Azure SQL Database cannot be migrated without modification
  - Check compatibility with SqlPackage or SSMS

Any database that runs on SQL Server 2005 or later may be suitable for migration to Azure SQL Database. However, the functionality of Azure SQL Database is a subset of the full functionality of SQL Server, because Azure SQL Database is only available at database level. There are also some restrictions due to the design of the Azure SQL Database service.

Important general limitations include:

- Windows Authentication is not supported. Note that, at the time of writing, authentication through Azure Active Directory, which can be federated with your on-premises Active Directory, is a preview feature. This offers a method to centrally manage database logins, although connection with a username and password is still required.
- Only TCP/IP connections are supported.
- SQL Server Agent is not supported. Note that you may connect an on-premises instance of the SQL Server Agent to Azure SQL Database.
- Distributed transactions are not supported.

For a full list of general restrictions for Azure SQL Database, see *Azure SQL Database General Limitations and Guidelines* in the Azure documentation:

**Azure SQL Database General Limitations and Guidelines**

http://aka.ms/fuyplz

Most, but not all, T-SQL language constructs are supported by Azure SQL Database. The list of features that are not supported includes:

- Cross-database queries
- Global temporary tables
- Linked servers
- OPENQUERY, OPENROWSET, OPENDATASOURCE, BULK INSERT
- .Net CLR integration

For full details of partially supported and unsupported T-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

An on-premises database that relies on any unsupported features is not a suitable candidate for migration to Azure SQL Database without modification. The migration tools can help you identify whether an on-premises database is dependent upon any of these features.

There are two ways to check the compatibility of an on-premises database:

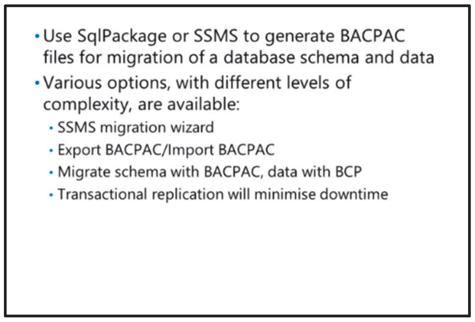
- **SqlPackage:** a command-line tool which can generate a report of detected compatibility issues.
- **SQL Server Management Studio**: the Export Data Tier Application wizard will report detected compatibility issues.

**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 SQL Server Books Online:

**SET TRANSACTION ISOLATION LEVEL (Transact-SQL)**

http://aka.ms/faim9a

## Migration Options



- Use SqlPackage or SSMS to generate BACPAC files for migration of a database schema and data
- Various options, with different levels of complexity, are available:
  - SSMS migration wizard
  - Export BACPAC/Import BACPAC
  - Migrate schema with BACPAC, data with BCP
  - Transactional replication will minimise downtime

To migrate an on-premises database to Azure SQL Database, you must duplicate its database schema and data to an empty Azure SQL Database before you change client connection strings to connect to the Azure SQL Database.

There are numerous ways to do this, but tools are available to make the process as simple as possible. These include:

- **SqlPackage:** a command-line tool which can generate migration files and execute migrations.

- **SSMS:** the Deploy Database to Microsoft Azure Database wizard, Export Data Tier Application wizard, and Import Data Tier Application wizard can generate migration files and execute migrations.

**Note:** Note that these are the same tools used to generate Azure SQL Database compatibility reports. When you generate a compatibility report, the tools are executed in an information-only mode. Compatibility checks are also carried out when a migration action takes place.

Depending on the size of your database and the acceptable duration of the service outage for migration, you may also opt to use general-purpose tools such as **bcp** and SQL Server Transactional Replication to assist with the process.

Both **SqlPackage** and SSMS use a file format called BACPAC, which contains a serialized copy of part or all of the database schema—and may optionally contain data from the exported tables. You select the schema objects and data to export into the BACPAC file when running the export tools.

Some alternative migration methods are:

- **SSMS Deploy Database to Microsoft Azure Database wizard**. This is the simplest option—the wizard will export all or part of a database schema to a BACPAC file, and then immediately import it to an Azure SQL Database. Because the export and import processes are serial, this is only suitable for small databases, or databases where long outage periods are acceptable.
- **Export to BACPAC then Import from BACPAC**. With this method, the export and import processes are run independently of one another, using either the SqlPackage tool or SSMS for the export and SqlPackage, SSMS, PowerShell, or the Azure portal for the import. This can improve reliability in scenarios where connectivity to Azure is limited or unreliable. It is considered suitable for medium to large databases.
- **Use BACPAC and BCP**. With this method, the database schema is exported from on-premises and imported to Azure SQL Database using a BACPAC file. Data is extracted and imported by using the **bcp** tool, which can transfer data in parallel. This method is more complex to manage and is only recommended for very large databases.

All of these methods result in a period of downtime while the database is migrated. In situations where the downtime period must be kept to a minimum, it is possible to use SQL Server Transactional Replication to duplicate schema and data from on-premises databases to Azure SQL Database. This is only supported where the transactional replication publisher and distributor run on specific patch levels of SQL Server 2012, SQL Server 2014 and SQL Server 2016.

For further information on methods for migrating databases to Azure SQL Database, see *SQL Server database migration to SQL Database in the cloud* in the Azure documentation:

**SQL Server database migration to SQL Database in the cloud**

http://aka.ms/h7fk4d

## Demonstration: Test Compatibility of an 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\Mod04 folder as Administrator.

2. Start a command prompt. Execute the following commands to test the TestPSDB database for compatibility with Azure SQL Database:

```
cd C:\Program Files (x86)\Microsoft SQL Server\130\DAC\bin
sqlpackage.exe /Action:Export /ssn:MIA-SQL /sdn:TestPSDB
/tf:D:\Demofiles\Mod04\TSQL.compatibility.bacpac /p:TableData=Stats.Tests
```

 The results of the compatibility test are returned.

## Demonstration: Migrate an 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 (or use the command prompt opened for the previous demonstration). 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\Mod04\TSQL.export.bacpac
```

2. Verify that the export BACPAC file exists at D:\Demofiles\Mod04\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:psUser
/tp:Pa^$^$w0rd /sf:D:\Demofiles\Mod04\TSQL.export.bacpac
```

   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.

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

| Statement | Answer |
|---|---|
| Azure SQL Database includes SQL Server Agent. True or False? | |

## Lab: Migrating SQL Server with Azure

### Scenario

As a pilot project for Adventure Works Cycles, you have been tasked with the migration of an SQL Server database, 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: **20765A-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, 20765A-MIA-DC, and 20765A-MIA-SQL VMs are both running, and then log on to 20765A-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: 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 which 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 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. 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, seen the reasons 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 VMs more suitable for you?

# Module 5

## Working with Databases

### Contents:

## Module Overview

One of the most important roles for database administrators working 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

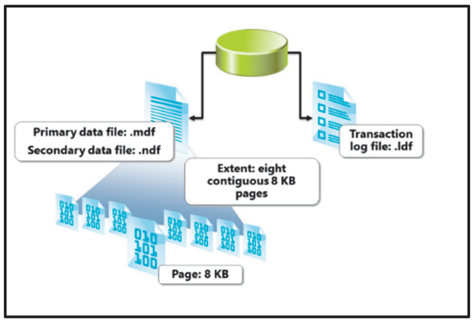
To effectively create and manage databases, you must understand SQL server files, file location, planning for growth, and how data is stored.

### 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


Primary data file: .mdf
Secondary data file: .ndf
Transaction log file: .ldf
Extent: eight contiguous 8 KB pages
Page: 8 KB

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, with data records being written to pages within those files.

#### Database Files

There are three types of database file 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 simply 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 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 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 is 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 uniquely identify a page in a database, 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*

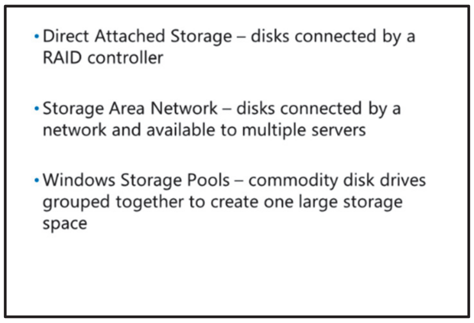
Groups of eight contiguous pages are referred to as an extent. SQL Server uses extents to simplify the management of data pages. There are two types of extents:

- **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 are 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

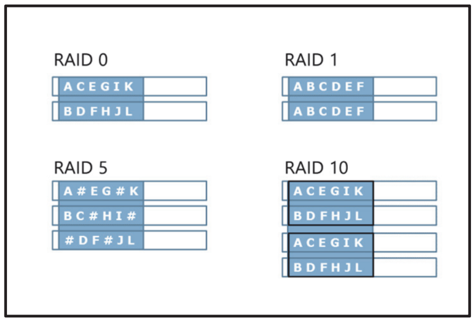


- Direct Attached Storage – disks connected by a RAID controller
- Storage Area Network – disks connected by a network and available to multiple servers
- Windows Storage Pools – commodity disk drives grouped together to create one large storage space

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 (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 may 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 in terms of disk space 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 non-fault 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:

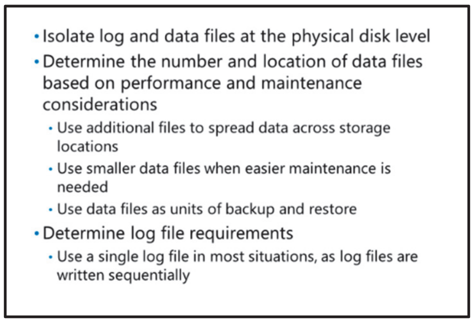
- Generally, RAID 10 offers the best combination of read/write performance and fault tolerance, but is the most costly solution.
- Write operations on RAID 5 can sometimes be relatively slow compared to RAID 1, because of the need to calculate parity data (RAID 5). If you have a high proportion of write 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



- 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, as log files are written sequentially

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, as 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.

#### *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.

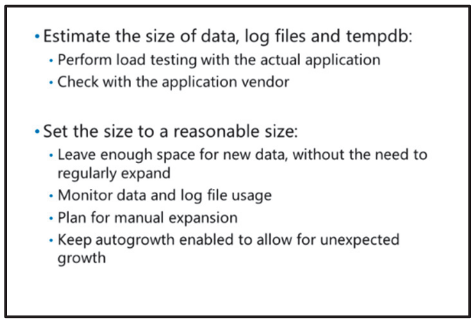
Allocating multiple data files provides a number of management advantages, including:

- 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



- 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

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 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.

### 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 is 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 as 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, as doing so may 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 | |
|---|---|
| **Which of these storage options does the following statement most closely refer to? “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

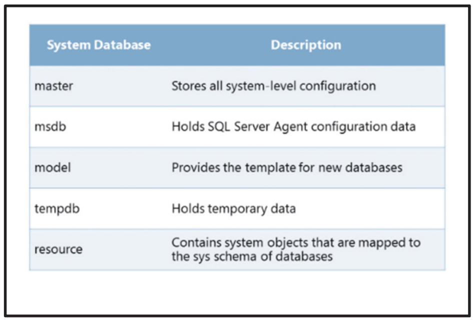
SQL Server uses system databases to maintain internal metadata. Database administrators should be familiar with the SQL Server system databases and how to manage them.

## 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



| 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 |

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.

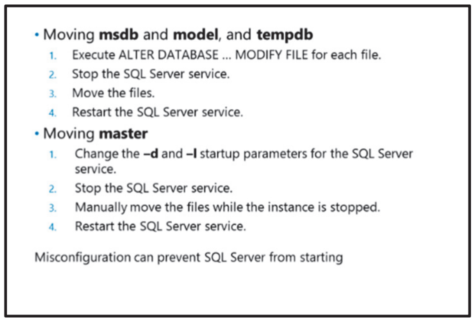
### tempdb

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.

### 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



- 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

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 as, 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:

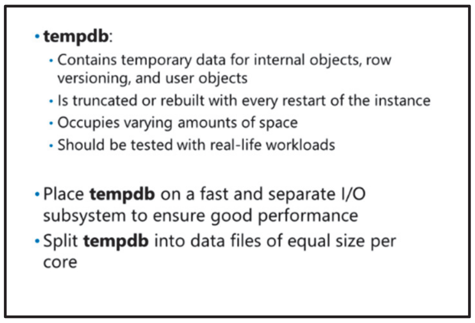
1. For each file to be moved, execute the ALTER DATABASE ... MODIFY FILE statement.
2. Stop the instance of SQL Server.
3. Move the files to the new location (this step is not necessary for tempdb, as its files are recreated automatically on startup—though you should delete the old tempdb files after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

## Considerations for tempdb



- **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

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.

**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 multi-statement 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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are running, and log on to 20765A-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 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** 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\Mod05 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**. 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**.
10. View the contents of **T:\** and note that the **tempdb.mdf** and **templog.ldf** files have been moved to this location.
11. 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 non-system 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.

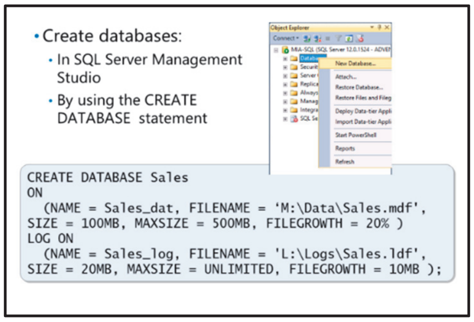
When creating databases, you should also consider where the data and logs will be stored on the file system. You may also want to change this or provide additional storage when the database is in use. When databases become larger, the data should be allocated across different volumes, rather than storing 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



- Create databases:
  - In SQL Server Management Studio
  - By using the CREATE DATABASE statement

```
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 );
```

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:

**Using the CREATE DATABASE statement**

```
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

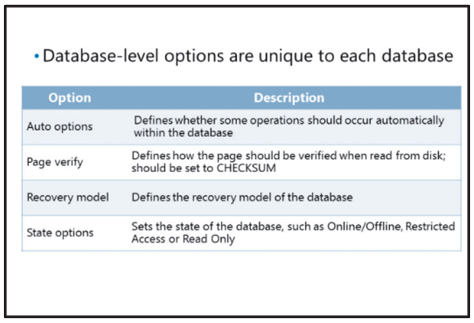
To **delete** (or "drop") a database, right-click it in Object Explorer and click **Delete** or use the DROP DATABASE Transact-SQL statement. Dropping a database automatically deletes all of its files.

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

**Dropping a database.**

```
DROP DATABASE Sales;
```

## Configuring Database Options



- 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 |

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 impact on any others. All database options are initially 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.

### Categories of Options

There are several categories of database options:

- **Auto Options**. 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**. 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**. 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 20765A: *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

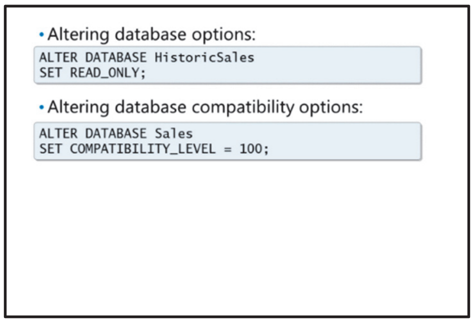
Create a Database by Using SQL Server Management Studio

1. Ensure that you have completed the previous demonstration. If not, start the 20765A-MIA-DC and 20765A-MIA-SQL virtual machines, log on to 20765A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**, and run D:\Demofiles\Mod05\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. Then change the **Path** and **File Name** by typing the following values:
    - **DemoDB1**:
        - **Path**: M:\Data\
        - **File Name**: DemoDB1.mdf
    - **DemoDB1_log**:
    - **Path**: L:\Logs\
    - **File Name**: DemoDB1.ldf
6. Click **OK** to create the new database.
7. Expand the **Databases** folder, and then right-click **DemoDB1** and click **Properties**.
8. On the **Options** tab, review the database options. 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\Mod05 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**. Then view the information that is returned.
4. Keep SQL Server Management Studio open for the next demonstration.

## Altering User Databases



- Altering database options:

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

- Altering database compatibility options:

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

You may need to modify a database when it is in use; for example, you might need to change the name or 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, 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 <option>**

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

**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 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.

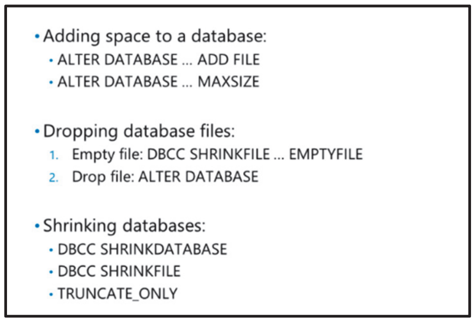
**ALTER DATABASE … SET COMPATIBILITY_LEVEL**

```
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



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

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

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 may have 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 may 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, as 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 which 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.

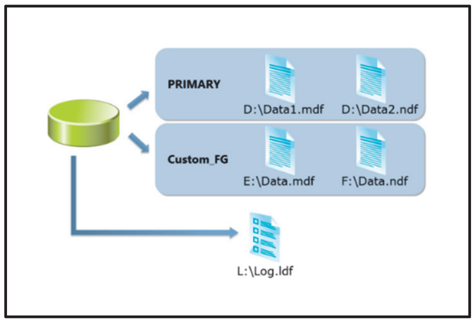
For more information on

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 may 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 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.

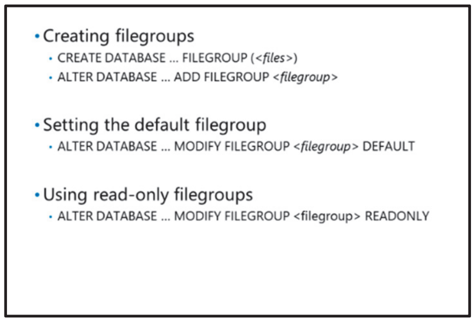
**Note:** Generally, you should use filegroups primarily to improve manageability and rely on storage device configuration for I/O performance. However, when a striped storage volume is not available, using a filegroup to spread data files across physical disks can be an effective alternative.

For more information on database files and filegroups, see:

**Database Files and Filegroups**

http://aka.ms/eql7p2

## Creating and Managing Filegroups



- 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

As a database administrator, you may 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 a Database with Multiple 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 = 'SatesTrans2', 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.

The following code example changes the default filegroup in the **Sales** database to the **Transactions** filegroup:

**Changing the Default Filegroup**

```
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:

**Making a Filegroup Read-Only**

```
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 |
|---|---|
| Database files can belong to many filegroups. True or false? | |

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

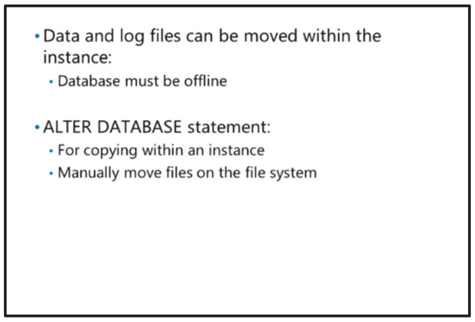
In addition to adding and removing files from a database, you may sometimes need to move database files, or even whole databases. You may also need to copy a database.

## 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



- 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

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.

### Using the ALTER DATABASE Statement

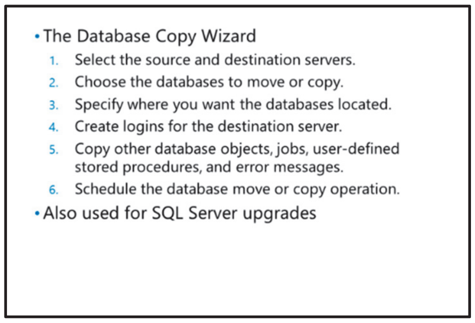
You can use the ALTER DATABASE statement to move database files within the same instance of SQL Server by including the MODIFY FILE clause in the statement.

The following example shows how to move the data file for the AdventureWorks database:

**Moving Database Files by Using the 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


- 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

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. This is covered in another module.

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.

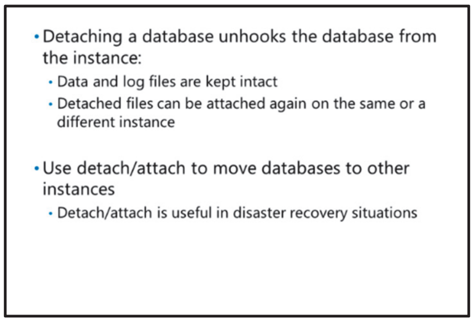
**Note:** You can also copy database metadata such as login information and other necessary **master** database objects.

For more information on using the Copy Database Wizard, see:

**Use the Copy Database Wizard**

http://aka.ms/gmdzdz

## Detaching and Attaching Databases


- 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

SQL Server provides you with detach and attach functionality which 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 simply 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.

### 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

SSMS also gives you the ability to attach databases. You can also do this by using the CREATE DATABASE … FOR ATTACH statement.

**Note:** You may find many references to the **sp_attach_db** and b**p_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*. You will see how to deal with this problem in a later module.

## 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**, 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**. Then connect to the **MIA-SQL\SQL2** database engine using Windows authentication.
2. In Object Explorer, under **MIA-SQL\SQL2**, expand **Databases** and view the databases on this instance.
3. In Object Explorer, under **MIA-SQL\SQL2**, right-click **Databases** and click **Attach**.
4. In the **Attach Databases** dialog box, click **Add**.
5. In the **Locate Database Files** dialog box, select the **M:\Data\DemoBD2.mdf** database file, then 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. Then click **OK**.
7. 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 |
|---|---|
| Databases that are configured for replication, mirrored, or in a suspect state, cannot be detached. True or false? | |

# Lesson 5
# Configuring the Buffer Pool Extension

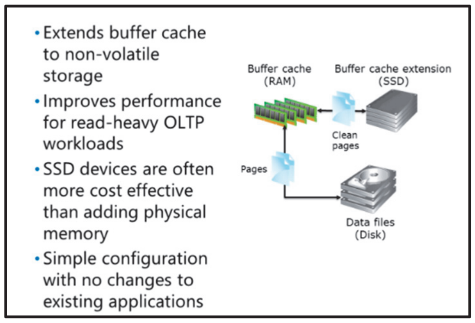
The topics in this module so far have discussed the storage of system and user database files. However, SQL Server also supports the use of high performance storage devices, such as solid-state disks (SSDs) to extend the buffer pool (the cache used to modify data pages in-memory).

### 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



- Extends buffer cache to non-volatile storage
- Improves performance for read-heavy OLTP workloads
- SSD devices are often more cost effective than adding physical memory
- Simple configuration with no changes to existing applications

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 non-volatile storage devices, such as Solid-State Disk (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 can be enabled easily, and requires no changes to existing applications.

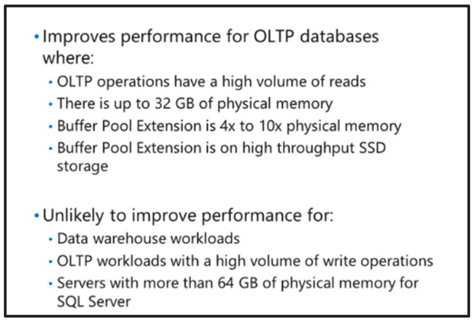
**Note:** The Buffer Pool Extension is not supported in all SQL Server 2016 versions.

For more information about Buffer Pool Extension, see:

**Buffer Pool Extension**

http://aka.ms/pnqvzg

## Considerations for Using the Buffer Pool Extension



- 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

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:

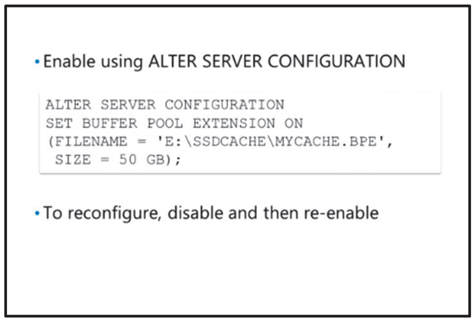
- Data warehouse workloads.
- OLTP workloads with a high volume of write operations.
- Servers on which more than 64 GB of physical memory is available to SQL Server.

### 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 may 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



- 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 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:

**Enabling the Buffer Pool Extension**

```
ALTER SERVER CONFIGURATION
SET BUFFER POOL EXTENSION ON
(FILENAME = 'E:\SSDCACHE\MYCACHE.BPE', SIZE
= 50 GB);
```

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 may 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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines, log on to 20765A-MIA-SQL as **ADVENTUREWORKS\**Student with the password **Pa$$w0rd**, and run D:\Demofiles\Mod05\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.sq**l in the D:\Demofiles\Mod05 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. Select the code under the comment **View buffer pool extension** details, then click **Execute**.
3. Use File Explorer to view the contents of the **T:\** folder and note that the **MyCache.bpe** file has been deleted.
4. 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, note that buffer pool extension is disabled.

## Check Your Knowledge

| Question | |
|---|---|
| **Which of these statements is not true about Buffer Pool Extensions?** | |
| Select the correct answer. | |
| | Extends buffer cache to non-volatile 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: **20765A-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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are both running, and then log on to 20765A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.

1. Run **Setup.cmd** in the D:\Labfiles\Lab05\Starter folder as **Administrator**.

1. Ensure that the 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are both running, and then log on to 20765A-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: 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 Nam**e: 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

1. 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

1. 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 |

| Logical Name | Filegroup | Initial Size | Growth | Path |
|---|---|---|---|---|
| InternetSales_data1 | SalesData | 100MB | 10MB / Unlimited | N:\Data\InternetSales_data1.ndf |
| 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\Lab05\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 includes 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 AWDataWarehouse Database

2. Configure Filegroups

### ▶ Task 1: Attach the AWDataWarehouse Database

1. Move **AWDataWarehouse.ldf** from the D:\Labfiles\Lab05\Starter\ folder to the L:\Logs\ folder, and then move the following files from the D:\Labfiles\Lab05\Starter\ folder to the M:\Data\ folder:
   - AWDataWarehouse.mdf
   - AWDataWarehouse_archive.ndf
   - AWDataWarehouse_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.

**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 6

## Database Storage Options

### Contents:

## 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 impacts 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

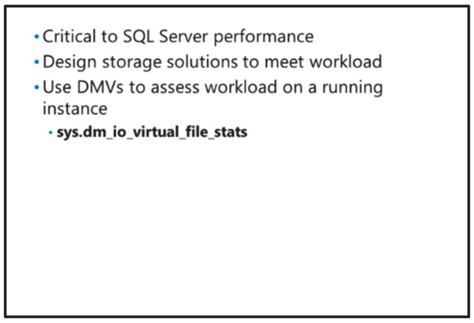
In this lesson, you will learn about different options available for SQL Server storage and how they may impact performance.

## 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



- 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**

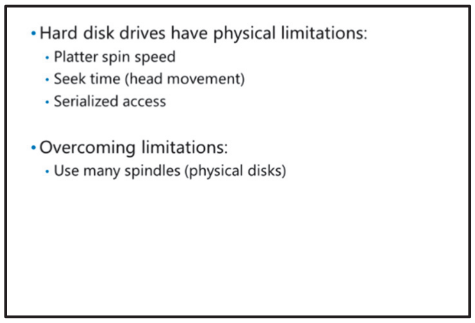
I/O performance has a direct impact on the overall performance of SQL Server, making it critical that the storage layer is designed and implemented in a way that delivers the best performance for the server's anticipated workload.

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.

## Number of Spindles



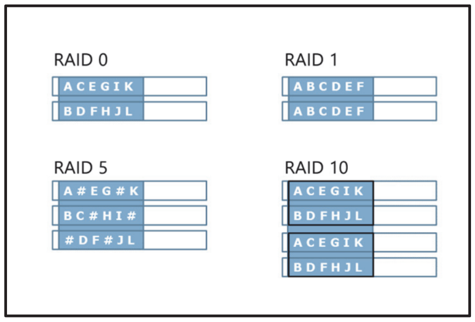
- Hard disk drives have physical limitations:
  - Platter spin speed
  - Seek time (head movement)
  - Serialized access
- Overcoming limitations:
  - Use many spindles (physical disks)

All hard disks have physical limitations that impact 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, may 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.

## 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 may be supported by third-party SANs.

Commonly used types of RAID include:

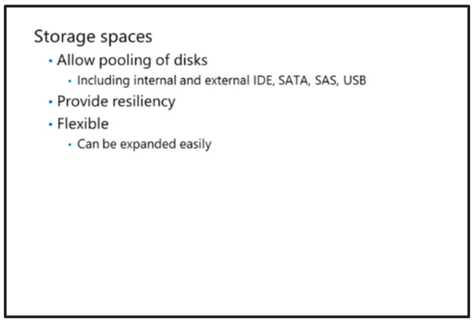
- **RAID 0, disk striping**. A stripe set consists of space from two or more disks that is combined into a single volume. The data is distributed evenly across all of the disks, which improves I/O performance, particularly when each disk device has its own hardware controller. RAID 0 offers no redundancy, 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 non-fault 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
- Allow pooling of disks
  - Including internal and external IDE, SATA, SAS, USB
- Provide resiliency
- Flexible
  - Can be expanded easily

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 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.
- **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.

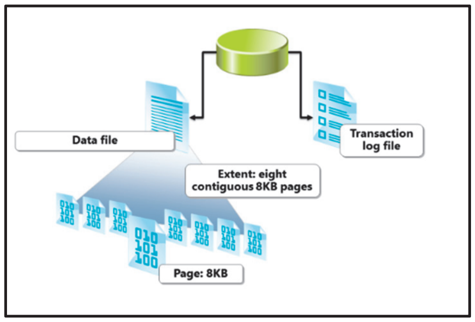
For more information on Storage Spaces, see:

**Storage Spaces Overview**

http://aka.ms/dyg4dk

## 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*

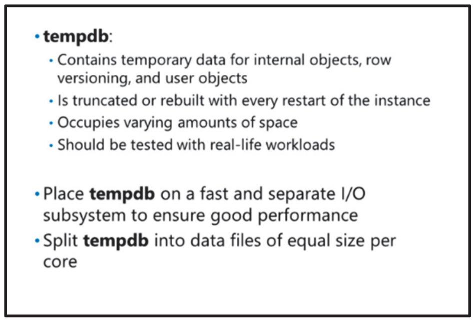
Groups of eight contiguous pages are referred to as an extent. SQL Server uses extents to simplify the management of data pages. There are two types of extents:

- 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



- **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

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.

- **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 multi-statement 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

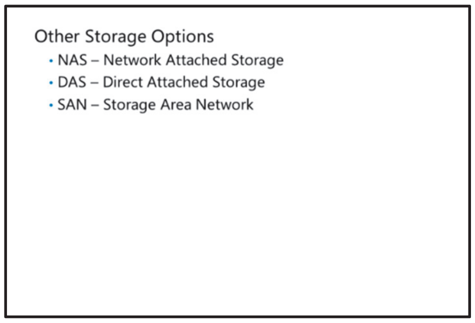
In this demonstration, you will see how to move **tempdb** files.

### Demonstration Steps

1. Ensure that the 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are running, and log on to 20765A-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 **tempdb.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
- NAS – Network Attached Storage
- DAS – Direct Attached Storage
- SAN – Storage Area Network

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.

**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

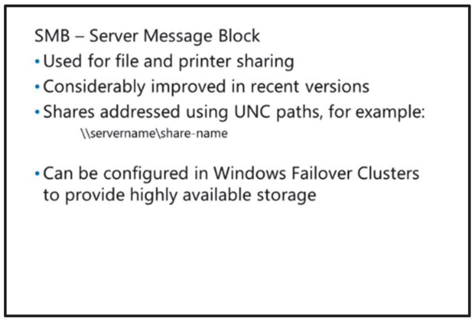
In this lesson, you will learn how about Server Message Block (SMB) Fileshare storage and the benefits of using it with SQL Server.

### 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?



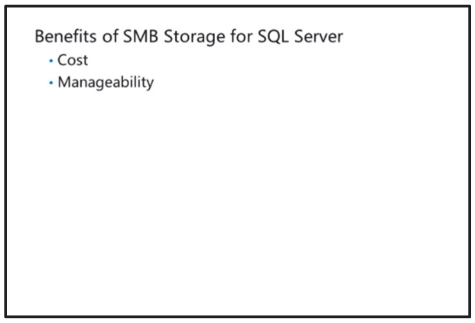
SMB – Server Message Block
- Used for file and printer sharing
- Considerably improved in recent versions
- Shares addressed using UNC paths, for example:
  \\servername\share-name
- Can be configured in Windows Failover Clusters to provide highly available storage

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 may not have 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.

### Benefits of SMB Fileshare

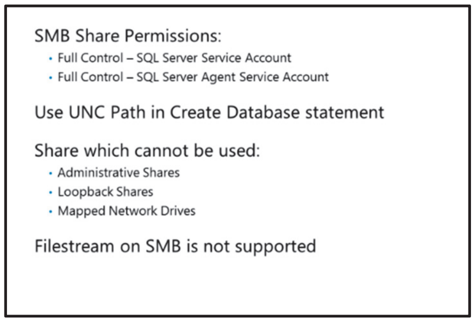


Benefits of SMB Storage for SQL Server
- Cost
- Manageability

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.

## Implementing SMB Fileshare


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

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.

The example below shows a CREATE DATABASE statement using an SMB share for storage:

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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are running, and log on to 20765A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Open Windows 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. Open the file **SMBDemo.sql** located in the D:\Demofiles\Mod06 folder and execute the code it contains.
5. In Windows 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. | |
| | CREATE DATABASE [Sales]<br>ON  PRIMARY<br>( NAME = N'Sales_Data', FILENAME =<br>N'\\SMBServer\d$\SMBShare\Sales_data.mdf' )<br>LOG ON<br>( NAME = N'Sales_Log', FILENAME =<br>N'\\SMBServer\d$\SMBShare\Sales_Log.ldf')<br>GO |
| | CREATE DATABASE [Sales]<br>ON  PRIMARY<br>( NAME = N'Sales_Data', FILENAME = N'\\127.0.0.1\SMBShare\Sales_data.mdf' )<br>LOG ON<br>( NAME = N'Sales_Log', FILENAME = N'\\127.0.0.1\SMBShare\Sales_Log.ldf')<br>GO |
| | CREATE DATABASE [Sales]<br>ON  PRIMARY<br>( NAME = N'Sales_Data', FILENAME = N'\\SMBServer\SMBShare\Sales_data.mdf'<br>)<br>LOG ON<br>( NAME = N'Sales_Log', FILENAME = N'\\SMBServer\SMBShare\Sales_Log.ldf')<br>GO |

# Lesson 3
# SQL Server Storage in Microsoft Azure

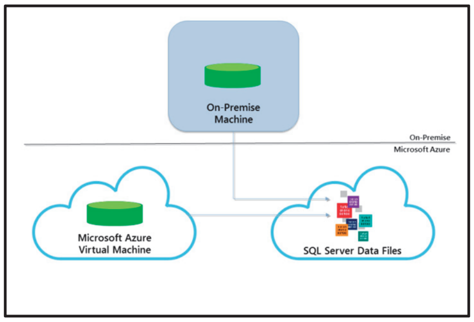
You can use Microsoft Azure to store SQL Server data files in the cloud. In this lesson, you will learn about the benefits of storing SQL Server data files in the cloud and how to implement this technology.

## 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 which store data files in Microsoft Azure storage as Microsoft Azure page blobs. On-premise 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-premise solutions; simplifies the provisioning of database storage; and facilitates the easy migration of databases between machines.

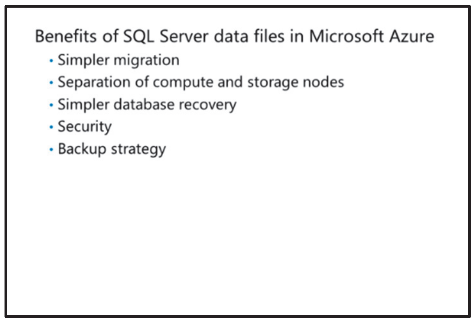
To use SQL Server data files in Microsoft Azure, you create a storage account, along with a container. You then create a SQL Server credential containing policies and the shared access signature essential to access the container. You can store page blobs inside the container, with each having a maximum 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

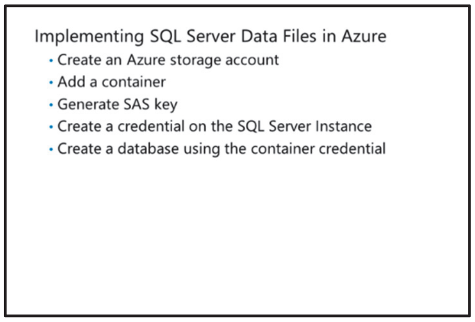
## Benefits of SQL Server Data Files in Microsoft Azure


Benefits of SQL Server data files in Microsoft Azure
- Simpler migration
- Separation of compute and storage nodes
- Simpler database recovery
- Security
- Backup strategy

The main benefits of storing SQL Server data files in Microsoft Azure are:

1. **Simpler Migration**: The migration of databases between on-premise 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-premise 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-premise 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.

## Implementing SQL Server Data Files in Azure


Implementing SQL Server Data Files in Azure
- Create an Azure storage account
- Add a container
- Generate SAS key
- Create a credential on the SQL Server Instance
- Create a database using the container credential

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. Example code is below:

```
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. An example CREATE DATABASE statement is below:

```
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 as the data is sourced from the Internet. |

# Lesson 4
# Stretch Database

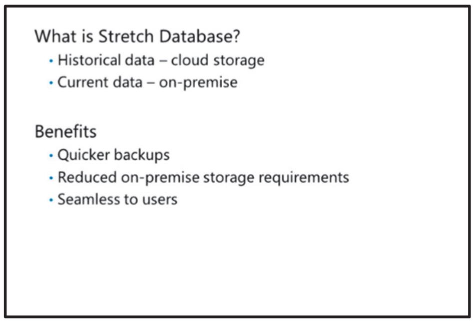
In this lesson, you will learn about Stretch Database, a feature you can use for the migration of historical, or cold, data to the cloud.

## 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?



What is Stretch Database?
- Historical data – cloud storage
- Current data – on-premise

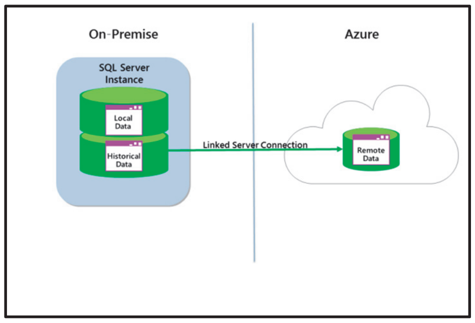
Benefits
- Quicker backups
- Reduced on-premise storage requirements
- Seamless to users

Stretch Database is a feature of SQL Server 2016 which means data can be split between on-premise storage and cloud storage. With Stretch Database, cold, historical data is kept in the cloud and active data is kept on-premise 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-premise storage requirements both for data and associated backups. Backups of on-premise 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 may be a small amount of additional latency associated with queries.

## Stretch Database Architecture



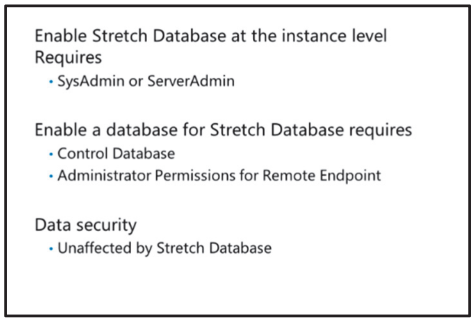
With Stretch Database, the storage and query processing for cold data is offloaded to the Microsoft Azure cloud. When you enable Stretch Database for a database, SQL Server creates a secure linked server that uses the remote Microsoft Azure instance as the endpoint. Enabling Stretch Database for a table 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-premise 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 may notice an increase in latency when querying data hosted in Azure.

## Stretch Database Security



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

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

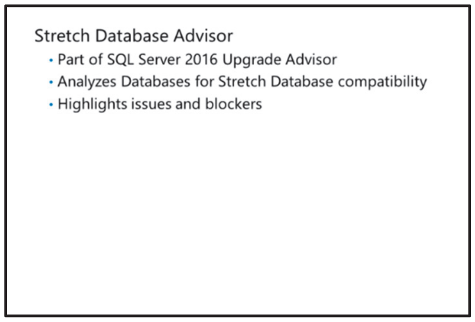
### Data Security

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.

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.



Stretch Database Advisor

- Part of SQL Server 2016 Upgrade Advisor
- Analyzes Databases for Stretch Database compatibility
- Highlights issues and blockers

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 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.

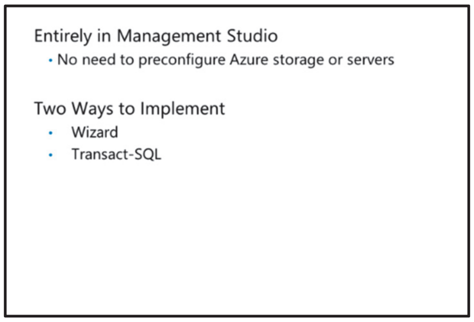
For more details on tables that may not be compatible with Stretch Database, see:

**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.

## Implement a Stretch Database



Entirely in Management Studio
- No need to preconfigure Azure storage or servers

Two Ways to Implement
- Wizard
- Transact-SQL

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

**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-premise 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: **20765A-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-premise 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 20765A-MIA-DC, 20765A-MIA-SQL and MSL-TMG1 virtual machines are running, and then log on to 20765A-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

1. 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

### Review Question(s)

**Question:** What are the advantages of SMB Fileshare storage over SAN storage?

# Module 7

## Performing Database Maintenance

### Contents:

## 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.

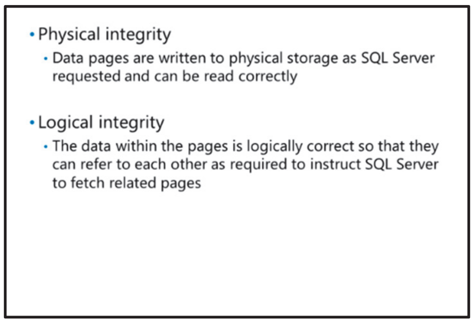
You can use the DBCC CHECKDB command to detect, and in some circumstances correct, database corruption. It is therefore important that you are familiar with how DBCC CHECKDB works.

### 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



- 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

There are two levels of database integrity, termed as follows:

- **Physical integrity**: Ensures that SQL Server can read and write data extents to the physical media.
- **Logical integrity**: 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.

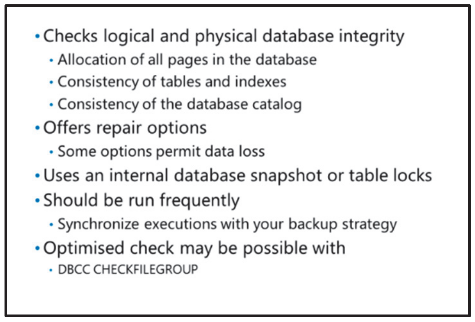
Without regular checking of the database files, any lack of integrity of the database might lead to bad information derived from it. Backup does *not* check the integrity—it only checks the page checksums—and that is only when you use the WITH CHECKSUM option on the BACKUP command. Although the CHECKSUM 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 that, if it's not checked as it's backed up, 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



- 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 may be possible with
  - DBCC CHECKFILEGROUP

SQL Server provides the DBCC utility, which supports a range of management facilities. In earlier documentation, you may see it referred to as the Database Consistency Checker. Although checking the consistency of databases, using the CHECKDB option, 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.

### 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, it is not always possible to 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 onwards 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 upon 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 provides an ESTIMATEONLY option.

### Backups and DBCC CHECKDB

It is 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 needing to be used around the clock, this may not be convenient; in this case, you should run the checks against a restored backup of the system instead.

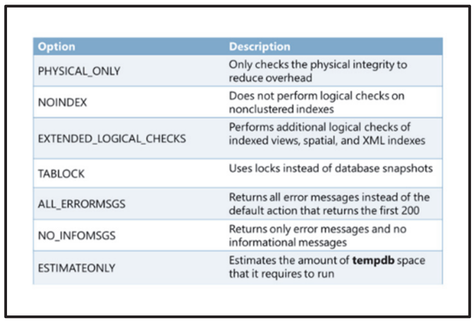
### MAXDOP override

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 non-user 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 non-optimal 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



| 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_ERRORMSGS | Returns all error messages instead of the default action that returns the first 200 |
| NO_INFOMSGS | Returns only error messages and no informational messages |
| ESTIMATEONLY | Estimates the amount of **tempdb** space that it requires to run |

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.

**NOINDEX**

You can use the NOINDEX option to specify not to perform the intensive checks of non-clustered 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 non-clustered 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.

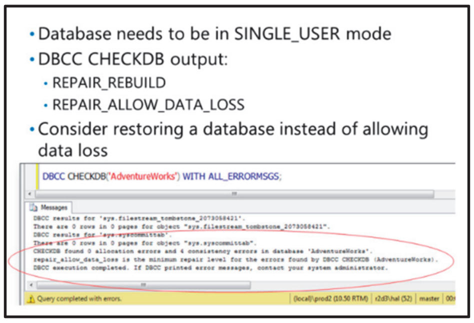
**ALL_ERRORMSGS and NO_INFOMSGS**

The ALL_ERRORMSGS 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. This way you can find out how much space the utility will need and avoid the risk of the process terminating part way through because **tempdb** is out of space.

## DBCC CHECKDB Repair Options



- Database needs to be in SINGLE_USER mode
- DBCC CHECKDB output:
  - REPAIR_REBUILD
  - REPAIR_ALLOW_DATA_LOSS
- Consider restoring a database instead of allowing data loss

```
DBCC CHECKDB('AdventureWorks') WITH ALL_ERRORMSGS;
```

Messages

```
DBCC results for 'sys.filestream_tombstone_2073058421'.
There are 0 rows in 0 pages for object "sys.filestream_tombstone_2073058421".
DBCC results for 'sys.syscommittab'.
There are 0 rows in 0 pages for object "sys.syscommittab".
CHECKDB found 0 allocation errors and 4 consistency errors in database 'AdventureWorks'.
repair_allow_data_loss is the minimum repair level for the errors found by DBCC CHECKDB (AdventureWorks).
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
```

Query completed with errors. | (local)\prod2 (10.50 RTM) | r2d3\hal (52) | master

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. You should back up a database before performing any repair option, if you can, and also find and resolve the reason for the corruption—otherwise it could well happen again soon after.

In addition to providing details of errors that have been found, 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** 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** 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 may 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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are running, and log on to 20765A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Demofiles\Mod07 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\Mod07\ 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 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

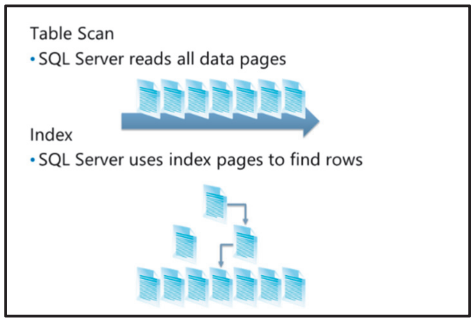


Table Scan
• SQL Server reads all data pages
Index
• SQL Server uses index pages to find rows

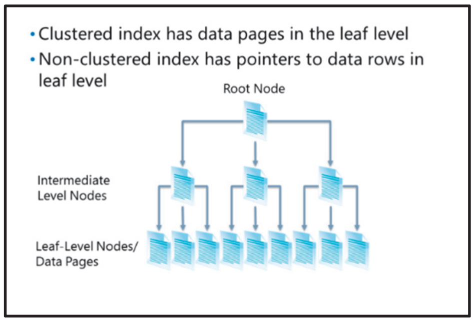
Whenever SQL Server needs to access data in a table, it calculates whether to read all the pages (known as a table scan) or use a seek operation to pick up individual pages, reducing the amount of effort required to locate the relevant rows.

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



- Clustered index has data pages in the leaf level
- Non-clustered index has pointers to data rows in leaf level

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 non-clustered index. For each clustered index, you can have many non-clustered 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 can 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 rest 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.

**Non-clustered Indexes**

A non-clustered 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 non-clustered 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 contain the actual data pages of the table as well.

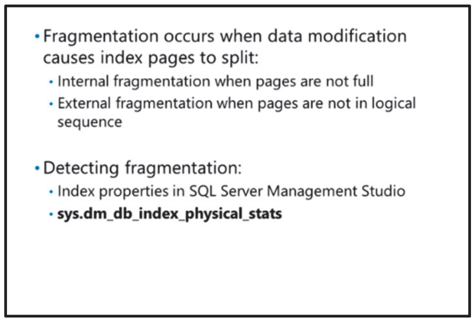
Index and data pages are linked and double-linked within a logical hierarchy across all pages at the same level of the hierarchy. This assists SQL Server to perform partial scan operations across an index, from a given start point, that is located using a seek operation.

**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.

## Index Fragmentation



- 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**

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.

**Internal vs. External Fragmentation**

There are two types of index fragmentation:

Internal fragmentation occurs when pages are not completely full of data. This often occurs 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 occurs when pages get 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 dynamic management view. 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**

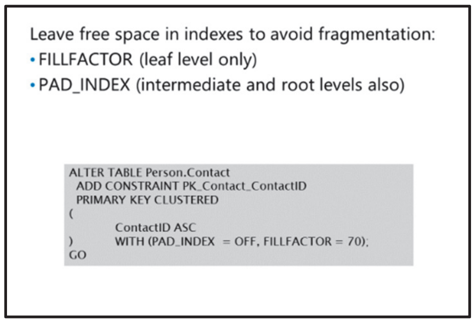
```
SELECT * FROM sys.dm_db_index_physical_stats
(  DB_ID() --Current database
,   OBJECT_ID('HumanResources.Employee') -- Object needing checking
,  NULL -- Index_id
,  NULL -- partition_number
,  'DETAILED' -- as opposed to LIMITED OR SAMPLED
);

-- 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/lzvjmq

SQL Server Management Studio (SSMS) also provides details of index fragmentation in the properties page for each index.

## FILLFACTOR and PAD_INDEX



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
```

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
```

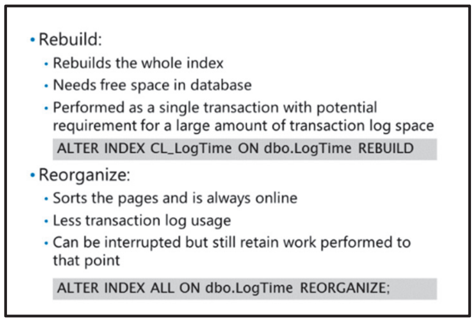
**Note:** The same action of the values 0 and 100 can seem confusing. While both lead to the same outcome, 100 indicates that a specific FILLFACTOR value is being set. The value zero indicates that no FILLFACTOR has been specified and that the default server value should be applied.

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 be allocated in the non-leaf levels of the index as well.

Although the use of the fill factor to pre-allocate space in the index pages can help in some situations, it is not a panacea for good performance; it may 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 will be able to update rows in place too, 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 get as much useful data when it needs to process it.

## Ongoing Maintenance of Indexes



- 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;`

When SQL Server updates indexes during data modifications, the index storage can fragment. SQL Server provides two options for removing fragmentation from clustered and non-clustered 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 ensure 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 non-clustered 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.

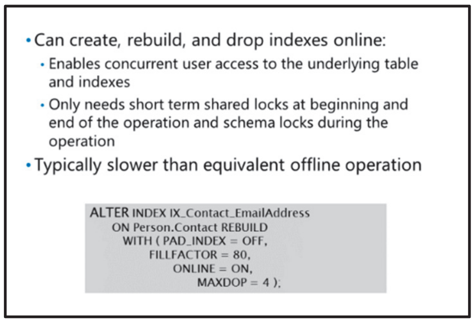
For heavily fragmented indexes (those with fragmentation greater than 30 percent) rebuilding is usually the most appropriate option to use. SQL Server maintenance plans include options to rebuild or reorganize indexes. If you do not use maintenance plans, it is important to create a job that regularly 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;
```

## Online Index Operations



- 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 );
```

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:

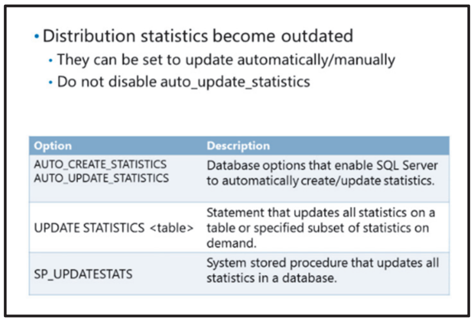
**Rebuilding an Index Online**

```
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



- 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. |

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 non-indexed columns when you enable the AUTO_CREATE_STATISTICS database option. You enable and disable this option by using the ALTER DATABASE statement.

The purpose of Statistics are 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
```

SQL Server automatically updates statistics when AUTO_UPDATE_STATISTICS is set. This is the default setting and it is best that you do not disable this option unless you really have to, because 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\Mod07\ 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 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.

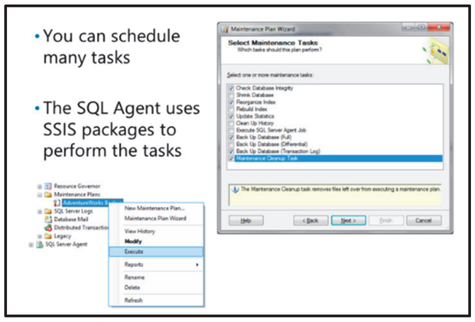
Although the Maintenance Plan Wizard makes this process easy to set up, it is important to realize that you can use the output of the wizard as a starting point for creating your own maintenance plans. Alternatively, you can create plans from scratch, using the SQL Server DMOs or system stored procedures.

### 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 schedule many tasks
- The SQL Agent uses SSIS packages to perform the tasks

Maintenance Plan Wizard

Select Maintenance Tasks
Which tasks should this plan perform?

Select one or more maintenance tasks:

- [✓] Check Database Integrity
- [ ] Shrink Database
- [✓] Reorganize Index
- [ ] Rebuild Index
- [✓] Update Statistics
- [ ] Clean Up History
- [ ] Execute SQL Server Agent Job
- [✓] Back Up Database (Full)
- [ ] Back Up Database (Differential)
- [✓] Back Up Database (Transaction Log)
- [✓] Maintenance Cleanup Task

The Maintenance Cleanup task removes files left over from executing a maintenance plan.

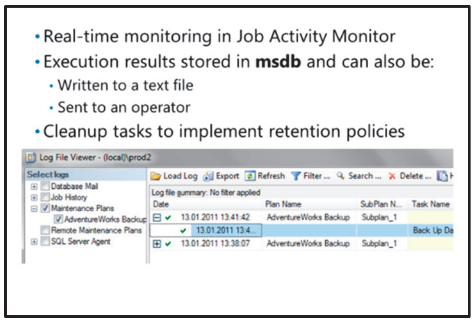
You can use the SQL Server Maintenance Plan Wizard to create SQL Server Agent jobs that perform routine database maintenance tasks, such as regularly backing up system and user databases. The user databases can be optimized and checked for consistency either at the same time, before, or after backups. The wizard creates SQL Server Integration Services (SSIS) packages for SQL Server Agent tasks to execute on the specified schedules. It is also able to integrate with other features of the SQL Agent service to facilitate monitoring and control of the package outcomes.

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. With combinations of these you can group and sequence tasks based on custom requirements.

## Monitoring Database Maintenance Plans



- 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

Log File Viewer - (local)\prod2

Select logs
- Database Mail
- Job History
- Maintenance Plans
  - AdventureWorks Backup
- Remote Maintenance Plans
- SQL Server Agent

Log file summary: No filter applied

| Date | Plan Name | SubPlan N... | Task Name |
|---|---|---|---|
| 13.01.2011 13:41:42 | AdventureWorks Backup | Subplan_1 | |
| 13.01.2011 13:4... | | | Back Up Da |
| 13.01.2011 13:38:07 | AdventureWorks Backup | Subplan_1 | |

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. You can inspect these tables, **dbo.sysmaintplan_log** and **dbo.sysmaintplan_log_detail**, just like any user table, 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.

**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**:
   - o Shrink Database
   - o Rebuild Index
   - o Update Statistics
6. On the **Select Maintenance Task Order** page, change the order of the tasks to **Rebuild Index**, **Shrink Database,** and then **Update Statistics**. Then click **Next**.
7. On the **Define Rebuild Index Task** page in the **Database(s)** list, click **AdventureWorks**, click **OK** to close the drop-down list box. Click **Next**.
8. On the **Define Shrink Database Task** page, in the **Database(s)** 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 **Database(s)** 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**.

### Categorize Activity

Your manager asks you to implement a maintenance solution that minimizes data loss and cost in addition to maximizing performance. Categorize each item by writing the appropriate category number to the right of each item.

| 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, including 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: **20765A-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 DBBC 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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are both running, and then log on to 20765A-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: 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\Lab07\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 the database server down 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 multi-user 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 DBBC 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\Lab07\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

1. Create a database maintenance plan for the **AdventureWorks** database. The maintenance plan should perform the following operations:
   - Perform a full database backup on a weekly basis.
   - Perform a differential backup on a daily basis.
   - Perform transaction log backups on an hourly basis.

### ▶ Task 2: Create a Database Integrity Check Maintenance Plan

1. Create a database maintenance plan for the **AdventureWorks** database. The maintenance plan should perform the following operations:
   - 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:

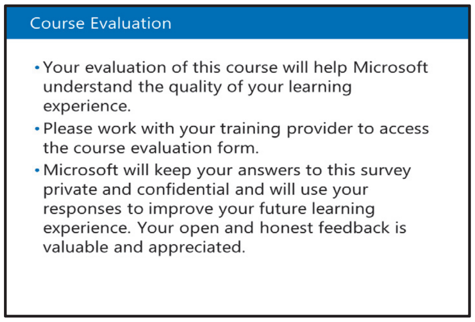
- Run DBCC CHECKDB regularly.
- Synchronize DBCC CHECKDB with your backup strategy.
- If corruption occurs, consider restoring the database from a backup, and only repair the database as a last resort.
- Defragment your indexes when necessary, but if they get beyond about 30 percent fragmentation, consider 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 a good idea 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.

## 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, 20765A-MIA-DC, and 20765A-MIA-SQL virtual machines are running, and then log on to 20765A-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, run **setup.exe**. In the **User Account Control** dialog box, click **Yes**.
2. In the SQL Server Installation Center, on the **Planning** page, click **Hardware and Software Requirements**.
3. 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 that has been checked. If the list of checks is not shown, click **Show Details**. If a warning about Windows Firewall is displayed, you can still 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 CTP3.3** 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 **Setup Role** page, ensure that **SQL Server Feature Installation** is selected, and then click **Next**.
9. On the **Feature Selection** page, under **Instance Features**, select **Database Engine Services**, and then click **Next**.
10. On the **Instance Configuration** page, ensure that **Named instance** is selected, in the **Named instance** box, type **SQLTEST**, and then click **Next**.
11. 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$$w0rd
    - **Startup Type**: Manual
12. On the **Collation** tab, ensure that **SQL_Latin1_General_CP1_CI_AS** is selected and click **Next**.
13. On the **Database Engine Configuration** page, on the **Server Configuration** tab, in the **Authentication Mode** section, select **Mixed Mode (SQL Server authentication and Windows authentication)**. Then enter and confirm the password, **Pa$$w0rd**.
14. Click **Add Current User**; this will add the user **ADVENTUREWORKS\Student (Student)** to the list of Administrators.
15. On the **Data Directories** tab, change the **User database directory** to **M:\SQLTEST\Data**.
16. Change the **User database log directory** to **L:\SQLTEST\Logs**.
17. On the **TempDB** tab, review the default values that have been selected for **tempdb** data files.
18. On the **FILESTREAM** tab, ensure that **Enable FILESTREAM for Transact-SQL access** is not selected, and then click **Next**.
19. On the **Ready to Install** page, review the summary, then click **Install** and wait for the installation to complete.
20. On the **Complete** page, click **Close**.
21. 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 Configuration Manager**, and then click **SQL Server 2016 CTP3.3 Configuration Manager**. In the **User Account Control** dialog box, click **Yes**.
2. In the left-hand pane of the **Sql Server Configuration Manager** window, click **SQL Server Services**.
3. In the right-hand pane, double-click **SQL Server (SQLTEST)**.
4. 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**. Verify that the value of the **Name** property is **MIA-SQL\SQLTEST** and click **Cancel**.
7. In Object Explorer, right-click **Test** and click **Stop**. In the **User Account Control** dialog box, click **Yes**. Then when prompted to confirm that you want to stop the MSSQL$SQLTEST service, click **Yes**.
8. 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, select **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 Books Online. 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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are both running, and then log on to 20765A-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**, and then 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**. If **Object Explorer** is not visible, on the **View** menu, click **Object Explorer**.
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.ssmssln**.
3. In Solution Explorer, double-click the query **Lab Exercise 01 - create login.sql**. If Solution Explorer is not visible, on the **View** menu, click **Solution Explorer**.
4. When the query window opens, 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**, 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 devise** 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. 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. When the query window opens, highlight the statement **USE TSQL;** and 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 **Properties** window, on the **Options** page, change the value of the **Compatibility level** box to **SQL Server 2016 (130)**, and then click **OK**.

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

Identify orphaned database users.

Repair orphaned database users.

Update the database compatibility level.

# Module 4: Deploying SQL Server on Microsoft Azure

# Lab: Migrating SQL Server with Azure

## Exercise 1: Check Database Compatibility

### ▶ Task 1: Prepare the Lab Environment

1. Ensure that the MSL-TMG1, 20765A-MIA-DC, and 20765A-MIA-SQL VMs are both running, and then log on to 20765A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Labfiles\Lab04\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. 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\Lab04 \salesapp1.bacpac**.
6. On the **Advanced** tab, clear the **Select All** check box. 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.
9. On the **Results** page, click **Close**.

### ▶ Task 3: Resolve the Failure by Removing a Stored Procedure

1. In SSMS, click **New Query** (or keyboard **Ctrl + N**).
2. Type the following:

```
USE salesapp1;
GO
DROP PROCEDURE dbo.up_CrossDatabaseQuery;
GO
```

3. Click **Execute** (or keyboard **F5**).

### ▶ 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. 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\Lab04 \salesapp1.bacpac**.
4. On the **Advanced** tab, clear the **Select All** check box. 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 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**, then click **Data + Storage**, then click **SQL Database**.
4. In the **Name** box on the **SQL Database** blade, type **empty**.
5. Under **Server** click **Configure required settings**, and then click **Create a new server**.
6. In the **Server name** box on the **New server** blade, 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**, click **West US** and select a region nearest your current geographical location. Then click **OK**.
10. On the **SQL Database** blade, verify that **Select source** has the value **Blank database.** 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**, 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 the Databases node.
3. Right-click **salesapp1**, point to **Tasks**, then click **Deploy Database to Microsoft Azure SQL Database**.
4. On the **Introduction** page, click **Next**.
5. On the **Deployment Settings** page, click **Connect…**.
6. 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 box, select **SQL Server Authentication**.
8. Type **salesappadmin** in the **Login** box and **Pa$$w0rd1** in the **Password** box, then click **Connect**, then click **Next.**
9. On the **Summary** page, verify the settings then click **Finish**. The migration process will take some time to complete.
10. Review the outcome of the wizard on the **Results** page, 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**, then click **Change Connection**.
6. 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** box, select **SQL Server Authentication**.
8. Type **salesappadmin** in the **Login** box and **Pa$$w0rd1** in the **Password** box, then click **Connect**.
9. In the **Available databases** list box, select **salesapp1** (keyboard **Ctrl+U**).

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 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 5: Working with Databases

# Lab: Managing Database Storage

## Exercise 1: Configuring tempdb Storage

### ▶ Task 1: Prepare the Lab Environment

1. Ensure that the 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are both running, and then log on to 20765A-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: 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, then click **Cancel**.
4. On the toolbar, click **New Query**.
5. Enter the following statements and click **Execute**; alternatively, you can open the **Configure TempDB.sql** script file in the D:\Labfiles\Lab05\Solution folder:

```
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**. Click Yes when prompted. When prompted to allow changes, to restart the service, and to stop any dependent services, click **Yes**.
7. View the contents of **T:\** and note that the **tempdb.mdf** and **templog.ldf** files have been moved to this location.
8. In SQL Server Management Studio, in Object Explorer, right-click **tempdb**, and click **Properties**.
9. On the **Files** page, verify that the file settings have been modified. Then click **Cancel**.
10. Save the script file as **Configure TempDB.sql** in the D:\Labfiles\Lab05\Starter folder.
11. 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;** alternatively, you can open the **Create HumanResources.sql** script file in the D:\Labfiles\Lab05\Solution folder):

```
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\Lab05\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**; alternatively, you can open the **Create InternetSales.sql** script file in the D:\Labfiles\Lab05\Solution folder):

```
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. Then select the statements you have just added, and click **Execute**:

```
ALTER DATABASE InternetSales
MODIFY FILEGROUP SalesData DEFAULT;
```

4. Save the script file as **Create InternetSales.sql** in the D:\Labfiles\Lab05\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\Lab05\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. Move **AWDataWarehouse.ldf** from the D:\Labfiles\Lab05\Starter\ folder to the L:\Logs\ folder.
2. Move the following files from the D:\Labfiles\Lab05\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** 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. 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**, and expand **Tables**. Then 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. 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** and press Enter to update the record. 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 and click **OK**. Then press Esc to cancel the update and close the **dbo.FactInternetSalesArchive** results.

**Results**: After this exercise, you should have attached the **AWDataWarehouse** database to MIA-SQL.

# 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 20765A-MIA-DC, 20765A-MIA-SQL and MSL-TMG1 virtual machines are running, and then log on to 20765A-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. Click **Yes** when prompted to confirm that you want to run the command file, 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, 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**.

   On the **Create Database Credentials** page, in the **Password** and **Confirm Password** boxes, type **Pa$$w0rd**, and then click **Next**.
3. On the **Select Database Tables** page, in the tables list, select the check box next to the table **OrderTracking**, and then click **Next**.
4. On the **Validate SQL Server Settings** page, note the warning concerning Unique/Primary keys, and then click **Next**.
5. On the **Select Azure Deployment** page, click **Sign In**, and then log in using your Azure account credentials as provided by your instructor.
6. In the **Select a subscription to use** box, select **Azure Pass**, in the **Select Azure Region** choose an appropriate region, ensure the **Create New Server** option is selected, and then click **Next**.
7. On the **Configure Azure Deployment** page, in the **Server Admin Login** box, type **Student**, in the **Password** and **Confirm Password** boxes, type **Pa$$w0rd**.
8. In the **Starting IP Address** and the **Ending IP Address** box, type the IP addresses as provided by your instructor, and then click **Next**.
9. 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.
10. 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: 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 20765A-MIA-DC and 20765A-MIA-SQL virtual machines are both running, and then log on to 20765A-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. Click **Yes** when prompted 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\Lab07\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 **-- Simulate an automatic checkpoint**, select the code, and then click **Execute**.
7. In **Solution Explorer**, double-click **Ex1-Shutdown.sql**.
8. 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**.
9. Close the **Ex1-Shutdown.sql** query pane without saving changes.
10. In Windows Explorer, navigate to the **D:\Labfiles\Lab07\Starter\Data** folder, right-click **CorruptDB_log**, and click **Rename**.
11. Type **CorruptDB__log_renamed** and press ENTER. This simulates losing the transaction log file in a disk failure.
12. Click **Start**, type **SQL Server Configuration**, and then click **SQL Server 2016 CTP3.3 Configuration Manager**.
13. In the **User Account Control** dialog box, click **Yes**.
14. In **SQL Server Configuration Manager**, under **SQL Server Configuration Manager (Local)**, click **SQL Server Services**.
15. In the right-hand pane, right-click **SQL Server (MSSQLSERVER)**, and then click **Start**.
16. Right-click **SQL Server Agent (MSSQLSERVER)**, and then click **Start**.
17. Close SQL Server Configuration Manager.

18. 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.
19. In the query pane, review the code under the comment **-- Check the status of the database**, select the code, and then click **Execute**.
20. In the query pane, review the code under the comment **-- Confirm that the database is not online**, select the code, and then click **Execute**.
21. In Object Explorer, right-click **MIA-SQL**, click **Refresh**, and then expand **Databases**. Note that **CorruptDB** shows as **Recovery Pending**.
22. 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**.
23. 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.
24. In the query pane, review the code under the comment **-- Set CorruptDB offline**, select the code, and then click **Execute**.
25. In Windows Explorer, navigate to the D:\Labfiles\Lab07\Starter\Data folder, right-click **CorruptDB_log_renamed**, and click **Rename**.
26. Type **CorruptDB__log** and press ENTER. This simulates replacing the lost transaction log file with a mirror copy.
27. 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**.
28. 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**.
29. 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**.
30. 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 69 products are in stock—the state that the data was in before the erroneous transaction was run.
31. 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 DBBC 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\Lab07\Starter\Ex2 folder, click **Ex2.ssmssln**, 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. Review the options on the **Destination** and **Options** tabs to see further changes possible. On the **General** tab, in the **Schedule** section, click **Change**. Review the default schedule, and then click **OK**.
7. On the **Define Back Up Database (Full) Task** page, click **Next**.
8. 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.
9. In the **Schedule** section, click **Change**.
10. In the **New Job Schedule** dialog box, in the **Frequency** section, in the **Occurs** drop-down list box, click **Daily** and then click **OK**.
11. On the **Define Back Up Database (Differential) Task** page, click **Next**.
12. 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.
13. In the **Schedule** section, click **Change**.
14. 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**.
15. On the **Define Back Up Database (Transaction Log) Task** page, click **Next**.
16. On the **Select Report Options** page, accept the default options, and then click **Next**.
17. 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. Wait until the maintenance plan succeeds, and then in the **Execute Maintenance Plan** dialog box, 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.