50400A:

Designing, Optimizing, and Maintaining a Database Administrative Solution for Microsoft® SQL Server® 2008

(Part – 2)

Acknowledgement

Sify Software Limited 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.

Shweta Baijal Mukherjee – Lead Content Developer

Shweta is a Senior Instructional Designer at Sify eLearning Services. She comes with more than five years of experience in the field of Instructional Designing. Currently, she is leading the content development team for the Microsoft ILT courses. She earned her Masters in Computer Applications in 2005 and her Bachelors in Computer Applications in 2002. She has extensive experience working on both technical and non-technical courses.

Kalpana Sharma – Content Developer

Kalpana is an Instructional Designer at Sify eLearning Services. She has three years of experience in Instructional Designing. She completed her Bachelors in Science in 2006. She has successfully completed her ABAP\4 certification in 2007. Her expertise is in designing and developing technical courses.

James Yip - Subject Matter Expert

James has been leading the content development for Eventus and Microsoft Learning since 2007. He has a background as a Project Manager and Solution Architect for a number of large scale projects using Microsoft technologies, such as Windows Server Systems and .NET Development. Before joining Eventus, he was into designing and deploying Microsoft technologies for over eight years.

Mark Fitzgerald – Technical Reviewer

Mark is the Business Intelligence Principal Technologist for QA Ltd in the UK. He holds an honors degree in Computer Studies and multiple certifications in SQL, PerformancePoint, and Windows. With more than 20 years experience in IT, he has become a lead instructor for all Microsoft SQL and BI products within the UK.

Contents

Lab Answer Keys

Appendix

About This Course

This section provides information about the course, the audience, student prerequisites, and the course objectives.

Course Description

This five-day instructor-led course is for IT professionals who design and maintain SQL Server databases. This course will help students to design, optimize, and maintain a database administrative solution for Microsoft SQL Server™ 2008.

Audience for this course

Students who take up this course should have three or more years of experience working on databases for two or more of the following phases in the product lifecycle—design, development, deployment, optimization, maintenance, and support:

- Administering databases
- Designing logical database schema solutions
- Defining high-availability solutions
- Automating administrative tasks
- Defining security solutions
- Monitoring and troubleshooting the database server
- Designing and executing deployments
- Defining the infrastructure (storage, hardware, and number of servers or instances, and so on)

Students can also be technical architects and consultants who design and implement SQL Server solutions.

Student Prerequisites

This course expects students to:

- Understand the tradeoffs among the different redundant storage types. For example, what raid levels mean, and how they differ from storage area networks (SANs).

- Understand how replication works and how replication is implemented.
- Be familiar with reading user requirements and business-need documents. For example, development project vision/mission statements or business analysis reports.
- Have some knowledge of how queries execute. Must be able to read a query execution plan.
- Have basic knowledge of the dependencies between system components.
- Be able to design a database to third normal form (3NF) and know the tradeoffs when backing out of the fully normalized design (denormalization).
- Be able to design for performance and business requirements in addition to being familiar with design models, such as Star and Snowflake schemas.
- Have monitoring and troubleshooting skills.
- Have knowledge of the operating system and platform, including how the operating system integrates with the database, what the platform or operating system can do, and how the interaction between the operating system and the database works. For example, how integrated authentication interacts with Active Directory® directory service.
- Have knowledge of application architecture; that is, how applications can be designed in three layers, what applications can do, interaction between applications and the database, interaction between the database, and the platform or operating system.
- Must already know how to use:
  - A data modeling tool
  - Microsoft Office Visio® (to create infrastructure diagrams)
  - Be familiar with SQL Server 2008 features, tools, and technologies.
  - Have a Microsoft Certified Technology Specialist: Microsoft SQL Server 2008 credential or equivalent experience.

Objectives of this Course

After completing this course, students will be able to:

- Design an administrative solution for SQL Server 2008.
- Deploy SQL Server 2008.
- Design the physical structure of SQL Server 2008.
- Design a strategy for maintaining SQL Server 2008.
- Design solutions for managing SQL Server 2008.
- Automate database managing strategy for SQL Server 2008.
- Design a strategy for securing databases in SQL Server 2008.
- Design a strategy for monitoring SQL Server 2008.
- Design a strategy for content distribution in SQL Server 2008.
- Design a strategy for replication in SQL Server 2008.
- Design a high-availability solution for SQL Server 2008.
- Design a strategy for backup and recovery solution in SQL Server 2008.

Course Outline

This section provides an outline of the course:

- **Module 1 - Designing an Administrative Solution for SQL Server 2008**: This module provides an overview of an administrative solution and how it helps the administrator.
- **Module 2 - Deploying SQL Server 2008**: This module describes the considerations for implementing a new version of SQL Server 2008.
- **Module 3 - Designing the Physical Structure of SQL Server 2008**: This module describes file placement considerations for different functionalities in SQL Server 2008 that help to improve performance and availability.
- **Module 4 - Designing a Strategy for Maintaining a Database in SQL Server 2008**: This module describes the importance of keeping the database up-to-date and the database performing in optimal state.

- **Module 5 - Designing Solutions for Managing SQL Server 2008**: This module describes the new approach of managing SQL Servers that has been introduced in SQL Server 2008.
- **Module 6** - **Automating the Database Management Strategy for SQL Server 2008**: This module describes the use of SQL Server Agent and scripts, such as VBScripts and SQL PowerShell, to automate some routine jobs that administrators often perform.
- **Module 7 - Designing a Strategy for Securing SQL Server 2008**: This module presents information related to database access control and how SQL Server manages keys and certificates.
- **Module 8 - Designing a Monitoring Strategy for SQL Server 2008**: This module describes the information on monitoring a SQL Server 2008 server to ensure that it is performing properly and is in a healthy state.
- **Module 9** - **Designing a Strategy for Content Distribution in SQL Server 2008**: This module describes the different methods of distributing data to multiple servers. The module also describes how to use remote query with linked server and the considerations for using distributed transactions on query span across multiple servers.
- **Module 10** - **Designing a Strategy for Replication in SQL Server 2008**: This module provides more in-depth discussions on using replication. The module also describes how to use replication, specifically snapshot and transactional replication, which is one-way replication, to replicate data to other servers.
- **Module 11** - **Designing a High-Availability Solution for SQL Server 2008**: This module describes the different high-availability features in SQL Server 2008. The module also covers the scenarios of using them in an organization.
- **Module 12** - **Designing a Backup and Recovery Strategy for SQL Server 2008**: This module describes the importance of having a backup of the database. In case of a disaster, backing up the database helps to recover data that cannot be protected by using high-availability solutions mentioned in the previous module.

Course Materials

The following materials are included with the course kit:

- **Course Handbook**. Presents technical information in a simple, direct manner. Only relevant information that enhances the learning experience in class has been presented.
- **PowerPoint Slide Decks**: Provide key points, relevant graphics, and illustrations to enhance learning. Also provides instructor notes and additional links for the instructor to effectively prepare for the class.
- **Labs**: Provide a real-world, hands-on experience in the application to help you apply the knowledge and skills learned in the content section of the module.
- **Lab Answer Keys**: Provide the step-by-step detailed steps on labs exercises.
- **Virtual Machine Build Guide**: Provides the detailed steps required to re-create the Virtual Machine and Server images with the necessary configuration.
- **Student Course Files**: Provide a self-extracting executable file named, **Allfiles.exe**, which contains the supporting files for the labs and demonstrations to work as required.

Virtual Machine Environment

This section provides 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 Virtual Server 2005 R2 SP1 to perform the labs.

Important: At the end of each lab, you must close the virtual machine and must not save any changes. To close a virtual machine without saving the changes, perform the following steps:

1. In the virtual machine, on the **Action** menu, click **Close**.

2. In the **Close** dialog box, in the **What do you want the virtual machine to do?** list, click **Turn off and delete changes**, and then click **OK**.

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

Virtual machine	Role
50400A-NYC-SQL1	SQL Server 2008

Software Configuration

The following software is installed on each virtual machine (VM):

- Microsoft SQL Server 2008 Developer Edition
- Microsoft Visual Studio 2008 Professional Edition
- Microsoft Office Excel 2007

Course Files

There are files associated with the labs in this course. The lab files are located in the folder, D:\Labfiles\ModXX, on the student computers.

Classroom Setup

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

Note: The content coverage of this course is about Microsoft products and technologies. Therefore, the content in this course has been paraphrased from Microsoft Web sites and other Web sites that discuss Microsoft products and technologies. Wherever it has been inappropriate to rewrite or paraphrase the content, the original content has been retained. All content sources have been acknowledged in the **Source** section of each topic.

END USER LICENSE AGREEMENT

This End-User License Agreement ("Agreement") is a legal agreement between Sify Software Limited ("Sify") and you ("Licensee"). By using the licensed content, you accept these terms. If you do not accept them, do not use the licensed content. If you comply with these license terms, you have the rights below.

1. Sify owns or licenses all intellectual property rights (including, without limitation, copyright, trademark, domain name, trade secret, and patent rights) in all materials provided by Sify in connection with this Agreement, including without limitation any and all materials or other works created by Sify relating to the delivery of Course, including any structure or code licensed, developed, or created by Sify or at the direction of Sify. The licensed content is licensed on a per copy per device basis.

2. Sify may change any of the terms in this Agreement. When it does so, it will update the date of the agreement at the top of the agreement.

3. Sify hereby grants a limited, non-exclusive license to Licensee to use the Course Materials for the sole and exclusive purpose of learning the subject matter. This license is personal to Licensee and non-transferable. Licensee may not sublicense, alter, or disclose the Course Materials to third parties in any form for any reason. Licensee is responsible for all uses of the Course Materials through Licensee's Login Name and Password, and for protecting and securing such Login Name and Password from unauthorized use and disclosure. Licensee will not share Licensee's Login Name and Password with any third party. In the event that Licensee becomes aware of or believes there has been any breach of the Course Materials, such as theft or unauthorized use of Licensee's Login Name and Password, Licensee will notify Sify immediately. This license will terminate immediately and without notice in the event Sify determines, at its sole discretion, that Licensee has failed to comply with the terms of the Agreement.

4. License commences the day Licensee registers for Course and runs for the specific term purchased, whether or not the Course is used.

5. INSTALLATION AND USE RIGHTS.

- **Licensed Device.** The licensed device is the device on which you use the licensed content. You may install and use one copy of the licensed content on the licensed device.
- **Portable Device.** You may install another copy on a portable device for use by the single primary user of the licensed device.
- **Separation of Components.** The components of the licensed content are licensed as a single unit. You may not separate the components and install them on different devices.
- **Third Party Programs.** The licensed content may contain third party programs. These license terms will apply to your use of those third party programs, unless other terms accompany those programs.
- **Feedback.** If you agree to give feedback about the licensed content to Sify, you give to Sify, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You will not give feedback that is subject to a license that requires Sify to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.
- **Confidential Information.** The licensed content, including any viewer, user interface, features and documentation that may be included with the licensed content, is confidential and proprietary to Sify and its suppliers.
  i. **Use.** For five years after installation of the licensed content or its commercial release, whichever is first, you may not disclose confidential information to third parties. You may disclose confidential information only to your employees and consultants who need to know the information. You must have written agreements with them that protect the confidential information at least as much as this agreement.
  ii. **Survival.** Your duty to protect confidential information survives this agreement.
  iii. **Exclusions**. You may disclose confidential information in response to a judicial or governmental order. You must first give written notice to Sify to allow it to seek a protective order or otherwise protect the information. Confidential information does not include information that
     - becomes publicly known through no wrongful act;
     - you received from a third party who did not breach confidentiality obligations to Sify or its suppliers; or
     - you developed independently.

6. ADDITIONAL LICENSING REQUIREMENTS AND/OR USE RIGHTS.

- **Media Elements and Templates.** You may use images, clip art, animations, sounds, music, shapes, video clips and templates provided with the licensed content solely for your personal training use.
- **Academic Materials.** If the licensed content contains academic materials (such as white papers, labs, tests, datasheets and FAQs), you may copy and use the academic materials. You may not make any modifications to the academic materials and you may not print any book (either electronic or print version) in its entirety. If you reproduce any academic materials, you agree that:
  - The use of the academic materials will be only for your personal reference or training use
  - You will not republish or post the academic materials on any network computer or broadcast in any media;
  - You will include the academic material's original copyright notice, or a copyright notice to Sify's benefit in the format provided below:

    Form of Notice:

  © 2010 Reprinted for personal reference use only with permission by Sify Software Limited. All rights reserved.

7. **INTERNET-BASED SERVICES.** Sify may provide Internet-based services with the licensed content. It may change or cancel them at any time. You may not use these services in any way that could harm them or impair anyone else's use of them. You may not use the services to try to gain unauthorized access to any service, data, account or network by any means.

8. **SCOPE OF LICENSE.** The licensed content is licensed, not sold. This agreement only gives you some rights to use the licensed content. Sify reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the licensed content only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the licensed content that only allow you to use it in certain ways. You may not
   - work around any technical limitations in the licensed content;
   - reverse engineer, decompile or disassemble the licensed content, except and only to the extent that applicable law expressly permits, despite this limitation;
   - make more copies of the licensed content than specified in this agreement or allowed by applicable law, despite this limitation;
   - publish the licensed content for others to copy;

- transfer the licensed content;
- allow others to access or use the licensed content;
- rent, lease or lend the licensed content; or
- use the licensed content for commercial licensed content hosting services.
- have Rights to change or modify the server software that may be included with the Licensed Content, including the Virtual Hard Disks does not give you any right to implement Sify patents or other Sify intellectual property in software or devices that may access the server.

9. **TRANSFER TO ANOTHER DEVICE.** You may uninstall the licensed content and install it on another device for your personal training use. You may not do so to share this license between devices.

10. **TRANSFER TO A THIRD PARTY.** You may not transfer those versions marked as 'Beta Content' or 'Pre-release' to a third party. For final versions, these terms apply: The first user of the licensed content may transfer it and this agreement directly to a third party. Before the transfer, that party must agree that this agreement applies to the transfer and use of the licensed content. The first user must uninstall the licensed content before transferring it separately from the device. The first user may not retain any copies.

11. **NOT FOR RESALE SOFTWARE /LICENSED CONTENT.** You may not sell software or licensed content marked as "NFR" or "Not for Resale."

12. As a participant in Course, Licensee owns any original material that Licensee creates and enters online as part of Course, including Licensee's comments and testimonials about the quality of Course and the instructor, and messages posted in chat and threaded discussion or sent via email to Course instructor ("Licensee's Content"). Licensee hereby gives Sify an unlimited, worldwide and royalty-free license for the use and exploitation of Licensee's Content.

13. Licensee agrees that Sify may collect, use, sell, license and otherwise distribute demographic data about users in aggregated form (so that no Licensee is personally identifiable) to assist in market evaluation and for other purposes as reasonably determined by Sify.

14. Some activities within Course may request users to submit information that is tabulated and stored in a database. The data obtained is used to produce statistics that are integral to the educational value of these activities. Anywhere the requested data is personal in nature, a user's specific responses remain confidential and are not associated with name or e-mail address in any way.

15. Sify will not disclose personally identifiable information associated with Licensee's use of the Website (e.g., name, address, access code) to any third parties not affiliated with Sify except to duly authorized investigative agencies for the limited purpose of investigating a good faith claim that a particular user has violated the restrictions on use of or access to Course.

16. **DISCLAIMER OF WARRANTY: THE LICENSED CONTENT IS LICENSED "AS-IS." YOU BEAR THE RISK OF USING IT. SIFY GIVES NO EXPRESS WARRANTIES, GUARANTEES OR CONDITIONS. YOU MAY HAVE ADDITIONAL CONSUMER RIGHTS UNDER YOUR LOCAL LAWS WHICH THIS AGREEMENT CANNOT CHANGE. TO THE EXTENT PERMITTED UNDER YOUR LOCAL LAWS, SIFY EXCLUDES THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. SIFY DOES NOT WARRANT OR GUARANTEE THAT COURSE WILL SATISFY THE INTENDED USE BY LICENSEE. SIFY IS NOT LIABLE FOR ANY INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES, OR FOR LOSS OF PROFITS, REVENUE, OR DATA, WHETHER IN AN ACTION IN CONTRACT, TORT, PRODUCT LIABILITY, STATUTE OR OTHERWISE, EVEN IF ADVISED OF THE POSSIBILITY OF THOSE DAMAGES. SIFY'S LIABILITY WITH RESPECT TO COURSE PROVIDED HEREUNDER WILL NOT EXCEED THE AMOUNT LICENSEE PAID FOR COURSE, EVEN IF ANY TERM OF THIS AGREEMENT FAILS OF ITS ESSENTIAL PURPOSE.**

17. **ENTIRE AGREEMENT.** This agreement, and the terms for supplements, updates, Internet-based services and support services that you use, are the entire agreement for the licensed content and support services.

18. It also applies even if Sify knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

19. Licensee hereby indemnifies and will defend and hold Sify harmless against all claims, damages, injuries or deaths, judgments, liabilities and expenses (including

legal costs and attorney fees) arising from or in connection with Licensee's breach of the conditions to this License, or Licensee's negligence or violation or alleged violation of Sify's proprietary information, trademark, copyright, or patent. The obligation to indemnify will survive the termination of this Agreement.

20. This is the entire Agreement between the parties as to the subject matter hereof and supersedes all other agreements, express or implied, written or oral. No modification of, or amendment to, this Agreement shall be binding unless in writing and signed by both parties. Invalidity or unenforceability of one or more provisions of this Agreement shall not affect any other provision of this Agreement. Neither failure nor delay on the part of any party to exercise any right, remedy, power, or privilege hereunder nor course of dealing between the parties shall operate as a waiver thereof, or of the exercise of any other right, remedy, power, or privilege.

21. This Agreement, and any disputes or disagreements concerning Course or Course Materials, will be governed by and construed according to the laws of India, without regard to its choice of law rules, and specifically excluding the U.N. Convention on Contracts for the International Sale of Goods. Any lawsuit filed regarding this Agreement must be filed in courts in Chennai, India. Any notices required or authorized to be given hereunder shall be deemed to be given when mailed by certified or registered mail, postage prepaid, as follows:

To Sify:
Legal Department
Sify Software Limited
2nd Floor, Tidel Park,
No.4, Rajiv Gandhi Salai,
Taramani, Chennai 600113, INDIA

Module 10

Designing a Strategy for Replication in SQL Server 2008

Contents:

Module Overview

- Designing a Database Replication Strategy
- Implementing Content Distribution by Using Replication
- Configuring a High-Availability Solution by Using Replication

An organization may have multiple branches and each branch may need to access the same set of data. Each branch can have applications running at that point to a central database server. This sharing can delay the response time of an application because all the applications will be accessing data that is stored centrally. Microsoft™ SQL Server™ provides replication technologies that help to copy and distribute data and database objects from one database to another and then synchronize between the databases to maintain consistency.

In this module, you will learn about the different replication technologies of SQL Server.

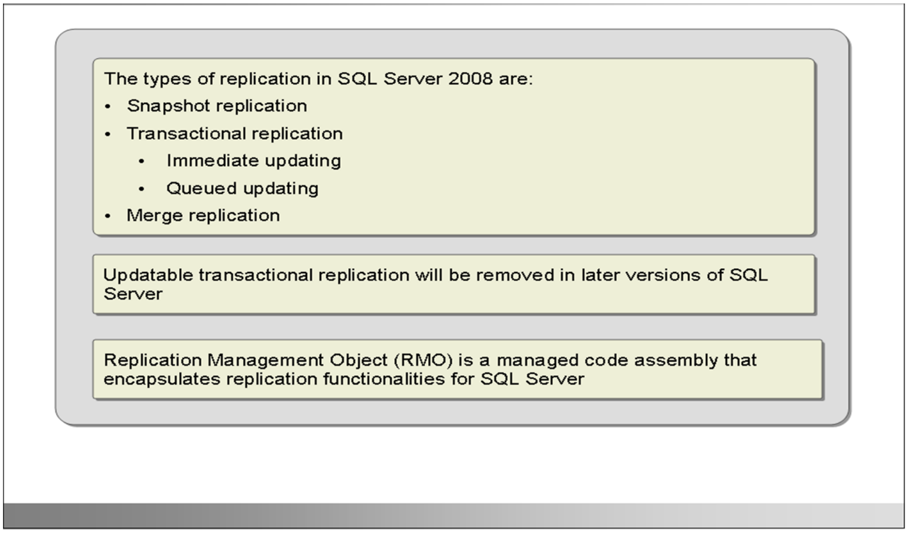
Lesson 1
Designing a Database Replication Strategy

- Types of Replication
- Business Scenarios for Using Replication
- Working of Replication
- Replication Agents in SQL Server 2008
- Guidelines for Using Replication

Replication needs vary depending on the business needs. Some organizations may need read-only access to the replicated data and other organizations may need read-write access to the replicated data. To decide the replication technologies, you need to consider the nature of the business application and how it works.

In this lesson, you will learn about the different replication technologies and how they work.

Types of Replication



Key Points

You can use replication to distribute data to remote or mobile users over local area network (LAN) and wide area networks (WAN), dial-up connections, wireless connections, and the Internet. SQL Server supports the following replication types:

Snapshot Replication

Snapshot replication distributes data exactly as it appears and does not monitor updates to the data. When synchronization occurs, the entire snapshot is generated and sent to the Subscriber.

Note: You can use snapshot replication by itself, but you can also use the snapshot process, which creates a copy of all of the objects and data specified by a publication, to provide the initial set of data and database objects for transactional and merge publications.

Scenarios for Using Snapshot Replication

You can use snapshot replication when at least one of the following is true:

- Data changes infrequently.
- Copies of data exist that are out of date with respect to the Publisher for a period of time.
- Small volumes of data have to be replicated.
- A large volume of changes occurs over a short period of time.

You can use snapshot replication when data changes are substantial, but infrequent.

For example, if a sales organization maintains a product price list and the prices are all updated at the same time once or twice each year, you must replicate the entire snapshot of data after it has changed. For certain types of data, you can also perform frequent snapshot replication. For example, if you update a relatively small table at the Publisher during the day, but some latency is acceptable, changes can be delivered at night as a snapshot.

Snapshot replication has a lower continuous overhead on the Publisher than transactional replication, because you do not track incremental changes. However, if the dataset set being replicated is very large, it will require substantial resources to generate and apply the snapshot. Consider the size of the entire data set and the frequency of changes to the data when evaluating whether to use snapshot replication.

Transactional Replication

Transactional replication typically starts with a snapshot of the publication database objects and data. As soon as the initial snapshot is taken, subsequent data changes and schema modifications made at the Publisher are usually delivered to the Subscriber as they occur in near real time. The data changes are applied to the Subscriber in the same order and within the same transaction boundaries as they occurred at the Publisher, guaranteeing transactional consistency within a publication.

For example, transactional replication can help applications replicate changes from corporate headquarters to branch offices whenever there is a change in the application at the corporate headquarters. Then, the application in the branch office will be able to read the new information.

Scenarios for Using Transactional Replication

You can use transactional replication in server-to-server environments and it is appropriate in each of the following cases:

- You want incremental changes to be propagated to Subscribers as they occur.
- The application requires low latency between the time changes are made at the Publisher and the changes appear at the Subscriber.
- The application requires access to intermediate data states. For example, if a row changes five times, transactional replication allows an application to respond to each change, such as firing a trigger, not just the net data change to the row.
- The Publisher has a very high volume of INSERT, UPDATE, and DELETE activities.
- The Publisher or Subscriber is a non-SQL Server database, such as Oracle.

By default, you should consider Subscribers to transactional publications as read-only, because changes are not propagated back to the Publisher. However, transactional replication does offer options that allow updates at the Subscriber.

Updatable Subscriptions

Transactional replication supports updates at Subscribers through updatable subscriptions and peer-to-peer replication. The following are the two types of updatable subscriptions:

- **Immediate updating**. You must connect the Publisher and Subscriber to update data at the Subscriber.
- **Queued updating**. You need not connect the Publisher and Subscriber to update data at the Subscriber. Updates can be made when the Subscriber or Publisher is offline.

When data is updated at a Subscriber, it is first propagated to the Publisher and then propagated to other Subscribers. If immediate updating is used, the changes are propagated immediately by using the two-phase commit protocol. If queued updating is used, the changes are stored in a queue; the queued transactions are then applied asynchronously at the Publisher whenever network connectivity is available. Because the updates are propagated asynchronously to the Publisher, the same data may have been updated by the Publisher or by another Subscriber and conflicts can occur when applying the updates. You can detect and resolve conflicts according to a conflict resolution policy that is set when creating the publication.

If you create a transactional publication with updatable subscriptions in the New Publication Wizard, both immediate updating and queued updating are enabled. If you create a publication with stored procedures, you can enable one or both options. When you create a subscription to the publication, you specify which update mode to use. You can then switch between update modes, if necessary.

Merge Replication

Just like transactional replication, merge replication starts with a snapshot of the publication database objects and data. You can use triggers to track subsequent data changes and schema modifications made at the Publisher and Subscribers. The Subscriber synchronizes with the Publisher when connected to the network and exchanges all rows that have changed between the Publisher and Subscriber since the last time synchronization occurred.

For example, you can use merge replication to replicate changes across various shops in your organization for the POS system. After a product is sold, the change in inventory would get replicated to other systems; so, other shops would always have the most up-to-date inventory records.

Scenarios for Using Merge Replication

You can use merge replication in server-to-client environments and it is appropriate in any of the following situations:

- Multiple Subscribers might update the same data at various times and propagate those changes to the Publisher and to other Subscribers.
- Subscribers need to receive data, make changes offline, and later synchronize changes with the Publisher and other subscribers.
- Each Subscriber requires a different partition of data.

- Conflicts might occur, and when they do, you need the ability to detect and resolve them.
- The application requires net data change, rather than access to intermediate data states.

Replication Management Objects (RMOs)

RMO is a managed code assembly that encapsulates replication functionalities for SQL Server. RMO is designed for programming all aspects of SQL Server replication. The RMO namespace is Microsoft.SqlServer.Replication, and it is implemented by the Microsoft.SqlServer.Rmo.dll, which is a Microsoft .NET Framework assembly. The Microsoft.SqlServer.Replication.dll assembly, which also belongs to the Microsoft.SqlServer.Replication namespace, implements a managed code interface for programming the various replication agents, such as Snapshot Agent, Distribution Agent, and Merge Agent. You can access its classes from RMO to synchronize subscriptions. You can use the classes in the Microsoft.SqlServer.Replication.BusinessLogicSupport namespace, implemented by the Microsoft.SqlServer.Replication.BusinessLogicSupport.dll assembly to create custom business logic for merge replication. This assembly is independent from RMO.

Question: Which replication types support the multimaster replication model?

Question: What should you do when replication fails?

Question: How can you find out the cause of the failure of replication?

Source:

- http://msdn.microsoft.com/en-us/library/ms151832.aspx
- http://msdn.microsoft.com/en-us/library/ms151176.aspx
- http://msdn.microsoft.com/en-us/library/ms151718(SQL.90).aspx

- http://msdn.microsoft.com/en-us/library/ms152746.aspx
- http://msdn.microsoft.com/en-us/library/ms146869.aspx

For more information, see Types of Replication Overview.

For more information, see Replication Management Objects Concepts.

Discussion: Business Scenarios for Using Replication

- Which replication technologies would you use if you want all branches worldwide to access a POS system?
- What replication technologies should you use when you want the mobile workforce to have access to up-to-date information?
- Which method should you use for a client and a server to communicate without a VPN connection?
- Which replication technologies should you use if you want to create a report daily based on real-time information stored in the databases of the branch offices of your organization?
- If you change the generation frequency for the report from daily to every two hours, which replication technology should you use?

Key Points

In this discussion, you are presented scenarios in which you are asked to determine the type of replication you will use in the various business scenarios for using replication. You and your classmates will discuss possible solutions to the scenario.

Scenario

In your organization, the Point of Sale (POS) system is being rolled out for all branches, worldwide. Because of the performance, having each POS terminal to connect directly to the centralized database is not possible.

Question: Which replication technologies would you use if you want all branches worldwide to access a POS system?

Scenario

You have a CRM application that supports a mobile workforce. From time to time, users are using a PDA to access customer information. The Director of Sales has informed you that the sales people have access to the Internet most of the time and the director wants them to have access to up-to-date information while they are traveling. All the PDAs used by the sales people have the CRM application, which uses SQL CE as the Database Engine. The PDA synchronizes the database with the central database when it synchronizes with the system by using ActiveSync®.

Question: What replication technologies should you use when you want the mobile workforce to have access to up-to-date information?

Question: Which method should you use for a client and a server to communicate without a VPN connection?

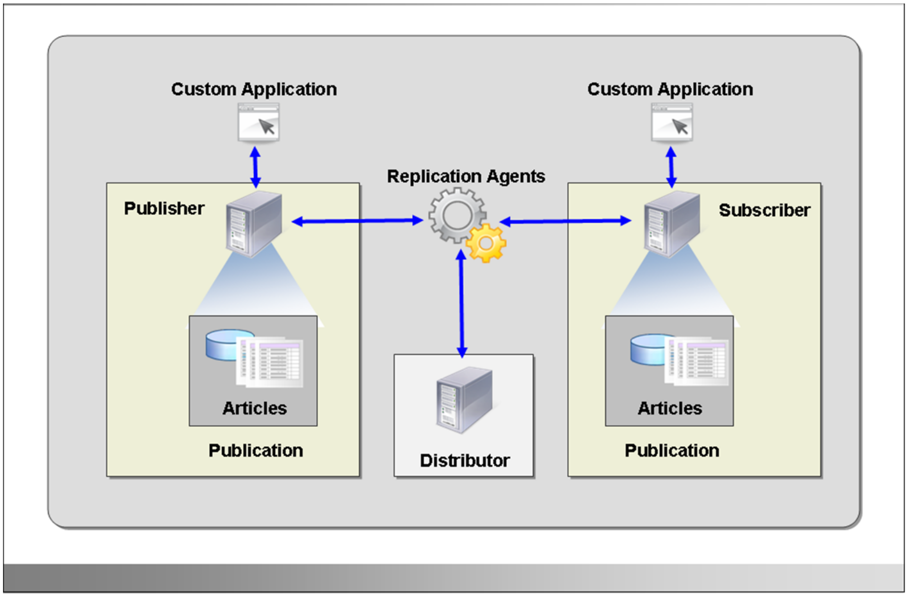
Scenario

You have a number of applications that only exist in the regional office. Your senior management requires real-time information about the business in regional offices, and this information only exists in those applications. The applications are running SQL Server 2008 and you have already created reports to extract information from those databases once every day. During a test run, you find that the performance is extremely slow because of the fact that the report needs to extract a large amount of data from the database in the regional offices.

Question: Which replication technologies should you use if you want to create a report daily based on real-time information stored in the databases of the branch offices of your organization?

Question: If you change the generation frequency for the report from daily to every two hours, which replication technology should you use?

Working of Replication



Key Points

Replication uses a publishing industry metaphor to represent the components in a replication topology, which include the Publisher, Distributor, Subscribers, publications, articles, and subscriptions.

A replication topology defines the relationship between servers and copies of data, and defines the logic that determines how data flows between servers. There are several replication processes referred to as agents that are responsible for copying and moving data between the Publisher and Subscribers.

In a replication model:

- A publisher produces one or more publications.
- A publication contains articles.
- The publisher either distributes the articles directly or uses a distributor.

- Subscribers receive publications to which they have subscribed.
- Replication Agents running on both the publisher and the subscriber help to retrieve and apply the changes.
- Depending on the type of replication, client applications can make changes to publisher or subscriber. Client applications can also read the replication information from the subscriber.

The components and processes involved in server role replication include:

Publisher

The Publisher is a database instance that makes data available to other locations through replication. The Publisher can have one or more publications, each defining a logically related set of objects and data to replicate.

Distributor

The Distributor is a database instance that acts as a store for replicating specific data associated with one or more Publishers. Each Publisher is associated with a single database, known as a distribution database, at the Distributor. The distribution database stores replication status data and metadata about the publication. In some cases, the publication acts as a queue for data moving from the Publisher to the Subscriber. In many cases, a single database server instance acts as both the Publisher and the Distributor. This is known as a local Distributor. When you configure the Publisher and the Distributor on separate database server instances, the Distributor is known as a remote Distributor.

Subscribers

A Subscriber is a database instance that receives replicated data. A Subscriber can receive data from multiple Publishers and publications. Depending on the type of replication you choose, the Subscriber can also pass data changes back to the Publisher or republish the data to other Subscribers.

Article

An article identifies a database object that a publication includes. A publication can contain different types of articles, including tables, views, stored procedures, and other objects. When you publish tables as articles, you can use filters to restrict the columns and rows of the data sent to Subscribers.

Publication

A publication is a collection of one or more articles from one database. The grouping of multiple articles into a publication makes it easier to specify a logically related set of database objects and data that are replicated as a unit.

Subscription

A subscription is a request for a copy of a publication to be delivered to a Subscriber. The subscription defines what publication will be received, where, and when. There are two types of subscriptions: push and pull.

The following table describes the characteristics of pull and push subscriptions and when to use them.

Subscription	Characteristics	When to use
Push Subscription	With a push subscription, the Publisher propagates changes to a Subscriber without a request from the Subscriber. Changes can be pushed to Subscribers on demand, continuously, or on a scheduled basis. The Distribution Agent or Merge Agent runs at the Distributor.	• Data will typically be synchronized continuously or on a frequently recurring schedule. • Publications require near real-time movement of data. • The higher processor overhead at the Distributor does not affect performance. • Most often used with snapshot and transactional replication.

Subscription	**Characteristics**	**When to use**
Pull Subscription	With a pull subscription, the Subscriber requests changes are made at the Publisher. Pull subscriptions allow the user at the Subscriber to determine when the data changes are synchronized. The Distribution Agent or the Merge Agent runs at the Subscriber.	• Data will typically be synchronized on demand or on a schedule, rather than continuously. • The publication has a large number of Subscribers. It would be too resource-intensive to run all the agents at the Distributor. • Subscribers are autonomous and disconnected. Subscribers can also be mobile. Subscribers will determine when they will connect and synchronize changes. • Most often used with merge replication.

Question: What is the purpose of the Distributor?

Question: Where does the Distributor reside?

Source:

- http://msdn.microsoft.com/en-us/library/ms151170.aspx
- http://technet.microsoft.com/en-us/library/ms152567.aspx

Replication Agents in SQL Server 2008

Replication Agent	Description
SQL Server Agent	• Hosts and schedules the agents used in replication • Controls and monitors operations outside the replication
Snapshot Agent	• Prepares schema and initial data files of published tables and other objects • Stores the snapshot files • Records information about synchronization in the distribution database
Log Reader Agent	Moves transactions marked for replication from the transaction log on the Publisher to the distribution database
Distribution Agent	Applies the initial snapshot to the Subscriber and moves transactions held in the distribution database to Subscribers
Merge Agent	Applies the initial snapshot to the Subscriber, and moves and reconciles incremental data changes that occur
Queue Reader Agent	Runs at the Distributor and moves changes made at the Subscriber back to the Publisher

Key Points

Replication uses a number of standalone programs, called agents, to carry out the tasks associated with tracking changes and distributing data. By default, replication agents run as jobs scheduled under SQL Server Agent, and SQL Server Agent must be running for the jobs to run. You can run replication agents from the command line and by using applications that use Replication Management Objects (RMO). You can administer replication agents from SQL Server Replication Monitor and SQL Server Management Studio (SSMS).

The following table describes the various replication agents available in SQL Server.

Replication agent	Description
SQL Server Agent	The SQL Server Agent hosts and schedules the agents used in replication and provides an easy way to run replication agents.

Replication agent	**Description**
	SQL Server Agent also controls and monitors operations outside the replication.
Snapshot Agent	The Snapshot Agent is typically used with all types of replication. It prepares schema and initial data files of published tables and other objects, stores the snapshot files, and records information about synchronization in the distribution database. The Snapshot Agent runs at the Distributor.
Log Reader Agent	The Log Reader Agent is used with transactional replication. It moves transactions marked for replication from the transaction log on the Publisher to the distribution database. Each database published by using transactional replication has its own Log Reader Agent that runs on the Distributor and connects to the Publisher. The Distributor can be on the same computer as the Publisher.
Distribution Agent	The Distribution Agent is used with snapshot replication and transactional replication. It applies the initial snapshot to the Subscriber and moves transactions held in the distribution database to Subscribers. The Distribution Agent runs at either the Distributor for push subscriptions or at the Subscriber for pull subscriptions.
Merge Agent	The Merge Agent is used with

Replication agent	Description
	merge replication. It applies the initial snapshot to the Subscriber and moves and reconciles incremental data changes that occur. Each merge subscription has its own Merge Agent that connects to both the Publisher and the Subscriber, and updates both. The Merge Agent runs at either the Distributor for push subscriptions or the Subscriber for pull subscriptions. By default, the Merge Agent uploads changes from the Subscriber to the Publisher and then downloads changes from the Publisher to the Subscriber.
Queue Reader Agent	The Queue Reader Agent is used with transactional replication, with the queued updating option. The agent runs at the Distributor and moves changes made at the Subscriber back to the Publisher. Unlike the Distribution Agent and the Merge Agent, only one instance of the Queue Reader Agent exists to service all Publishers and publications for a given distribution database.

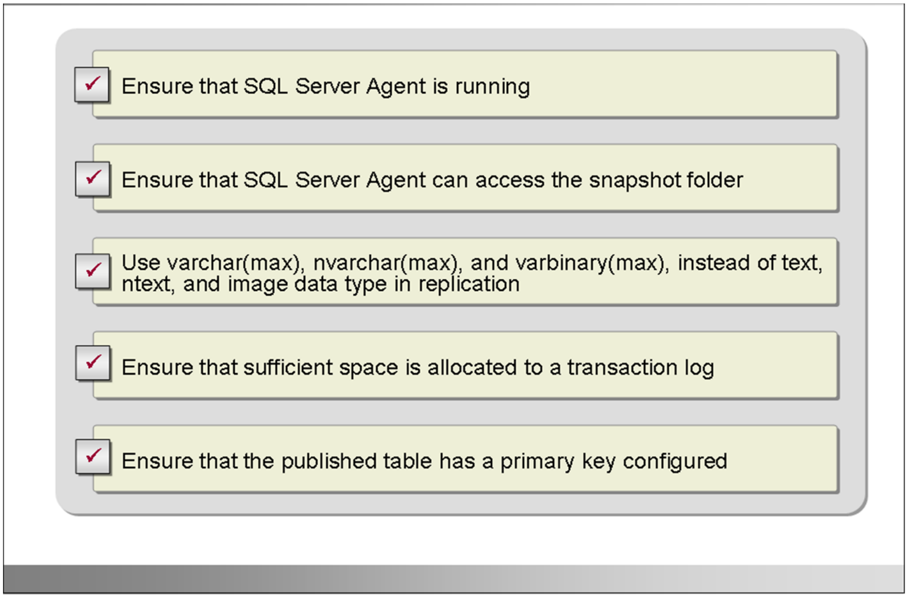
Question: Why should SQL Server Agent be on when you configure SQL Server replication?

Question: How can you determine which agent is configured in each SQL Server?

Source:

- http://technet.microsoft.com/en-us/library/ms152501.aspx

Guidelines for Using Replication



Key Points

You need to consider the following guidelines while using replication:

- **Ensure that SQL Server Agent is running**. Replication uses stand-alone programs, called agents, to perform the tasks associated with tracking changes and distributing data. By default, replication agents run as jobs scheduled under SQL Server Agent. SQL Server Agent must be running for the jobs to run. By default, the SQL Server Agent service is disabled when SQL Server is installed, unless you explicitly choose to autostart the service during installation.
- **Ensure that SQL Server Agent can access the snapshot**

folder. The snapshot folder is simply a folder that you have designated as a share; agents that read from and write to this folder must have sufficient permissions to access it. Prior to implementing replication, test to ensure that the replication agents will be able to connect to the snapshot folder. Log on to the account that will be used by each agent and then attempt to access the snapshot folder.

- **Consider using varchar(max), nvarchar(max), and varbinary(max), instead of text, ntext, and image, in replication**. Updates to text, ntext, and image columns are replicated only if you update the column explicitly by an UPDATE statement. The UPDATE statement causes a trigger to update metadata, ensuring that the transaction is propagated to other Subscribers. Using only the WRITETEXT and UPDATETEXT operations does not propagate the change to other sites. If your application uses WRITETEXT and UPDATETEXT to update the text or ntext columns, explicitly add a dummy UPDATE statement after the WRITETEXT or UPDATETEXT operations within the same transaction, to send the trigger and thereby guarantee that the change is propagated to other sites.

To avoid the preceding issues, you should use the data types varchar(max), nvarchar(max), varbinary(max), instead of text, ntext, and image data types, respectively. You should avoid using the text, ntext, and image data types.

- **Ensure that sufficient space is allocated to a transaction log**. For each database that will be published by using transactional replication, ensure that sufficient space has been allocated for the transaction log. The transaction log of a published database might require more space than the log of an identical unpublished database because the log records are not truncated until they have been moved to the distribution database.

If the distribution database is unavailable, or if the Log Reader Agent is not running, the transaction log of a publication database continues to increase. The transaction log of a published database might require more space than the log of an identical unpublished database because the log records are not truncated until they have been moved to the distribution database. You should set the transaction log file to autogrow, so that the log can accommodate these circumstances.

- **Ensure that the published table has a primary key configured**. A primary key is used to uniquely identify a row in a table and is used by replication to identify the row during the replication operations. SQL Server replication will not able to replicate content if the table does not have a primary key.

Monitoring Replication

The steps to monitor replication include:

1. Open the Microsoft SQL Server Management Studio window.
2. Using the Microsoft SQL Server Management Studio window, run the New Publication Wizard and add a publishing instance to the replication monitor.
3. Using the Replication Monitor window, monitor the status of the publication.

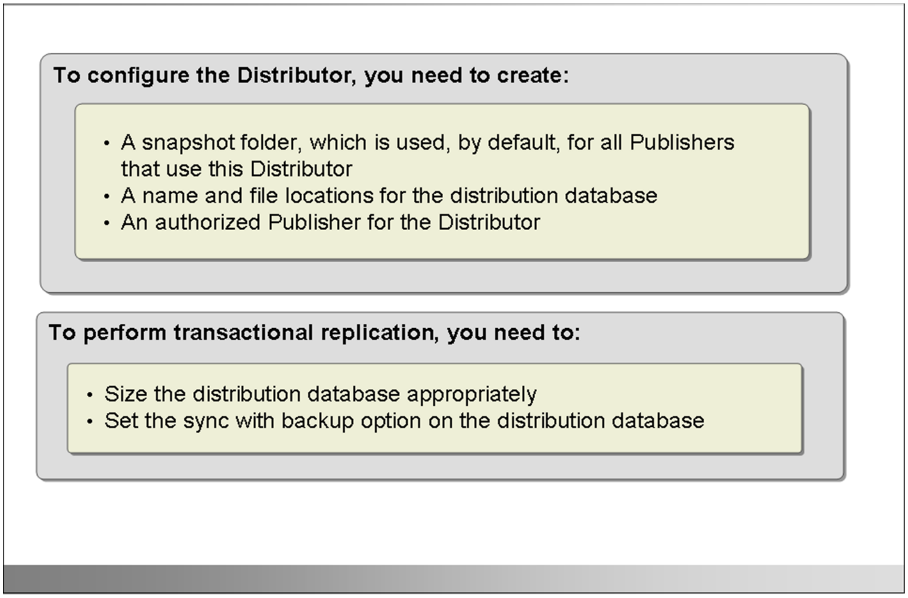
Question: What should you consider when designing a table schema that will be replicated?

Question: What is the purpose of the snapshot folder?

Source:

- http://msdn.microsoft.com/en-us/library/aa256081(SQL.80).aspx
- http://msdn.microsoft.com/en-us/library/ms151206.aspx
- http://msdn.microsoft.com/en-us/library/ms151254.aspx

Considerations for Configuring Distribution



Key Points

The Distributor is a server that contains the distribution database, which stores metadata and history data for all types of replication and transactions for transactional replication. To set up replication, you must configure a Distributor. Each Publisher can be assigned to only a single Distributor instance, but multiple publishers can share a Distributor. The Distributor uses the following additional resources on the server:

- Additional disk space if the snapshot files for the publication are stored on the Distributor.
- Additional disk space to store the distribution database.
- Additional processor usage by replication agents for push subscriptions running on the Distributor.

The server you select as the Distributor should have adequate disk space and processor power to support replication and any other activities on that server.

To configure the Distributor, you need to create:

- A snapshot folder, which is used, by default, for all Publishers that use this Distributor. Ensure that this folder is already shared and has the appropriate permissions set.

- A name and file locations for the distribution database. The distribution database cannot be renamed after it is created. To use a different name for the database, you must disable distribution and reconfigure it.

- An authorized Publisher for the Distributor. If you specify Publishers other than the instance on which the Distributor runs, you must also specify a password for the connections the Publishers make to the remote Distributor.

After you configure distribution, you need to perform the following tasks for transactional replication:

- Size the distribution database appropriately.

- Set the sync with backup option on the distribution database.

Remote Distributors

Remote Distributors are common in transactional replication than they are in merge replication for two reasons:

- The Distributor plays a larger role in transactional replication because all replicated transactions are written to and read from the distribution database.

- Merge replication topologies typically use pull subscriptions, so agents run at each Subscriber, rather than all running at the Distributor.

You can use a remote Distributor to:

- Perform offload processing to another computer if you want minimal impact from replication on the Publisher.

- Configure a centralized Distributor for multiple Publishers.

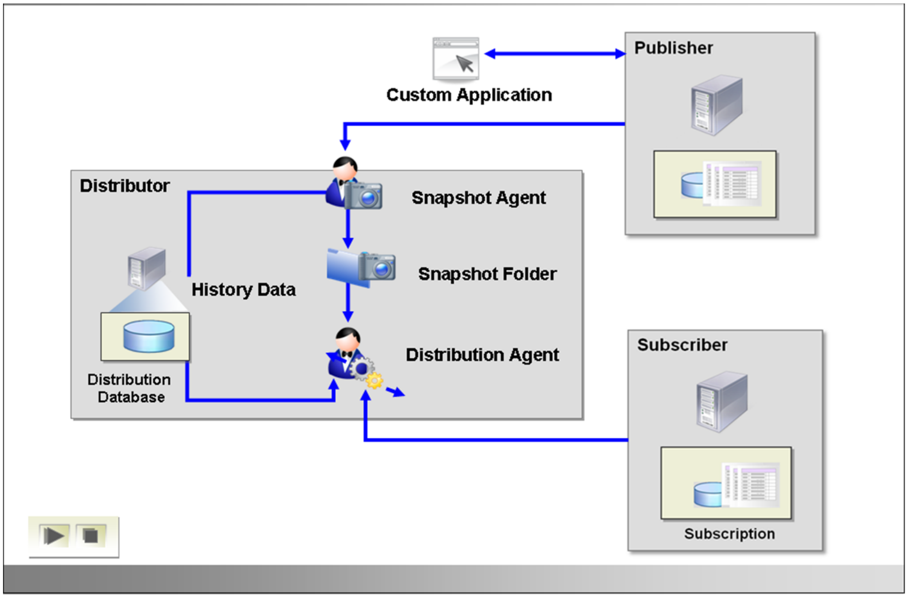
Lesson 2
Implementing Content Distribution by Using Replication

- Working of Snapshot Replication
- How To Set Up Snapshot Replication
- Working of Transactional Replication
- How To Set Up Transactional Replication
- Filtering Publication Data
- Considerations for Filtering Data
- Making Schema Changes on Publication Databases
- Configuring Heterogeneous Replication
- Overview of Oracle Publishing
- Considerations for Using Heterogeneous Database Replication

Content distribution helps you to bring the data closer to the client application for read-only purpose by using replication, which helps to improve server performance. You can distribute content by using either snapshot replication or transactional replication. You need to consider the various factors to determine the type of replication to use for content distribution.

In this lesson, you will learn about snapshot replication and transactional replication, and set up both replication types. You will learn how to publish data in replication. You will configure heterogeneous replications. You will also learn the considerations for filtering data and the guidelines for heterogeneous replications.

Working of Snapshot Replication



Key Points

By default, snapshot replication, transactional replication, and merge replication use a snapshot to initialize Subscribers. The SQL Server Snapshot Agent always generates the snapshot files. However, the agent that delivers the files differs depending on the type of replication that you use. The Snapshot Agent runs at the Distributor. The Distribution Agent and the Merge Agent run at the Distributor for push subscriptions, or at Subscribers for pull subscriptions.

You can generate and apply snapshots either immediately after you create the subscription or according to a schedule set at the time when you created the publication. The Snapshot Agent prepares snapshot files containing the schema and data of published tables and database objects. The Snapshot Agent also stores the files in the snapshot folder for the Publisher, and records tracking information in the distribution database on the Distributor. You need to specify a default snapshot folder when you configure a Distributor. You can also specify an alternate location for a publication, instead of, or in addition to the default.

Snapshot Agent

For merge replication, a snapshot is generated every time you run the Snapshot Agent. For transactional replication, snapshot generation depends on the setting of the publication property, immediate_sync. If the property is set to TRUE, the default when using the New Publication Wizard, a snapshot is generated every time you run the Snapshot Agent, and it can be applied to a Subscriber at any time. If the property is set to FALSE, the default when using sp_addpublication, the snapshot is generated only if you add a new subscription since the last time the Snapshot Agent was run. Subscribers must wait for the Snapshot Agent to complete creating the snapshot before they can synchronize.

The Snapshot Agent performs the following steps to generate snapshots:

1. Establishes a connection from the Distributor to the Publisher, and takes locks on published tables, if necessary. The Snapshot Agent takes locks based on the following conditions:
    - For merge publications, the Snapshot Agent does not hold any locks on the table.
    - For transactional publications, by default, the Snapshot Agent takes locks only during the initial phase of the snapshot generation.
    - For snapshot publications, locks are held during the entire snapshot generation process.
2. Writes a copy of the table schema for each article to a .sch file. If you publish other database objects, such as indexes, constraints, stored procedures, views, and user-defined functions, you need to generate additional script files.
3. Copies the data from the published table at the Publisher and writes the data to the snapshot folder. The snapshot is generated as a set of bulk copy program (BCP) files.
4. For snapshot and transactional publications, the Snapshot Agent appends rows to the MSrepl_commands and MSrepl_transactions tables in the distribution database. The entries in the MSrepl_commands table contain commands indicating the location of .sch and .bcp files, any other snapshot files, and

references to any presnapshot or postsnapshot scripts. The entries in the MSrepl_transactions table are commands relevant to synchronizing the Subscriber.

5. Releases any locks on published tables.

During snapshot generation, you cannot make schema changes on published tables. After the snapshot files are generated, you can view them in the snapshot folder by using Windows®Explorer.

Distribution Agent and Merge Agent

For snapshot publications, each time the Distribution Agent runs for the publication, it moves a new snapshot to each Subscriber that has not yet been synchronized, has been marked for reinitialization, or have the publication modified.

For snapshot and transactional replication, the Distribution Agent performs the following steps:

1. Establishes a connection to the Distributor.
2. Examines the MSrepl_commands and MSrepl_transactions tables in the distribution database on the Distributor. The agent reads the location of the snapshot files from the first table and Subscriber synchronization commands from both tables.
3. Applies the schema and commands to the subscription database.

For an unfiltered merge replication publication, the Merge Agent performs the following steps:

1. Establishes a connection to the Publisher.
2. Examines the sysmergeschemachange table on the Publisher and determines whether there is a new snapshot that should be applied at the Subscriber.
3. If a new snapshot is available, the Merge Agent applies the snapshot files to the subscription database from the location specified in sysmergeschemachange.

Question: How does snapshot replication help other replication types?

Source:

- http://msdn.microsoft.com/en-us/library/ms151734.aspx

Demonstration: How To Set Up Snapshot Replication

In this demonstration, you will see how to:

Set up snapshot replication

Key Points

The steps to set up snapshot replication are:

1. On the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. In the Microsoft SQL Server Management Studio window, by using the **Connect to Server** dialog box, connect to the default instance of SQL Server Database Engine.
3. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **Replication**, right-click the **Local Publications** folder, and then click **New Publication**. The New Publication Wizard launches.
4. In the **New Publication Wizard**, click **Next**.
5. On the **Distributor** page, click **Next**.
6. On the **Snapshot Folder** page, click **Next**.

7. On the **Publication Database** page, under **Databases**, ensure that **QuantamCorp** is selected as the publication database, and then click **Next**.
8. On the **Publication Type** page, under **Publication type**, ensure that the **Snapshot publication is selected**, and then click **Next**.
9. On the **Articles** page, under **Objects to publish**, expand Tables, and select any one table, and then click **Next**.
10. On the **Filter Table Rows** page, click **Next**.
11. On the **Snapshot Agent** page, select the **Create a snapshot immediately and keep the snapshot available to initialize subscriptions** check box, and then click **Next**.
12. On the **Agent Security** page, click **Security Settings**. The **Snapshot Agent Security** dialog box appears.
13. In the **Snapshot Agent Security** dialog box, ensure that the **Run under the following Windows account** option is clicked.
14. Under **Run under the following Windows account**, in the **Process account** box, type **NYC-SQl1\Administrator**.
15. In the **Password** box, type **Pa$$w0rd**, and in the **Confirm Password** box, type **Pa$$w0rd**, and then click **OK**.
16. In the New Publication Wizard, on the **Agent Security** page, click **Next**.
17. On the **Wizard Actions** page, click **Next**.
18. On the **Complete the Wizard** page, in the **Publication name** box, type **Snapshot**, and then click **Finish**.
19. On the **Creating Publication** page, click **Close**.
20. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **Local Publications**, right-click **[QuantamCorp]: Snapshot**, and then click **New Subscriptions**. The New Subscription Wizard launches.
21. In the **New Subscription Wizard**, click **Next**.
22. On the **Publication** page, click **Next**.
23. On the **Distribution Agent Location** page, click **Next**.

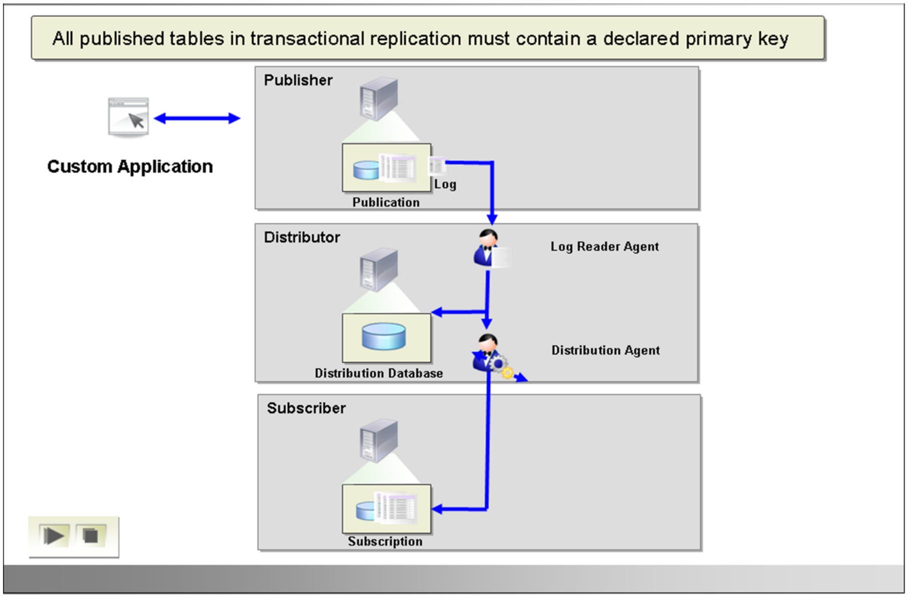
24. On the **Subscribers** page, click the **Add Subscriber** drop-down arrow, and then click **Add SQL Server Subscriber**. The **Connect to Server** dialog box appears.
25. In the **Connect to Server** dialog box, in the **Server name** box, type or select **NYC-SQL1\DEVELOPMENT**, and then click **Connect**.
26. On the **Subscribers** page, in the **Subscription Database** list for NYC-SQL1\DEVELOPMENT, ensure that **QuantamCorp** is selected by default, and then click **Next**.
27. On the **Distribution Agent Security** page, click the **ellipsis** button, in the NYC-SQL1\DEVELOPMENT, agent for subscriber row. The **Distribution Agent Security** dialog box appears.
28. In the **Distribution Agent Security** dialog box, click **Run under the SQL Server Agent service account (This is not a recommended security best practice.)**, and then click **OK**.
29. On the **Distribution Agent Security** page, click **Next**.
30. On the **Synchronization Schedule** page, click **Next**.
31. On the **Initialize Subscriptions** page, click **Next**.
32. On the **Wizard Actions** page, click **Next**.
33. On the **Complete the Wizard** page, click **Finish**.
34. On the **Creating Subscriptions** page, click **Close**.
35. In the Object Explorer pane, right-click **Replication**, and then click **Launch Replication Monitor**. The Replication Monitor window appears.
36. Ensurte that NYC-SQL1 is selected, the server instance, at the tree node.
37. In the **Publications** tab, review the information regarding the publications on the added instance.

38. Close the Replication Monitor window.

Question: Explain the difference between the pull and push replication in the New Subscription Wizard?

Question: What is the advantage of using snapshot replication, compared to other replication types?

Working of Transactional Replication



Key Points

All published tables in a transactional replication must contain a declared primary key. The primary key is used by the replication agent to identify the changes and used for applying the changes to the replica. Primary keys are also used for merge replication.

Incremental changes made at the Publisher flow to Subscribers according to the schedule of the Distribution Agent, which can run continuously for minimal latency, or at scheduled intervals. When transactional replication is used without immediate updating or queue updating options, you may avoid update conflicts because changes to the data must be made at the Publisher. All Subscribers achieve the same values as the Publisher. If you use immediate updating or queue updating options with transactional replication, you can make updates at the Subscriber with queue updating. In such situations, a conflict may occur.

Initial Dataset

Before a new transactional replication Subscriber can receive incremental changes from a Publisher, the Subscriber must contain tables with the same schema and data as the tables at the Publisher. The initial dataset is usually a snapshot that is created by the Snapshot Agent and distributed and applied by the Distribution Agent. The initial dataset can also be supplied through a backup, or other means such as SQL Server Integration Services (SSIS). When snapshots are distributed and applied to Subscribers, only those Subscribers waiting for initial snapshots are affected. Other initialized Subscribers to that publication are unaffected.

Concurrent Snapshot Processing

Snapshot replication places shared locks on all tables published as part of replication for the duration of snapshot generation. This can prevent you from making updates on the publishing tables. Concurrent snapshot processing, the default with transactional replication, does not hold the share locks in place during the entire snapshot generation. This allows users to continue working uninterrupted while replication creates initial snapshot files.

Snapshot Agent

The procedure used by the Snapshot Agent to implement the initial snapshot in transactional replication is the same as that used in snapshot replication. After the snapshot files have been generated, you can view them in the snapshot folder by using Windows Explorer.

Log Reader Agent

The Log Reader Agent runs at the Distributor. It typically runs continuously, but can also run according to a schedule you establish.

When executing, the Log Reader Agent:

1. Reads the publication transaction log (the same database log used for transaction tracking and recovery during regular SQL Server Database Engine operations) and identifies any INSERT, UPDATE, and DELETE statements, or other modifications made to the data in transactions that have been marked for replication.

2. Copies those transactions in batches to the distribution database at the Distributor. The Log Reader Agent uses the internal stored procedure, sp_replcmds, to get the next set of commands marked for replication from the log. The distribution database

then becomes the store-and-forward queue from which changes are sent to Subscribers. Only committed transactions are sent to the distribution database.

After the entire batch of transactions has been written successfully to the distribution database, it is committed.

3. Calls sp_repldone to mark where replication was last completed.
4. Marks the rows in the transaction log that are ready to be purged. Rows still waiting to be replicated are not purged.

Transaction commands are stored in the distribution database until they are propagated to all Subscribers or until the maximum distribution retention period has been reached. Subscribers receive transactions in the same order in which they were applied at the Publisher.

Distribution Agent

The Distribution Agent runs at the Distributor for push subscriptions and at the Subscriber for pull subscriptions. The agent moves transactions from the distribution database to the Subscriber. If a subscription is marked for validation, the Distribution Agent also checks whether data at the Publisher and Subscriber match.

Question: What is the purpose of the Log Reader Agent?

Question: What is the first step for setting up a transactional replication?

Source:

- http://msdn.microsoft.com/en-us/library/ms151706.aspx

Demonstration: How To Set Up Transactional Replication

In this demonstration, you will see how to:

Set up transactional replication

Key Points

The steps to set up transactional replication are:

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **Replication**, right-click the **Local Publications** folder, and then click **New Publication**. The New Publication Wizard launches.
2. In the **New Publication Wizard**, click **Next**.
3. On the **Publication Database** page, under **Databases**, ensure that **QuantamCorp** is selected as the publication database, and then click **Next**.
4. On the Publication Type page, under Publication type, click **Transactional publication**, and then click **Next**.
5. On the Articles page, under **Objects to publish**, expand Tables, and select any one table, and then click **Next**.

6. On the **Filter Table Rows** page, click **Next**.
7. On the **Snapshot Agent** page, select the **Create a snapshot immediately and keep the snapshot available to initialize subscriptions** check box, and then click **Next**.
8. On the **Agent Security** page, click **Security Settings**. The **Snapshot Agent Security** dialog box appears.
9. In the **Snapshot Agent Security** dialog box, click **Run under the SQL Server Agent service account (This is not a recommended security best practice.)**, and then click **OK**.
10. In the New Publication Wizard, on the **Agent Security** page, click **Next**.
11. On the **Wizard Actions** page, click **Next**.
12. On the **Complete the Wizard** page, in the **Publication name** box, type **Transaction**, and then click **Finish**.
13. On the **Creating Publication** page, click **Close**.
14. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **Replication**, under **Local Publications**, right-click **[QuantamCorp]: Transaction**, and then click **New Subscriptions**. The New Subscription Wizard launches.
15. In the **New Subscription Wizard**, click **Next**.
16. On the **Publication** page, ensure that **Transaction** is selected, and then click **Next**.
17. On the **Distribution Agent Location** page, click **Next**.
18. On the **Subscribers** page, click **Add Subscriber**, and then click **Add SQL Server Subscriber**. The **Connect to Server** dialog box appears.
19. In the **Connect to Server** dialog box, in the **Server name** box, type or select **NYC-SQL1\DEVELOPMENT**, the instance by which you want to create the subscription, and then click **Connect**.
20. On the **Subscribers** page, in the **Subscription Database** list for NYC-SQL1\DEVELOPMENT, click **QuantamCorp**, and then click **Next**.

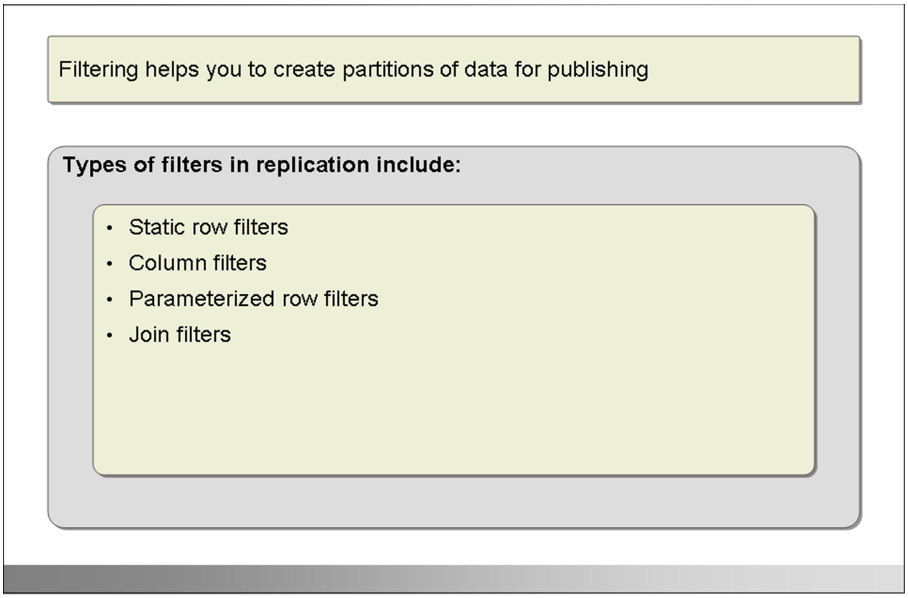
21. On the **Distribution Agent Security** page, click the **ellipsis** button, in the NYC-SQL1\DEVELOPMENT agent for subscriber. The **Distribution Agent Security** dialog box appears.
22. In the **Distribution Agent Security** dialog box, click **Run under the SQL Server Agent service account (This is not a recommended security best practice.)**, and then click **OK**.
23. On the **Distribution Agent Security** page, click **Next**.
24. On the **Synchronization Schedule** page, click **Next**.
25. On the **Initialize Subscriptions** page, click **Next**.
26. On the **Wizard Actions** page, click **Next**.
27. On the **Complete the Wizard** page, click **Finish**.
28. On the **Creating Subscriptions** page, click **Close**.

Note: After completing this demonstration, turn off the virtual machine and discard the changes.

Question: Does transactional replication replicate the entire database?

Question: Do you need to configure snapshot replication along with transactional replication for initializing the database schema?

Filtering Publication Data



Key Points

Filtering helps you to create partitions of data for publishing. You can do the following by filtering published data:

- Minimize the amount of data sent over the network.
- Reduce the amount of storage space required at the Subscriber.
- Customize publications and applications based on individual Subscriber requirements.
- Avoid or reduce conflicts if Subscribers are updating data. This is because different data partitions can be sent to different Subscribers.
- Avoid transmitting sensitive data. You can use row filters and column filters to restrict a Subscriber's access to data. You need to follow security considerations for merge replication if you use a parameterized filter that includes HOST_NAME().

Types of Filters

Replication offers four types of filters:

- **Static row filters**. These filters are available with all replication types.
  By using static row filters, you can choose a subset of rows for publishing. All Subscribers to a filtered publication receive the same subset of rows for the filtered table. A static row filter uses a WHERE clause to select the appropriate data to be published. You need to specify the final part of the WHERE clause. Static row filters result in a single set of data for each publication.
- **Column filters**. These filters are available with all replication types.
  By using column filters, you can choose a subset of columns for publishing. After a publication is created, you can use column filtering to drop a column from an existing publication, but retain the column in the table at the Publisher and include an existing column in the publication. For other changes, such as adding a new column to a table and then adding it to the published article, you need to use schema change replication.

  The following table describes the types of columns that you cannot filter out of certain types of publications.

Column type	Type of publication and options
Primary key column	You need primary key columns for all tables in transactional publications. However, you do not need primary keys for tables in merge publications; but if a primary key column is present, you cannot filter it.
Foreign key column	You cannot set up a filter on a column with a foreign key in the wizard, but you can do it with a stored procedure. You can filter foreign key columns by using Transact-SQL (T-SQL) stored procedures.

Column type	Type of publication and options
The rowguid column	If you are publishing a table in a merge publication and that table already contains a column of data type uniqueidentifier with the ROWGUIDCOL property set, replication can use this column instead of creating an additional column named, rowguid. In this case, the existing column must be published.
The msrepl_tran_version column	Snapshot or transactional publications that allow updatable subscriptions.
Columns that do not allow NULL and do not have default values or the IDENTITY property set	Snapshot or transactional publications that allow updatable subscriptions.
Columns with unique constraints or indexes	Snapshot or transactional publications that allow updatable subscriptions.
All columns in a SQL Server 7.0 merge publication	Columns cannot be filtered in SQL Server 7.0 merge publications.
Timestamp	SQL Server 7.0 snapshot or transactional publications that allow updatable subscriptions.

- **Parameterized row filters**. These filters are available only with merge replication. By using parameterized row filters, you can choose a subset of rows for publishing. Unlike static filters that send the same subset of rows to every Subscriber, parameterized row filters use a data value supplied by the Subscriber to send Subscribers different subsets of rows.

A parameterized row filter uses a WHERE clause to select the appropriate data to be published. Rather than specifying a literal value in the clause, you specify one or both of the following system functions: SUSER_SNAME() and HOST_NAME(). User-defined functions can also be used, but they must include SUSER_SNAME() or HOST_NAME() in the body of the function, or evaluate one of these system functions (such as MyUDF(SUSER_SNAME()). If a user-defined function includes SUSER_SNAME() or HOST_NAME() in the body of the function, you cannot pass parameters to the function.

- **Join filters**. These filters are available only with merge replication.
  By using join filters, you can extend a row filter from one published table to another.

Question: What is the purpose of configuring filtering in a Publisher?

Question: What is the difference between static row filtering and parameterized row filtering?

Source:

- http://msdn.microsoft.com/en-us/library/ms151775.aspx
- http://msdn.microsoft.com/en-us/library/ms152478.aspx

Considerations for Filtering Data

- ✓ Include all columns referenced in row filters in a publication
- ✓ Reinitialize subscriptions, if a filter is added or changed after subscriptions have been initialized
- ✓ Consider the size of the column included in a replication
- ✓ Consider the data type of the column included in a row filter or a join filter
- ✓ Apply caution while replicating an indexed view as a view or table

Key Points

Consider the following for filtering data:

- **Include all columns referenced in row filters in a publication**. This means that you cannot use a column filter to exclude a column that is used in a row filter. Failure to include all columns in a row filter will cause the replication to fail.
- **Reinitialize subscriptions, if a filter is added or changed after subscriptions have been initialized**. This helps to ensure that the databases in all the Subscribers are updated to allow storing additional data.
- **Consider the size of the column included in a replication**. The maximum size allowed for a column used in a filter is 1,024 bytes for an article in a merge publication, and 8,000 bytes for an article in a transactional publication which is the maximum supported data length in SQL Server replication

- **Consider the data type of the column included in a row filter or a join filter**. Columns with the following data types cannot be referenced in row filters or join filters:
  - varchar(max)and nvarchar(max)
  - varbinary(max)
  - text and ntext
  - image
  - XML
  - Uniform data transfer (UDT)
- **Apply caution while replicating an indexed view as a view or table**. Transactional replication allows you to replicate an indexed view as a view or as a table. If you replicate the view as a table, you cannot filter columns from the table.

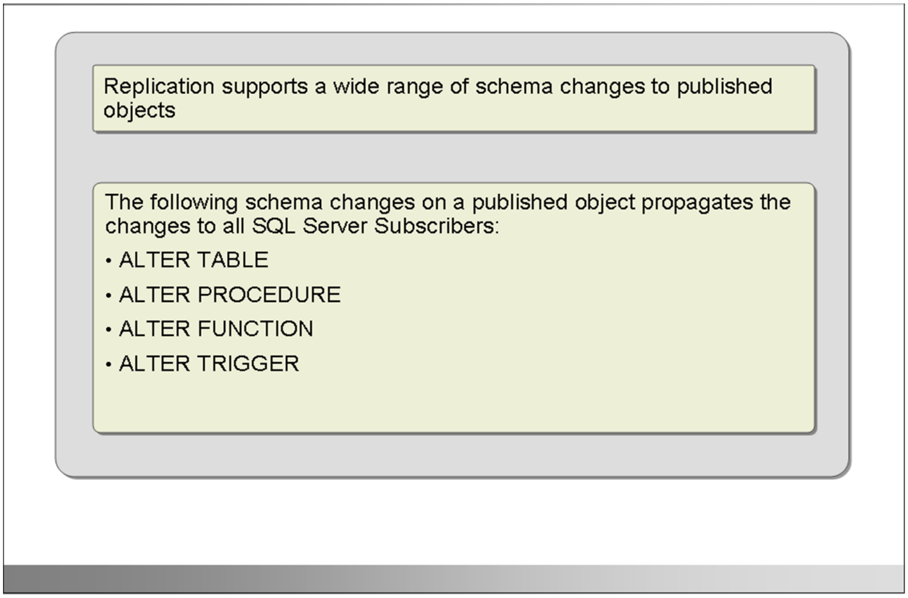
Question: What should you do if you modify the filtering?

Question: What is the restriction when you are publishing an indexed view?

Source:

- http://msdn.microsoft.com/en-us/library/ms151775.aspx

Making Schema Changes on Publication Databases



Key Points

Replication supports a wide range of schema changes to published objects. When you make any of the following schema changes on the appropriate published object at a Microsoft SQL Server Publisher, the change is propagated by default to all SQL Server Subscribers:

- ALTER TABLE
- ALTER PROCEDURE
- ALTER FUNCTION
- ALTER TRIGGER

You can use ALTER TRIGGER only for data manipulation language (DML) triggers because data definition language (DDL) triggers cannot be replicated.

You should not use ALTER TABLE SET LOCK ESCALATION if schema change replication is enabled and a topology includes SQL Server 2005 or SQL Server Compact 3.5 Subscribers.ALTER VIEW

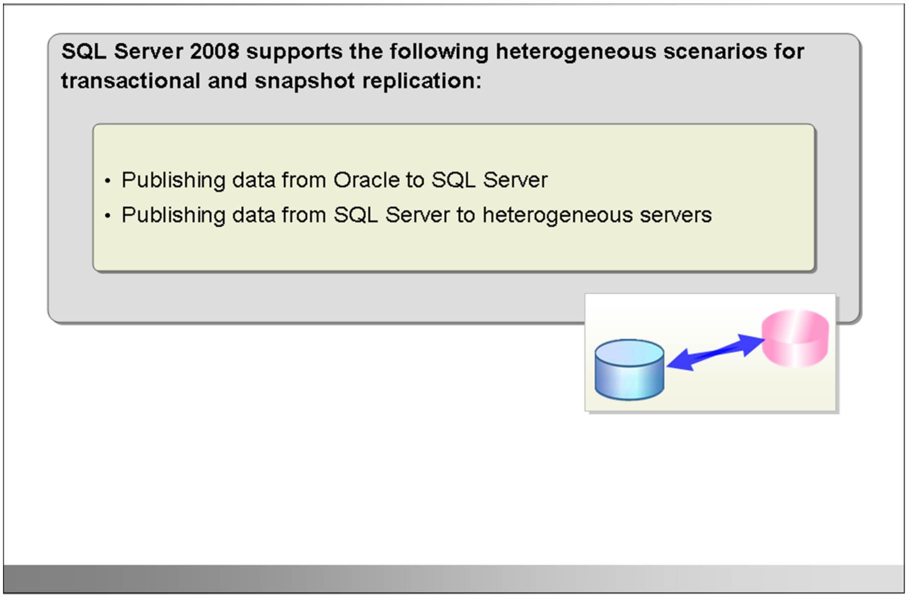
Important: You must make schema changes to tables by using T-SQL or SQL Server Management Objects (SMO). When schema changes are made in SQL Server Management Studio (SSMS), SSMS attempts to drop and re-create the table. You cannot drop published objects, therefore the schema change fails.

For transactional replication and merge replication, schema changes are propagated incrementally when the Distribution Agent or Merge Agent is run. For snapshot replication, schema changes are propagated when a new snapshot is applied at the Subscriber. In snapshot replication, a new copy of the schema is sent to the Subscriber each time synchronization occurs. Therefore, all schema changes to previously published objects are automatically propagated with synchronization.

Source:

- http://technet.microsoft.com/en-us/library/ms151870.aspx

Configuring Heterogeneous Replication



Key Points

SQL Server 2008 supports the following heterogeneous scenarios for transactional and snapshot replication:

- Publishing data from Oracle to SQL Server.
- Publishing data from SQL Server to non-SQL Server Subscribers.

Publishing Data from Oracle

You can use SQL Server 2008 to publish data from Oracle while retaining many features of SQL Server 2008.

You can publish data from Oracle for the following scenarios:

- **Microsoft .NET Framework application deployments**. These are applications developed with Microsoft Visual Studio® and SQL Server, and they operate on data replicated from a non-SQL Server database.

- **Data warehousing staging servers**. You need to ensure that SQL Server staging databases are synchronized with a non-SQL Server database.
- **Migration to SQL Server**. You need to test the application in real time against SQL Server, while replicating the changes of source system. Also, you need to switch to SQL Server when satisfied with the migration.

Publishing Data to Non-SQL Server Subscribers

The following non-SQL Server databases are supported as Subscribers for snapshot and transactional publications:

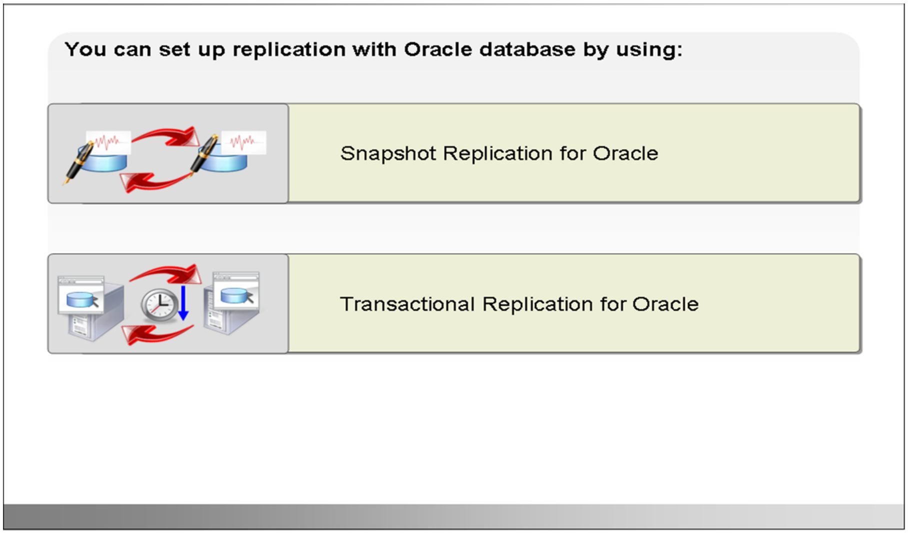
- Oracle for all platforms that Oracle supports
- IBM DB2 for AS400, MVS, Unix, Linux, and Windows

Question: What is the disadvantage of using Heterogeneous replication?

Source:

- http://msdn.microsoft.com/en-us/library/ms151149.aspx

Overview of Oracle Publishing



Key Points

Beginning with Microsoft SQL Server 2005, you can include Oracle Publishers in your replication topology, starting with Oracle version 9i. You can deploy publishing servers on any Oracle supported hardware and operating system. The feature is built on the foundation of SQL Server snapshot replication and transactional replication, providing similar performance and usability.

Snapshot Replication for Oracle

You can implement Oracle snapshot publications similar to SQL Server snapshot publications. When the Snapshot Agent runs for an Oracle publication, it connects to the Oracle Publisher and processes each table in the publication. When processing each table, the agent retrieves the table rows and creates schema scripts, which are then stored on the publication's snapshot share. The entire set of data is created each time the Snapshot Agent runs, so change tracking triggers are not added to the Oracle tables as they are with transactional replication. Snapshot replication provides a convenient way to migrate data with minimal impact on the publishing system.

Transactional Replication for Oracle

You can implement Oracle transactional publications are implemented by using the transactional publishing architecture of SQL Serve. However, changes are tracked by using a combination of database triggers on the Oracle database and the Log Reader Agent. Subscribers to an Oracle transactional publication are automatically initialized by using snapshot replication; subsequent changes are tracked and delivered to Subscribers as they occur through the Log Reader Agent. When you create an Oracle publication, triggers and tracking tables are created for each published table within the Oracle database. When you make data changes to the published tables, the database triggers on the tables fire and insert information into the replication tracking tables for each modified row. The Log Reader Agent on the SQL Server Distributor then moves the data change information from the tracking tables to the distribution database on the Distributor. Finally, as in standard transactional replication, the Distribution Agent moves changes from the Distributor to the Subscribers.

Question: What is the version of Oracle supported to function as a Publisher?

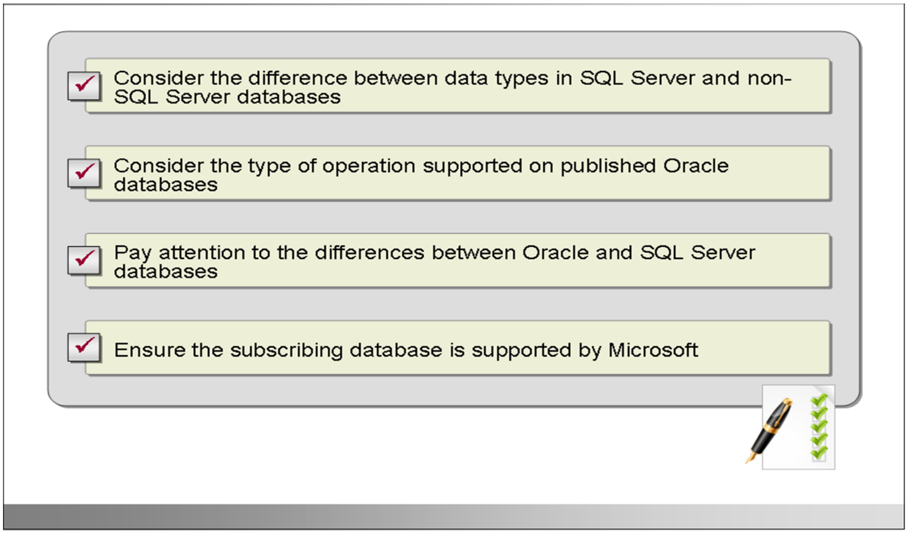
Source:

- http://msdn.microsoft.com/en-us/library/ms151229.aspx

For more information, see How Snapshot Replication Works.

For more information, see How Transactional Replication Works.

Considerations for Using Heterogeneous Database Replication



Key Points

The considerations for using heterogeneous database replication are:

- **Consider the difference between data types in SQL Server and non-SQL Server databases**. Data types in Oracle and SQL Server do not always match. Wherever applicable, the matching data type is selected automatically while publishing an Oracle table. If a matching data type is not available, alternative data type mappings are provided.
- **Consider the type of operation supported on published Oracle databases**. The following actions do not require special considerations:
  - Rebuilding indexes on published tables
  - Adding user triggers to a published table
  - Moving a published table requires you to stop all activity on the published tables.

The following actions require you to drop the publication, perform the operation, and then re-create the publication:

- Truncating a published table
- Renaming a published table
- Adding a column to a published table
- Dropping or modifying a column that is published for replication
- Performing non-logged operations

- **Pay attention to differences between Oracle and SQL Server**. Oracle has different maximum size limits for some objects. Any objects created in the Oracle publication database should adhere to the maximum size limits for the corresponding objects in SQL Server.

By default, Oracle object names are created in upper case. Ensure that you supply the names of Oracle objects in upper case when publishing them through a SQL Server Distributor, if they are upper case on the Oracle database. Failure to specify the objects in the correct case may result in an error message indicating that the object cannot be found.

Oracle has a slightly different SQL dialect from SQL Server. Therefore, row filters should be written in Oracle-compliant syntax.

- **Ensure that the subscribing database is supported by Microsoft**. The following non-SQL Server Subscribers can subscribe to snapshot publication and transactional publication by using push subscriptions. Subscriptions are supported for the two most recent versions of the following databases.

The subscriptions use the most recent version of the OLE DB provider appropriate for each of the following databases:

- **Oracle database**. Subscriptions are supported by using the Oracle OLE DB provider for all the platforms supported by Oracle.

- **IBM DB2**. Subscriptions are supported by using the Microsoft Host Integration Server OLE DB provider for the MVS, AS400, Unix, Linux, and Windows platforms.

Question: What is the key consideration for configuring Heterogeneous replication?

Question: Which non-SQL publisher is supported by SQL Server Heterogeneous replication?

Source:

- http://msdn.microsoft.com/en-us/library/ms151859.aspx

For more information, see Maximum Capacity Specifications for Server SQL.

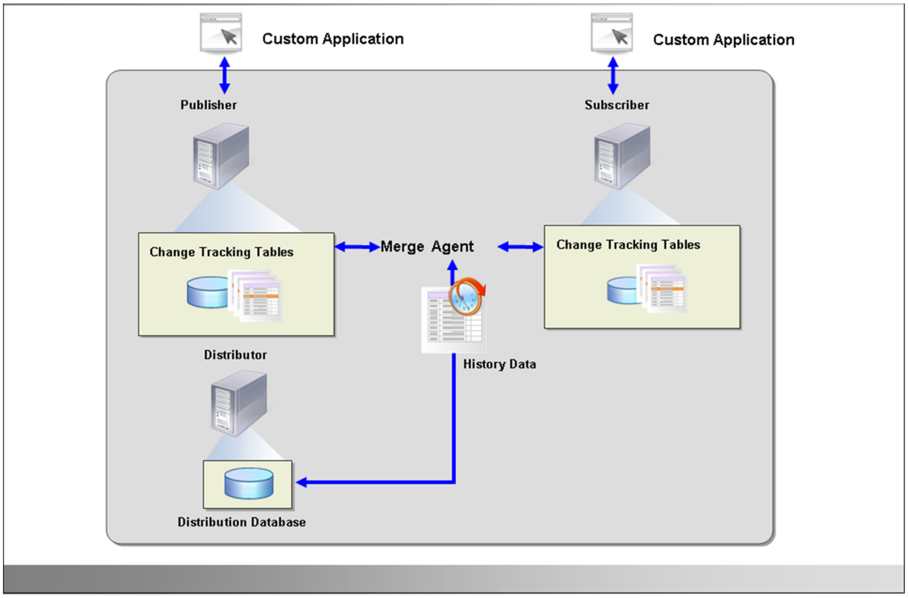
Lesson 3
Configuring a High-Availability Solution by Using Replication

- Working of Merge Replication
- Working of Web Synchronization
- Topologies for Web Synchronization
- How To Set Up Merge Replication
- Peer-to-Peer (P2P) Replication
- How To Set Up P2P Replication
- How Merge Replication Detects and Resolves Conflicts
- Considerations for Using Replication for High Availability

To increase the high-availability of a database solution, you can consider storing data in multiple locations. This reduces the risk of data loss in case of unexpected events. With replication, you can keep multiple copies of data, stored in multiple locations. This allows users to update the data by using the local server. It also helps you distribute content across multiple locations and allows application in each location to update the database.

In this lesson, you will learn about the different types of replication that help in creating a high availability solution and how each type works. You will also learn how SQL Server resolves conflicts in merge replication. In addition, you will learn the considerations for using replication for high availability.

Working of Merge Replication



Key Points

Merge replication, like transactional replication, starts with a snapshot of the publication database objects and data. You can track subsequent data changes and schema modifications made at the Publisher and Subscribers with the help of triggers. The Subscriber synchronizes with the Publisher when connected to the network and exchanges all rows that have changed between the Publisher and Subscriber since the last time the synchronization occurred.

The steps of how merge replication works include:

1. Merge replication is implemented by the SQL Server Snapshot Agent and Merge Agent.
    - If the publication is unfiltered or uses static filters, the Snapshot Agent creates a single snapshot.
    - If the publication uses parameterized filters, the Snapshot Agent creates a snapshot for each partition of data.

2. The Publisher sends the replication data and the information about the publication to the distributor for sending to the Subscribers.
3. The Merge Agent applies the initial snapshots to the Subscribers.
4. After a publication or subscription has been initialized, merge replication tracks and enumerates all changes made by custom applications to the data in published tables. All tracked changes will be stored in the change tracking tables.
5. The Merge Agent then detects and resolves any conflicts according to rules that you configure.

In merge replication, changes are tracked through triggers, which replication creates for each published table in the publication and subscription databases.

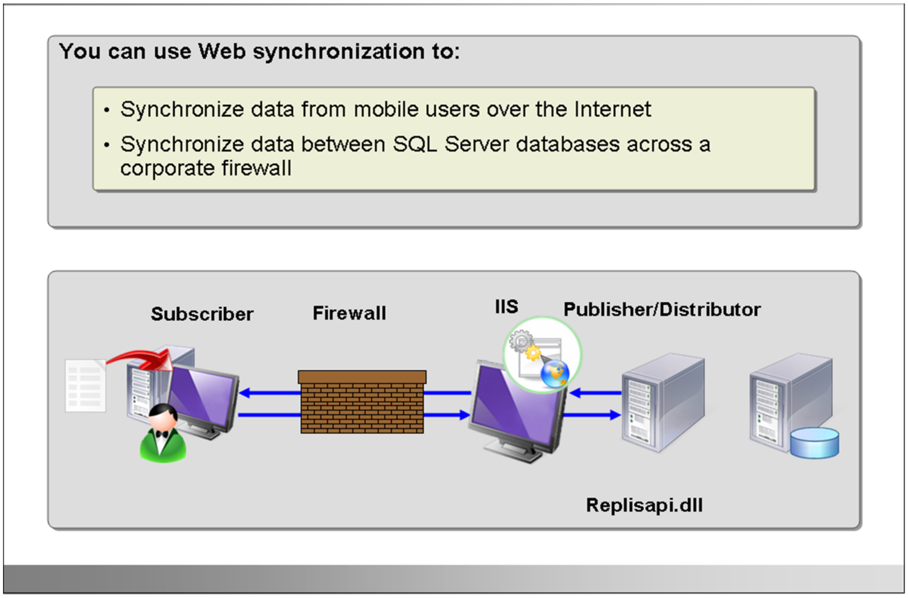
Question: What is the key difference between merge replication and snapshot or transactional replication?

Question: Why should you consider using merge replication over transactional replication when updating Subscriber?

Source:

- http://msdn.microsoft.com/en-us/library/ms152746(SQL.90).aspx
- http://msdn.microsoft.com/en-us/library/ms151329.aspx
- http://msdn.microsoft.com/en-us/library/ms151789.aspx

Working of Web Synchronization



Key Points

Web synchronization for merge replication lets you replicate data by using the HTTPS protocol. You can use Web synchronization to:

- Synchronize data from mobile users over the Internet.
- Synchronize data between SQL Server databases across a corporate firewall.

Web synchronization is designed to synchronize data with portable computers, handheld devices, and other clients. Web synchronization is not intended for high-volume server-to-server applications.

Working of Web Synchronization

To use Web synchronization, you need to use SQL Server 2005 or SQL Server 2008. When you use Web synchronization, updates at the Subscriber are packaged and sent as an XML message to the computer that is running Internet Information Server (IIS) by using the Hypertext Transfer Protocol Secure (HTTPS) protocol. The computer that is running IIS then sends the commands to the

Publisher in a binary format, typically by using TCP/IP. Updates at the Publisher are sent to the computer that is running IIS and then packaged as an XML message for delivery to the Subscriber.

Only pull subscriptions can use Web synchronization. Therefore, a Merge Agent will always run on the Subscriber. This Merge Agent can be the standard Merge Agent, the Merge Agent ActiveX® control, or an application that provides synchronization through Replication Management Objects (RMO). To specify the location of the computer that is running IIS, use the *–InternetUrl* parameter for the Merge Agent.

You should configure the SQL Server Replication Listener, Replisapi.dll, on the computer that is running IIS and is responsible for handling messages that are sent to the server from the Publisher and Subscribers. Each node in the topology handles the XML data stream by using the Merge Replication Reconciler, Replrec.dll.

Synchronization Process

The following steps occur during synchronization:

1. The Merge Agent is started at the Subscriber. The agent:
    a. Makes an SQL connection to the subscription database.
    b. Extracts any changes from the database.
    c. Makes an HTTPS request to the computer that is running IIS.
    d. Uploads data changes as an XML message.
2. The SQL Server Replication Listener and Merge Replication Reconciler that are hosted on the computer that is running IIS do the following:
    a. Respond to the HTTPS request.
    b. Make an SQL connection to the publication database.
    c. Apply the upload changes to the publication database.
    d. Extract the download changes for the Subscriber.
    e. Send an HTTPS response back to the Merge Agent.
3. The Merge Agent at the Subscriber then accepts the HTTPS response and applies the download changes to the subscription database.

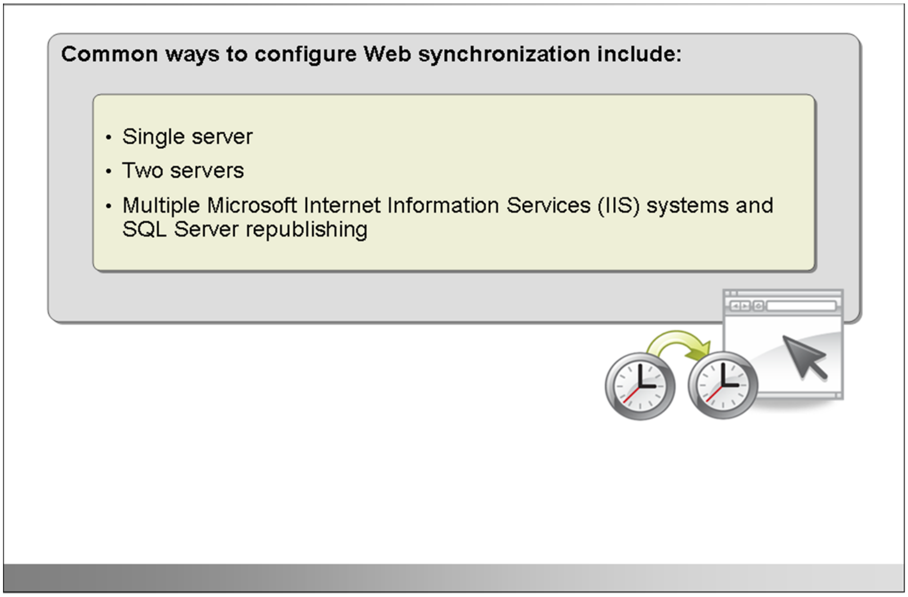
Question: What is the requirement for configuring Web synchronization?

Question: What is the benefit of using Web synchronization over traditional transport protocol?

Source:

- http://msdn.microsoft.com/en-us/library/ms152746(SQL.90).aspx

Topologies for Web Synchronization



Key Points

You can choose from a variety of Web synchronization replication topologies. Common ways to configure Web synchronization include:

Single server

In the simplest topology, IIS, the SQL Server Publisher, and the SQL Server Distributor all reside on a single server. Subscribers synchronize by connecting to IIS on the Publisher. The Publisher can be located behind a firewall.

The following illustration shows the single server topology.

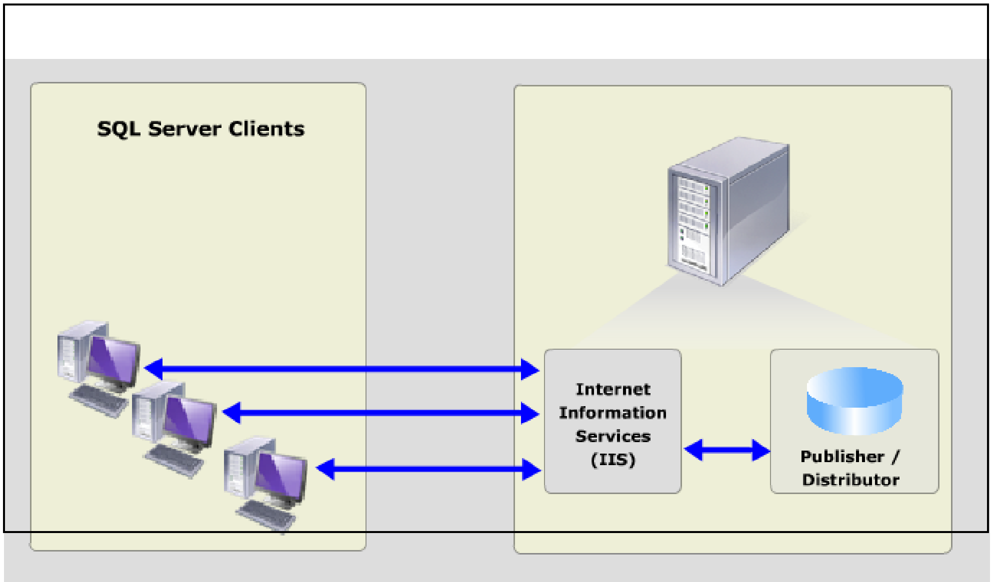


Figure 10.1. Single Server Topology

Note: You should use this configuration only for intranet scenarios. For other scenarios, you should ensure that the IIS server and SQL Server Publisher/Distributor are on separate computers.

Two servers

You can place IIS on one server and configure the SQL Server Publisher and Distributor on another server. The server running IIS can be isolated from the Internet by a firewall. Subscribers synchronize by connecting to IIS.

The following illustration shows two servers topology.

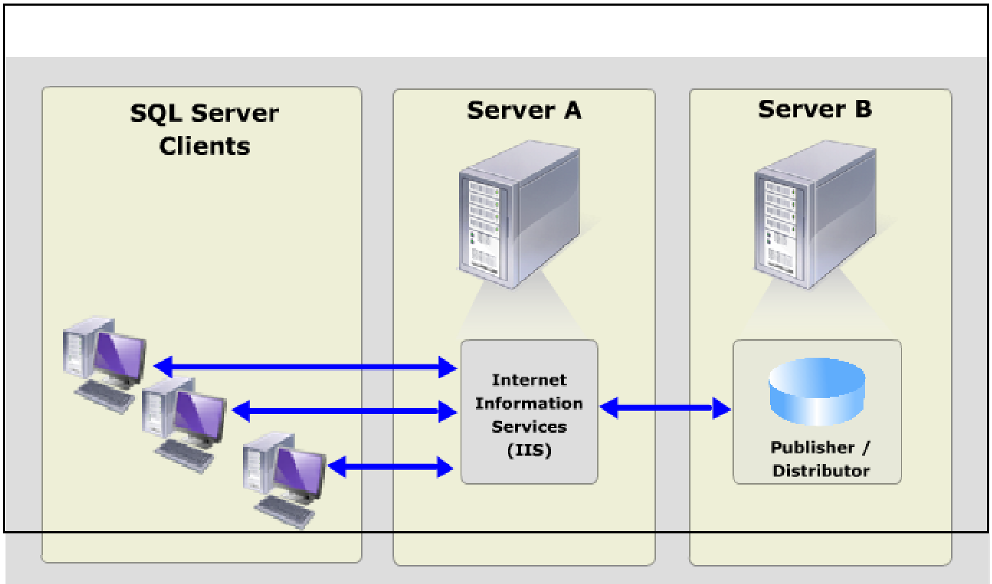


Figure 10.2. Two Servers Topology

Multiple IIS systems and SQL Server republishing

If you need to support a very large number of Subscribers that synchronize at the same time, you can partition the work across multiple computers running IIS.

If further load balancing is required on the computer running SQL Server, you can create a republishing hierarchy on multiple computers. The top-level Publisher publishes data to Subscribers, which in turn republish the data, load balancing the requests from the Subscribers.

The following illustration shows the multiple IIS systems and SQL Server republishing topology.

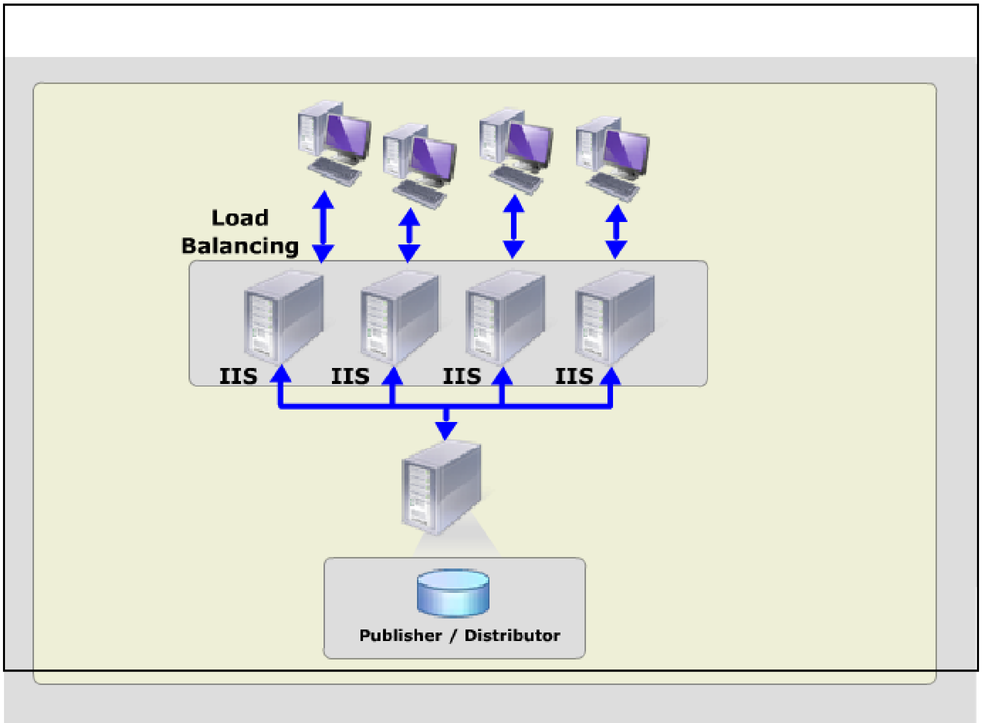


Figure 10.3. Multiple IIS Systems and SQL Server Topology

Note: Subscribers can only synchronize with a specific Publisher. For example, a Subscriber to republisher A cannot synchronize with republisher B when A is not available.

Source:

- http://technet.microsoft.com/en-us/library/ms151811(SQL.90).aspx

Demonstration: How To Set Up Merge Replication

In this demonstration, you will see how to:

Set up merge replication in SSMS

Key Points

The steps to set up merge replication are:

1. On the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. In the Microsoft SQL Server Management Studio window, by using the **Connect to Server** dialog box, connect to the default instance of SQL Server Database Engine.
3. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **Replication**, right-click the **Local Publications** folder, and then click **New Publication**. The New Publication Wizard launches.
4. In the New Publication Wizard, click **Next**.
5. On the **Distributor** page, click **Next**.
6. On the **Snapshot Folder** page, click **Next**.

7. On the **Publication Database** page, under **Databases**, click **QuantamCorp** as the publication database, and then click **Next**.
8. On the Publication Type page, under Publication type, click **Merge publication**, and then click **Next**.
9. On the **Subscriber Types** page, ensure that the **SQL Server 2008** check box is selected, and then click **Next**.
10. On the **Articles** page, under **Objects to publish**, expand Tables, and select any one table, and then click **Next**.
11. On the **Filter Table Rows** page, click **Next**.
12. On the **Snapshot Agent** page, click **Next**.
13. On the **Agent Security** page, click **Security Settings**. The **Snapshot Agent Security** dialog box appears.
14. In the **Snapshot Agent Security** dialog box, click the **Run under the SQL Server Agent service account (This is not a recommended security best practice.)** option, and then click **OK**.
15. In the New Publication Wizard, on the **Agent Security** page, click **Next**.
16. On the **Wizard Actions** page, click **Next**.
17. On the **Complete the Wizard** page, in the **Publication name** box, type **Merge**, and then click **Finish**.
18. On the **Creating Publication** page, click **Close**.
19. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **Replication**, expand **Local Publications**, right-click **[QuantamCorp]: Merge**, and then click **New Subscriptions**. The New Subscription Wizard launches.
20. In the New Subscription Wizard, click **Next**.
21. On the **Publication** page, ensure that **Merge** is selected, and then click **Next**.
22. On the **Merge Agent Location** page, click **Next**.

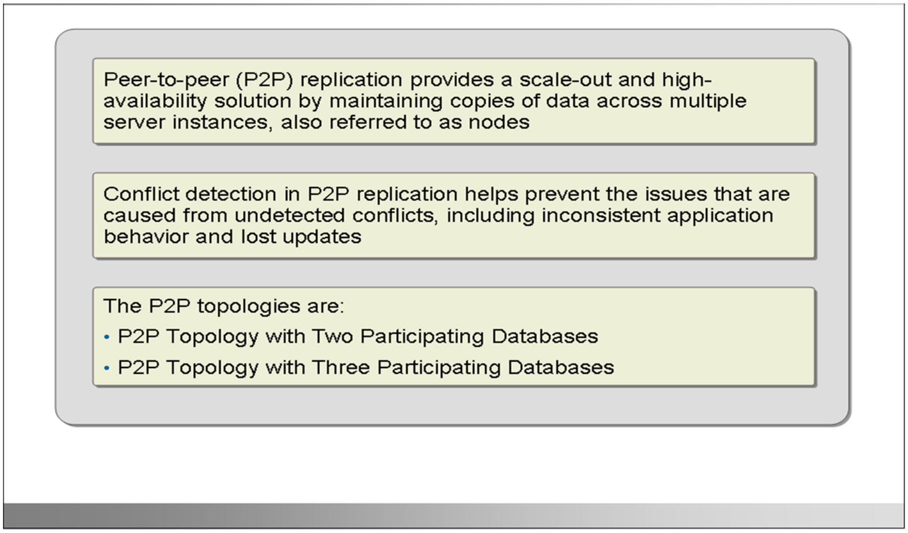
23. On the **Subscribers** page, click **Add SQL Server Subscriber**. The **Connect to Server** dialog box appears.
24. In the **Connect to Server** dialog box, in the **Server name** box, type or select **NYC-SQL1\DEVELOPMENT**, the instance by which you want to create the subscription, and then click **Connect**.
25. On the **Subscribers** page, in the **Subscription Database** list for NYC-SQL1\DEVELOPMENT, click **QuantamCorp**, and then click **Next**.
26. On the **Merge Agent Security** page, click the **ellipsis** button, in the NC-SQL1\DEVELOPMENT agent for subscriber. The **Merge Agent Security** dialog box appears.
27. In the **Merge Agent Security** dialog box, click **Run under the SQL Server Agent service account (This is not a recommended security best practice.)**, and then click **OK**.
28. On the **Merge Agent Security** page, click **Next**.
29. On the **Synchronization Schedule** page, click **Next**.
30. On the **Initialize Subscriptions** page, click **Next**.
31. On the **Subscription Type** page, click **Next**.
32. On the **Wizard Actions** page, click **Next**.
33. On the **Complete the Wizard** page, click **Finish**.
34. On the **Creating Subscriptions** page, click **Close**.

Note: After completing this demonstration, turn off the virtual machine and discard the changes.

Question: What is the difference between transactional replication and merge replication?

Question: What do you need to consider when using merge replication in terms of filtering?

P2P Replication



Key Points

Peer-to-peer (P2P) replication provides a scale-out and high-availability solution by maintaining copies of data across multiple server instances, also referred to as nodes. P2P replication is based on transactional replication. It propagates transactionally consistent changes in near real-time. This enables applications that require scale-out of read operations to distribute the reads from clients across multiple nodes. Because data is maintained across the nodes in near real-time, P2P replication provides data redundancy, which increases the availability of data.

Although P2P replication enables scaling out of read operations, write performance for the topology is similar to that of a single node. This is because all inserts, updates, and deletes are propagated to all nodes. Replication recognizes when a change has been applied to a given node and prevents changes from cycling through the nodes more than one time. You should ensure that write operations for each row are performed at only the node, for the following reasons:

- If a row is modified at more than one node, it can cause a conflict or even a lost update when the row is propagated to other nodes.
- There is usually some latency involved when changes are replicated. For applications that require the latest change to be reflected immediately, dynamically load balancing the application across multiple nodes can cause problems.

Conflict Detection in P2P Replication

P2P replication in SQL Server 2008 introduces the option to enable conflict detection across a P2P topology. This option helps prevent the issues that are caused from undetected conflicts, including inconsistent application behavior and lost updates. By enabling this option, a conflicting change is treated as a critical error that causes the failure of the Distribution Agent. This is the default behavior. In case of a conflict, the topology remains in an inconsistent state until the conflict is resolved manually and the data is made consistent across the topology.

P2P Topologies

The P2P topologies are:

- P2P Topology with Two Participating Databases
- P2P Topology with Three Participating Databases

P2P Topology with Two Participating Databases

You can use a P2P topology with two databases for different applications, from Web sites to workgroup applications. This kind of a topology provides the following benefits:

- Improved read performance, because reads are spread out over two servers.
- Higher availability if maintenance is required or in case of failure at one node.

The following illustration shows a P2P topology with two participating databases.

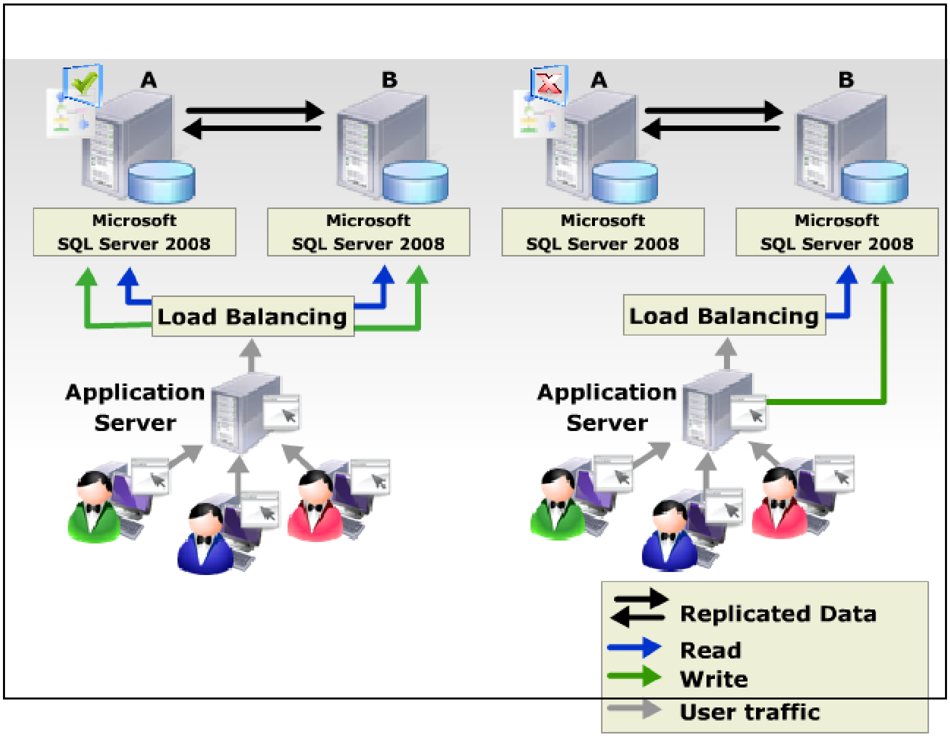


Figure 10.4. P2P Topology with Two Participating Databases

In the illustrations, the read activity is load-balanced between the participating databases, but updates are handled differently:

- On the left, updates are partitioned between the two servers. If the database contained a product catalog, you could have a custom application direct updates to node A for product names that start with A through M, and direct updates to node B for product names that start with N through Z. Updates are then replicated to the other node.

-
- On the right, all updates are directed to node B. From there, updates are replicated to node A. If node B is offline, such as for maintenance, the application server can direct all activity to node A. When node B is back online, updates can flow to it, and the application server can move all updates back to node B or keep directing them to node A.

P2P Topology with Three Participating Databases

Each location has a database and an application server, which are used by the support engineers as they enter and update information about customer calls. The topology is partitioned by time. Therefore, updates occur only at the node that is currently open for business, and then the updates flow to the other participating databases.

A P2P topology with three participating databases provides the following benefits:

- Each location can insert, update, or delete data independently and can also share the data because it is replicated to all other participating databases.
- This topology provides higher availability in case of failure or to allow maintenance at one or more of the participating databases.

The following illustration shows a P2P topology with three participating databases

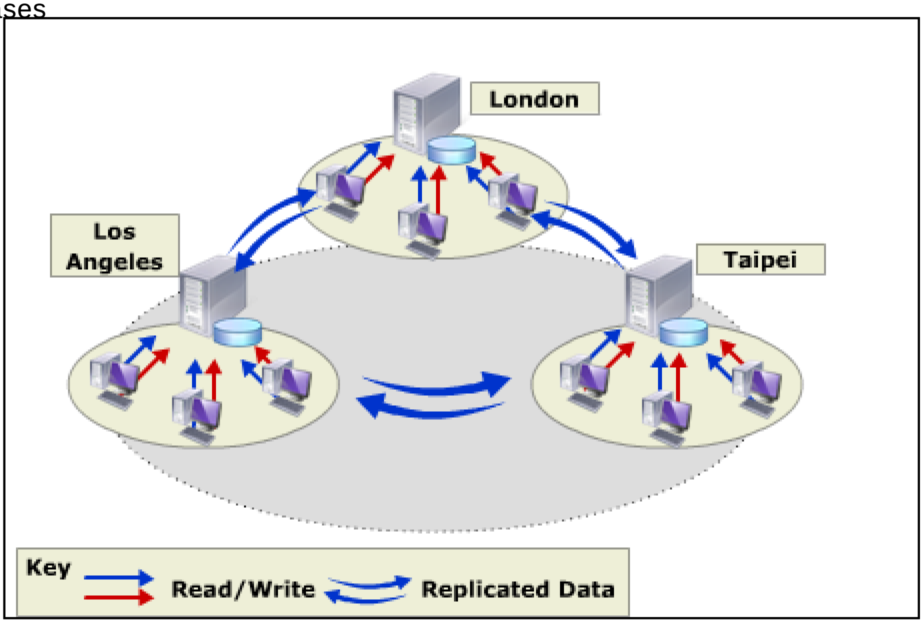


Figure 10.5. P2P Topology with Three Participating Databases

The preceding illustration shows three participating databases that provide data for a worldwide software support organization, with offices in Los Angeles, London, and Taipei. The support engineers at each office take customer calls, and enter and update information about each customer call. The time zones for the three offices are eight hours apart, so there is no overlap in the workday. As the Taipei office closes, the London office is opening for the day. If a call is still in progress as one office is closing, the call is transferred to a representative in the next office to open.

Question: What is the benefit of using P2P replication?

Question: What is the difference between P2P replication and normal transactional replication with updating Subscriber?

Source:

- http://technet.microsoft.com/en-us/library/ms151196.aspx

Demonstration: How To Set Up P2P Replication

In this demonstration, you will see how to:

- Set up P2P replication
- Change P2P replication topologies

Key Points

The steps to set up P2P replication are:

1. On the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. In the Microsoft SQL Server Management Studio window, by using the **Connect to Server** dialog box, connect to the default instance of SQL Server Database Engine.
3. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **Replication**, right-click the **Local Publications** folder, and then click **New Publication**. The New Publication Wizard launches.
4. In the **New Publication Wizard**, click **Next**.
5. On the **Distributor** page, click **Next**.
6. On the **Snapshot Folder** page, click **Next**.

7. On the **Publication Database** page, under **Databases**, click **QuantamCorp** as the publication database, and then click **Next**.
8. On the **Publication Type** page, under **Publication type**, click **Transactional publication**, and then **Next**.
9. On the **Articles** page, under **Objects to publish**, expand Tables, and select any one table, and then click **Next**.
10. On the **Filter Table Rows** page, click **Next**.
11. On the **Snapshot Agent** page, select the **Create a snapshot immediately and keep the snapshot available to initialize subscriptions** check box, and then click **Next**.
12. On the **Agent Security** page, click **Security Settings**. The **Snapshot Agent Security** dialog box appears.
13. Under **Run under the following Windows account**, in the **Process account** box, type **NYC-SQL1\ADMINISTRATOR**.
14. In the **Password** box, type **Pa$$w0rd**, and in the **Confirm Password** box, type **Pa$$w0rd**, and then click **OK**.
15. In the New Publication Wizard, on the **Agent Security** page, click **Next**.
16. On the **Wizard Actions** page, click **Next**.
17. On the **Complete the Wizard** page, in the **Publication name** box, type **Publication Properties**, and then click **Finish**.
18. On the **Creating Publication** page, click **Close**.
19. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, click **Connect**, and then click **Database Engine**. The **Connect to Server** dialog box appears.
20. In the **Connect to Server** dialog box, logon to another SQL instance with the following credentials:
    - Server type: **Database Engine**
    - Server name: **NYC-SQL1\DEVELOPMENT**
    - Authentication: **Windows Authentication**
21. Click **Connect**.

22. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **NYC-SQL1\DEVELOPMENT**, expand **Replication**, right-click **Replication**, and then click **Configure Distribution**. The Configure Distribution Wizard launches.

23. On the **Configure Distribution Wizard** page, click **Next**.

24. On the **Distributor** page, click **Next**.

25. On the **Snapshot Folder** page, in the **Snapshot folder** box, type **\\NYC-SQL1\Repl_DEVELOPMENT**, and then click **Next**.

26. On the **Distribution Database** page, click **Next**.

27. On the **Publishers** page, click **Next**.

28. On the **Wizard Actions** page, click **Next**.

29. On the **Complete the Wizard** page, click **Finish**.

30. On the **Configuring** page, click **Close**.

31. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **NYC-SQL1**, under **Replication**, expand **Local Publications**, right-click **[QuantamCorp]: Publication Properties**, and then click **Properties**. The **Publication Properties – Publication Properties** dialog box appears.

32. In the **Publication Properties - Publication Properties** dialog box, in the Console pane, under **Select a page**, click **Subscription Options**. In the work pane, under **Peer-to-Peer Replication**, in the **Allow peer-to-peer subscriptions** list, click **True**, and then click **OK**.

33. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **Replication**, under **Local Publications**, right-click **[QuantamCorp]: Publication Properties**, and then click **Configure Peer-To-Peer Topology**. The Configure Peer-To-Peer Topology Wizard launches.

34. On the **Configure Peer-To-Peer Topology Wizard** page, click **Next**.

35. On the **Publication** page, under **Databases and publications**, ensure that the appropriate publication is selected, and then click **Next**.

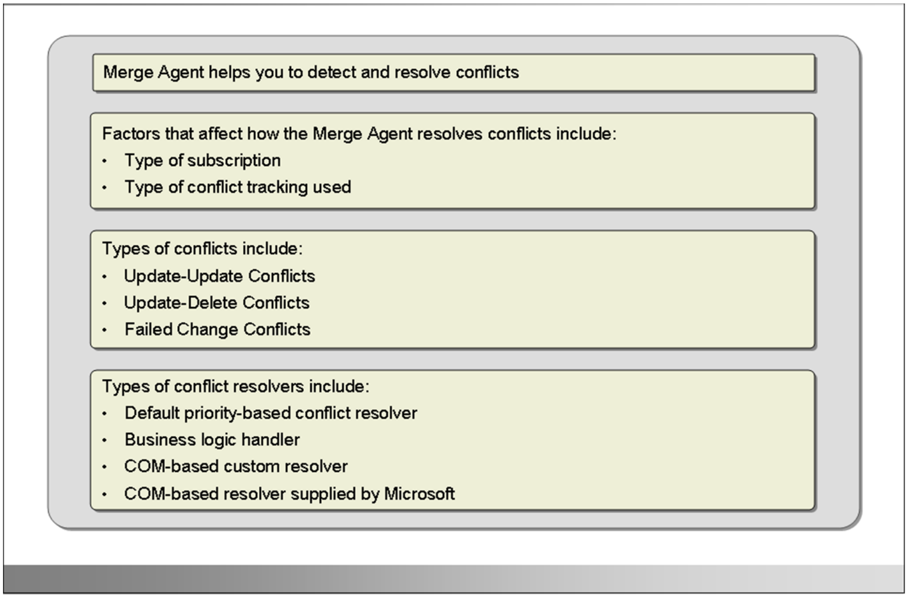
36. On the **Configure Topology** page, right–click anywhere in the white space, and then click **Add a New Peer Node**. The **Connect to Server** dialog box appears.
37. In the **Connect to Server** dialog box, in the **Server name** box, type **NYC-SQL1\DEVELOPMENT** or select the SQL Server instance with a distribution configured to create a peer node, and then click **Connect**. The **Add a New Peer Node** dialog box appears.
38. In the **Add a New Peer Node** dialog box, in the **Server Database** drop down list, click **QuantamCorp**.
39. In the **Add a New Peer Node** dialog box, select the **Connect to ALL displayed nodes** check box, in the **Peer Originator ID** box, type **3**, and then click **OK**.
40. On the **Configure Topology** page, click **Next**.
41. On the **Log Reader Agent Security** page, click the **ellipsis** button in the Log Reader Agent that you created. The **Log Reader Agent Security** dialog box appears.
42. In the **Log Reader Agent Security** dialog box, in the **Process account** box, type **NYC-SQL1\Administrator**.
43. In the **Password** box, type **Pa$$w0rd**.
44. In the **Confirm Password** box, type **Pa$$w0rd**, and then click **OK**.
45. In the Configure Peer-To-Peer Topology Wizard, on the **Log Reader Agent Security** page, click **Next**.
46. On the **Distribution Agent Security** page, click the **ellipsis** button, against **NYC-SQL1**. The **Distribution Agent Security** dialog box appears.
47. The **Distribution Agent Security** dialog box, under **Run under the following Windows account**, in the **Process account** box, type **NYC-SQL1\Administrator**.
48. In the **Password** box, type **Pa$$w0rd**, and in the **Confirm Password** box, type **Pa$$w0rd**, and then click **OK**.
49. On the Distribution Agent Security page, select the **Use the first peer's security settings for all other peers** check box.

50. On the **Configure Peer-To-Peer Topology Wizard**, on the **Distribution Agent Security** page, click **Next**.
51. On the **New Peer Initialization** page, click **Next**.
52. On the **Complete the Wizard** page, click **Finish**.
53. On the **Building the Peer-To-Peer Topology** page, click **Close**.

Question: What is the benefit of using P2P replication?

Question: What should you do if a SQL Server fails in P2P replication topology?

How Merge Replication Detects and Resolves Conflicts



Key Points

With Merge replication, multiple nodes can make autonomous data changes, so situations exist in which a change made at one node may conflict with a change made to the same data at another node. In other situations, the Merge Agent encounters an error such as a constraint violation and cannot propagate a change made at a particular node to another node.

The Merge Agent detects such conflicts by using the lineage column of the MSmerge_contents system table. If column-level tracking is enabled for an article, the Merge Agent also uses the COLV1 column. These columns contain metadata about when a row or column is inserted or updated, and about which nodes in a merge replication topology made changes to the row or column. You can use the system stored procedure, sp_showrowreplicainfo (T-SQL), to view this metadata.

As the Merge Agent enumerates changes to be applied during synchronization, it compares the metadata for each row at the Publisher and Subscriber. The Merge Agent uses this metadata to determine if a row or column has changed at more than one node in the topology, which indicates a potential conflict. After a conflict is detected, the Merge Agent starts the conflict resolver specified for the article with a conflict and uses the resolver to determine the conflict winner. The winning row is applied at the Publisher and Subscriber, and the data from the losing row is written to a conflict table.

The Merge Agent resolves conflicts automatically and immediately unless you have chosen interactive conflict resolution for the article. If you manually change the winning row for a conflict by using the merge replication Conflict Viewer, the Merge Agent applies the winning version of the row to the losing server during the next synchronization.

Factors that Affect Conflict Resolution

Factors that affect how the Merge Agent resolves a detected conflict include:

- **Type of subscription**. This refers to whether the subscription type is a client or a server. Pull or push subscriptions do not affect conflict resolution.
- **Type of conflict tracking used**. This refers to whether conflicts are tracked at row-level, column-level, or at logical record-level.

Types of Conflicts

Although the majority of conflicts are related to updates, there are other conflict types. Such types of conflicts can occur during the upload phase or the download phase of merge processing. Upload processing is the first reconciliation of changes performed in a particular merge session. The Merge Agent replicates changes from the Subscriber up to the Publisher during this phase. Conflicts detected during this processing are referred to as upload conflicts. Download processing involves moving changes from the Publisher to the Subscriber, and occurs after upload processing. Conflicts during this phase of processing are referred to as download conflicts. Some of the conflict types include:

- **Update-Update conflicts**. The Merge Agent detects update-update conflicts when an update to a row, column, or logical record at one node conflicts with another update to the same row at another node. In this case, the default resolver sends the winning version of the row to the losing node and logs the losing row version in the article conflict table.

- **Update-Delete conflicts**. The Merge Agent detects update-delete conflicts when an update of data at one node conflicts with a delete at another. In this case, the Merge Agent updates a row. However, when the Merge Agent searches for that row at the destination, it cannot find the row because it has been deleted. If the winner is the node that updated the row, the delete at the losing node is discarded and the Merge Agent sends the newly updated row to the conflict loser.
- **Failed Change conflicts**. The Merge Agent raises these conflicts when it cannot apply a particular change. This typically occurs because of a difference in constraint definitions between the Publisher and Subscriber, and the use of the NOT FOR REPLICATION (NFR) property on the constraint.

Types of Resolvers

In merge replication, conflict resolution takes place at the article level. For publications composed of several articles, you can have different conflict resolvers serving different articles, or the same conflict resolver serving one article, several articles, or all the articles comprising a publication.

If you plan to use the default priority-based conflict resolver, you do not have to set the resolver property of an article. If you want to use an article resolver instead of the default resolver, you must set the resolver property for the article that will use it by selecting an available resolver on the Publisher. Any specific information that needs to be passed to the resolver can also be specified in the resolver information property.

Merge replication offers four types of conflict resolvers:

- **The default priority-based conflict resolver**. The default resolution mechanism depends on whether a subscription is a client subscription or a server subscription. You assign priority values to individual Subscribers that use server subscriptions. Changes made at the node with the highest priority win any conflicts. For client subscriptions, the first change written to the

Publisher wins the conflict. After a subscription is created, it cannot be changed from one type to another.

- **A business logic handler**. The business logic handler framework allows you to write a managed code assembly that is called during the merge synchronization process. The assembly includes business logic that can respond to conflicts and a number of other conditions during synchronization.
- **A COM-based custom resolver**. Merge replication provides an API for writing resolvers as COM objects in languages such as Microsoft Visual C++® or Microsoft Visual Basic®.
- **A COM-based resolver supplied by Microsoft**. Microsoft SQL Server includes a number of COM-based resolvers.

Resolving Conflicts with Interactive Resolver

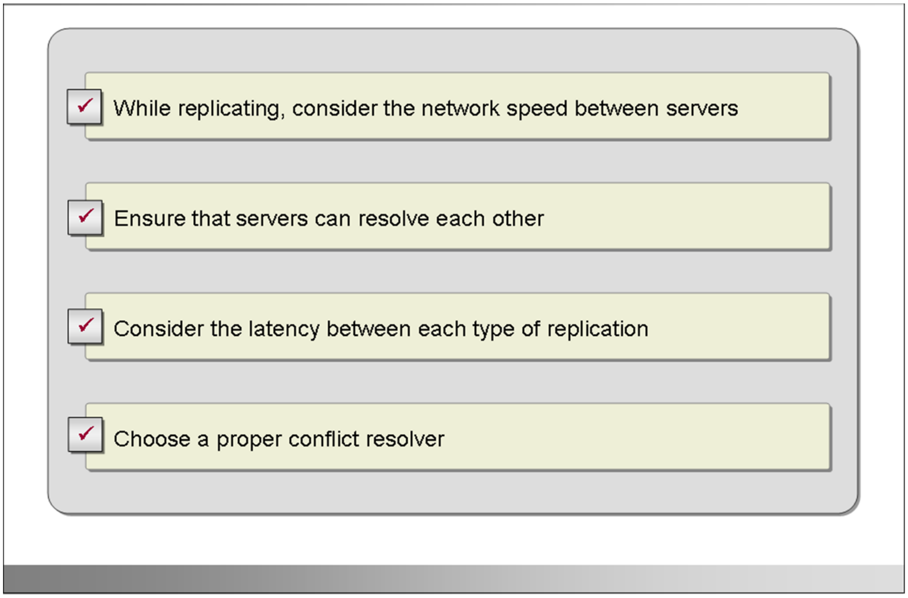
SQL Server replication provides an Interactive Resolver, which allows you to resolve conflicts manually during on-demand synchronization in Microsoft Windows Synchronization Manager.

Question: What is the purpose of using the conflict viewer when troubleshooting replication?

Source:

- http://msdn.microsoft.com/en-us/library/ms151749.aspx
- http://msdn.microsoft.com/en-us/library/ms151257.aspx
- http://msdn.microsoft.com/en-us/library/aa179421(SQL.80).aspx

Considerations for Using Replication for High Availability



Key Points

When using replication for high availability, you should take into account the following considerations:

- **Consider the network speed between servers**. When replicating over a slow connection, you can customize the profiles for replication agents. The available configuration options vary with the agent. In many cases, the network speed is an important factor when applying the initial snapshot. The volume of incremental data changes might be low, but the volume of data initially distributed might be high.
- **Ensure that servers can resolve each other**. When specifying server instance names participating in replication, you must supply the name in the format of the SQL Server registered server instance name. For example, you must use the SQL Server instance name when specifying Publisher or Subscriber

parameters to replication stored procedures or the Replication Agent connection settings at the command line. If the network name for the SQL Server instance differs from the registered instance name, replication connections by agents will not succeed. If the network name of the instance and the SQL Server instance name differ, consider adding the SQL Server instance name as a valid network name. One method to set an alternative network name is to add it to the local hosts file. The local hosts file is located by default at WINDOWS\system32\drivers\etc or WINNT\system32\drivers\etc. For example, if the computer name is comp1 and the computer has an IP address of 10.193.17.129, and the instance name is inst1/instname, add the 10.193.17.129 inst1 entry to the hosts file.

- **Consider the latency between each type of replication.** Replication uses the publish-subscribe model, allowing a primary server, or Publisher, to distribute data to one or more secondary servers, or Subscribers. Replication allows real-time availability and scalability across these servers. It supports filtering to provide a subset of data at Subscribers, and allows partitioned updates. Subscribers are online and available for reporting or other functions, without query recovery. SQL Server offers three types of replication—snapshot, transactional, and merge. Transactional replication provides the lowest latency and is most commonly used for high availability.
- **Choose a proper conflict resolver**. Choose the proper conflict resolver that works best for the type of conflict that could occur in your environment. If you select the wrong conflict resolver, the data quality can deteriorate and lead to storage of incorrect information.

Question: What is the importance of latency between each type of replication?

Question: What's the importance of name resolution in setting up a replication?

Source:

- http://msdn.microsoft.com/en-us/library/ms152479.aspx

- http://msdn.microsoft.com/en-us/library/ms190202.aspx

Lab 10: Designing a Strategy for Replication in SQL Server 2008

- Exercise 1: Designing a Replication Strategy (Discussion)
- Exercise 2: Setting Up Snapshot Replication
- Exercise 3: Setting Up Peer-to-Peer Replication

Logon Information

Virtual machine	50400A-NYC-SQL1
User name	Administrator
Password	Pa$$w0rd

Estimated time: 60 minutes

Objectives:

After completing this lab, you will be able to:

- Design a replication strategy.
- Set up snapshot replication.
- Set up a peer-to-peer replication.

Scenario

QuantamCorp has deployed the first two SQL Server 2008 servers in their organization and will add more to replace their existing installations of SQL Server 2005. QuantamCorp wants to make SQL Server 2008 the primary database engine for their organization. To increase its business growth, the company has decided to implement the application in other regional offices in China, Singapore, U.K.,and Japan. For performance reasons, the application team recommends that a local copy of the database should be used, instead of a centralized database. You are assigned as the architectof the project. After evaluating different options, you decide toconfigure both snapshot replication and peer-to-peer replication forthe application. Snapshot replication will be used for data that isstored in a database, which contains mainly read-only data. Peer-to-peer replication will be used for databases that users in regionaloffices will be updating.

You are assigned as the architect of this project. You need to achieve the following:

- Determine the need for replication.
- Determine the type of replication to use for a particular database.
- Implement snapshot replication.
- Implement peer-to-peer transactional replication.
- Implement replication over the Web.

Exercise 1: Designing a Replication Strategy (Discussion)

Scenario

Your application team recently created an application that requires intense read access to the SQL Server database. During initialtesting, the database is located in corporate headquarters and theapplication that requires access to this database performs poorly.You are required to fix this poor performance issue. You find out thatthe poor performance is due to a large volume of data transferbetween the application and the database. The large volume of datatransfer would also adversely affect the network bandwidth. The application team informs you that you are not allowed to change the application logic because of the limited timeframe. You have to come up with a solution to resolve this performance issue withoutmodifying the database logic.

The application uses the App_Lookup and the App_Data. App_Lookup databases for storing the data generated by a batch job. The batch job runs at midnight on every Saturday. No data updates will be performed to the App_Lookup database during the week. The App_Data database stores all transactional data, and hence, requires intense read and write access when the application is running.

Based on the above scenario, answer the following questions:

1. Which strategy would you propose to address the performance issue with the application?
2. How would you replicate the App_Lookup database?
3. How would you replicate the App_Data database?

Exercise 2: Setting Up Snapshot Replication

Scenario

You have a database that is used to store look up information, like product category, which only gets updated once every week. You want to reduce the network bandwidth usage by replicating the database only after it gets refreshed. To do this, you choose to use snapshot replication.

The main tasks for this exercise are as follows:

1. Create a test database.
2. Create a snapshot publication.
3. Create subscriptions.
4. Verify the snapshot replication.

Task 1: Create a test database.

1. Open the Microsoft SQL Server Management Studio window.
2. In the Connect to Server dialog box, type the following to connect to Microsoft SQL Server Management Studio:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Using the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a script to create a test database, **Mod10** on the **NYC-SQL1** instance, with the following structure.

File name	**Settings**
Mod10_Primary.mdf	Current size: 10 Maximum size: 50 Filegrowth: 5
Mod10_Log.ldf	Current size: 10 Maximum size: 50 Filegrowth: 5

Create a test table named, **[dbo].[CallDetails],** with the following structure and insert records as per details given in the following table.

Field name	Data type	Value	Record
call_date	datetimes	NOT NULL	A date between 2004-01-01 and 2009-12-31

Field name	Data type	Value	Record
Caller	varchar(20)	NOT NULL	Peter
Duration	Int	NULL	30
Product	Int	NOT NULL	A random number between one and three
cs_rep	Int	NOT NULL	A random number between one and five
additional_info	nchar(4000)	NULL	Call to 'x', where 'x' is a random number between one and 100,000

Note: There will be one record for each date.

Note: Browse to the **D:\Labfiles\Mod10** folder and select the **E2_T1.sql** file to execute the required T-SQL code.

Task 2: Create a snapshot publication.

1. Using the Microsoft SQL Server Management Studio window, start the **New Publication Wizard**.
2. Run the **New Publication Wizard** and create a snapshot publication named, **Mod10 Snapshot Pub,** to publish the **Mod10** database with the following Snapshot Agent security settings:
   - Distributor: **NYC-SQL1**
   - Snapshot Folder: **\\NYC-SQL1\Repl**
   - Publication Database: **Mod10**
   - Publication Type: **Snapshot publication**

- Articles: **All tables**
- Snapshot Agent: **Create a snapshot immediately and keep the snapshot available to initialize subscriptions**
- Process account: **NYC-SQL1\Administrator**
- Password: **Pa$$w0rd**

Task 3: Create subscriptions.

1. Using the Microsoft SQL Server Management Studio window, start the **New Subscription Wizard**.
2. Run the **New Subscription Wizard** to create a subscription with the following settings:
   - Add a SQL Server subscriber with the following credentials:
     - Server type: **Database Engine**
     - Server name: **NYC-SQL1\DEVELOPMENT**
     - Authentication: **Windows Authentication**
   - Run the Distribution Agent Security under the following Windows account:
     - Process account: **NYC-SQL1\Administrator**
     - Password: **Pa$$w0rd**

Task 4: Verify the snapshot replication.

1. Using the Microsoft SQL Server Management Studio window, open the Replication Monitor window.
2. In the Replication Monitor window, verify if the snapshot is running.
3. Close the Replication Monitor window.
4. Using the Microsoft SQL Server Management Studio window, log on to SQL instance with the following credentials:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1\DEVELOPMENT**
   - Authentication: **Windows Authentication**

5. Using the Microsoft SQL Server Management Studio window, select the first 1,000 records from the **dbo.CallDetails** table.
6. Close the SQL Server Management Studio window.

Results: After completing this exercise, you should have created a test database, created a snapshot publication, created subscriptions, and verified the snapshot replication.

Exercise 3: Setting Up Peer-to-Peer Replication

Scenario

You have an application database with a number of servers around the world. You are required to replicate the data between these offices as the management wants the users to view changes that occurred in other offices in a reasonable time. You decide to use P2P replication which helps to reduce the need to manually change the replication topology if any of the servers involved in the replication fails.

The main tasks for this exercise are as follows:

1. Create a test database.
2. Initialize a database schema.
3. Create a publication.
4. Configure Distribution Agent.
5. Enable peer-to-peer replication.
6. Configure peer-to-peer topology.
7. Verify the replication setup.

Task 1: Create a test database.

1. Using the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a script to create a test database, **Mod10_P2P** on the **NYC-SQL1** instance, with the following structure.

File name	**Settings**
Mod10_P2P_Primary.mdf	Current size: 10

File name	Settings
	Maximum size: 50 Filegrowth: 5
Mod10_P2P_Log.ldf	Current size: 10 Maximum size: 50 Filegrowth: 5

Create a test table named, **[dbo].[CallDetails]**, with the following structure:

Field name	Data type	Value
Id	Uniqueidentifier	Default value of this column is NEWID().
Call_date	Datetime	NOT NULL
Caller	Varchar(20)	NOT NULL
Duration	Int	NULL
Product	Int	NOT NULL
Cs_rep	Int	NOT NULL

2. Allow only the duration column to have NULL values. Create a constraint named, **PK_CallDetails,** on the primary key **id**.

Note: Browse to the **D:\Labfiles\Mod10** folder and select the **E3_T1.sql** file to execute the required T-SQL code.

Task 2: Initialize a database schema.

1. Using the Microsoft SQL Server Management Studio window, copy the **Mod10_P2P** database from **NYC-SQL1** to **NYC-SQL1\DEVELOPMENT**, by using the **Copy Database Wizard**.
2. Using the Microsoft SQL Server Management Studio window, copy the **Mod10_P2P** database from **NYC-SQL1** to

NYC-SQL1\INSTANCE3, by using **Copy Database Wizard**.

Task 3: Create a publication.

1. Using the Microsoft SQL Server Management Studio window, in **NYC-SQL1**, start the **New Publication Wizard** used for replication.
2. Run the **New Publication Wizard** and create a transactional publication named, **Mod10_P2P Trans Pub,** to publish the **Mod10_P2P** database with the following Snapshot Agent security settings:
   - Select the **Create a snapshot immediately and keep the snapshot available to initialize subscriptions** check box.
   - Process account: **NYC-SQL1\Administrator**
   - Password: **Pa$$w0rd**

Task 4: Configure Distribution Agent.

1. In the Microsoft SQL Server Management Studio window, open the Connect to Server dialog box.
2. In the Connect to Server dialog box, type the following to connect to Microsoft SQL Server Management Studio:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1\DEVELOPMENT**
   - Authentication: **Windows Authentication**
3. On the **NYC-SQL1\DEVELOPMENT** instance, by using the Microsoft SQL Server Management Studio window, start the Configure Distribution Wizard to configure Distribution Agent with **\\NYC-SQL1\Repl_DEVELOPMENT** as the snapshot folder.
4. In the Microsoft SQL Server Management Studio window, open the Connect to Server dialog box.
5. In the Connect to Server dialog box, type the following to connect to Microsoft SQL Server Management Studio:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1\INSTANCE3**
   - Authentication: **Windows Authentication**

6. On the **NYC-SQL1\INSTANCE3** instance, by using the Microsoft SQL Server Management Studio window, start the Configure Distribution Wizard to configure Distribution Agent with **\\NYC-SQL1\Repl_INSTANCE3** as the snapshot folder.
7. On the **NYC-SQL1\DEVELOPMENT** instance, in the Microsoft SQL Server Management Studio window, start the Configure Distribution Wizard to configure the Distribution Agent with **\\NYC-SQL1\Repl_DEVELOPMENT** as the snapshot folder.
8. On the **NYC-SQL1\INSTANCE3** instance, in the Microsoft SQL Server Management Studio window, start the Configure Distribution Wizard to configure the Distribution Agent with **\\NYC-SQL1\Repl_INSTANCE3** as the snapshot folder.

Task 5: Enable peer-to-peer replication.

1. Using the Microsoft SQL Server Management Studio window, in the NYC-SQL1 instance, open the **Publication Properties – Mod10_P2P Trans Pub** dialog box.
2. In the **Publication Properties – Mod10_P2P Trans Pub** dialog box, enable the **[Mod10_P2P]: Mod10_P2P Trans Pub** peer-to-peer replication.

Task 6: Configure peer-to-peer topology.

1. Using the Microsoft SQL Server Management Studio window, in the NYC-SQL1 instance, start the Configure Peer-to-Peer Topology Wizard.
2. Using the Configure Peer-to-Peer Topology Wizard, connect to the **NYC-SQL1\DEVELOPMENT** server, and configure the peer-to-peer topology for **[Mod10_P2P]: Mod10_P2P Trans Pub** with the following settings:
   - Add a new peer node, **Mod10_P2P,** with the **Peer Originator ID** as **3**.
   - Set **NYC-SQL1\Administrator** as the Process account, and **Pa$$w0rd** as the password for **Log Reader Agent**.
   - Set **NYC-SQL1\Administrator** as the Process account, and **Pa$$w0rd** as the password for **Distribution Agent**.

3. Using the Configure Peer-to-Peer Topology Wizard connect to the **NYC-SQL1\INSTANCE3** server, and configure peer-to-peer topology for **[Mod10_P2P]: Mod10_P2P Trans Pub** with the following settings:
   - Add a new peer node **Mod10_P2P** with the **Peer Originator ID** as **4**.
   - Set **NYC-SQL1\Administrator** as the Process account, and **Pa$$w0rd** as the password for **Log Reader Agent**.
   - Set **NYC-SQL1\Administrator** as the Process account, and **Pa$$w0rd** as the password for **Distribution Agent**.

Task 7: Verify the replication setup.

1. Using the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a script to generate sample data for verifying the replication setup in NYC-SQL1.

Field name	Record
Call_date	A date between 2004-01-01 and 2009-12-31
Caller	Peter
Duration	30
Product	A random number between one and three
cs_rep	A random number between one and five

2. Verify if the records created in the **dbo.CallDetails** table from the NYC-SQL1 instance are replicated in the NYC-SQL1\DEVELOPMENT instance.
3. Verify if the records created in the **dbo.CallDetails** table from the NYC-SQL1 instance are replicated in the NYC-SQL1\INSTANCE3 instance.

4. Using the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a script to generate sample data for verifying the replication setup in NYC-SQL1\DEVELOPMENT.

Field name	Record
Call_date	A date between 2000-01-01 and 2003-12-31
Caller	Peter
Duration	30
Product	A random number between one and three.
cs_rep	A random number between one and five.

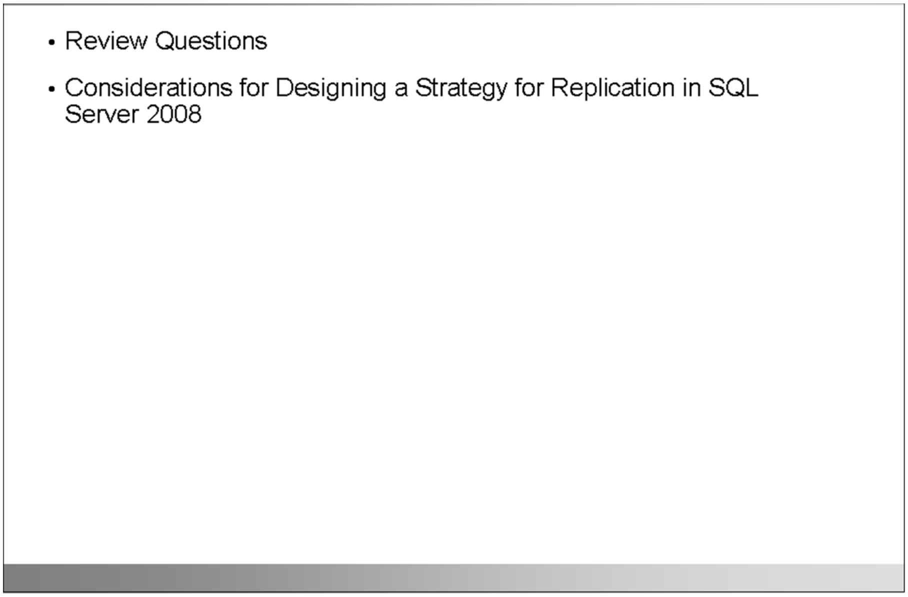
5. Verify if the records created in the **dbo.CallDetails** table from the **NYC-SQL1** instance are replicated in the NYC-SQL1 instance.

6. Verify if the records created in the **dbo.CallDetails** table from the NYC-SQL1 instance are replicated in the NYC-SQL1\INSTANCE3 instance.

Note: Browse to the **D:\Labfiles\Mod10** folder and select the **E3_T7.sql** file to execute the required T-SQL code.

Results: After completing this exercise, you should have created a test database, initialized a database schema, created a publication, configured the Distribution Agent, enabled peer-to-peer replication, configured peer-to-peer topology, and verified the replication setup.

Module Review and Takeaways



Review Questions

1. You are administrating a database for a CRM application to replicate a customer's information to SQL Server Express database installed on the laptop of each sales person. Which feature should you use to allow a sales person to get the latest information without using VPN?
2. You need to select a replication topology for the POS application. One of the key features of this POS application is the ability to see the inventory of other branches. You want to select the replication topology to replicate changes between each SQL Server installed in each branch of your organization. Which replication technology should you use?
3. You need to configure transactional replication to replicate stock information to each retail shop. But you only want to send the stock information to the shop in which the product is available for selling. Which feature of transactional replication will you use?

4. You are troubleshooting a failure in replication for an application. What is the first thing you should look at when the replication is not happening?

5. You have an Oracle database for storing customer information. Your customer service department is using an application, which supports only the Oracle database. You need to ensure that the latest customer information is available whenever the procurement department requires it. The procurement department is using an application that uses SQL Server. What should you do?

Considerations for Designing a Strategy for Replication in SQL Server 2008

When designing a strategy for replication in SQL Server 2008, consider the following:

- Use snapshot replication if you need to update the data at the scheduled time.
- Use transactional replication and merge replication if you need the transfer changes to other nodes in real-time.
- Use transactional replication if all Subscribers are getting a read-only copy of data.
- Use merge replication if each Subscriber has a different data set and you are consolidated the data into a single database.
- Use Web synchronization for a Subscriber to communicate with the Publisher without using VPN.
- Use the domain user account for the replication account for the Subscriber to access the snapshot folder for obtaining the snapshot file.
- Use P2P replication to reduce the convergence time for changes to reach all the servers.
- Use P2P replication to achieve high availability.

- Use Heterogeneous replication to obtain information from the non-SQL Publisher.
- Use an appropriate conflict resolver to resolve possible conflicts that may occur when synchronizing data between each server.

Note: The use of updating the Subscriber could be removed in the future version of SQL Server.

Module 11

Designing a High-Availability Solution for SQL Server 2008

Contents:

Module Overview

- Introduction to High Availability
- Implementing Log Shipping
- Implementing Database Mirroring
- Implementing Failover Clustering
- Designing a High-Availability Strategy

Organizations generally have inappropriate expectations regarding availability targets. They might demand high levels of availability because they may not be aware of the cost implications. Therefore, it is necessary to identify high-availability solutions that improve the availability of servers or databases. These solutions should also mask the effects of a hardware or software failure, minimize the perceived downtime for users, and make provisions for disaster recovery (DR) site setup and data backup. Some options for creating high availability for a server or database include failover clustering, database mirroring, log shipping, and replication.

In this module, you will learn how to implement the various high-availability tools such as log shipping, database mirroring, and failover clustering. You will also learn the guidelines to implement a high-availability solution.

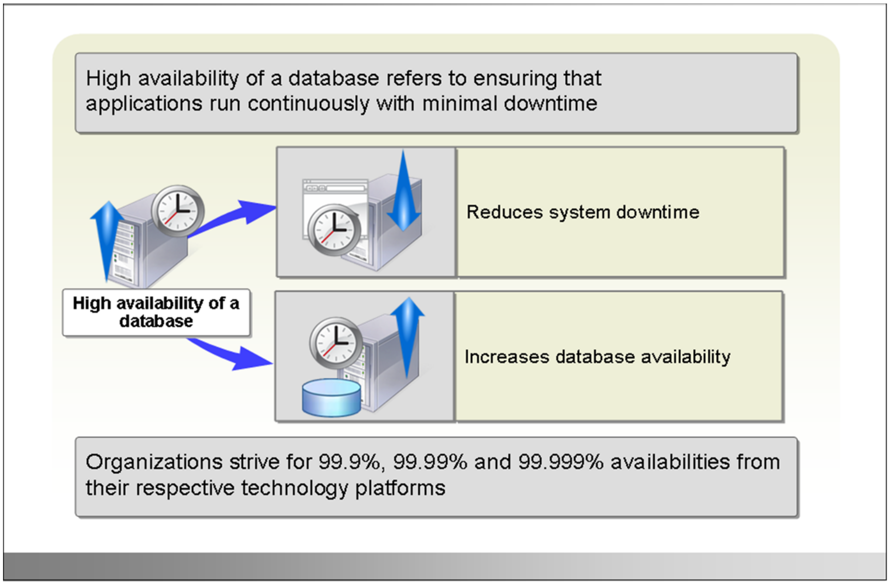
Lesson 1
Introduction to High Availability

- Need for High-Availability Solutions
- Business Scenarios for Using High-Availability Solutions
- High-Availability Solutions in SQL Server 2008

In a business enterprise, a high-availability solution helps to reduce the effects of a hardware or software failure. It helps to maintain the availability of applications so that the perceived downtime for users is minimized. You need to plan the recovery by providing a high-availability solution which is a mix of software and hardware planning, solution layout, and a set of good processes.

In this lesson, you will learn about the business needs for implementing a high-availability solution in an organization. You will also learn about the various high-availability solutions available in SQL Server™ 2008.

Need for High-Availability Solutions



Key Points

High availability of a database refers to ensuring that applications run continuously with minimal downtime. Business organizations use their information technology infrastructure to provide competitive advantage, increase productivity, and provide uninterrupted service to their customers. Reliance on server-based systems to accelerate business processes requires the servers to keep running continuously. Many database applications, especially enterprise business applications, require minimum system downtime. Therefore, you should consider maximizing the availability of applications while designing and configuring the technical infrastructure of your organization. Ensuring high availability of database guarantees that you can rely on the applications for continuous service.

A high availability solution:

- **Reduces system downtime**. System downtime impacts business cycles and customer satisfaction. System downtime may be unexpected or planned. You can plan system downtime for maintenance tasks, such as rebuilding an index, running an application, or upgrading a system. However, it is more challenging to protect your systems against unexpected downtimes, such as network failures, natural disasters, and human errors. Lower availability and higher downtime results in higher business costs because of reduced business activities.

- **Increases database availability**. Increased availability of a database ensures that you can continue critical business operations in case of disasters or system failure. SQL Server 2008 provides many high availability solutions, such as database mirroring, log shipping, replication and failover clustering, to achieve the appropriate level of availability.

Downtime Measurements at Different Availability Rates

You can measure availability as a percentage of downtime in a given year. The following table shows the accepted downtime for a particular percentage of availability.

Availability	Annual Downtime
99.9%	8 hours, 45 minutes
99.99%	53 minutes
99.999%	5 minutes

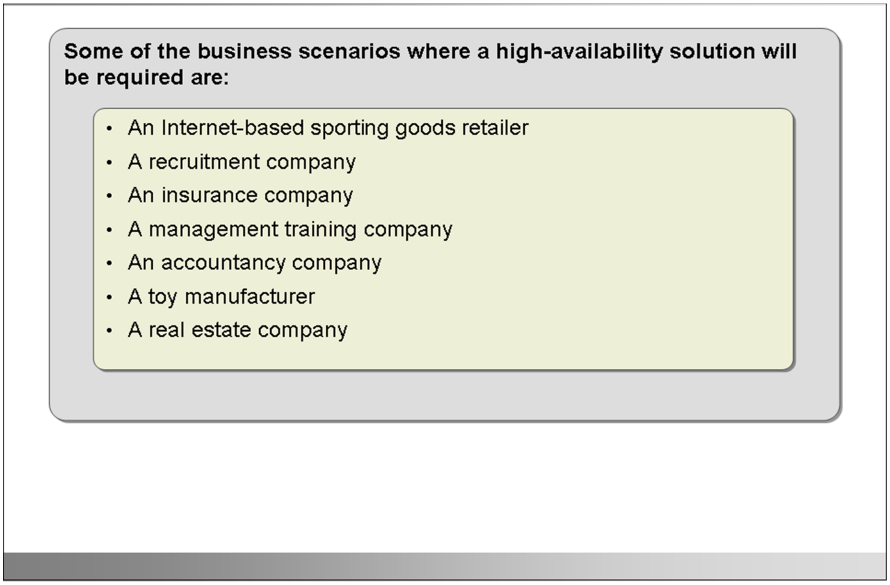
In the table, an application achieves 99.999% availability if it has a maximum downtime of five minutes in a year. Similarly, to achieve 99.99% availability, the application downtime must not exceed 53 minutes in a year.

Many organizations strive for 99.9%, 99.99% and 99.999% availabilities from their respective technology platforms. This is not unusual because many database applications are mission-critical and demand minimum downtime and quick recovery. SQL Server 2008 offers a wide range of technologies to minimize downtime and achieve the appropriate level of availability. Therefore, system maintenance and operational procedures required for mission-critical applications can occur with minimum downtime.

Source:

- download.microsoft.com/download/a/c/d/.../SQL2008HA.docx

- Business Scenarios for Using High-Availability Solutions



Key Points

Many businesses strive to achieve high availability. The appropriate solution for a particular scenario depends on a number of factors, like cost, manageability, and the solution that works best in one case might not be the most suitable solution for another case. For example, consider the following scenarios:

- An Internet-based sporting goods retailer wants to provide near-continuous availability to its customers. The Web servers are in a Web farm, but database failures cause errors resulting in the company losing customers. The company has an approved standard server and ideally wants to avoid major costs, infrastructure changes, or application re-writes.
- A recruitment company wants to reduce the user errors on its client database. The company would also like to have a warm standby server that could be brought online quickly, although instant failover is unnecessary.

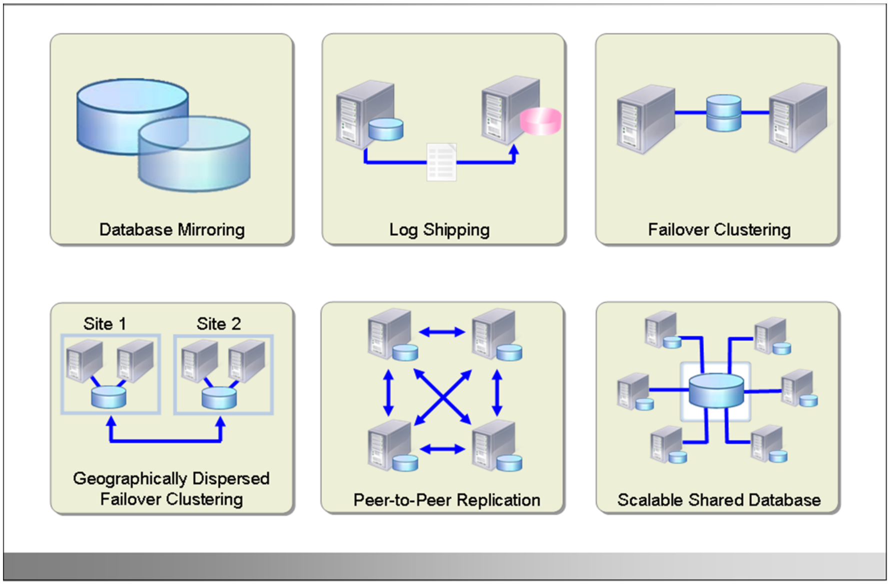
- An insurance company wants to provide near-continuous availability for its claims database without affecting the system performance. The documentation to support the claim, which is currently stored in the file system, must also be made available.
- A management training company has three sites with a sales team at each site. Each sales team enters records into the Orders database for its own site. To improve performance, each site requires access to all orders for the company, ideally on its local server. In the event of a failure, to provide availability, each site must be able to switch to the server at another site.
- An accountancy company wants to provide availability to its clients, even if there is wide-ranging failure, such as a natural disaster. Information does not need to be made instantly available, but it should be available within hours rather than days. The company would also like to retain a daily version of its databases for regulatory compliance. The copy data does not need to be instantly available, but must be available for a number of years.
- A toy manufacturer wants to run reports against its sales database. However, because the database is constantly in use, reports are often blocked by currently running transactions. The company has looked into reporting against uncommitted transactions, but this has resulted in incorrect results because some of the transactions did not complete.
- A real estate company wants to protect its system against user error by being able to roll records back to the values of the previous day. The company would also like to be able to run reports against the previous day's data that are unaffected by transactions taking place currently. Ideally, the company would like to implement a solution that has the minimum possible disk overhead.

With such a wide range of availability requirements, businesses need a database platform that provides flexible options for high-availability solutions.

Source:

- download.microsoft.com/download/a/c/d/.../SQL2008HA.docx

High-Availability Solutions in SQL Server 2008



Key Points

In SQL Server 2008, there are many technologies to increase the availability of the system. These solutions include database mirroring, log shipping, failover clustering, and scalable shared databases. Some of these solutions are available only in the Enterprise Edition of SQL Server 2008.

Database Mirroring

In database mirroring, there is another server that serves as the mirror server and as changes are applied to the principal database, they are automatically applied to the mirror.

You can use database mirroring in conjunction with replication to provide availability for the publication database. In database mirroring, there are two copies of a single database that typically reside on different computers. At any given time, only one copy of the database can service user's requests. This copy is known as the principal database. Updates made by clients to the principal database are applied to the other copy of the database, known as the mirror database. Mirroring involves applying the transaction log from every insert, update, or delete made on the principal database, onto the mirror database.

In the event of a failure on the principal server, client applications are automatically redirected to the mirror server without any changes to the application.

Log Shipping

The log shipping technology provides a warm standby server. In log shipping, there are two copies of a single database that typically reside on different computers. At any given time, only one copy of the database is available to clients. This copy is known as the primary database. Updates made by clients to the primary database are propagated by means of log shipping to the other copy of the database, known as the secondary database. Log shipping involves applying the transaction log from every insert, update, or delete made on the primary database, onto the secondary database.

While database mirroring can only have one mirror server, log shipping can have many secondary servers, which increases the level of protection. Log shipping can also take advantage of backup compression to reduce the size of the log files.

Failover Clustering

A failover cluster is a group of independent computers that work together to increase the availability of applications and services. The clustered servers (called nodes) are connected by physical cables and by software. If one of the cluster nodes fails, another node begins to provide service. This process is known as failover. Users experience a minimum of disruption in service.

Failover clustering provides high-availability support for an entire instance of SQL Server and just not the databases. A failover cluster is a combination of one or more nodes or servers, with two or more shared disks and a quorum.

The failover cluster feature is not available in Windows® Web Server 2008 or Windows Server® 2008 Standard.

Geographically-Dispersed Failover Clustering

It provides server-level redundancy on a certified Microsoft™ Geographically-Dispersed Cluster Services configuration with storage area network (SAN) replication and a virtual local area network (VLAN). If the site, the server node, or the disks fail, the complete redundancy of systems and disks enables the failover cluster to handle subsequent activities on another site. This configuration removes the risk of failure of a shared disk array, which would prevent a standard cluster configuration from working. Server nodes have to be on the same subnet and SQL Server 2008 does not support OR dependencies between IP addresses.

Peer-to-Peer Replication

It enables multiple databases to be replicated with each other. You can make changes to any database and you can apply it to the replication topology close to real time. You must design applications to connect to another node in the topology if their principal server is unavailable.

Peer-to-Peer replication uses a publish-subscribe model. This lets a primary server, referred to as the Publisher, distribute data to one or more secondary servers, or Subscribers. Replication enables real-time availability and scalability across these servers. It supports filtering to provide a subset of data to Subscribers, and allows for partitioned updates. Subscribers are online and are available for reporting or other functions, without query recovery. SQL Server offers three types of replication: snapshot, transactional, and merge. Transactional replication provides the lowest latency and is usually used for high availability.

Scalable Shared Database

It is a feature to scale out a read-only database built exclusively for reporting. The reporting database must reside on a set of dedicated, read-only volumes whose primary purpose is hosting the database. By using commodity hardware for servers and volumes, you can scale out a reporting database that provides the same view of the reporting data on multiple reporting servers.

Source:

- http://msdn.microsoft.com/en-us/library/ms151799.aspx
- http://msdn.microsoft.com/en-us/library/ms151224.aspx
- download.microsoft.com/download/a/c/d/.../SQL2008HA.docx
- http://technet.microsoft.com/en-us/library/cc731844(WS.10).aspx
- http://msdn.microsoft.com/en-us/library/ms190202.aspx

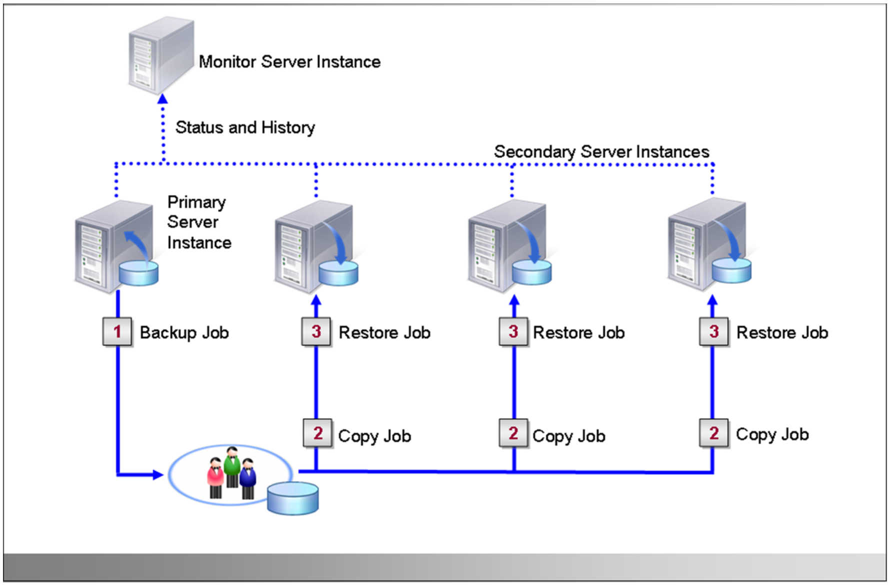
Lesson 2
Implementing Log Shipping

- Architecture of Log Shipping
- Log Shipping Jobs
- Process of Failover by Using Log Shipping
- How To Set Up Log Shipping
- Methods of Monitoring Log Shipping

Log shipping is one of the solutions for high-availability for business critical application. Log shipping is an automated process for backing up, restoring, copying the transaction logs, and synchronizing the database for distributed database server applications.

In this lesson, you will learn about the architecture of log shipping and about the log shipping jobs. You will also learn how to set up log shipping and the methods of monitoring log shipping.

Architecture of Log Shipping



Key Points

The following steps are performed to configure log shipping:

1. The primary server instance runs the backup job to back up the transaction log on the primary database. This server instance then places the log backup into a primary log-backup file, which it sends to the backup folder.

2. Each of the three secondary server instances runs its own copy job to copy the primary log-backup file to its own local destination folder.

3. Each secondary server instance runs its own restore job to restore the log backup from the local destination folder onto the local secondary database.

The primary and secondary server instances send their own history and status to the monitor server instance. The log can be shipped to multiple secondary server instances. In such cases, you need to copy the transaction log and restore the backup of the log on the secondary server multiple times, once for each secondary server. A log shipping configuration does not automatically fail over from the primary server to the secondary server. If the primary database becomes unavailable, any of the secondary databases should be brought online manually.

Primary Server and Database

The primary server in a log shipping configuration is the instance of the SQL Server Database Engine that is your production server. The primary database is the database on the primary server that you want to back up to another server. All administration of the log shipping configuration through SSMS is performed from the primary database. The primary database must use the full or bulk-logged recovery model. Switching the database to simple recovery will cause log shipping to stop functioning.

Secondary Server and Database

The secondary server in a log shipping configuration is the server where you want to keep a warm standby copy of your primary database. A secondary server can contain backup copies of databases from several different primary servers. For example, a department could have five servers, each running a mission-critical database system. Rather than having five separate secondary servers, a single secondary server could be used. The backups from the five primary systems could be loaded onto the single backup system, reducing the number of resources required, and the cost. It is unlikely that more than one primary system would fail at the same time. Additionally, to cover the remote chance that more than one primary system becomes unavailable at the same time, the secondary server could be of higher specification than the primary servers.

You need to initialize the secondary database by restoring a full backup of the primary database. You can complete the restore by using either the NORECOVERY option or the STANDBY option. You can do this manually or through SSMS.

Monitor Server

The optional monitor server tracks all details of log shipping, including:

- When the transaction log on the primary database was last backed up.

- When the secondary servers last copied and restored the back up files.
- Information about any backup failure alerts.

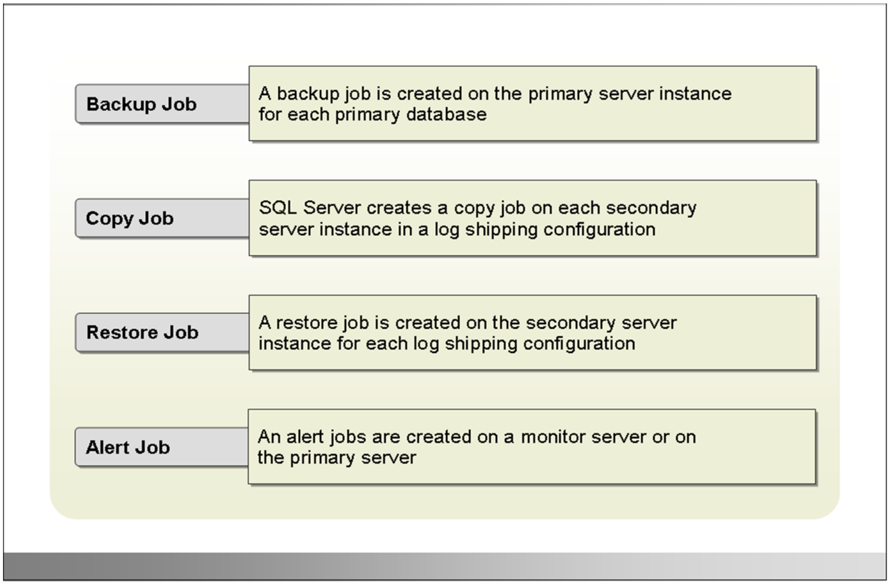
The monitor server should be a server that is separate from the primary or secondary servers, to avoid losing critical information and disrupting monitoring if the primary or secondary server is lost. A single monitor server can monitor multiple log shipping configurations and share a single alert job.

Question: What is the technology behind log shipping?

Source:

- http://msdn.microsoft.com/en-us/library/ms187103(SQL.105).aspx
- http://msdn.microsoft.com/en-us/library/ms187103(SQL.90).aspx

Log Shipping Jobs



Key Points

Log shipping allows you to automatically send transaction log backups from a primary database on a primary server instance to one or more secondary databases on separate secondary server instances. The transaction log backups are applied to each of the secondary databases individually. Optional third server instances, known as the monitor server, records the history and status of backup and restore operations, and optionally, raises alerts if these operations fail to occur as scheduled.

Log shipping involves four jobs, which are handled by dedicated SQL Server Agent jobs. These jobs include the backup job, the copy job, the restore job, and the alert job.

The user controls how frequently log backups are taken, copied to each secondary server, and applied to the secondary database. To reduce the work required to bring a secondary server online, for example, after the production system fails, you can copy and restore each transaction log backup soon after it is created. Alternatively, perhaps on a second secondary server, you can delay applying transaction log backups to the secondary database. This delay provides an interval during which you can notice and respond to a failure on the primary server, such as accidental deletion of critical data.

The four types of jobs are:

- **Backup job**. A backup job is created on the primary server instance for each primary database. It performs the backup operation, logs history to the local server and the monitor server, and deletes old backup files and history information. By default, this job will run every 15 minutes, but the interval is customizable.
  - When log shipping is enabled, the SQL Server Agent job category, "Log Shipping Backup," is created on the primary server instance. SQL Server 2008 Enterprise supports backup compression. When creating a log shipping configuration, you can control the backup compression behavior of log backups.
- **Copy job**. SQL Server creates a copy job on each secondary server instance in a log shipping configuration. This job copies the backup files from the primary server to a configurable destination on the secondary server and logs history on the secondary server and on the monitor server. The copy job schedule, which is customizable, should approximate the backup schedule.

When you enable log shipping, the SQL Server Agent job category, "Log Shipping Copy," is created on the secondary server instance.

- **Restore job**. A restore job is created on the secondary server instance for each log shipping configuration. This job restores the copied backup files to the secondary databases. It logs history on the local server and the monitor server, and deletes old files and old history information. The SQL Server job category, "Log Shipping Restore," is created on the secondary server instance when you enable log shipping.

On a given secondary server instance, the restore job can be scheduled as frequently as the copy job, or the restore job can be delayed. Scheduling these jobs with the same frequency keeps the secondary database as closely aligned with the primary database as possible to create a warm standby database.

In contrast, delaying restore jobs, perhaps by several hours, can be useful in the event of a serious user error, such as a dropped table or inappropriately deleted table row. If the time of the error is known, you can move that secondary database forward to a time soon before the error. Then, you can export the lost data and import it back into the primary database.

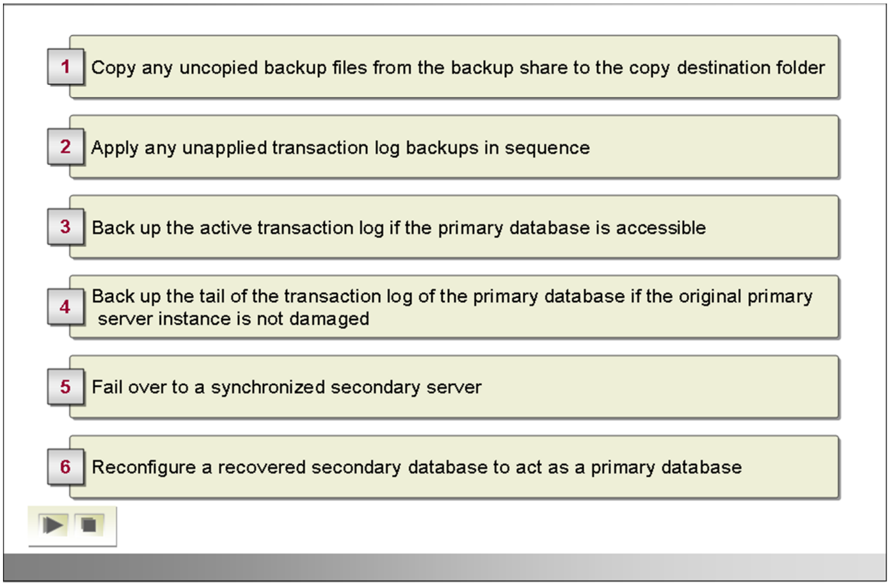
- **Alert job**. If a monitor server is used, an alert job is created on the monitor server instance. This alert job is shared by the primary and secondary databases of all log shipping configurations by using the monitor server instance. Any change to the alert job, such as rescheduling, disabling, or enabling the job, affects all databases using that monitor server. This job raises alerts for primary and secondary databases when backup and restore operations have not completed successfully within specified thresholds. You must configure these alerts to have an operator receive notification of the log shipping failure. The SQL Server Agent job category, "Log Shipping Alert," is created on the monitor server instance when log shipping is enabled.

If a monitor server is not used, alert jobs are created locally on the primary server instance and on each secondary server instance. The alert job on the primary server instance raises errors when backup operations have not completed successfully within a specified threshold. The alert job on the secondary server instance raises errors when local copy and restore operations have not completed successfully within a specified threshold.

Source:

- http://msdn.microsoft.com/en-us/library/ms187103.aspx

Process of Failover by Using Log Shipping



Key Points

Typically, the primary and secondary databases are unsynchronized, because the primary database continues to be updated after its latest backup job. In some cases, recent transaction log backups are not copied to the secondary server instances, or some copied log backups are not applied to the secondary database. You must begin by synchronizing all the secondary databases with the primary database, if possible.

Use the following process to fail over to a secondary database:

1. Copy any uncopied backup files from the backup share to the copy destination folder of each secondary server.
2. Apply any unapplied transaction log backups in sequence to each secondary database.
3. If the primary database is accessible, back up the active transaction log and apply the log backup to the secondary databases.

4. If the original primary server instance is not damaged, back up the tail of the transaction log of the primary database by using the WITH NORECOVERY option. This leaves the database in the restoring state and therefore unavailable to users. Eventually, you will be able to roll this database forward by applying transaction log backups from the replacement primary database.
5. After the secondary servers are synchronized, you can fail over to whichever one you prefer by recovering its secondary database and redirecting clients to that server instance. Recovering puts the database into a consistent state and brings it online.

Note: When you make a secondary database available, you should ensure that its metadata is consistent with the metadata of the original primary database.

6. After you have recovered a secondary database, you can reconfigure it to act as a primary database for other secondary databases.

Source:

- http://msdn.microsoft.com/en-us/library/ms191233.aspx

Demonstration: How To Set Up Log Shipping

In this demonstration, you will see how to:

Set up log shipping between two server instances

Key Points

The steps to set up log shipping between two servers are:

1. On the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. To log on to SQL Server Management Studio, in the Connect to Server dialog box, type the following:
    1. Server type: **Database Engine**
    2. Server name: **NYC-SQL1**
    3. Authentication: **Windows Authentication**
3. Click **Connect**.
4. In Windows Explorer, browse to the **D:\Demofiles** folder.
5. On the **File** menu of the Demofiles window, point to **New**, click **Folder**, type **LogShipping**, and then press ENTER.

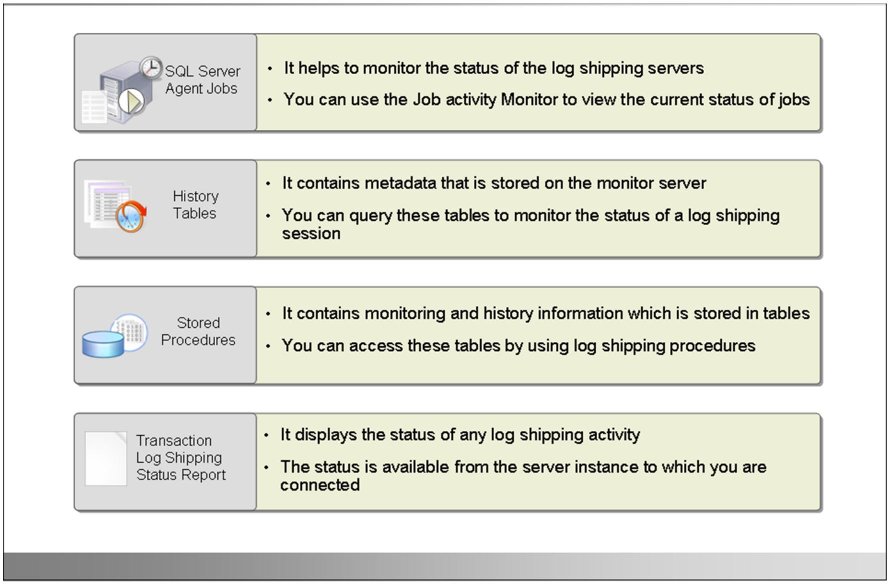
6. In the Demofiles window, right-click **LogShipping**, and then click **Share**. The **File Sharing** dialog box appears.
7. In the **File Sharing** dialog box, click **Share**. The Network discovery and file sharing message box appears.
8. In the **Network discovery and file sharing** message box, click **Yes, turn on network discovery and file sharing for all public networks**.
9. When the folder is shared, in the **File Sharing** dialog box, click **Done**.
10. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **Databases**, right-click **QuantamCorp**, and then click **Properties**. The **Database Properties - QuantamCorp** dialog box appears.
11. In the **Database Properties - QuantamCorp** dialog box, in the console pane, under **Select a page**, click **Options**.
12. On the **Options** page, in the work pane, ensure that the **Recovery model** is set to **Full**.
13. In the console pane, under **Select a page**, click **Transaction Log Shipping**.
14. On the **Transaction Log Shipping** page, select the **Enable this as a primary database in a log shipping configuration** check box.
15. Under **Transaction log backups**, click **Backup Settings**. The **Transaction Log Backup Settings** dialog box appears.
16. In the **Transaction Log Backup Settings** dialog box, in the **Network path to backup folder** box, type \\NYC-SQL1\LogShipping, and then click **OK**.
17. In the **Database Properties - QuantamCorp** dialog box, under **Secondary server instances and databases**, click **Add**. The **Secondary Database Settings** dialog box appears.
18. In the **Secondary Database Settings** dialog box, click the **Connect** button, besides the **Secondary server instance** box. The **Connect to Server** dialog box appears.

19. In the **Connect to Server** dialog box, in the **Server name** box, type **NYC-SQL1\DEVELOPMENT** the server name to which you want to ship the log, and then click **Connect**.
20. In the **Secondary Database Settings** dialog box, on the **Initialize Secondary Database** tab, select the **Yes, generate a full backup of the primary database and restore it into the secondary database (and create the secondary database if it doesn't exist)** check box.
21. On the **Copy Files** tab, in the **Destination folder for copied files** box, type **\\NYC-SQL1\ReplData**, and then click **OK**.
22. In the **Database Properties - QuantamCorp** dialog box, click **OK** to begin the configuration process. The **Save Log Shipping Configuration** dialog box appears.
23. In the **Save Log Shipping Configuration** dialog box, click **Close**.

Question: What should you configure on the secondary instance server before setting up log shipping?

Question: Why should SQL Server Agent be running when log shipping is configured?

Methods of Monitoring Log Shipping



Key Points

You can monitor log shipping by using SQL Server Agent jobs, history tables, stored procedures, and in SSMS.

SQL Server Agent Jobs

Log shipping relies on SQL Server Agent jobs to perform most operations. Thus keeping SQL Server Agent running to ensure log shipping related jobs are working properly becomes an essential part for monitoring log shipping. You can use the Job Activity Monitor in SQL Server Agent to view the current state of jobs. If the SQL Server Agent service unexpectedly terminates, you can refer to the sysjobactivity table to see which jobs were being run when the service terminated.

To view the job activity:

1. In the Microsoft SQL Server Management Studio window, connect to an instance of the SQL Server Database Engine, and then expand that instance.
2. In the Object Explorer pane, under the SQL Server instance, expand **SQL Server Agent**.
3. Right-click **Job Activity Monitor** and click **View Job Activity**. The Job Activity Monitor window appears.
4. In the **Job Activity Monitor** window, you can view details about each job that is defined for this server.
5. In the Work pane, right-click a job to start it, stop it, enable, or disable it, refresh its status as displayed in the Job Activity Monitor, delete it, or view its history or properties.
6. To start, stop, enable or disable, or refresh multiple jobs, select multiple rows in the Job Activity Monitor, and right-click your selection.
7. To update the Job Activity Monitor, click the **Refresh** button.
8. To view fewer rows, click **Filter**, and then specify the filter parameters.

After you have configured log shipping, you can monitor information about the status of all the log shipping servers. The history and the status of log shipping operations are always saved locally by the log shipping jobs. The history and status of the backup operation are stored at the primary server, and the history and status of the copy and restore operations are stored at the secondary server. If you have implemented a remote monitor server, this information is also stored on the monitor server.

You can configure alerts that will run if log shipping operations fail to occur as scheduled. Errors are raised by an alert job that watches the status of the backup and restore operations. You can define alerts that notify an operator when these errors are raised. If a monitor server is configured, one alert job runs on the monitor server that raises errors for all operations in the log shipping configuration. If a monitor server is not specified, an alert job runs on the primary server instance, which monitors the backup operation. If a monitor server is not specified, an alert job also runs on each secondary server instance to monitor the local copy and restore operations.

History Tables

The monitoring history tables contain metadata that is stored on the monitor server. A copy of information specific to a given primary or secondary server is also stored locally.

You can query these tables to monitor the status of a log shipping session. For example, to learn the status of log shipping, check the status and history of the backup job, copy job, and restore job.

The following table shows the history tables you can view specific log shipping history and error details by querying them.

Table	Description
log_shipping_monitor_alert	This table stores the alert job ID.
log_shipping_monitor_error_detail	This table stores error details for log shipping jobs. You can query this table and see the errors for an agent session. Optionally, you can sort the errors by the date and time at which each was logged. Each error is logged as a sequence of exceptions, and multiple errors (sequences) can be logged per agent session.
log_shipping_monitor_history_detail	This table contains history details for log shipping agents. You can query this table to see the history details for an agent session.
log_shipping_monitor_primary	This table stores one monitor record for the primary database in each log shipping configuration, including information about the last backup file and last restored file that is useful for monitoring.

Table	Description
log_shipping_monitor_secondary	This table stores one monitor record for each secondary database, including information about the last backup file and last restored file that is useful for monitoring.

Stored Procedures

Monitoring and history information is stored in tables in the msdb database, which can be accessed by using log shipping stored procedures.

The following table shows the stored procedures that you can run on the servers.

Stored procedure	Description	Run this procedure on
sp_help_log_shipping_monitor_primary	It returns monitor records for the specified primary database from the log_shipping_monitor_primary table.	Monitor server or primary server
sp_help_log_shipping_monitor_secondary	It returns monitor records for the specified secondary database from the log_shipping_monitor_secondary table.	Monitor server or secondary server
sp_help_log_shipping_alert_job	It returns the job ID of the alert job.	Monitor server, or primary or secondary server, if no monitor is defined
sp_help_log_shipping_primary_database	It retrieves primary database settings and displays the values from	Primary server

Stored procedure	Description	Run this procedureon
	the log_shipping_primary_databases and log_shipping_monitor_primary tables.	
sp_help_log_shipping_primary_secondary	It retrieves secondary database names for a primary database.	Primary server
sp_help_log_shipping_secondary_database	It retrieves secondary-database settings from the log_shipping_secondary, log_shipping_secondary_databases, and log_shipping_monitor_secondary tables.	Secondary server
sp_help_log_shipping_secondary_primary (Transact-SQL)	This stored procedure retrieves the settings for a given primary database on the secondary server.	Secondary server

Transaction Log Shipping Status Report

You can view the Transaction Log Shipping Status report in SSMS by running a status report at a monitor server, primary server, or secondary server. You can obtain complete information about the log shipping configuration by viewing the report at the monitor server instance.

The report displays the status of any log shipping activity that is available from the server instance to which you are connected. If that server instance is involved in multiple configurations in different roles (such as serving as a monitor for one database and a secondary for another database), the displayed results contain the information of every configuration from the perspective of each role. If the stored procedure can connect to the monitor server instance for a given log shipping configuration, the report displays additional status for that configuration.

The following table lists the information that you can view for each role performed by the current server instance.

Role	Information displayed
Monitor	The status report displays the name and status of every primary server and secondary server that uses this server instance as its monitor server.
Primary	The status report displays the status and name of the current server instance (as the primary server), along with the primary database name for each primary database. The report displays the status of the backup job. The report also contains a row for each of the corresponding secondary servers. If the configuration uses a monitor server and the stored procedure can connect to the monitor, these rows display the copy status and restore status for the most recent log backup.
Secondary	The status report displays the status and name of the current server instance (as the secondary server), along with the secondary database name for each secondary database. The report displays the status of the copy and restore jobs at the secondary server. The report also contains a row for the corresponding primary server. If the configuration uses a monitor server and the stored procedure can connect to the monitor, this row displays the status of the most recent log backup.

The information displayed depends on whether the server instance is a monitor server, primary server, or secondary server. If information is not available, the corresponding cells are grayed out.

The report calls the stored procedure, sp_help_log_shipping_monitor, to get the data.

Question: If log shipping is not functioning as expected, what should you check first?

Question: What should you use to check the log shipping activities?

Source:

- http://msdn.microsoft.com/en-us/library/ms187449.aspx
- http://msdn.microsoft.com/en-us/library/ms190224.aspx
- http://technet.microsoft.com/en-us/library/ms181149.aspx

Lesson 3
Implementing Database Mirroring

- What Is Database Mirroring?
- Options to Handle Principal Server Failure
- Operating Modes of Database Mirroring
- Process of Implementing Database Mirroring
- How To Set Up Database Mirroring in a High-Availability Scenario

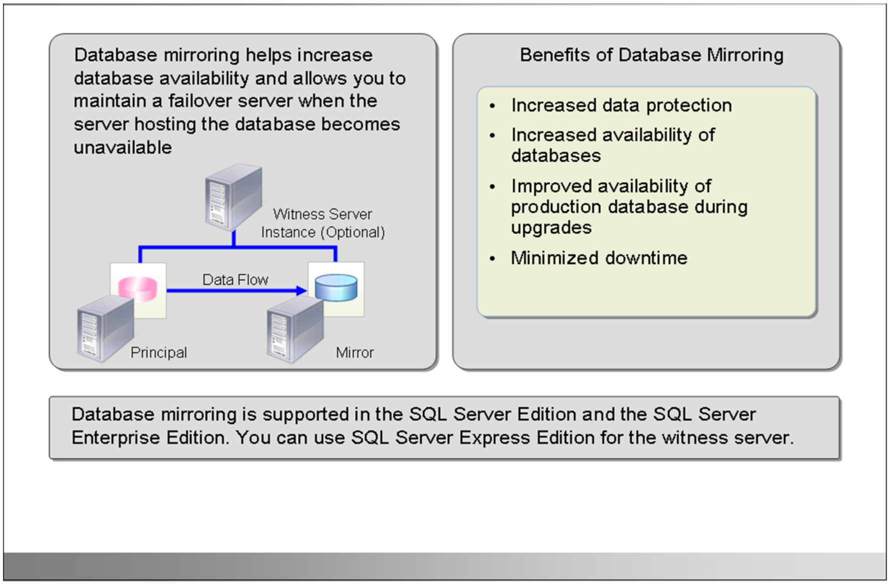
Database mirroring is a replication feature that helps to provide businesses of all sizes with flexible high-availability databases without requiring specialized or dedicated storage equipments. Database mirroring provides a hot standby server that supports rapid failover with no loss of data from committed transactions. During a typical mirroring session, after a production server fails, client applications can recover quickly by reconnecting to the standby server.

In this lesson, you will learn about database mirroring, synchronous and asynchronous database mirroring, and the process for implementing database mirroring.

Source:

- http://msdn.microsoft.com/en-us/library/bb934127.aspx

What Is Database Mirroring?



Key Points

Database mirroring increases data availability by allowing you to maintain a failover server, when the server that is hosting the database becomes unavailable. You can implement mirroring on each database and with databases that use the full recovery model.

Database Mirroring Session

Database mirroring maintains two copies of a single database that must reside on different server instances of the SQL Server Database Engine. These server instances reside on computers in different locations. Based on the configuration and state of the mirroring session, database mirroring comprises of two mandatory server instances and a third, optional server instance. The server instances include:

- **Principal server**. This server role serves as the database to clients.

- **Mirror server**. This server role is available as a hot or warm standby server.
- **Witness Server**. This server role is optional. This instance helps to implement automatic failure detection and failover.

In a database mirroring session, the principal and mirror servers communicate and cooperate as partners. These two partners perform complementary roles in the session—the principal role and the mirror role. Each partner is described as owning its current role. The partner that owns the principal role is known as the principal server, and its copy of the database is the current principal database. The partner that owns the mirror role is known as the mirror server, and its copy of the database is the current mirror database. When database mirroring is deployed in a production environment, the production database is the principal database.

Database mirroring involves redoing all insert, update, and delete operations that occur on the principal database, onto the mirror database as quickly as possible. Redoing is accomplished by sending a stream of active transaction log records to the mirror server, which applies log records to the mirror database, in sequence, as quickly as possible. Database mirroring works at the level of the physical log record. In SQL Server 2008, the principal server compresses the stream of transaction log records before sending it to the mirror server. This log compression occurs in all mirroring sessions.

Benefits of Database Mirroring

- Database mirroring offers the following benefits:
- **Increases data protection**. Database mirroring provides complete or almost complete redundancy of the data, depending on whether the operating mode is high-safety or high-performance. A database mirroring partner running on SQL Server 2008 Enterprise automatically tries to resolve certain types of errors that prevent reading a data page. The partner that is unable to read a page requests a fresh copy from the other partner. If this request succeeds, the unreadable page is replaced by the copy, thereby resolving the error.
- **Increases availability of a database**. In the event of a disaster, in high-safety mode with automatic failover, you can bring the standby copy of the database online without data loss.

- **Improves the availability of the production database during upgrades**. You can have the passive server continue to serve user request while upgrading the production or active server.
- **Helps to minimize downtime**. To minimize downtime for a mirrored database, you can sequentially upgrade the instances of SQL Server that are participating in a database mirroring session. This will incur the downtime of only a single failover. This form of upgrade is called a rolling upgrade.

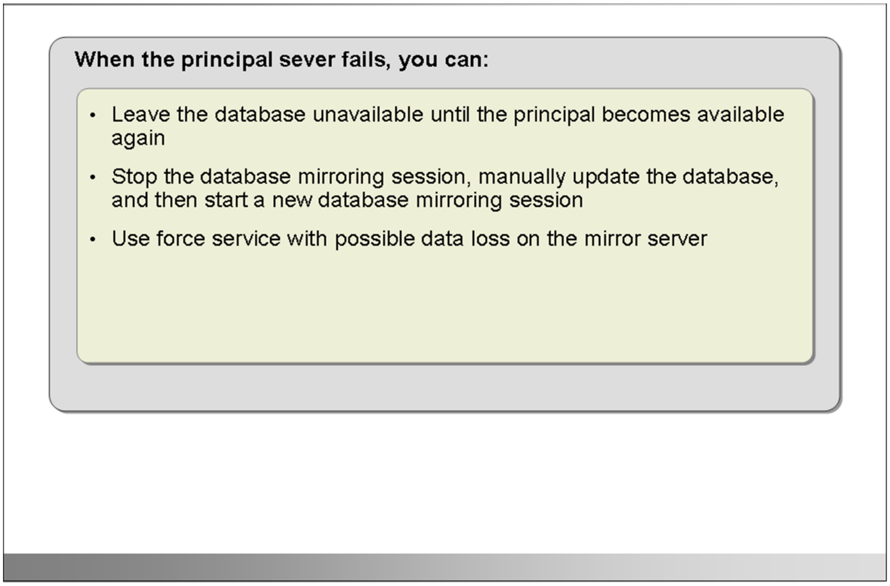
Database Mirroring Support
SQL Server Standard Edition and SQL Server Enterprise Editions support database mirroring partners and witness servers. However, the partners must use the same edition. Only SQL Server Enterprise Edition supports asynchronous database mirroring (high-performance mode). SQL Server Workgroup Edition and SQL Server Express Edition support witness servers.

Question: If you want the data to get committed to both active and passive server at the same time, which operating mode would you use in database mirroring?

Source:

- http://msdn.microsoft.com/en-us/library/ms189852.aspx
- http://msdn.microsoft.com/en-us/library/ms190202.aspx

Options to Handle Principal Server Failure



Key Points

When the principal sever fails, you can:

- **Leave the database unavailable until the principal becomes available again**.
  If the principal database and its transaction log are intact, this choice preserves all the committed transactions at the expense of availability.
- **Stop the database mirroring session, manually update the database, and then start a new database mirroring session**. If the principal database is lost but the principal server is still running, immediately attempt to back up the tail of the log on the principal database. If the tail-log backup succeeds, removing mirroring may be your best alternative. After removing mirroring, you can restore the log onto the former mirror database, which preserves all the data.
- **Use force service with possible data loss on the mirror**

server. Forced service is strictly a disaster recovery method and should be used sparingly. Forcing service is possible only if the principal server is down, the session is asynchronous, and either the session does not have any witness or the witness is connected to the mirror server. Forcing service causes the mirror server to assume the role of principal and serve its copy of the database for clients. When service is forced, whatever transaction logs the principal has not yet sent to the mirror server are lost. Therefore, you should limit forced service to situations where possible data loss is acceptable and immediate database availability is critical.

Source:

- http://msdn.microsoft.com/en-us/library/ms187110.aspx

Operating Modes of Database Mirroring

The operating modes of database mirroring are:

- Asynchronous. In this mode, the transactions are committed without waiting for the mirror server to write the log to disk.
- Synchronous. In this mode, a transaction is committed on both partners, but at the cost of increased transaction latency.

Operating Mode	Transaction Safety	Witness Server Support	Automatic Failover Support
Asynchronous (High-Performance Mode)	OFF	No	No
Synchronous (High-Safety Mode)	FULL	Yes	Yes

Key Points

A database mirroring session runs in either asynchronous or synchronous mode. Asynchronous database mirroring is referred to as a high-performance mode and synchronous database mirroring is referred to as a high-safety mode.

Under asynchronous operation, the transactions are committed without waiting for the mirror server to write the log to disk; thereby maximizing the performance. Under the synchronous operation, a transaction is committed on both partners, but at the cost of increased transaction latency.

Whether an operating mode is asynchronous or synchronous depends on the transaction safety setting. If you exclusively use SSMS to configure database mirroring, transaction safety settings are configured automatically when you select the operation mode.

If you use T-SQL to configure database mirroring, you must set the transaction safety. Transaction safety is controlled by the SAFETY property of the ALTER DATABASE statement. On a database that is being mirrored, SAFETY is either FULL or OFF.

Asynchronous Database Mirroring (High-Performance Mode)

When transaction safety is set to OFF, the database mirroring session operates asynchronously. Asynchronous operation supports only the high-performance mode. This mode enhances performance at the expense of high availability. The high-performance mode uses just the principal server and the mirror server. Problems on the mirror server never impact the principal server. On the loss of the principal server, the mirror database is marked DISCONNECTED, but is available as a warm standby.

The mirror server attempts to keep up with the log records sent by the principal server. However, the mirror database might lag somewhat behind the principal database. However, the gap between the databases is small. It can become substantial if the principal server is under a heavy work load or the system of the mirror server is over loaded.

You can use the high-performance mode in a disaster-recovery scenario where:

- The principal and mirror servers are separated by a significant distance.
- Small errors might occur, which might impact the performance of the principal server.

Impact of a Witness on the High-Performance Mode

If you use T-SQL to configure the high-performance mode, whenever the SAFETY property is set to OFF, you should set the WITNESS to OFF. A witness can coexist with the high-performance mode, but the witness does not provide any benefit and introduces a risk.

If the witness is disconnected from the session when either partner goes down, the database becomes unavailable. This is because even though the high-performance mode does not require a witness, if one is set, the session requires a quorum consisting of two or more server instances. If the session losses quorum, it cannot serve the database.

When a witness is set in a high-performance mode session, the enforcement of quorum means that:

- If the mirror server is lost, the principal server must be connected to the witness. Otherwise, the principal server takes its database offline until either the witness or mirror server rejoins the session.
- If the principal server is lost, forcing service to the mirror server requires that the mirror server be connected to the witness.

Synchronous Database Mirroring (High-Safety Mode)

When transaction safety is set to FULL, the database mirroring session runs in a high-safety mode and operates synchronously after an initial synchronizing phase. To achieve synchronous operation for a session, the mirror server must synchronize the mirror database with the principal database. When the session begins, the principal server begins sending its active log to the mirror server. The mirror server writes all the incoming log records to disk as quickly as possible. As soon as all the received log records have been written to disk, the databases are synchronized. As long as the partners remain in communication, the databases remain synchronized.

After synchronization finishes, every transaction committed on the principal database is also committed on the mirror server. This guarantees protection of the data. This is achieved by waiting to commit a transaction on the principal database, until the principal server receives a message from the mirror server stating that it has stored the transaction's log to disk.

The time required for synchronization depends essentially on how far the mirror database was behind the principal database at the start of the session, the work load on the principal database, and the speed of the mirror system. After a session is synchronized, the hardened log that is yet to be redone on the mirror database remains in the redo queue.

As soon as the mirror database becomes synchronized, the state of both the copies of the database changes to SYNCHRONIZED.

You can maintain the synchronous operation in the following manner:

- On receiving a transaction from a client, the principal server writes the log for that transaction to the transaction log.
- The principal server writes the transaction to the database, and concurrently, sends the log record to the mirror server. The principal server waits for an acknowledgement from the mirror server before confirming either of the following to the client—a transaction commit or a rollback.
- The mirror server hardens the log to disk and returns an acknowledgement to the principal server.
- On receiving the acknowledgement from the mirror server, the principal server sends a confirmation message to the client.
- All the committed transactions are guaranteed to be written to disk on the mirror server.

High-safety Mode Without Automatic Failover

The configuration of high-safety mode without automatic failover consists of only the two partners. When the partners are connected and the database is already synchronized, manual failover is supported. If the mirror server instance goes down, the principal server instance is unaffected and runs exposed. If the principal server is lost, the mirror is suspended, but service can be forced to the mirror server with possible data loss.

High-Safety Mode with Automatic Failover

Automatic failover provides high availability by ensuring that the database is still served after the loss of one server. Automatic failover requires that the session possess a third server instance, the *witness*, which ideally resides on a third computer.

Unlike the two partners, the witness does not serve the database. The witness simply supports automatic failover by verifying whether the principal server is up and functioning. The mirror server initiates automatic failover only if the mirror and the witness remain connected to each other after both have been disconnected from the principal server.

When a witness is set, the session requires *quorum*—a relationship between at least two server instances that allows the database to be made available.

Automatic failover requires the following conditions:

- The database is already synchronized.
- The failure occurs while all three server instances are connected, and the witness and mirror server remain connected.
- The loss of a partner has the following effect:
- If the principal server becomes unavailable under the above conditions, automatic failover occurs. The mirror server switches to the role of principal, and it offers its database as the principal database.
- If the principal server becomes unavailable when those conditions are not met, forcing service (with possible data loss) might be possible.
- If only the mirror server becomes unavailable, the principal and witness continue.
- If the session loses its witness, the quorum requires both partners. If either partner loses the quorum, both partners lose the quorum, and the database becomes unavailable until the quorum is re-established. This quorum requirement ensures that in the absence of a witness, the database never runs *exposed*, that is, without being mirrored.

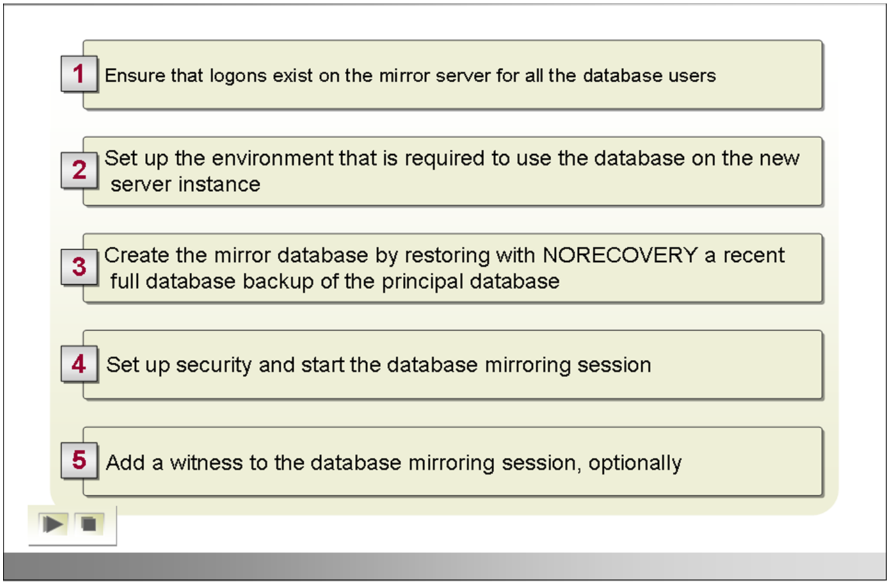
Question: What is the consideration for using synchronous database mirroring with the transaction safety mode?

Question: What is the requirement for using automatic failover?

Source:

- http://msdn.microsoft.com/en-us/library/ms189852.aspx
- http://msdn.microsoft.com/en-us/library/ms187110.aspx
- http://msdn.microsoft.com/en-us/library/ms179344.aspx

Process of Implementing Database Mirroring



Key Points

While implementing database mirroring for standby and high-availability scenarios, perform the following steps:

1. **Ensure that you have logons existing on the mirror server for all the database users**. For two server instances to communicate in a database mirroring session, the logon account of each instance requires access to the other instance. Also, each logon account requires connect permission to the database mirroring endpoint of the other instance.

2. **Set up the environment that is required to use the database on the new server instance**. Before making a database available on the other server instance, you must set up the environment that is required to use the database on the new server instance.

3. **Create the mirror database by restoring with NORECOVERY a recent full database backup of the principal database**.

Ensure that the principal database was already using the full recovery model when the backup was taken. The mirror database must have the same name as the principal database, and they cannot be renamed during a database mirroring session. You must restore all the log backups done since the full database backup. Before you can start mirroring on the partners, you should create a current log backup on the original database and restore it to the future mirror database.

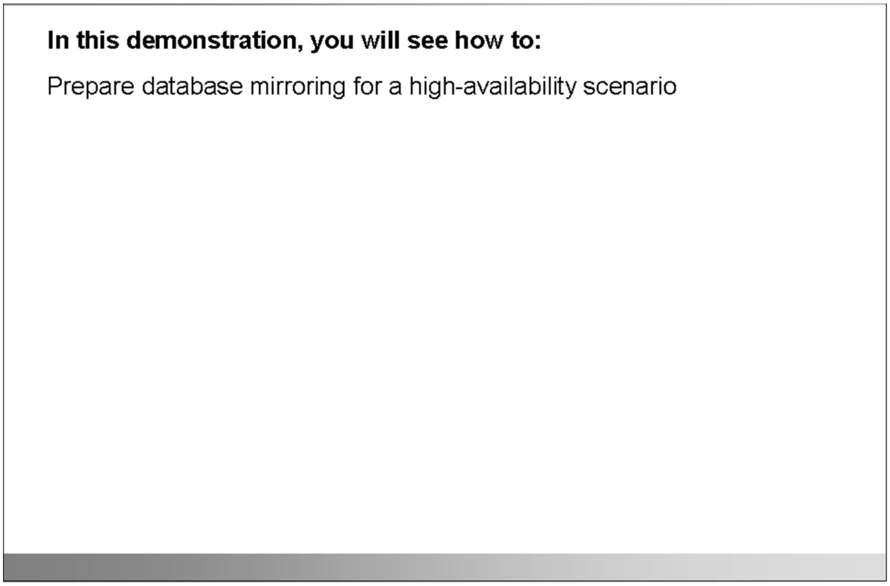
Note: Complete the remaining setup steps as soon as you can after taking the backup of the principal database.

4. **Set up the security and start the database mirroring session**. You can set up mirroring by using either Transact-SQL or the Database Mirroring Wizard.
5. **Add a witness to the session, optionally**. You can add the witness by using either Transact-SQL or the Database Mirroring Wizard. The database owner can turn off the witness for a database at any time. Turning off the witness is equivalent to having no witness, and automatic failover cannot occur.

Source:

- http://msdn.microsoft.com/en-us/library/ms366346.aspx
- http://msdn.microsoft.com/en-us/library/ms365599.aspx

Demonstration: How To Prepare Database Mirroring in a High-Availability Scenario



Key Points

The steps to prepare database mirroring for a high-availability scenario are:

1. On the **Start** menu, click **Server Manager**. The Server Manager console appears.
2. In the tree pane of the Server Manager console, expand **Configuration**, expand **Local Users and Groups**, and then click **Users**.
3. In the tree pane of the Server Manager console, right-click **Users**, and then click **New User**. The **New User** dialog box appears.
4. In the **User name** box of the **New User** dialog box, type **Ben Darton**, in the **Password** and **Confirm password** boxes, type **Pa$$word**, click to clear the **User must change password at next logon**, and then click **Create**.

5. Switch to the Microsoft SQL Server Management Studio window.
6. In Windows Explorer, browse to the **D:\Demofiles\Mod11** folder, and then double-click the **M11_Demo2.sql** file.
7. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, right-click the **NYC-SQL1 (SQL Server 10.0.1600 – NYC-SQL1\Administrator)** node, and then click **New Query**.
8. Copy the **Create an endpoint on the principal server instance** code from the **M11_Demo2.sql** file and paste it in the Query Editor pane for the NYC-SQL1 instance, and then click **Execute**.
9. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, click **Connect**, and then click **Database Engine**. The **Connect to Server** dialog box appears.
10. In the **Connect to Server** dialog box, in the **Server name** box, type or select the name of the principal server, **NYC-SQL1\DEVELOPMENT**, and then click **Connect**.
11. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, right-click the **NYC-SQL1\DEVELOPMENT (SQL Server 10.0.1600 – NYC-SQL1\Administrator)** node, and then click **New Query**.
12. Copy the **Create an endpoint on the partner server instance** code from the **M11_Demo2.sql** file and paste it in the Query Editor pane for the NYC-SQL1\DEVELOPMENT instance, and then click **Execute**.
13. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, click **Connect**, and then click **Database Engine**. The **Connect to Server** dialog box appears.
14. In the **Connect to Server** dialog box, in the **Server name** box, type or select the name of the principal server, **NYC-SQL1\INSTANCE3**, and then click **Connect**.
15. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, right-click the **NYC-SQL1\INSTANCE3 (SQL Server 10.0.1600 – NYC-SQL1\Administrator)** node, and then click **New Query**.

16. Copy the **Create an endpoint on the witness server instance** code from the **M11_Demo2.sql** file and paste it in the Query Editor pane for the NYC-SQL1\INSTANCE3 instance, and then click **Execute**.

Question: Why should certificates be used for setting up database mirroring?

Question: How do servers authenticate each other's identity in a database mirroring session?

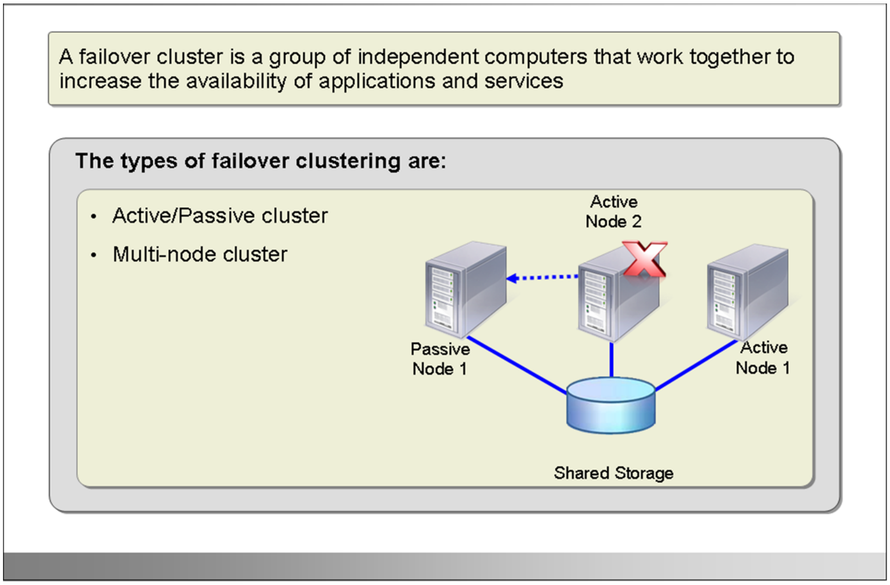
Lesson 4
Implementing Failover Clustering

- What Is Failover Clustering?
- Quorum Modes for a Failover Cluster
- What Is a Geographically Dispersed Failover Clustering?
- Considerations for Setting Up Geographically Dispersed Failover Clustering
- Considerations for Setting Up a Failover Cluster
- Considerations for Using Disk and File Share Quorum
- How To Set Up SQL Server 2008 Failover Cluster

Failover Clustering is a technology toolset which is designed to help businesses meet their availability and uptime goals. A failover cluster contains one or more clustered servers called nodes. It also contains a configuration of shared cluster disks that are set up for use within the cluster. Failover clustering can protect against hardware and software failures by failing over resources from one server to another as required.

In this lesson, you will learn about failover clustering and geographically-dispersed failover clustering. You will also learn about the hardware requirements for failover clustering and how to implement a failover cluster in SQL Server 2008.

What Is Failover Clustering?



Key Points

SQL Server failover clustering provides high-availability support for an entire SQL Server instance. SQL Server failover clusters are built on top of Windows Server failover clusters. To create a SQL Server failover cluster, you need to first create the underlying Windows Server failover cluster.

A SQL Server failover cluster instance appears on the network as a single computer, but has functionality that provides failover from one node to another if the current node becomes unavailable. For example, during a non-disk hardware failure, operating system failure, or planned operating system upgrade, you can configure an instance of SQL Server on one node of a failover cluster to fail over to any other node in the disk group.

A failover cluster does not protect against disk failure. You can use failover clustering to reduce system downtime and provide higher application availability. Failover clustering is supported in SQL Server Enterprise Edition and SQL Server Developer Edition, and with some restrictions, in SQL Server Standard Edition.

A SQL Server failover cluster, also known as a failover cluster instance, consists of the following:

- One or more Windows Server failover cluster nodes
- A cluster resource group dedicated for the SQL Server failover cluster, which contains the following:
  - Network name to access the SQL Server failover cluster
  - IP addresses
  - Shared disks used for the SQL Server database and log storage
  - Resource dynamic-link libraries (DLLs) that control the SQL Server failover behavior
  - Check-pointed registry keys that are automatically synchronized across the failover cluster nodes

A SQL Server failover cluster appears on the network as a single SQL Server instance on a single computer. Internally, only one of the nodes owns the cluster resource group at a time, serving all the client requests for that failover cluster instance. In case of a failure, or a planned upgrade, the group ownership is moved to another node in the failover cluster. This process is called failover. By using the Windows Server failover cluster functionality, SQL Server failover cluster provides high availability through redundancy at the instance level.

Types of Failover Clustering

The types of failover clustering are:

Active / Passive Cluster

In Active/Passive Cluster, the node that is serving the user request is the active node and the node that is standby is the passive node. By creating a failover cluster and configuring a SQL Server 2008 failover cluster, you can help provide high availability for the service or application. This design does not include a disaster recovery option as part of the failover cluster, which makes this design less expensive and simpler to deploy.

For the maximum availability of any server, it is important to follow the best practices for server management. For example, carefully managing the physical environment of the servers, testing software changes before placing them into production, and carefully keeping track of software updates and configuration changes on all clustered servers.

A failover cluster usually includes a storage unit that is physically connected to all the servers in the cluster, although any given volume in the storage is only accessed by one server at a time.

The following illustration shows failover on a two-node cluster supporting an application.

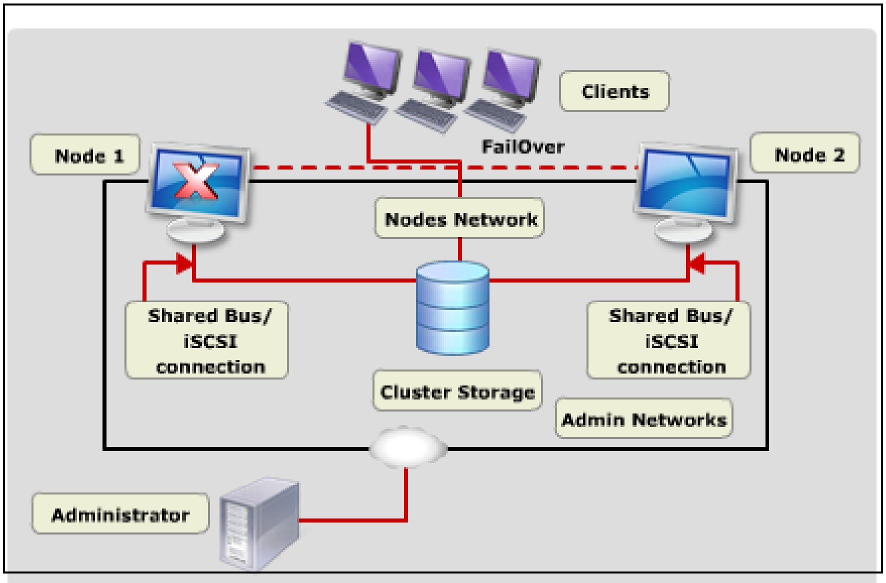


Figure 11.1. Active/Passive Cluster

Multi-Node Cluster

This design starts with the basic idea of clustering a service or application and expands it to include multiple services and applications on one cluster.

With any design that involves multiple services and applications on a cluster, it is crucial to consider server loads and server capacity. Any given node on the cluster must have enough capacity to perform reasonably well with the load it will carry in a typical failover situation. It is also important to consider the relative importance of each service or application in relation to your organization’s goals. If one of the services or applications is significantly less important, it might be more appropriate not to run that application as a clustered application. Instead, you might place it on a nonclustered server. This would mean that you could maximize the server capacity devoted to supporting the clustered applications, so that if one of the clustered servers had minor performance problems, the clustered applications would be affected only minimally, or not at all.

The following illustration shows four clustered applications on a three-node cluster.

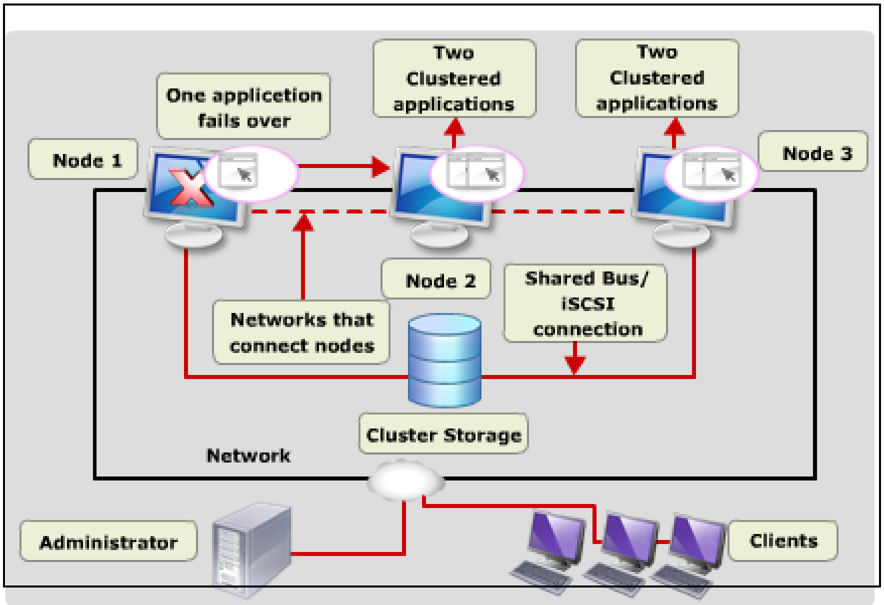


Figure 11.2. Multi-Node Cluster

Source:

- http://msdn.microsoft.com/en-us/library/ms189134.aspx
- http://msdn.microsoft.com/en-us/library/ms190202.aspx
- http://technet.microsoft.com/en-us/library/dd197609(WS.10).aspx
- http://technet.microsoft.com/en-us/library/dd197459(WS.10).aspx

Quorum Modes for a Failover Cluster

Description of Cluster	Quorum Recommendation
Odd number of nodes	Node Majority
Even number of nodes (but not a multisite cluster)	Node and Disk Majority
Even number of nodes, multisite cluster	Node and File Share Majority
Even number of nodes, no shared storage	Node and File Share Majority

Key Points

There have been significant improvements to the quorum model in SQL Server 2008. In Windows Server 2008, a majority of "votes" is what determines whether a cluster achieves quorum. Nodes can vote, and where appropriate, either a disk in cluster storage called a disk witness or a file share called a file share witness can vote. There is also a quorum mode called No Majority: Disk Only, which functions like the disk-based quorum in Windows Server 2003. Aside from that mode, there is no single point of failure with the quorum modes, because what matters is the number of votes, not whether a particular element is available to vote.

This new quorum model is flexible and you can choose the mode best suited to your cluster.

Important: In most situations, it is best to use the quorum mode selected by the cluster software. If you run the quorum configuration wizard, the quorum mode that the wizard lists as "recommended" is the quorum mode chosen by the cluster software. You must change the quorum configuration only if you have determined that the change is appropriate for your cluster.

There are four quorum modes:

- **Node Majority.** Each node that is available and in communication can vote. The cluster functions only with a majority of the votes, that is, more than half.
- **Node and Disk Majority.** Each node plus a designated disk in the cluster storage (the "disk witness") can vote, whenever they are available and in communication. The cluster functions only with a majority of the votes, that is, more than half.
- **Node and File Share Majority.** Each node plus a designated file share created by the administrator (the "file share witness") can vote, whenever they are available and in communication. The cluster functions only with a majority of the votes, that is, more than half.
- **No Majority: Disk Only**. The cluster has quorum if one node is available and in communication with a specific disk in the cluster storage. Only the nodes that are also in communication with that disk can join the cluster.

Diagrams of quorum modes

The following diagrams show how each of the quorum modes affects a cluster to achieve quorum.

Node Majority

The following diagram illustrates node majority used for a cluster with an odd number of nodes.

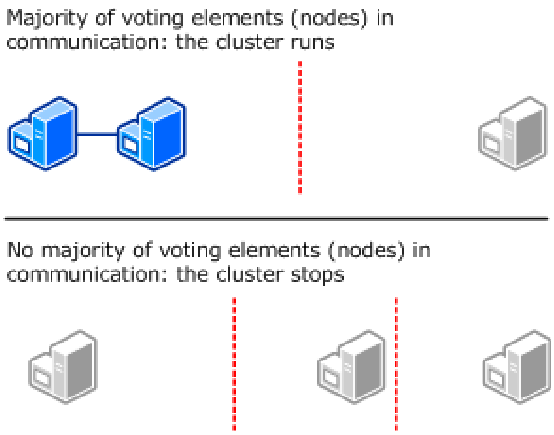


Figure 11.3. Node Majority

Node and Disk Majority

The following diagram illustrates node and disk majority used for a cluster with an even number of nodes. Each node can vote, as can the disk witness.

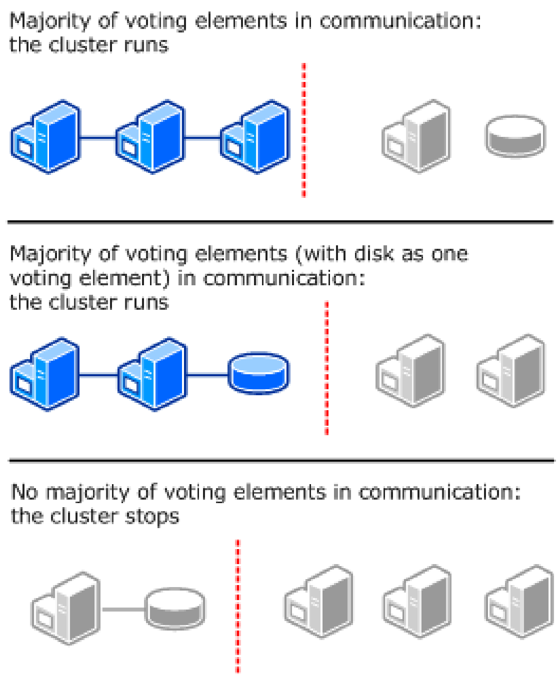


Figure 11.4. Node and Disk Majority

The following diagram shows how the disk witness also contains a replica of the cluster configuration database in a cluster that uses Node and Disk Majority.

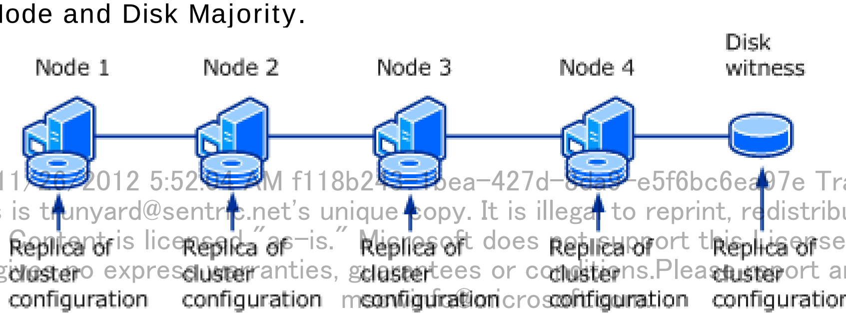


Figure 11.5. Node and Disk Majority with Replica of Cluster

Node and File Share Majority

The following diagram shows Node and File Share Majority used for a cluster with an even number of nodes and a situation where having a file share witness works better than having a disk witness. Each node can vote, as can the file share witness.

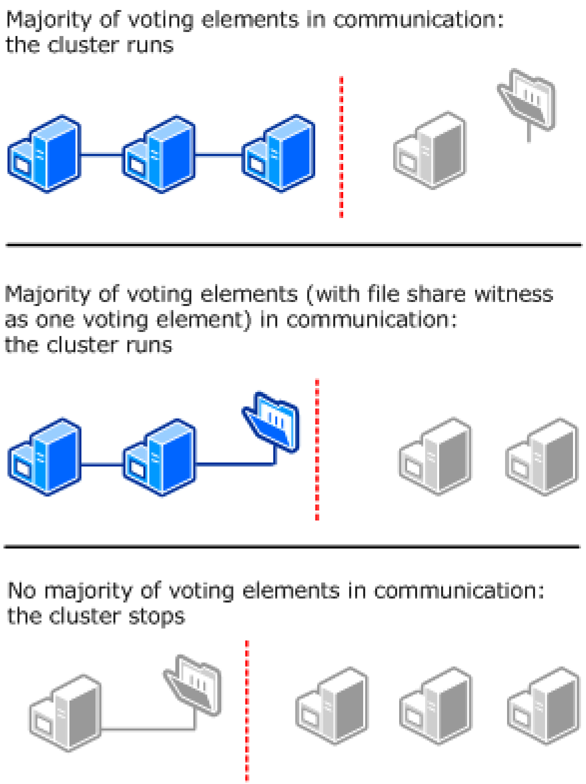


Figure 11.6. Node and File Share Majority

The following diagram shows how the file share witness can vote, but does not contain a replica of the cluster configuration database. Note that the file share witness does contain information about which version of the cluster configuration database is the most recent.

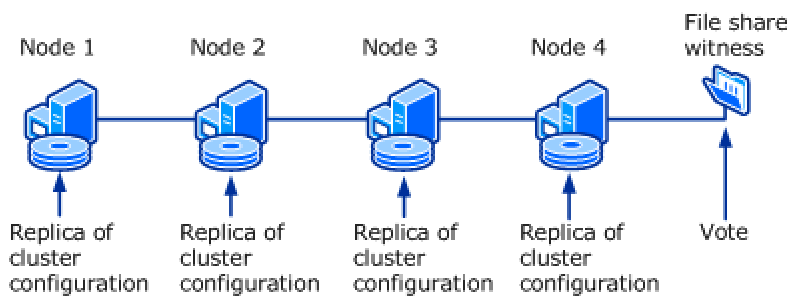


Figure 11.7. Node Majority with Replica of Cluster

No Majority: Disk Only

The following illustration shows how a cluster that uses the disk as the only determiner of quorum can run even if only one node is available and in communication with the quorum disk. It also shows how the cluster cannot run if the quorum disk is not available. For this cluster, which has an odd number of nodes, Node Majority is the recommended quorum mode.

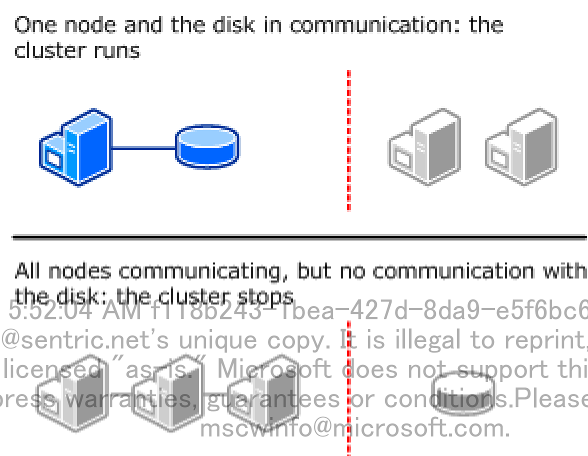


Figure 11.8. No Majority: Disk Only

Benefits of achieving quorum

When network problems occur, they can interfere with communication between cluster nodes. A small set of nodes might be able to communicate together across a functioning part of a network but not be able to communicate with a different set of nodes in another part of the network. This can cause serious issues. In this "split" situation, at least one of the sets of nodes must stop running as a cluster.

To prevent the issues that are caused by a split in the cluster, the cluster software requires that any set of nodes running as a cluster must use a voting algorithm to determine whether, at a given time, that set has quorum. Because a given cluster has a specific set of nodes and a specific quorum configuration, the cluster will know how many "votes" constitutes a majority. If the number drops below the majority, the cluster stops running. Nodes will still listen for the presence of other nodes, in case another node appears again on the network, but the nodes will not begin to function as a cluster until the quorum exists again.

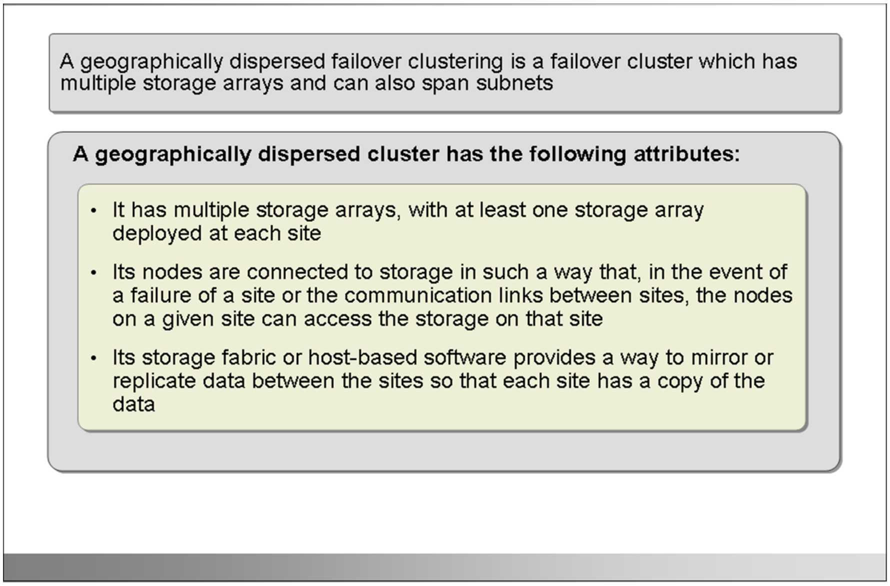
For example, in a five node cluster that is using a node majority, consider if nodes 1, 2, and 3 can communicate with each other but not with nodes 4 and 5. Nodes 1, 2, and 3 constitute a majority, and they continue running as a cluster. Nodes 4 and 5, being a minority, stop running as a cluster. If node 3 loses communication with other nodes, all nodes stop running as a cluster. However, all functioning nodes will continue to listen for communication, so that when the network begins working again, the cluster can form and begin to run.

Question: What is the key difference between the quorum in Windows 2003 and Windows 2008?

Source:

- http://technet.microsoft.com/ja-jp/library/cc770620(WS.10).aspx

What Is a Geographically Dispersed Failover Clustering?



Key Points

A geographically dispersed failover clustering is a failover cluster which has multiple storage arrays. It can also span subnets. A geographically dispersed or multisite cluster is a Windows Server 2008 failover cluster that has the following attributes:

- **It has multiple storage arrays, with at least one storage array deployed at each site**. This ensures that in the event of failure of any one site, the other site or sites will have local copies of the data that they can use to continue to provide the services and applications.

- **Its nodes are connected to storage in such a way that in the event of a failure of a site or the communication links between sites, the nodes on a given site can access the storage on that site**. In other words, in a two-site configuration, the nodes in Site 1 are connected to the storage in Site 1 directly, and the nodes in Site 2 are connected to the storage in Site 2 directly. In the event of a failure, the nodes in Site 1 can continue, without accessing the storage on Site 2.
- **Its storage fabric or host-based software provides a way to mirror or replicate data between the sites so that each site has a copy of the data**. There is no shared mass storage that all the nodes access and data must thus be replicated between separate storage arrays to which each node is attached.

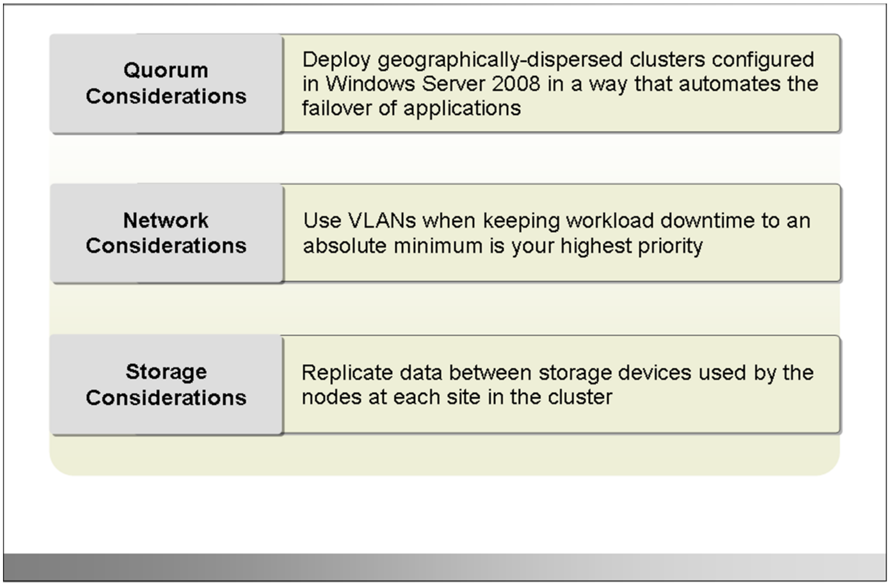
  Because of their extreme disaster tolerance, multisite clusters should be thought of as both a high-availability solution and a disaster recovery solution. The automatic failover of multisite clustering means that your backup data on the other nodes of the cluster is available in moments upon failure of your primary site. Because there is automatic failover, you can quickly fail back to your primary site after your servers there have been restored.

Question: What are the key requirements for setting up a geographically-dispersed cluster?

Source:

- download.microsoft.com/.../WS2008%20Multi%20Site%20Clustering.doc

Considerations for Setting Up Geographically Dispersed Failover Clustering



Key Points

When you set up a multi-site cluster, you need to consider the quorum model, and the network, and data storage for the cluster.

Quorum Considerations

You can deploy geographically-dispersed clusters configured in Windows Server 2008 in a way that automates the failover of applications in situations where the following occur:

- Communication between sites has failed and one site is still functioning.
- The other site is down and is not available to run applications.

You can create multi-site clusters by using Node and File Share Majority and Node Majority quorum models.

A cluster quorum configured to use a Node and File Share Majority is an excellent solution for multi-site clusters. The file share witness can reside at a third site, independent of either site, hosting a cluster node for high disaster resilience. Moreover, a single file server can serve as a witness to multiple clusters with each cluster using a separate file share witness on the file server.

If you do not want a file share witness at a site independent of your cluster sites, you can still use a multi-site cluster with a Node Majority cluster configuration. A node-majority cluster consists of three of more cluster nodes without shared storage.

Network Considerations

A major improvement to clustering in Windows Server 2008 is that cluster nodes can now reside on different subnets. Cluster nodes in Windows Server 2008 can communicate across network routers. This means that you need not stretch virtual local area networks (VLANs) to connect geographically-separated cluster nodes, far reducing the complexity and cost of setting up and maintaining multi-site clusters.

One consideration for subnet-spanning clusters can be the client response time. Client computers cannot connect to passive nodes unless the Domain Name System (DNS) servers update records to point to the new server hosting. For this reason, you can use VLANs when keeping workload downtime to an absolute minimum is your highest priority.

Storage Considerations

You need to replicate data between storage devices used by the nodes at each site in the cluster because there is no shared mass storage for all of the nodes in a multi-site cluster.

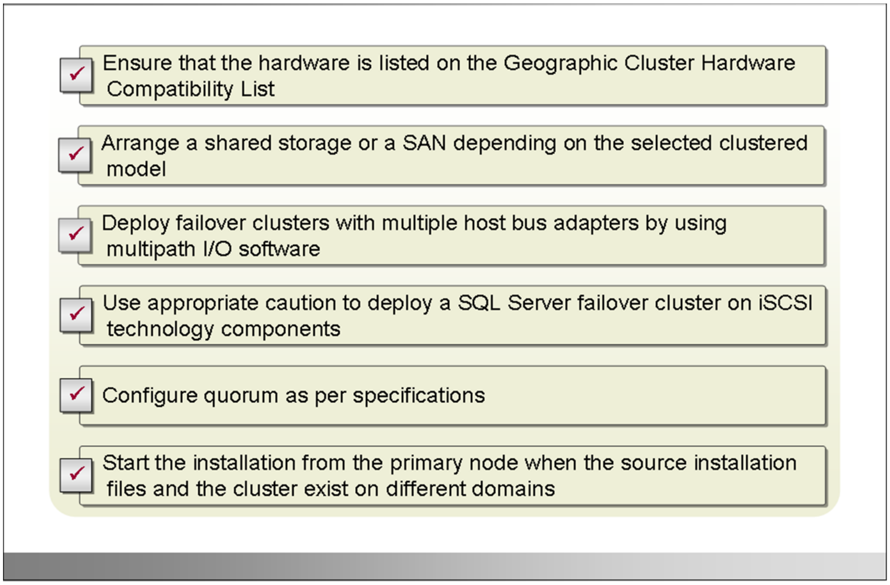
Consider the case of a cluster dispersed between two sites, Site A and Site B. Traditionally, when a cluster node at Site A makes a change to its local storage, the changes are replicated to Site B. The LUNs of the storage at Site A would generally be in a Read-Write mode, while the replica at Site B would be in a Read-Only mode to ensure data consistency. When a failover occurs, Site B must switch its replica LUNs from Read-Only to Read-Write, and then replicate its changes to the other site (Site A).

Data replication is the backbone of geographically-dispersed clustering. You can replicate data between sites within a multi-site cluster by different techniques, and at different levels in the clustering infrastructure. At the hardware level or at the block level, replication is performed by the storage controllers or by mirroring the software. At the file-system level when replicating file system changes, the host software performs the replication. Finally, at the application level, the applications themselves can replicate data.

Source:

- download.microsoft.com/.../WS2008%20Multi%20Site%20Clustering.doc

Considerations for Setting Up a Failover Cluster



Key Points

Before you begin the failover cluster installation process, you need to consider the following:

- **Ensure that the hardware is listed on the Geographic Cluster Hardware Compatibility List**. If the cluster solution includes geographically-dispersed cluster nodes, you must verify additional hardware components such as network latency and shared disk support. All hardware included in a geographically-dispersed failover cluster must be on the Geographic Cluster Hardware Compatibility List.
- **Arrange a shared storage or a SAN depending on the selected clustered model**. When deploying a SAN with a failover cluster, servers from different clusters must not be able to access the same storage devices. In most cases, a LUN that is used for one set of cluster servers should be isolated from all other servers through LUN masking or zoning.

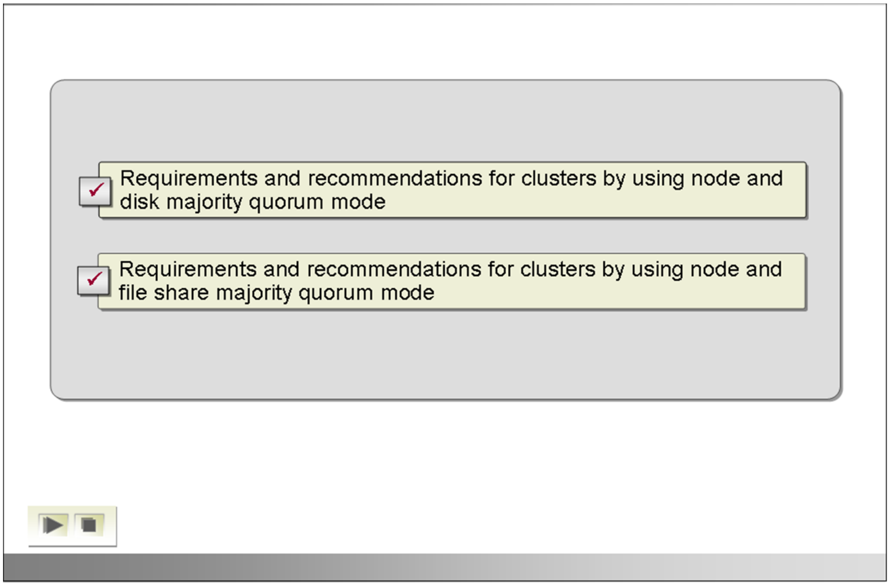
- **Deploy failover clusters with multiple host bus adapters by using multipath I/O software**. In a highly-available storage fabric, you can deploy failover clusters with multiple host bus adapters by using multipath I/O software. This provides the highest level of redundancy and availability. For Windows Server 2008, your multipath solution must be based on Microsoft Multipath I/O (MPIO). Your hardware vendor will usually supply an MPIO device-specific module (DSM) for your hardware, although Windows Server 2008 includes one or more DSMs as part of the operating system.
- **Use appropriate caution to deploy a SQL Server failover cluster on iSCSI technology components**. If you deploy a SQL Server failover cluster on iSCSI technology components, you should use appropriate caution. Microsoft supports Microsoft SQL Server when it is deployed on iSCSI technology components that have received the "Designed for Windows" Logo Program qualification. SQL Server installations that use iSCSI will require these iSCSI hardware devices in addition to the network adapters that are required for typical network communications.
- **Configure quorum as per specifications**. You need to consider quorum disk resource sharing. In a server cluster, the quorum disk contains a master copy of the server cluster configuration. It can be used as a tie-breaker if all network communication fails between cluster nodes. Depending on the type of server cluster you implement, the quorum disk might or might not be a physical disk on the shared cluster disk array. It is best to reserve an entire cluster disk for use as the quorum disk. Resources other than the quorum resource may be permitted to access the quorum disk.
  However, making the quorum resource share the same disk with other resources forces you to choose between two undesirable alternatives. You must either configure the resource so that its failure does not affect the group, or allow the group to be affected by the other resource's failures. In the first case, you lose failover support for the resource; in the second, the quorum resource fails over along with the rest of the group that contains both the quorum resource and the failed resource. As a result, the entire cluster is offline for as long as it takes the group to fail over.

- **Start the installation from the primary node when the source installation files and the cluster exist on different domains**. To install a SQL Server failover cluster when the source installation files and the cluster exist on different domains, copy the installation files to the primary node of the cluster, then start the installation from the primary node.
domains. To install a SQL Server failover cluster when the source installation files and the cluster exist on different domains, copy the installation files to the primary node of the cluster, then start the installation from the primary node.

Source:

- http://msdn.microsoft.com/en-us/library/ms189910.aspx
- http://technet.microsoft.com/en-us/library/cc771404.aspx
- http://support.microsoft.com/kb/833770

Considerations for Using Disk and File Share Quorum



Key Points

To use disk and file share quorum, consider the following:

Requirements and Recommendations for Clusters by Using Node and Disk Majority Quorum Mode

When using the Node and Disk Majority mode, review the following requirements and recommendations for the disk witness:

- Use a small LUN that is at least 512 MB in size.
- Choose a basic disk with a single volume.
- Ensure that the LUN is dedicated to the disk witness. It must not contain any other user or application data.
- Choose whether to assign a drive letter to the LUN based on the needs of your cluster. The LUN does not have to have a drive letter (to conserve drive letters for applications).
- Add the LUN to the set of disks that the cluster can use.

- Ensure that the LUN has been verified by using the Validate a Configuration Wizard.
- Configure the LUN with hardware Redundant Array of Inexpensive Disks (RAID) for fault tolerance.
- Do not back up the disk witness or the data on it. Backing up the disk witness can add to the I/O activity on the disk and decrease its performance, which could potentially cause it to fail.
- Avoid all antivirus scanning on the disk witness.
- Format the LUN with the NTFS file system.

If there is a disk witness configured, but bringing that disk online will not achieve a quorum, then it remains offline. If bringing that disk online will achieve the quorum, then it is brought online by the cluster software.

Note: These requirements and recommendations also apply to the quorum disk for the No Majority: Disk Only mode.

Requirements and Recommendations for Clusters by Using Node and File Share Majority Quorum Mode

When using the Node and File Share Majority mode, review the following for the file share witness:

- Use a Server Message Block (SMB) share on a Windows Server 2003 or Windows Server 2008 file server.
- Ensure that the file share has a minimum of 5 MB of free space.
- Ensure that the file share is dedicated to the cluster and is not used in other ways (including storage of user or application data).
- Do not place the share on a node that is a member of this cluster or will become a member of this cluster in the future.
- Place the share on a file server that has multiple file shares servicing different purposes. This may include multiple file share witnesses, each one a dedicated share. You can even place the share on a clustered file server (in a different cluster), which would typically be a clustered file server containing multiple file shares servicing different purposes.

- Co-locate the external file share at one of the sites where a node or nodes are located for a multi-site cluster.
- Place the file share on a server that is a member of a domain, in the same forest as the cluster nodes.
- For the folder that the file share uses, ensure that the administrator has the Full Control share and Windows NT® File System (NTFS) permissions.
- Do not use a file share that is part of a Distributed File System (DFS) Namespace.

Question: What is the key hardware requirement for setting up a failover cluster?

Source:

- http://technet.microsoft.com/en-us/library/cc770620(WS.10).aspx

Demonstration: How To Verify the Setup of SQL Server 2008 Failover Cluster

In this demonstration, you will see how to:

Verify the setup of failover clustering

Key Points

Note: Before performing this step, you need to perform some perquisite tasks. Refer the Trainer Preparation Guide of this course for how to perform the prerequisite tasks.

The steps to verify the setup of failover clustering are:

1. In the Failover Cluster Management window, in the work area for the SQL Server (MSSQLServer) service, in the **Resources and Applications** area, under **Other Resources**, right-click **SQL Server**, and then click **Properties**. The **SQL Server Properties** dialog box appears.
2. In the **SQL Server Properties** dialog box, on the **Dependencies** tab, review dependency setting of the cluster.
3. On the **Policies** tab, in the **Response to resource failure** area, review response setting.

4. In the Failover Cluster Management window, in the work pane for the SQL Server (MSSQLServer) service, in the **Resources and Applications** area, under **Other Resources**, right-click **SQL Server**, point to **More Actions**, and then click **Simulate failure of this resource**.
5. In the **Please confirm action** box, click **Simulate Failure of SQL Server**, and then wait until the SQL Server resource displays the online status.
6. Under **Other Resources**, right-click **SQL Server**, and then click **Properties**. The **SQL Server Properties** dialog box appears.
7. In the **SQL Server Properties** dialog box, on the **General** tab, ensure that **SQL Server** is listed in the **Resource Name** box, and then click **OK**.
8. In the Failover Cluster Management window, in the navigation pane, under the NYC-FC.QuantamCorp.com node, click **Nodes**. Ensure that NYC-SQl1 is listed in the work pane.

Question: Why should you specify an additional IP during setup?

Question: What is the key feature of the setup process for a failover cluster?

Lesson 5
Designing a High-Availability Strategy

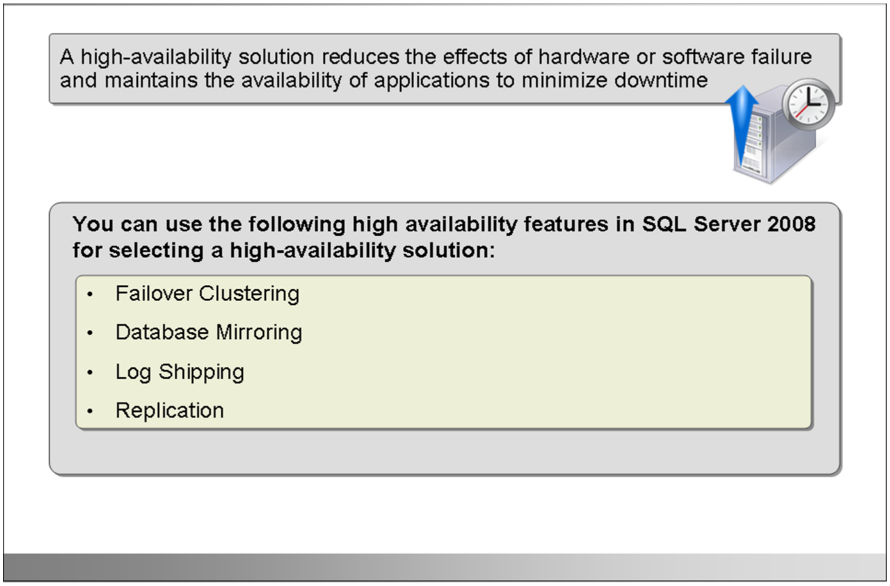
- Selecting a High-Availability Solution
- Business Scenarios for Implementing High-Availability Solutions
- Interoperability and Coexistence of High-Availability Features in SQL Server 2008

Key Points

The high-availability options are interoperable and their features can be used independently or in groups. The choice of the high-availability solution depends on the business scenario.

In this lesson, you will learn about the advantages and disadvantages of some of the solutions. You will also learn about the solution to implement in various business scenarios. You will learn about how the different options work together.

Selecting a High-Availability Solution



Key Points

SQL Server provides several options for creating high availability for a server or database. By using server clustering or database mirroring, you can provide high-availability support for an entire instance of SQL Server. You may opt to create a secondary database to serve as a backup of the primary database with no recovery, or with standby. Data could also be distributed to one or more secondary servers.

You can use the following high availability features in SQL Server 2008 for selecting a high-availability solution:

Failover Clustering

Failover clustering provides automatic detection and failover, manual failover, and transparent client redirect.

However, failover clustering has some constraints, as follows:

- It operates at the server instance scope.
- It requires signed hardware.

- It provides no reporting while on standby mode.
- It utilizes a single copy of the database.
- It does not protect against disk failure.

Database Mirroring

Like failover clustering, database mirroring also provides automatic detection and failover, manual failover, and transparent client redirect. It offers a substantive increase in availability by overcoming a few limitations of failover clustering.

- Database mirroring provides additional benefits, which include the following:
    - It operates at the database scope.
    - It uses a single, duplicate copy of the database.
    - It uses standard servers.
    - It provides limited reporting on the mirror server by using database snapshots.
    - It provides zero work loss through delayed commit on the principal database during synchronous operation.

Log Shipping

Log shipping can be an alternative to database mirroring. Although similar in concept, asynchronous database mirroring and log shipping have some key differences. Unlike database mirroring, log shipping supports multiple secondary databases on multiple server instances for a single primary database. It also allows the user to specify a delay between the time required for creating a log backup of the primary database and the time required for restoring the log backup. Asynchronous database mirroring has the potential advantage over log shipping of a shorter time difference when a change is made in the primary database and when that change is reflected to the mirror database. An advantage of database mirroring over log shipping is that high-safety mode is a no data loss configuration that is supported as a simple failover strategy.

Replication

Replication is supported in all editions of SQL Server. This feature uses a publish-subscribe model, allowing a primary server to distribute data to one or more secondary servers.

Replication provides the following benefits:

- It allows filtering in the database to provide a subset of data at the secondary databases.
- It allows more than one redundant copy of the database.
- It allows real-time availability and scalability across multiple databases.
- It allows complete availability of the secondary databases for reporting or other functions, without query recovery.

Question: What should you consider when selecting a high-availability solution?

Source:

- http://msdn.microsoft.com/en-us/library/ms190202(SQL.90).aspx
- http://msdn.microsoft.com/en-us/library/bb510414.aspx

Discussion: Business Scenarios for Implementing High-Availability Solutions

- What database availability technologies should you use if you want to minimize the downtime of SQL Server to ensure at least 99.9% availability?
- What database availability technologies should you use to ensure SQL Servers can continue to provide services even if the production site fails?
- What database availability technologies should you use to ensure that SQL Server can continue to provide services even if the production server fails without adding extra hardware?
- What database availability technologies should you use if you cannot implement failover clustering in branch offices because of cost and space limitation?

Key Points

In this discussion, you are presented scenarios in which you are asked to determine the high-availability solution you will use for the various business requirements. You and your classmates will discuss possible solutions to the scenario.

Scenario

You are planning a high-availability solution for your organization. The following are the different business requirements:

Requirement 1

You need to minimize the downtime of your SQL Server to ensure that it can achieve at least 99.9% availability.

Question: What database availability technologies should you use if you want to minimize the downtime of SQL Server to ensure at least 99.9% availability?

Requirement 2

You need to ensure that the SQL Servers can continue to provide services even if the production site fails.

Question: What database availability technologies should you use to ensure SQL Servers can continue to provide services even if the production site fails?

Requirement 3

You need to ensure SQL Servers can continue to provide services, even if the production server fails, but without adding extra hardware.

Question: What database availability technologies should you use to ensure that SQL Server can continue to provide services even if the production server fails without adding extra hardware?

Requirement 4

In some branch offices, it is not possible to implement failover clustering because of cost and space limitation.

Question: What database availability technologies should you use if you cannot implement failover clustering in branch offices because of cost and space limitation?

Interoperability and Coexistence of High-Availability Features in SQL Server 2008

Feature of SQL Server	DB Mirroring	Log Shipping
Failover Cluster	✓	✗
Full-Text Catalog	✓	✗
DB Snapshot	✓	✓
Replication	✓	✓

Key Points

SQL Server provides several options for creating high availability for a server or database. By implementing the high-availability solutions, you can reduce system downtime and provide higher application availability. You can maintain a single standby database, or mirror database, for a corresponding production database. You can replicate data on secondary servers, thereby enabling real-time availability and scalability across these servers. The scalable shared database feature also lets you scale out a read-only database built exclusively for reporting. The high-availability solutions and their features can be used independently or as a package to fulfill the high-availability needs.

For example, database mirroring can be used with the following features or components of SQL Server:

- Failover clustering
- Log shipping
- Full-text catalogs

- Database snapshots
- Replication
    - Log shipping can be used with the following features or components of SQL Server:
- Database mirroring
- Replication

Database Mirroring and Failover Clustering

Typically, when mirroring is used with clustering, both the principal server and the mirror server reside on clusters. The principal server runs on the failover clustered instance of one cluster and the mirror server runs on the failover clustered instance of a different cluster.

You can establish a mirroring session in which one partner resides on the failover clustered instance of a cluster and the other partner resides on a separate, computer.

Database Mirroring and Log Shipping

The principal database in a mirroring session can also act as the primary database in a log shipping configuration, or vice versa, because the log shipping backup share is intact. The database mirroring session can run either in the synchronous mode or the asynchronous mode.

Typically, when combining log shipping and database mirroring, the mirroring session is established before log shipping, although this is not required. Then, the current principal database is configured as the log shipping principal database, along with one or more remote secondary databases. Also, the mirror database must be configured as a log shipping mirror database. The log shipping secondary databases should be on different server instances than either the principal server or mirror server.

Database Mirroring and Full-Text Indexing

To mirror a database that has a full-text catalog, use backup as usual to create a full database backup of the principal database, and then restore the backup to copy the database to the mirror server.

Database Mirroring and Database Snapshot

To use a mirror database for reporting, you can create a database snapshot on the mirror database and direct client connection requests to the most recent snapshot. A database snapshot is a static, read-only, transaction-consistent snapshot of its source database as it existed at the moment of the snapshot's creation. To create a database snapshot on a mirror database, the database must be in the synchronized mirroring state.

Unlike the mirror database itself, a database snapshot is accessible to clients. As long as the mirror server is communicating with the principal server, you can direct reporting clients to connect to a snapshot. Note that because a database snapshot is static, new data is not available. To make relatively recent data available to your users, you must create a new database snapshot periodically, and have applications direct incoming client connections to the newest snapshot.

Database Mirroring and Replication

Database mirroring can be used in conjunction with replication to provide availability for the publication database. Database mirroring involves two copies of a single database that typically reside on different computers. At any given time, only one copy of the database is currently available to clients. This copy is known as the principal database. Updates made by clients to the principal database are applied on the other copy of the database, known as the mirror database. Mirroring involves applying the transaction log from every insertion, update, or deletion made on the principal database, onto the mirror database.

Log Shipping and Replication

Log shipping involves two copies of a single database that typically reside on different computers. At any given time, only one copy of the database is currently available to clients. This copy is known as the primary database. Updates made by clients to the primary database are propagated by means of log shipping to the other copy of the database, known as the secondary database. Log shipping involves applying the transaction log from every insert, update, or delete made on the primary database, onto the secondary database.

Log shipping can be used in conjunction with replication, with the following behavior:

- Replication does not continue after a log shipping failover. If a failover occurs, replication agents do not connect to the secondary, so transactions are not replicated to Subscribers. If a failback to the primary occurs, replication resumes. All transactions that log shipping copies from the secondary back to the primary are replicated to Subscribers.
- Replication continues if the primary database is lost. If the primary is permanently lost, the secondary can be renamed so that replication can continue.

Question: You have a replication setup on a database and you want to increase the availability of the publishing database. Which high-availability solution should you use to reduce the downtime?

Source:

- http://technet.microsoft.com/en-us/library/ms191309.aspx
- http://msdn.microsoft.com/en-us/library/ms187016.aspx
- http://technet.microsoft.com/en-us/library/ms191182.aspx
- http://technet.microsoft.com/en-us/library/ms175511.aspx
- http://msdn.microsoft.com/en-us/library/ms151799.aspx
- http://msdn.microsoft.com/en-us/library/ms151224.aspx

For more information, see High Availability: Interoperability and Coexistence.

Lab 11: Designing a High-Availability Solution for SQL Server 2008

- Exercise 1: Designing a High-Availability Strategy (Discussion)
- Exercise 2: Implementing Database Mirroring
- Exercise 3: Designing a Failover Cluster

Logon Information

Virtual machine	50400A-NYC-SQL1
User name	Administrator
Password	Pa$$w0rd

Estimated time: 60 minutes

Objectives:

After completing this lab, you will be able to:

- Design a high-level availability strategy.
- Implement database mirroring.
- Design a failover cluster.

Scenario

QuantamCorp has deployed the first two SQL Server 2008 servers. It will continue to deploy more to replace their existing installations of SQL Server 2005. QuantamCorp wants SQL Server 2008 to be the primary database engine for their organization. The application team is creating an Internet application. However, the system must be available most of the time. You need to evaluate various high-availability options and determine which one is best suited for the application. You have read the product documentation and decided to set up a test lab for database mirroring and a failover cluster. This setup will help the application team to plug their application to the configured mirroring clusters and failover clusters, and select the appropriate clusters that match the application's requirements.

You are assigned as the architect of this project. You need to achieve the following:

- Determine the best high-availability solution for a given scenario.
- Determine the impact of implementing different high-availability options.
- Implement database mirroring.
- Perform a manual or an automatic switch over on a database mirroring cluster.
- Design a failover cluster configuration.
- Verify the setup of the failover cluster.

Exercise 1: Designing a High-Availability Strategy (Discussion)

Scenario

Your company provides multiple applications for use to anonymous Internet users. A huge amount of cash is generated from these applications. Hence, you should always ensure the availability of these applications. The redundancy system in your organization is in place, with redundancy in all aspects related to the applications. Your organization's requirement for availability is to ensure that the application would experience a down time for less than five minutes in a single server failure event.

As per the regulatory requirements, you need to ensure that the applications can resume services as soon as possible in the event of a site failure. Currently, all servers associated with Web applications are located at the corporate headquarter in New York. Your company had decided to create a disaster recovery (DR) site in its regional office at Hong Kong. You need to provide a solution to ensure that all the databases are replicated in real time. In addition, you also need to ensure that the switching over of active or passive server roles does not happen automatically. You also need to provide easy switch over between servers on the production and the DR site.

Based on the above scenario, answer the following questions:

1. What would you suggest to the application team to meet their availability needs for their application databases in the company's corporate headquarter?
2. What would you suggest to meet the requirements for setting up a DR site?
3. You application team wants to roll out a small-scale application. The application will provide the same level of availability with no budget required to implement failover clustering. How would you meet this need of the application team?
4. Which mode should use for the database mirroring session between the production site and the DR site?

Exercise 2: Implementing Database Mirroring

Scenario

You want to implement a DR site for your application as per regulatory requirements. You need real-time replication support so that all changes happening at the production site will get replicated to the standby server. You plan to implement database mirroring to replicate databases to the DR site. You also plan to implement automatic failover by adding a witness server so that users get routed to the DR server if the production server fails.

The main tasks for this exercise are as follows:

1. Configure NYC-SQL1 for outbound connections.
2. Configure NYC-SQL1\DEVELOPMENT for outbound connections.

3. Configure NYC-SQL1\INSTANCE3 for outbound connections.
4. Configure NYC-SQL1 for inbound connections.
5. Configure NYC-SQL1\DEVELOPMENT for inbound connections.
6. Configure NYC-SQL1\INSTANCE3 for database mirroring.
7. Create a test database and a test table and add records to the table.
8. Configure NYC-SQL1 for database mirroring.
9. Configure NYC-SQL1\DEVELOPMENT for database mirroring.
10. Enable database mirroring.
11. Perform an automatic and manual switch over.

Task 1: Configure NYC-SQL1 for outbound connections.

1. Open the Microsoft SQL Server Management Studio window and connect to the server, with the following details:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
2. In the Query Editor pane of the Microsoft SQL Server Management Studio window, run a Transact-SQL (T-SQL) code to create an encryption key with the password, **Pa$$w0rd**, for the **master** database.
3. Run a T-SQL code to create an encryption certificate named, **pri_cert**, for the **master** database.
4. Run a T-SQL code to create a database mirroring endpoint named, **pri_endpoint**, with the following settings:
   - State: **Started**
   - TCP listener port: **8081**
   - TCP listener IP: **ALL**
   - Database mirroring authentication: **pri_cert**
   - Database mirroring encryption: **AES algorithm**
   - Role: **ALL**

5. Create a backup of the **pri_cert** certificate,named **pri_cert.cer**.in the **D:\Data** folder.

Note: Browse to the **D:\Labfiles\Mod11** folder and select the **E2_T1.sql** file to execute the required T-SQL code.

Task 2: Configure NYC-SQL1\DEVELOPMENT for outbound connections.

1. Using the Microsoft SQL Server Management Studio window, connect to the server, with the following details:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1\DEVELOPMENT**
   - Authentication: **Windows Authentication**
2. In the Query Editor pane of the Microsoft SQL Server Management Studio window, run a T-SQL code to create an encryption key with the password, **Pa$$w0rd**, for the **master** database.
3. Run a T-SQL code to create an encryption certificate named, **sb_cert**, for the **master** database.
4. Run a T-SQL code to create a database mirroring endpoint named, **sb_endpoint**, with the following settings:
   - State: **Started**
   - TCP listener port: **8082**
   - TCP listener IP: **ALL**
   - Database mirroring authentication: **sb_cert**
   - Database mirroring encryption: **AES algorithm**
   - Role: **ALL**
5. Create a backup of the **sb_cert** certificate,named **sb_cert.cer**.in the **D:\Data** folder.

Note: Browse to the **D:\Labfiles\Mod11** folder and select the **E2_T2.sql** file to execute the required T-SQL code.

Task 3: Configure NYC-SQL1\INSTANCE3 for outbound connections.

1. Using the Microsoft SQL Server Management Studio window, connect to the server, with the following details:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1\INSTANCE3**
   - Authentication: **Windows Authentication**
2. In the Query Editor pane of the Microsoft SQL Server Management Studio window, run a T-SQL code to create an encryption key with the password, **Pa$$w0rd**, for the **master** database.
3. Run a T-SQL code to create an encryption certificate named, **witness_cert**, for the **master** database.
4. Run a T-SQL code to create a database mirroring endpoint named, **witness_endpoint**, with the following settings:
   - State: **Started**
   - TCP listener port: **8083**
   - TCP listener IP: **ALL**
   - Database mirroring authentication: **witness_cert**
   - Database mirroring encryption: **AES algorithm**
   - Role: **Witness**
5. Create a backup of the **witness_cert** certificate,named **witness_cert.cer**.in the **D:\Data** folder.

Note: Browse to the **D:\Labfiles\Mod11** folder and select the **E2_T3.sql** file to execute the required T-SQL code.

Task 4: Configure NYC-SQL1 for inbound connections.

1. In the NYC-SQL1 instance, in the Microsoft SQL Server Studio window, in the Query Editor pane, run a T-SQL code to create a SQL logon, **mirror_logon**, with the password, **Pa$$w0rd**.Create the user, **mirror_user**, for the **mirror_logon** account.
2. Run a T-SQL code to add the **sb_cert** and the **mirror_cert** certificates to the **mirror_user** user.
3. Run a T-SQL code to grant the **CONNECT** permission to the **pri_endpoint** endpoint for the **mirror_logon** account.

Note: Browse to the **D:\Labfiles\Mod11** folder and select the **E2_T4.sql** file to execute the required T-SQL code.

Task 5: Configure NYC-SQL1\DEVELOPMENT for inbound connections.

1. In the NYC-SQL1\DEVELOPMENT instance, in the Microsoft SQL Server Studio window, in the Query Editor pane, run a T-SQL code to create an SQL logon, **mirror_logon**, with the password, **Pa$$w0rd**. Create the **mirror_user** user for the **mirror_logon** account.
2. Run a T-SQL code to add the **pri_cert** and the **witness_cert** certificates to the **mirror_user** user.
3. Run a T-SQL code to grant the **CONNECT** permission to the **sb_endpoint** endpoint for the **mirror_logon** account.

Note: Browse to the **D:\Labfiles\Mod11** folder and select the **E2_T5.sql** file to execute the required T-SQL code.

Task 6: Configure NYC-SQL1\INSTANCE3 for database mirroring.

1. In the NYC-SQL1\DEVELOPMENT instance, in the Query Editor pane of the Microsoft SQL Server Studio window, create a logon for database mirroring named, **mirror_logon**, with the password, **Pa$$w0rd**. Also, create a user **mirror_user** for the **mirror_logon** account.
2. Run a T-SQL code to attach the **pri_cert** and the **sb_cert** certificates for the **mirror_user** user authorization.
3. Run a T-SQL code to grant the **CONNECT** permission to the **witness_endpoint** endpoint for the **mirror_logon** account.

Note: Browse to the **D:\Labfiles\Mod11** folder and select the **E2_T6.sql** file to execute the required T-SQL code.

Task 7: Create a test database and a test table and add records to the table.

1. In the NYC-SQL1 instance, in the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a code to create a test database with the following structure.

File name	Settings
Mod11_Primary.mdf	Current size: 10 Maximum size: 50 Filegrowth: 5
Mod11_Log.ldf	Current size: 10 Maximum size: 50 Filegrowth: 5

In addition, create a test table named, **[dbo].[CallDetails]** and add records in the table as follows..

Field name	Data type	Value	Record
call_date	Datetime	NOT NULL	Records from the date 2004-01-01 to 2009-12-31
caller	varchar(20)	NOT NULL	Peter
duration	int	NULL	30
product	int	NOT NULL	A random number between one and three
cs_rep	int	NOT NULL	A random number between one and five
additional_info	Nchar(4000)	NULL	Call to ‘x’,

Field name	Data type	Value	Record
			where 'x' is a random number between one and 1,00,000

Note: Browse to the **D:\Labfiles\Mod11** folder and select the **E2_T7.sql** file to execute the required T-SQL code.

Task 8: Configure NYC-SQL1 for database mirroring.

1. In the NYC-SQL1 instance, set full recovery for the **Mod11** database.
2. Run a code to back up the primary file of the **Mod11** database at **D:\Data\Mod11.bak**, and back up the log file at **D:\Data\Mod11_log.bak**.

Note: Browse to the **D:\Labfiles\Mod11** folder and select the **E2_T8.sql** file to execute the required T-SQL code.

Task 9: Configure NYC-SQL1\DEVELOPMENT for database mirroring.

1. In the NYC-SQL1\DEVELOPMENT instance, for the Mod11 database, restore the Primary_Data file to **D:\Data\Mod11.bak**, restore the Primary_log file to **D:\Data\Mod11_standby_Data.mdf**, and the database log to **D:\Data\Mod11_Log.bak**.

Note: Browse to the **D:\Labfiles\Mod11** folder and select the **E2_T9.sql** file to execute the required T-SQL code.

Task 10: Enable database mirroring.

1. In the NYC-SQL1\DEVELOPMENT instance, in the Query Editor pane of the Microsoft SQL Server Studio window, set the partner value of **Mod11** database to **TCP://nyc-sql1:8081**.

2. In the NYC-SQL1 instance, in the Query Editor pane of the Microsoft SQL Server Studio window, set the partner value of **Mod11** to **TCP://nyc-sql1:8082**.
3. In the NYC-SQL1\INSTANCE3 instance, in the Query Editor pane of the Microsoft SQL Server Studio window, set the partner value of **Mod11** to **TCP://nyc-sql1:8083**.

Task 11: Perform an automatic and manual switch over.

1. In the Microsoft SQL Server Studio window, start the NYC-SQL1 instance.
2. In the Query Editor pane for the NYC-SQL1 instance, run a code to perform a failover on Mod11.
3. Verify if the Mod11 database is included in the database list for the NYC-SQL1 instance.
4. Verify if the Mod11 database is included in the database list for the NYC-SQL1\DEVELOPMENT instance. Also verify if the Mod11 database is disabled and involved in database mirroring session.

Results: After completing this exercise, you should have configured NYC-SQL1, NYC-SQL1\DEVELOPMENT, and NYC-SQL1\INSTANCE3 instances for outbound connections. You should have then configured the NYC-SQL1, and the NYC-SQL1\DEVELOPMENT instances for inbound connections, and configure the NYC-SQL1\INSTANCE3 instance for database mirroring. In addition, you should have also created a test database and a test table, and configured the NYC-SQL1 and the NYC-SQL1\DEVELOPMENT instances for database mirroring. Moreover, you should have enabled database mirroring and performed an automatic and manual switch over of the database.

Exercise 3: Designing a Failover Cluster

Scenario

Your application requires high-availability support to reduce the possibility of being unable to support mission critical applications. Your team has decided to implement failover cluster to provide the level of availability required by the application. You are now assigned the task to design this failover cluster.

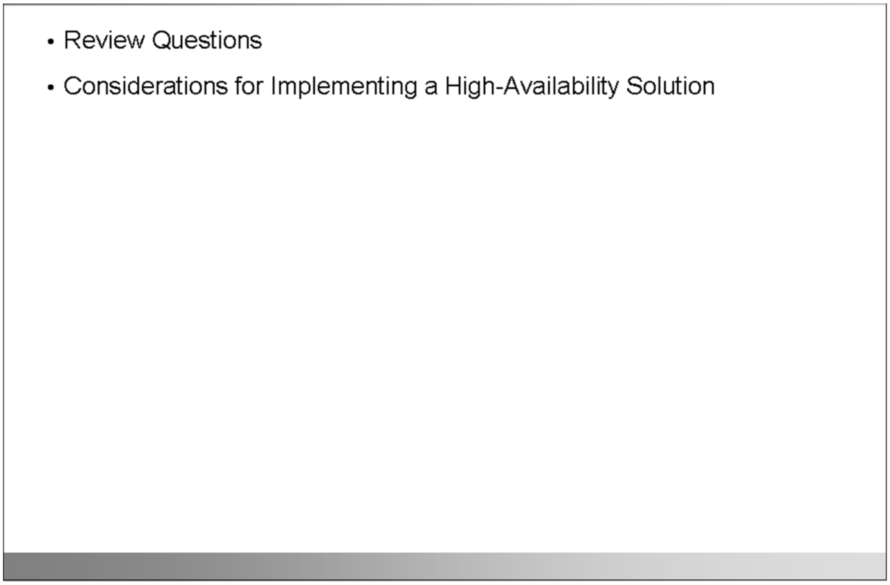
The main task for this exercise is to determine the design of an SQL failover cluster.

Task 1: Determine the design of an SQL failover cluster.

Based on the above scenario, answer the following questions:

1. Because of limited number of resources, you only have two Windows 2008 servers available to use as the cluster node. Your team member wants to know the pre-requisites for setting up the failover cluster. What would you suggest?
2. How would you setup the shared folder as a quorum to reduce the chances of occurrence of a single point of system failure?
3. Initially, you were planning to use your company's storage area network (SAN) as a shared storage and a quorum. Your storage team has now informed you that the capacity of the SAN is full and it will take some time to expand its capacity. However, you must setup the cluster and release it to the application team to use before storage team can expand the storage capacity. What are the alternatives that you can consider?
4. Your team is planning to have a real-time copy of the data available at the backup site to increase the availability of the database. However, the data stored on the backup site should always be available. Also, the data would be used by the Internet application in case the production server fails or the primary site fails. What should you propose to achieve this?

Module Reviews and Takeaways



Review Questions

1. Your company is selecting an availability solution for SQL Server 2008. You have a limited budget and want to place a passive server offsite. Which availability solution should you use?

2. Your company is considering using failover clustering for SQL Server 2008. Which operating system should you use to build this failover clustering?

3. Your company requires a high-availability solution that provides automatic failover within the same site and support for automatic switch over to a disaster recovery site. Which solution should you use?

4. You are implementing database mirroring for replicating a database from a production to a disaster recovery site. During testing, you realize that the performance of the database is degraded. What should you check to determine the cause of the issue?

5. You have a replicated database in place and you want to introduce an availability solution to the production server. You want to reduce the downtime for implementing the solution. Which availability solution should you use?

Considerations for Implementing a High-Availability Solution

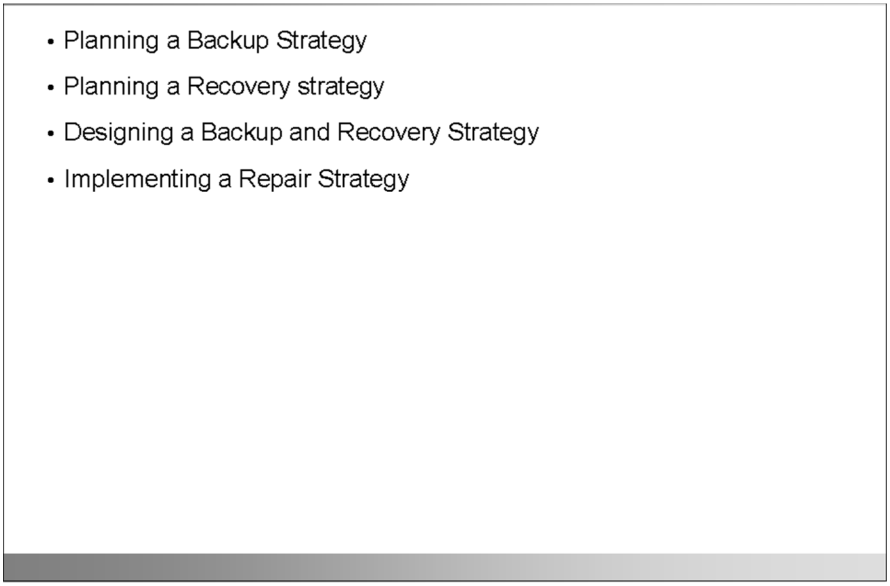
- When implementing a high-level solution, consider the following:
- Determine the availability requirements of the organization, such as permitted downtime and whether automatic failover is required.
- Determine the different high-availability solutions, such as database mirroring, failover clustering, log shipping, and replication.
- Ensure that the availability solutions are compatible with the solutions implemented.
- Consider using database mirroring over log shipping for new developments as log shipping may be removed in future versions of SQL Server.
- Ensure that SQL Server Agent is started for log shipping to work.
- Ensure that both the active and the passive server are in sync before you can consider database mirroring.
- Consider the network latency between servers if you are using the synchronous mode with transaction safety.
- Configure a witness server if you need automatic failover support.
- Use Windows 2008 as the platform to implement failover clustering to remove the possible single point of failure caused by quorum setup in Windows 2003.
- Consider using a geographically-dispersed cluster if you want to have a solution that can help to ensure services, if there is a site failure.

Module 12

Designing a Backup and Recovery Strategy for SQL Server 2008

Contents:

Module Overview



When you design an administrative solution for Microsoft™ SQL Server™ 2008, you also need to plan a backup and recovery strategy, which will restore the whole infrastructure when a disaster happens.

In this module, you will learn about the backup and recovery strategy in SQL Server 2008.

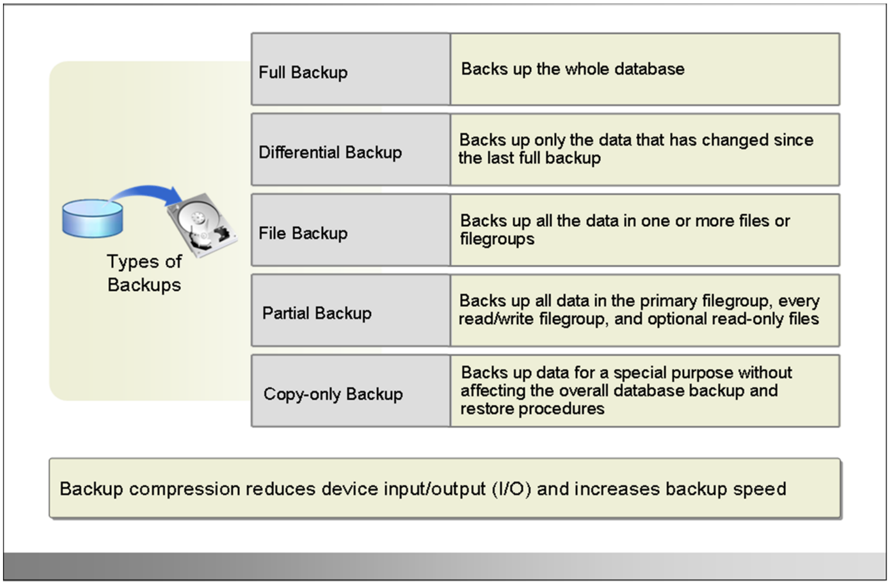
Lesson 1
Planning a Backup Strategy

- Types of Backups in SQL Server 2008
- Using Snapshot Backups in SQL Server 2008
- Transactional Log Backup
- What Is Marked Transaction?
- Backup Devices in SQL Server
- Backup Media in SQL Server

The SQL Server backup component provides an important safeguard for protecting critical data stored in SQL Server databases. A well-planned backup strategy helps to reduce the impact to system performance during backup and also helps to reduce the recovery time when failure happens.

In this lesson, you will learn about the various types of backup, backup devices, and backup media in SQL Server 2008.

Types of Backups in SQL Server 2008



Key Points

There are different types of backup and backup strategies supported in SQL Server 2008. You can back up a whole or partial SQL Server database or individual files or filegroups of the database. You cannot create table-level backups.

Full Backup

A full database backup backs up the whole database. This includes part of the transaction log so that the full database backup can be recovered. Full database backups represent the database at the time the backup finished.

Database backups are easy to use. A full database backup contains all the data in the database. For a small database that can be backed up quickly, the best practice is to use full database backups. However, as the database becomes larger, full backups take more time to finish and require more storage space. Therefore, for a large database, you might want to supplement full database backups with differential backups.

You can re-create a whole database in one step by restoring the database from a full database backup to any location. Some part of the transaction log is included in the backup, which helps you recover the database to the time when the last backup was finished. When the database is recovered, uncommitted transactions are rolled back. The restored database matches the state of the original database when the restored backup is finished.

The restore operation creates the database in the specified destination. If the database already exists in the specified location, the restore operation overwrites the existing database. To avoid overwriting an existing database, specify a different name for the restored database.

Differential Backup

A differential backup is based on the most recent full backup of the data that is included in the differential backup. A differential backup captures only the data that has changed since the full backup. This is known as the base of the differential. A differential backup includes only the data that has changed since the differential base. In SQL Server 2008, differential file backups can be very fast because the SQL Server Database Engine tracks the changes that were made since the differential base was created.

Each major type of file backup can serve as the base for a series of differential backups, such as:

- Differential database backups
- Differential partial backups
- Differential file backups

As a best practice, the scope of a differential backup should be the same as the scope of its base. Therefore, a differential file backup should be based on a file backup that covers the same set of files or filegroups, or both.

When using differential backup, evaluate the following considerations:

- **When taking a differential backup of a read-only database**. For read-only databases, full backups that are used alone are easier to manage than when they are used with differential backups.
- **When intermixing full and file backups, which creates multibase differentials**. You can create a multibase differential backup whose scope is larger than a single-base back up. However, multibase differential backups are complex to restore and should be avoided, except by expert users.
- **When taking a differential partial backup after changing the IsReadOnly property of a filegroup**. If only some of the data captured by a partial backup has changed, a differential partial backup is smaller than the base, and is faster to create. For a large database, taking differential backup facilitates making frequent backups of the data to decrease the risk of data loss.

File Backup

You can back up and restore individual files in a database. When you use file backups, the speed of recovery increases because you only need to restore damaged files, without restoring the rest of the database.

A full file backup backs up all the data in one or more files or filegroups. Under the full recovery model, a complete set of full file backups, together with enough log backups to span all the file backups, is the equivalent to a full database backup.

You can also back up an entire filegroup. In a BACKUP or RESTORE statement, you can specify a whole filegroup, instead of individually specifying each constituent file. Be aware that if any file of a filegroup is offline, the whole filegroup is offline and you cannot back up that filegroup.

File backups offer the following advantages over database backups:

- Recovery from isolated media failures is faster. The damaged file or files can be quickly restored.
- Flexibility in scheduling and media handling is increased when compared to full database backups, which can be unmanageable

for large databases. The increased flexibility of file or filegroup backups is also useful for large databases that contain data that has varying update characteristics.

File backups have the following disadvantages when compared to database backups:

- There is an additional administrative complexity in using file backups. A media failure can make a complete database unrecoverable if a damaged file lacks a backup. You must therefore maintain a complete set of file backups, and for the full or bulk-logged recovery model, one or more log backups covering minimally the interval between the first full file backup and last full file backup.
- It is time consuming to maintain and keep track of a complete set of file backups that might outweigh the space requirements of full database backups.
- Under the full recovery model, the number of log backups that you have to restore is reduced by using differential file backups.
- To maximize the advantages of using file backups, consider the layout of the data on disk and usage patterns.

Use the following guidelines for backing up files:

- Back up frequently modified data often.
- Back up infrequently modified data less often.
- Back up read-only data one time.

Only one file backup operation can occur at a time. You can back up multiple files in one operation, but this might extend the recovery time if you only have to restore a single file. This is because the whole backup is read to locate that file.

The file backup operation can be explained with the help of the following recovery models:

- **Simple Recovery Model**. Under the simple recovery model, you need to backup read/write files together. This makes sure that the database can be restored to a consistent point in time. Instead of individually specifying each read/write file or filegroup, use the READ_WRITE_FILEGROUPS option. This option backs up all the read/write filegroups in the database. A backup that is created by specifying READ_WRITE_FILEGROUPS is known as a partial backup.
- **Full Recovery Model**. Under the full recovery model, you must back up the transaction log, regardless of the rest of your backup strategy. A complete set of full file backups, together with enough log backups to span all the file backups from the start of the first file backup, is the equivalent of a full database backup.

Restoring a database using just file and log backups can be complex. Therefore, if it is possible, it is a best practice to perform a full database backup and start the log backups before the first file backup.

For both the database backup and file or filgroup backup, SQL Server 2008 supports the full backup, differential backup, partial backup, and copy only backups.

Partial Backup

Partial backups were introduced in SQL Server 2005. Partial backups are designed for use under the simple recovery model to improve flexibility for backing up very large databases that contain one or more read-only filegroups. However, partial backups work on all databases, regardless of the recovery model.

A partial backup resembles a full database backup, but a partial backup does not contain all the filegroups. Instead, a partial backup contains all the data in the primary filegroup, every read/write filegroup, and any optionally-specified read-only files. Partial backups are useful whenever you want to exclude read-only filegroups. A partial backup of a read-only database contains only the primary filegroup.

To create a partial backup, use the BACKUP statement and specify the READ_WRITE_FILEGROUPS option. You can also specify any read-only file or filegroup to include it in a partial backup. The following is the basic syntax for creating a partial backup is:

```
BACKUP DATABASE database_name READ_WRITE_FILEGROUPS [ ,
FILEGROUP = { logical_filegroup_name |
@logical_filegroup_name_var } [ ,...n ] ] TO
<backup_device>
```

Copy-Only Backup

A copy-only backup is a SQL Server backup that is independent of the sequence of conventional SQL Server backups. Usually, taking a backup changes the database and affects how later backups are restored. However, occasionally, it is useful to take a backup for a special purpose without affecting the overall backup and restore procedures for the database. For this purpose, copy-only backups were introduced SQL Server 2005.

The different types of copy-only backups are as follows:

- **Copy-only full backups (all recovery models)**. A copy-only full backup cannot serve as a differential base or differential backup and does not affect the differential base.
- **Copy-only log backups (full recovery model and bulk-logged recovery model only)**. A copy-only log backup preserves the existing log archive point, and therefore, does not affect the sequencing of regular log backups. Copy-only log backups are typically unnecessary. Instead, you can create another routine, current log backup, by using the WITH NORECOVERY option, and then use that backup together with all other previous log backups that are required for the restore sequence. However, a copy-only log backup can be created for performing an online restore.

The transaction log is never truncated after a copy-only backup. Copy-only backups are recorded in the is_copy_only column of the backupset table.

Backup Compression

A compressed backup is smaller than an uncompressed backup of the same data. Therefore, compressing a backup typically requires less device I/O, and therefore, usually increases the backup speed significantly.

By default, compression significantly increases CPU usage, and the additional CPU consumed by the compression process might adversely impact concurrent operations. Therefore, you might want to create low-priority compressed backups in a session whose CPU usage is limited by Resource Governor.

To calculate the compression ratio of a backup, use the values for the backup in the backup_size and compressed_backup_size columns of the backupset history table, as follows:

backup_size:compressed_backup_size

For example, a 3:1 compression ratio indicates that you are saving about 66 percent on disk space. To query on these columns, you can use the following Transact-SQL statement:

```
SELECT backup_size/compressed_backup_size FROM
msdb..backupset;
```

The following restrictions apply to compressed backups:

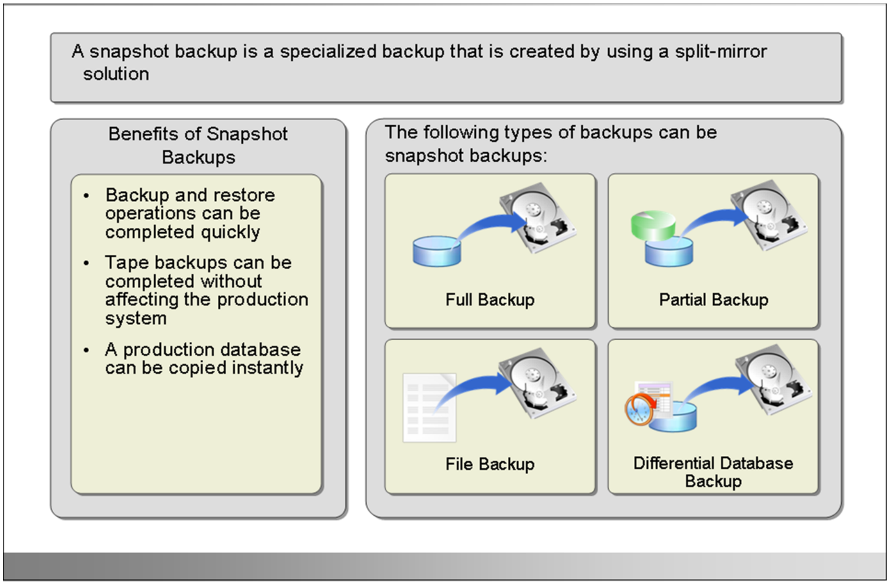
- Compressed and uncompressed backups cannot co-exist in a media set.
- Previous versions of SQL Server cannot read compressed backups.
- NTbackups cannot share a tape with compressed SQL Server backups.

Question: Why should you create a full/differential database backup after changing the IsReadOnly property of a filegroup?

Source:

- http://msdn.microsoft.com/en-us/library/ms186289.aspx
- http://msdn.microsoft.com/en-us/library/ms175526.aspx
- http://msdn.microsoft.com/en-us/library/ms190218.aspx
- http://msdn.microsoft.com/en-us/library/ms189860.aspx
- http://msdn.microsoft.com/en-us/library/ms191495%28SQL.90%29.aspx
- http://technet.microsoft.com/en-us/library/bb964719.aspx

Using Snapshot Backups in SQL Server 2008



Key Points

A snapshot backup is a specialized backup that is created almost instantaneously by using a split-mirror solution obtained from an independent hardware and software vendor. Snapshot backups minimize or eliminate the use of SQL Server resources to accomplish the backup. This is especially useful for moderate to very large databases in which availability is very important.

Benefits of Snapshot Backup

Snapshot backups have the following primary benefits:

- You can create a backup quickly with little or no effect on the server.
- You can perform a restore operation from a disk backup just as quickly.
- You can perform a backup to tape by another host without affceting on the production system.

- You can instantly create a copy of a production database for reporting or testing.

The instantaneous copying is typically accomplished by splitting a mirrored set of disks or by creating a copy of a disk block when it is written. This preserves the original disk. At restore time, the original disk is made available immediately and synchronization of the underlying disks occurs in the background. This synchronization results in almost instantaneous restore operations.

Volume Shadow Copy Service (VSS) in Windows® can also used by third-party backup software to provide better performance over backup.

Types of Backups

Only the following types of backups can be snapshot backups:

- Full backups
- Partial backups
- File backups
- Differential database backups. These are supported only when the vendor uses the VSS interface.

Snapshot backups are functionally equivalent to the corresponding conventional backups. You can use snapshot backups in restore sequences with non-snapshot full backups, differential backups, and log backups. Like other backups, snapshot backups are tracked in the msdb database, where snapshot backups are identified by backupset.is_snapshot = 1.

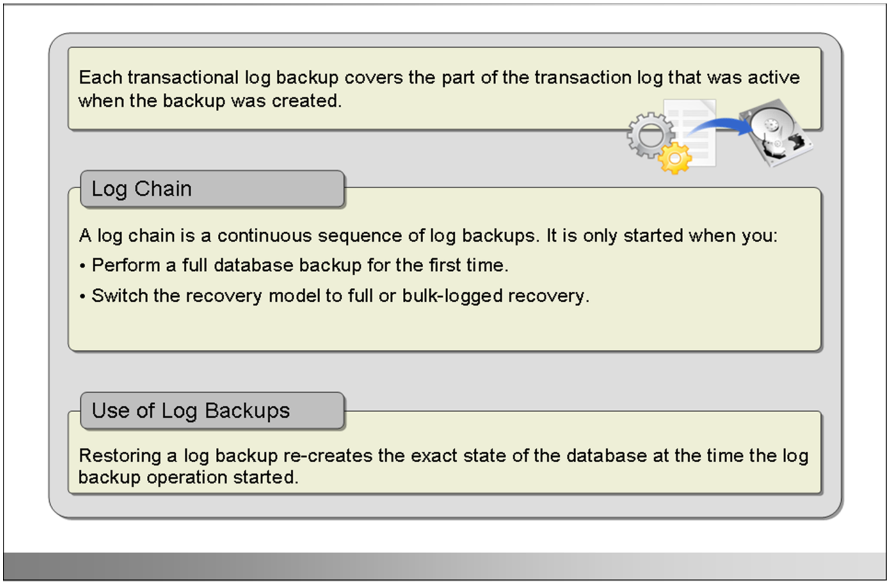
SQL Server does not support online restore from a snapshot backup. Restoring a snapshot backup takes the database offline automatically. A piecemeal restore can incorporate snapshot backups, but all the restore sequences are offline restores.

Question: What is the difference between a snapshot backup and a traditional backup?

Source:

- http://msdn.microsoft.com/en-us/library/ms189548.aspx

Transactional Log Backup



Key Points

You require regular transactional log backups or log backups under the full recovery model or bulk-logged recovery model. Each log backup covers the part of the transaction log that was active when the backup was created, and it includes all log records that were not backed up in a previous log backup. An uninterrupted sequence of log backups contains the complete log chain of the database, which is said to be unbroken. Under the full recovery model, and sometimes under the bulk-logged recovery model, an unbroken log chain lets you restore the database to any point in time.

Before you can create the first log backup, you must create a full backup, such as a database backup. Thereafter, backing up the transaction log regularly is necessary, not only to minimize work-loss exposure but also to enable truncation of the transaction log. Typically, the transaction log is truncated after every conventional log backup. However, log truncation can be delayed.

In SQL Server 2008, you can back up the log while any full backup is running.

The appropriate frequency for taking log backups depends on your tolerance for work-loss exposure, balanced by how many log backups you can store, manage, and potentially restore. Taking a log backup every 15 to 30 minutes might be enough. If your business requires that you minimize work-loss exposure, consider taking log backups more frequently. More frequent log backups have the added advantage of increasing the frequency of log truncation, resulting in smaller log files.

To limit the number of log backups that you need to restore, you must routinely back up your data. For example, you can schedule weekly full database backup and daily differential database backups.

Log Chain

A continuous sequence of log backups is called a *log chain*. A log chain starts with a full backup of the database. Usually, a new log chain is only started when the database is backed up for the first time or after the recovery model is switched from simple recovery to full or bulk-logged recovery.

Unless you choose to overwrite existing backup sets when creating a full database backup, the existing log chain remains intact. With the log chain intact, you can restore your database from any full database backup in the media set, followed by all subsequent log backups up through your recovery point. The recovery point could be the end of the last log backup or a specific recovery point in any of the log backups.

To restore a database up to the point of failure, the log chain must be intact. There must be an unbroken sequence of transaction log backups up to the point of failure. The point where this sequence of log must start depends on the type of data backups you are restoring: database, partial, or file. For a database or partial backup, the sequence of log backups must extend from the end of a database or partial backup. For a set of file backups, the sequence of log backups must extend from the start of a full set of file backups.

If you are using only file backups, you must back up the log from the beginning of the first full file backup. You can start taking log backups immediately after the first full file backup. This is because the first log backup can take a long time. While the log is being backed up, you back up the other files. To restore the database from only file backups, the set of full file backups must be augmented with one or more log backups that cover the interval between the first and last file backup.

Taking regular transaction log backups is necessary. In addition to letting you restore the backed-up transactions, a log backup truncates the log to remove the backed up log records from the log file. If you do not back up the log frequently enough, the log files can fill up.

Use of Log Backups

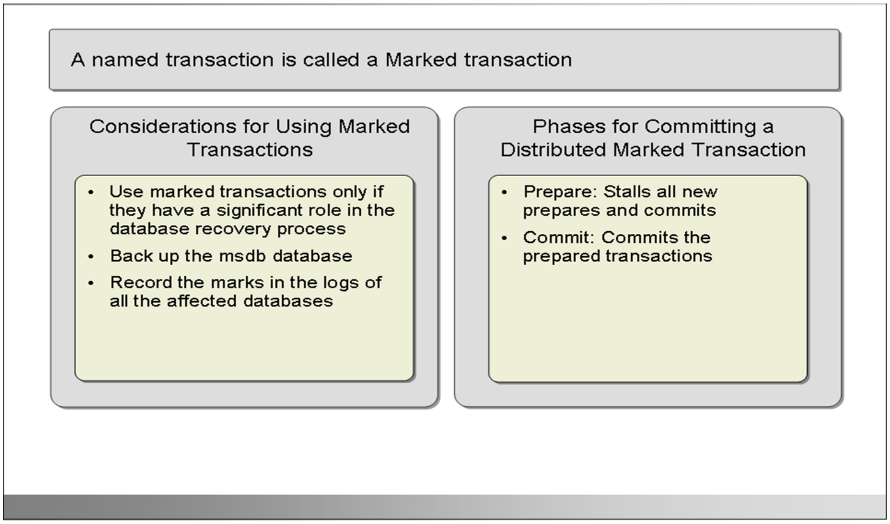
Restoring log backup rolls forward the changes that were recorded in the transaction log to re-create the exact state of the database at the time the log backup operation started. When you restore a database, you will have to restore the log backups that were created after the full database backup that you restore, or from the start of the first file backup that you restore. Typically, after you restore the most recent data or differential backup, you must restore a series of log backups until you reach your recovery point. Then, you recover the database. This rolls back all transactions that were incomplete when the recovery started and brings the database online. After the database has been recovered, you cannot restore any more backups.

Question: What should you do before you can create the first transaction log backup?

Source:

- http://technet.microsoft.com/en-us/library/ms175477.aspx
- http://msdn.microsoft.com/en-us/library/ms190440.aspx

What Is a Marked Transaction?



Key Points

An explicit transaction is called a marked transaction when used with marked option. A marked transaction is a transaction that is named, such as a time/date stamp. When you make related updates to two or more databases, related databases, you can use transaction marks to recover them to a logically consistent point. However, this recovery loses any transaction that is committed after the mark that was used as the recovery point. Marking transactions is suitable only when you are testing related databases or when you are willing to lose recently committed transactions.

Routinely marking related transactions in every related database establishes a series of common recovery points in the databases. The transaction marks are recorded in the transaction log and included in log backups. In the event of a disaster, you can restore each of the databases to the same transaction mark to recover them to a consistent point.

Note: You can create log backups on the different databases independently of each other.

Recovering related databases in the following scenarios requires that you have already marked transactions in every related database:

One or more transaction logs are destroyed. You have to restore the set of databases to a consistent state at the time of your last log backup.

You have to restore the entire set of databases to a mutually consistent state at some earlier point in time.

Important: You can recover related databases only to a marked transaction, not to a specific point in time.

A typical scenario for using marked transactions includes the following steps:

1. Create a full or differential database backup of each of the related databases.
2. Mark a transaction block in all the databases.
3. Back up the transaction log for all the databases.
4. Restore database backups WITH NORECOVERY.
5. Restore logs WITH STOPATMARK.

Considerations for Using Marked Transactions

Before inserting named marks into the transaction log, consider the following:

- Because transaction marks consume log space, use them only for transactions that play a significant role in the database recovery strategy.
- Back up the msdb database because after a marked transaction commits, a row is inserted in the logmarkhistory table in msdb.
- If a marked transaction spans multiple databases on the same database server or on different servers, the marks must be recorded in the logs of all the affected databases.

Phases for Committing a Distributed Marked Transaction

Committing a distributed transaction occurs in two phases: prepare and commit. When a marked transaction is committed, the commit log record for each database in the marked transaction is placed in the log at a point where there are no in-doubt transactions in any of the logs. At this point, it is guaranteed that there are no transactions that appear as committed in one log, but not committed in another log.

The following steps accomplish this during the commit of a marked transaction:

1. Prepare phase of a marking transaction stalls all new prepares and commits.
2. Only commits of already prepared transactions are allowed to continue.
3. Marking transaction then waits for all prepared transactions to drain (with time-out).
4. The marked transaction is prepared and committed.
5. The stall of new prepares and commits is removed.
6. The stalls generated by marked transactions that span multiple databases can reduce the transaction processing performance of the server.

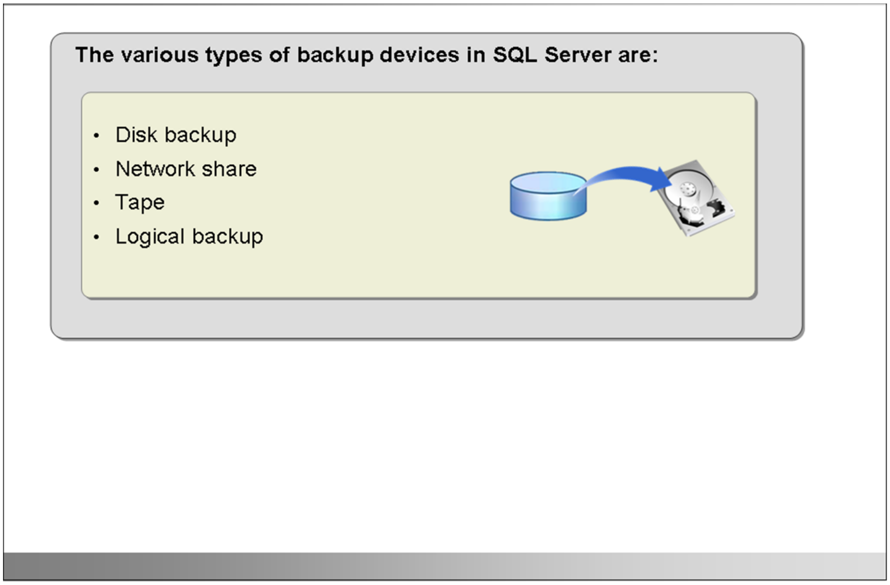
You should not run concurrent marked transactions. It is rare but possible that if you commit a distributed marked transaction with other distributed marked transactions at the same time, a deadlock might occur. When this happens, the marking transaction is chosen as the deadlock victim and is rolled back. When this error occurs, the application can retry the marked transaction. When multiple marked transactions try to commit concurrently, there is a higher probability of a deadlock state.

Question: What is the purpose of creating a marked transaction?

Source:

- http://msdn.microsoft.com/en-us/library/ms187014.aspx

Backup Devices in SQL Server



Key Points

During a backup operation, the data being backed up is written to a physical backup device. A physical backup device is either a tape drive or a disk file that is provided by the operating system. A backup can be written into a maximum of 64 devices. If a backup requires multiple backup devices, the devices must all correspond to a single type of device (disk or tape).

Backup Devices

The various backup devices available in SQL Server 2008 are:

- **Disk Backup**. A disk backup device is a hard disk or other disk storage media that contains one or more backup files. If a disk file fills while a backup operation is appending a backup to the media set, the backup operation fails. You can determine the maximum size of a backup file by the free disk space available on the disk device; therefore, the appropriate size for a backup disk device depends on the size of your backups. A disk backup device could be a simple disk device, such as an ATA drive. Alternatively, you could use a hot-swappable disk drive that would let you transparently replace a full disk on the drive with an empty disk. A backup disk can be a local disk on the server or a remote disk that is a shared network resource.

A backup file is a regular operating system file. If a disk file fills while a backup operation is appending a backup to the media set, the backup operation fails. You can determine the maximum size of a backup file by the free disk space available on the disk device; therefore, the appropriate size for a backup disk device depends on the size of your backups. When you are specifying a backup file, you should enter its full path and file name. If you specify only the file name or a relative path when you are backing up to a file, the backup file is put in the default backup directory. The default backup directory is C:\Program Files\Microsoft SQL Server\MSSQL.n\MSSQL\Backup, where n is the number of the server instance. Therefore, for the default server instance, the default backup directory is C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Backup.

To avoid ambiguity, especially in scripts, we recommend that you explicitly specify the path of the backup directory in every DISK clause.

SQL Server management tools are very flexible at handling disk backup devices because they automatically generate a time-stamped name on the disk file.

To specify a physical disk device in a BACKUP statement, following is the basic syntax:

```
BACKUP { DATABASE | LOG } database_name

   FROM DISK = { 'physical_backup_device_name' |
   @physical_backup_device_name_var }
```

For example,

```
BACKUP DATABASE QuantamCorp2008

   FROM DISK = 'Z:\SQLServerBackups\QuantamCorp.bak';
```

- **Network Share**. For SQL Server to access a remote disk file, the SQL Server service account must have access to the network share. This includes having the permissions needed for backup operations to write to the network share and for restore operations to read from it. The availability of network drives and permissions depends on the context is which SQL Server service is running:

To back up to a network drive when SQL Server is running in a domain user account, you need to map the shared drive as a network drive in the session where SQL Server is running. If you start Sqlservr.exe from command line, SQL Server sees any network drives you have mapped in your logon session.

You can connect with the network service account by using the computer account instead of a domain user. To enable backups from specific computers to a shared drive, grant access to the computer accounts. As long as the Sqlservr.exe process that is writing the backup has access, it is irrelevant whether the user sending the BACKUP command has access.

To specify a network share in a backup or restore command, you should use the fully qualified Universal Naming Convention (UNC) Name of the file for the backup device. A UNC name has the form *Systemname**ShareName**Path**FileName*.

For example,

```
BACKUP DATABASE QuantamCorp2008

   TO DISK =
   '\\BackupSystem\BackupDisk1\AW_backups\QuantamCorpData.
   Bak';

GO
```

- **Tape**. Backing up SQL Server data to tape requires that the tape drive or drives be supported by the Microsoft Windows operating system.

When a tape drive is used, a backup operation may fill one tape and continue onto another tape. Each tape contains a media header. The first media used is called the *initial tape*. Each successive tape is known as a *continuation tape* and has a media sequence number that is one higher than the previous tape. For example, a media set associated with four tape devices contains at least four initial tapes (and, if the database does not fit, four series of continuation tapes). When appending a backup set, you must mount the last tape in the series. If the last tape is not mounted, the Database Engine scans forward to the end of the mounted tape and then requires that you change the tape. At that point, mount the last tape.

You can use tape backup devices, such as disk devices, with a few exceptions. The tape device must be connected physically to the computer that is running an instance of SQL Server. Backing up to remote tape devices is not supported. If a tape backup device is filled during the backup operation, but more data must still be written, SQL Server prompts for a new tape and continues the backup operation after a new tape is loaded.

- **Logical Backup**. A logical backup device is an optional, user-defined name that points to a specific physical backup device (a disk file or tape drive). A logical backup device lets you use indirection when referencing the corresponding physical backup device.

When you define a logical backup device, you need to assign a logical name to a physical device. For example, you can define a logical device, QuantamCorpBackups, to point to the Z:\SQLServerBackups\QuantamCorp.bak file or the \\.\tape0 tape drive. Backup and restore commands can then specify QuantamCorpBackups as the backup device, instead of DISK = 'Z:\SQLServerBackups\QuantamCorp.bak' or TAPE = '\\.\tape0'.

You need to assign a unique logical device name among all the logical backup devices on the server instance. To view the existing logical device names, query the sys.backup_devices catalog view. This view displays the name of each logical backup device and describes the type and physical file name or path of the corresponding physical backup device.

After a logical backup device is defined in a BACKUP or RESTORE command, you can specify the logical backup device, instead of the physical name of the device.

For example, the following statement backs up the QuantamCorp2008 database to the QuantamCorpBackups logical backup device.

```
BACKUP DATABASE QuantamCorp2008
    TO QuantamCorpBackups;
```

One advantage of using a logical backup device is that it is simple to use than a long path. Using a logical backup device can help if you plan to write a series of backups to the same path or to a tape device. Logical backup devices are especially useful for identifying tape backup devices.

Source:

- http://msdn.microsoft.com/en-us/library/ms179313.aspx

Backup Media in SQL Server

Backup Media	Description
Media Sets	The backups on a set of one or more backup media compose a single media set
Media Families	Backups created on a single nonmirrored device or a set of mirrored devices in a media set constitute a media family
Backup Sets	The backup set is described in terms of the media set to which the backup belongs

Key Points

The backup media is the physical storage used by the backup device to store the database, transaction log, or file backup. It can be either a file or a tape.

Media Sets

The backups on a set of one or more backup media compose a single media set. A media set is an ordered collection of backup media, tapes, or disk files to which one or more backup operations have written by using a fixed type and number of backup devices. A given media set uses either tape drives or disk drives, but not both. For example, the backup devices associated with a media set might be three tape drives named \\.\TAPE0, \\.\TAPE1, and \\.\TAPE2. That media set contains only tapes, starting with a minimum of three tapes (one per drive). The type and number of backup devices are established when a media set is created, and they cannot be changed. However, if necessary, between backup and restore operations, a given device can be replaced with a device of the same type.

A media set is created on the backup media during a backup operation by formatting the backup media. After formatting, each file or tape contains a media header for the media set and is ready to receive backup content. With the header in place, the backup operation proceeds to back up the specified data to the backup media on all backup devices specified for the operation.

Media Families

Backups created on a single nonmirrored device or a set of mirrored devices in a media set constitute a media family. The number of backup devices used for the media set determines the number of media families in a media set. For example, if a media set uses two nonmirrored backup devices, the media set contains two media families.

Each tape or disk in a media family is assigned a media sequence number. The media sequence number of a disk is always 1. In a tape media family, the sequence number of the initial tape is 1, the sequence number of the second tape is 2, and so forth.

Backup Sets

A successful backup operation adds a single backup set to the media set. The backup set is described in terms of the media set to which the backup belongs. If the backup media consists of only one media family, that family contains the entire backup set. If the backup media consists of multiple media families, the backup set is distributed among them. On each medium, the backup set contains a header that describes the backup set. Typically, after a media set is created, subsequent backup operations, one after another, append their backup sets to the media set. All of the media used by a backup set make up the media set, regardless of the number of media or backup devices involved. Backup sets are sequentially numbered by their position in the media set, allowing you to specify which backup set to restore.

Every backup operation to a media set must write to the same number and type of backup devices. With multiple devices, as with the first backup set, the content of every subsequent backup set is distributed among the backup media on all of the devices.

Source:

- http://msdn.microsoft.com/en-us/library/ms178062.aspx

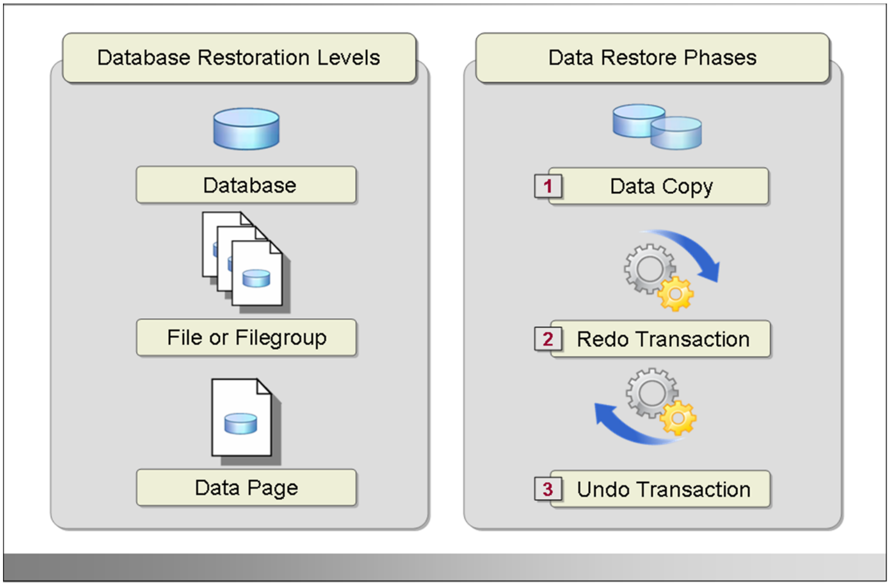
Lesson 2
Planning a Recovery Strategy

- Restoring a Database
- What Are Recovery Models?
- Considerations for Restoring Transaction Logs
- Process of Restoring a Transaction Log
- How To Restore a Database by Using File Backups
- How To Perform an Online Restore

If you create backup copies for the database, you need a well-defined recovery plan and recovery strategy to recover the system. You need to plan the recovery of a system carefully to ensure that you can recover a system within the shortest time. You may plan for the need to recover a system as part of the system itself while the database is still online. This planning helps to minimize the chances of downtime of the system.

In this lesson, you will learn how to restore a database, and the various recovery models available in SQL Server 2008. You will also learn to restore a transaction log and a database by using file backups. Finally, you will learn how to perform an online restore.

Restoring a Database



Key Points

Restoring a database is the process of copying data from a backup, applying logged transactions to the data, and rolling it forward to the target recovery point. A data or differential backup contains sufficient transaction log records to allow rolling forward the active transactions as part of restoring each backup. Each backup also contains sufficient log to roll back uncommitted transactions and bring the database into a transactionally consistent and usable state. The process of rolling forward uncommitted transactions, if any, and bringing the database online is known as recovery.

Data Restoration Levels

SQL Server 2008 supports restoring data at the following levels:

- **Database or a complete database restore**. The whole database is restored and recovered, and the database is offline for the duration of the restore and recovery operations.
- **File or filegroup restore**. A data file or a set of files is restored and recovered. During a file restore, the filegroups that contain

the files are automatically offline for the duration of the restore. Any attempt to access an offline filegroup causes an error.

- **Data page or a page restore**. Under the full recovery model or bulk-logged recovery model, you can restore individual data pages. You can perform page restores on any database, regardless of the number of filegroups.

Data Restore Phases

You can implement each restore scenario by using one or more restore operations, called a *restore sequence*. Each operation corresponds to an individual Transact-SQL RESTORE statement. A restore sequence moves affected data through one or more phases of restore.

Restoring a database is a multiphase process.

The possible phases of a restore scenario are:

1. **Data copy phase**. The data copy phase initializes the contents of the database, files, or pages being restored. This phase is performed by restore database, restore file, and restore page operations by using full or differential backups. The data copy phase involves copying data from one or more full backups or differential backups, and then resetting the contents of the affected database, files, or pages to the time that they were captured by those backups.

2. **Roll forward or redo transaction**. Roll forward is the process of redoing logged changes to the data in the roll forward set to bring the data forward in time. To accomplish roll forward, the SQL Server Database Engine processes log backups as they are restored, starting with the log that is contained in full backups. Usually, if data was read-only when it was backed up and has remained read-only, roll forward is unnecessary and can be skipped.

The goal of roll forward is to return the data to its original state at the recovery point. The recovery point is the point to which the user specifies the set of data to be recovered. Under the full recovery model, you can specify the recovery point as a particular point in time, a marked transaction, or a log sequence number. Under the bulk-logged recovery model, you can restore to a point in time only if no bulk operations have been performed since the previous log backup.

In the roll forward phase, data is always rolled forward to a point that is consistent with the state of the database at the recovery point.

If the primary file is being restored, the recovery point determines the state of the whole database. For example, if a database is being recovered to a point in time just before a table was accidentally dropped, the whole database must be restored to the same point in time. If you are not restoring a primary file, you need to know the database state and roll forward the restored data to a recovery point that is consistent with the database.

However, the database might contain changes made by transactions that are uncommitted at the recovery point. For online restore, you can recover data to a point in time consistent with the current state of the online part of the database.

A differential backup skips forward to when the differential backup was taken. You can overwrite pages in the roll forward set with any more recent ones from the differential backup.

After the redo or roll forward phase has rolled forward all the log transactions, a database usually contains changes made by transactions that are uncommitted at the recovery point. This makes the rolled forward data inconsistent. The recovery process opens the transaction log to identify uncommitted transactions. Uncommitted transactions are undone by being rolled back, unless they hold locks that prevent other transactions from viewing inconsistent data. This is the undo or the roll back phase. If the data is already consistent at the start of the recovery process, the undo phase is skipped. After the database is consistent, recovery brings the database online.

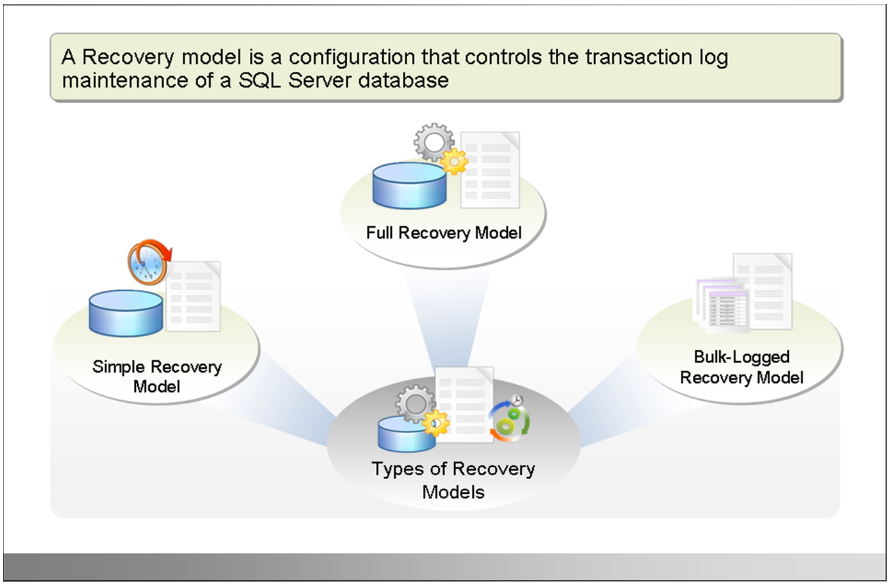
After one or more backups have been restored, recoveries usually include both the redo and undo phases. Every full and differential backup contains enough transaction log records to allow for the backup data to be recovered to a self-consistent state.

3. **Roll back or undo transaction**. After the redo phase has rolled forward all the log transactions, a database typically contains changes made by transactions that are uncommitted at the recovery point. This makes the rolled forward data transactionally inconsistent. The recovery process opens the transaction log to identify uncommitted transactions. The uncommitted transactions are undone by being rolled back, unless they hold locks that prevent other transactions from viewing transactionally inconsistent data. This step is called the undo or roll back phase.

Source:

- http://msdn.microsoft.com/en-us/library/ms191455%28SQL.90%29.aspx
- http://msdn.microsoft.com/en-us/library/ms191455%28SQL.90%29.aspx
- http://msdn.microsoft.com/en-us/library/ms191455.aspx
- http://technet.microsoft.com/en-us/library/bb795689.aspx

What Are Recovery Models?



Key Points

A recovery model is a configuration in SQL Server database that is designed to control transaction log maintenance of a SQL Server database. There are three types of recovery models available in SQL Server—simple, full, and bulk-logged. A database usually uses the full recovery model or simple recovery model.

Following are the types of recovery models:

- **Simple Recovery Model**. In a simple recovery model, the database is backed up at regular intervals. In this model, the transaction log is not backed up. The simple recovery model minimizes the administrative overhead for the transaction log because the transaction log is not backed up. The simple recovery model risks significant work-loss exposure if the database is damaged. Data is recoverable only to the most recent backup of the lost data. Therefore, under the simple recovery model, the backup intervals should be short enough to prevent the loss of significant amounts of data. However, the intervals should be sufficiently long to keep the backup overhead from affecting production work. Generally, for a user database, the simple recovery model is useful for test and development databases or for databases containing mostly read-only data, such as a data warehouse. The simple recovery model is inappropriate for production systems where loss of recent changes is unacceptable.

The simple recovery model provides the simplest form of backup and restore. This recovery model supports both database backups and file backups, but does not support log backups. Backup is easy to manage because the transaction log is never backed up. However, if there are no log backups, a database can be restored only to the end of the most recent backup of the data. If a failure were to occur, updates that are made after the most recent backup of the data are lost.

Under the simple recovery model, work-loss exposure increases over time until the next full or differential backup is taken. In contrast to a full backup, a differential backup includes only the changes made since the previous full backup. Therefore, you can schedule backups frequently enough to avoid losing data without your backups becoming unmanageable.

- **Full Recovery Model**. In this model, both the database and the transaction logs are backed up. This model logs all transactions and retains the transaction log records until they are backed up. The full recovery model allows a database to be recovered to the point of failure, assuming that the tail of the log can be backed up after the failure. This model also supports restoring individual data pages.

The full recovery model uses log backups to prevent data loss in the broadest range of failure scenarios. The model also uses log backups for backing and restoring the transaction log. The advantage of using log backups is that they let you restore a database to any point of time that is contained within a log backup. You can use a series of log backups to roll a database forward to any point in time that is contained in one of the log backups. To minimize your restore time, you can supplement each full backup with a series of differential backups of the same data.

If you can back up the active log after a disaster occurs; you can restore the database up to the point of failure without data loss. The disadvantages of using log backups are that they require storage space and increase restore time and complexity.

For a database that regularly uses the full recovery model, you can optimize certain bulk operations by temporarily using the bulk-logged recovery model. The bulk-logged recovery model incurs several restrictions that make it not unsuitable for day-to-day operations.

After the first full database backup is completed and regular log backups start, the potential work-loss exposure is narrowed to the time between when the database is damaged and the most recent regular log backup. Therefore, you should take log backups frequently enough to keep your work-loss exposure within the confines required by your business.

- **Bulk-Logged Recovery Model**. This recovery model bulk-logs most bulk operations. It is intended solely as an adjunct to the full recovery model. For certain large-scale bulk operations, such as bulk import or index creation, switching temporarily to the bulk-logged recovery model increases performance and reduces log space consumption. Log backups are still required. Similar to the full recovery model, the bulk-logged recovery model retains transaction log records until after they are backed up. The tradeoffs are bigger log backups and increased work-loss exposure because the bulk-logged recovery model does not support point-in-time recovery.

The bulk-logged recovery model is a special-purpose recovery model that you should use only intermittently to improve the performance of certain large-scale bulk operations.

You should minimize your use of the bulk-logged recovery model. The best practice is to switch to the bulk-logged recovery model right before a set of bulk operations, perform the operations, and then switch back to the full recovery model.

Compared to the full recovery model, which logs all transactions completely, the bulk-logged recovery model minimally logs bulk operations, although fully logging other transactions. The bulk-logged recovery model protects against media failure, and for bulk operations, provides the best performance and least log space usage.

However, the bulk-logged recovery model increases the risk of data loss for these bulk-copy operations, because bulk logging operations prevents recapturing changes on a transaction-by-transaction basis. If a log backup contains any bulk-logged operations, you cannot restore to a point-in-time within that log backup; you can restore only the whole log backup.

Under the bulk-logged recovery model, if a log backup covers any bulk operations, the log backup contains both log records and the data pages that were changed by bulk operations. This is necessary to capture the results of the bulk-logged operations. The incorporated data extents can make a log backup very large. Additionally, backing up the log requires access to the data files that contain the bulk-logged transactions. If any affected database file is inaccessible, the transaction log cannot be backed up and all operations committed in that log are lost.

To track the data pages, a log backup operation relies on a bulk-changes bitmap page that contains a bit for every extent. For each extent updated by a bulk-logged operation since the last log backup, the bit is set to 1 in the bitmap. The data extents are copied into the log followed by the log data.

The bulk-logged recovery model has the following backup restrictions:

- If a filegroup that contains bulk-logged changes is made read-only before a log backup is performed, all subsequent log backups contain the extents changed by the bulk-logged operations. Such log backups are larger and take longer to complete than under the full recovery model.
  To avoid this situation, before you make the filegroup read-only, you need to switch the database to the full recovery model and back up the log. You should then make the filegroup read-only.

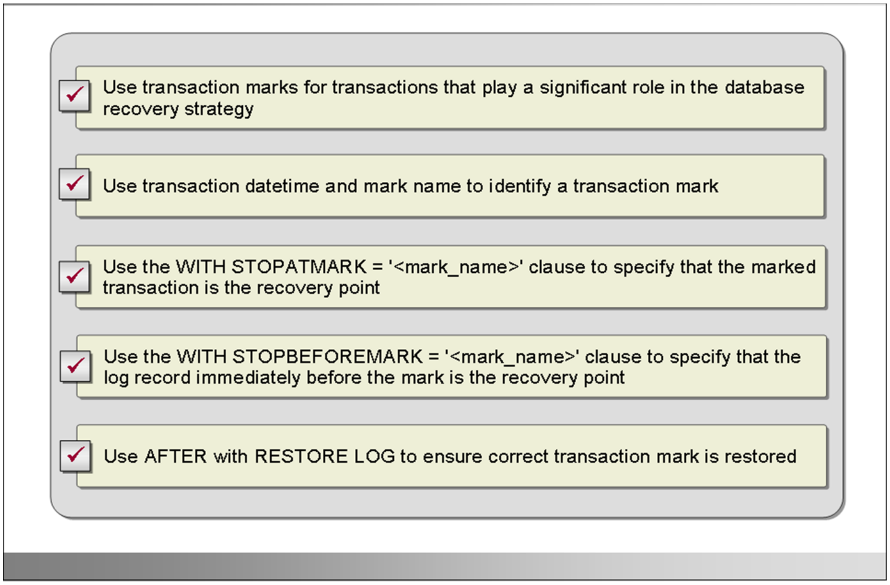
- If bulk operations were performed since the last log backup, bulk changes exist in the database. In this case, all files must be either online or defunct when the log backups are performed. This is because backing up a log that contains bulk-logged operations requires access to the data files that contain the bulk-logged transactions.

Question: When should you consider using a simple recovery model?

Source:

- http://technet.microsoft.com/en-us/library/ms190217.aspx
- http://msdn.microsoft.com/en-us/library/ms190692.aspx

Considerations for Restoring Transaction Logs



Key Points

Using SQL Server, you can insert named marks into the transaction log to allow recovery to that specific mark. Log marks are transaction specific and are inserted only if their associated transaction commits. As a result, marks can be tied to specific work, and you can recover to a point that includes or excludes this work.

When using marked transactions, you need to consider the following:

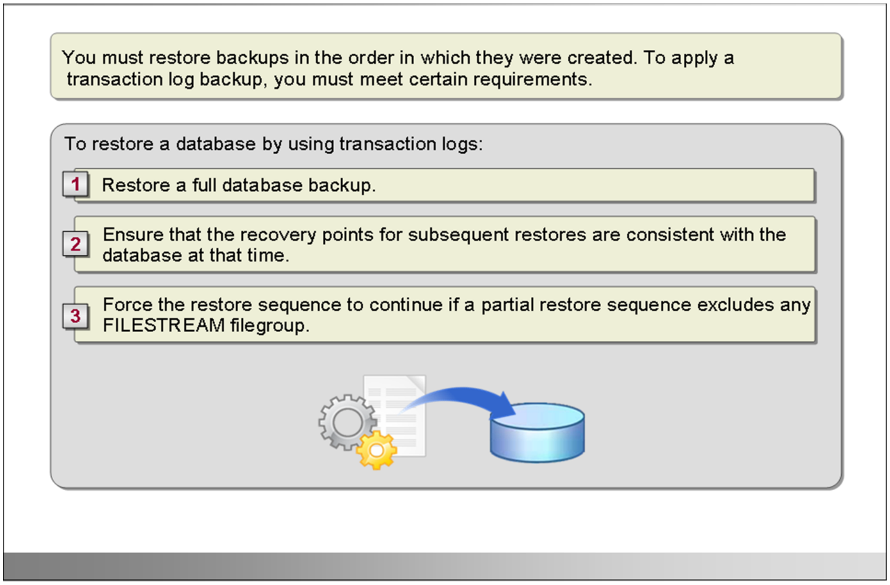
- **Use transaction marks for transactions that play a significant role in the database recovery strategy**. After a marked transaction commits, a row is inserted in the logmarkhistory table in msdb. The entries are stored permenantly and would not be cleared. The more frequent use of marked transaction, the more space is required for msdb database. Therefore, you need to use them only for transactions that play a significant role in the database recovery strategy. Mark transactions on all database/servers included in the transaction. If a marked transaction spans multiple databases on the same database server or on different servers, the marks must be recorded in the logs of all the affected databases. This is to ensure all database would be restored to the same consistent point when restoring to the transaction mark.

- **Use transaction datetime and mark name to identify a transaction mark**. The transaction log records the mark name, description, database, user, datetime information, and the log sequence number (LSN). The mark is named the same as the transaction. The optional description is a textual description of the mark, not the mark name. You should use the datetime information with the mark name to uniquely identify the mark.
- **Use the WITH STOPATMARK = '<mark_name>' clause to specify that the marked transaction is the recovery point**. STOPATMARK rolls forward to the mark and includes the marked transaction in the roll forward.
- **Use the WITH STOPBEFOREMARK = '<mark_name>' clause to specify that the log record immediately before the mark is the recovery point**. STOPBEFOREMARK rolls forward to the mark and excludes marked the transaction from the roll forward.
- **Use AFTER with RESTORE LOG to ensure correct transaction mark is restored**. If AFTER datetime is omitted, roll forward stops at the first mark that has the specified name. If AFTER datetime is specified, roll forward stops at the first mark that has the specified name, exactly at or after datetime.

Source:

- http://technet.microsoft.com/en-us/library/ms188623.aspx

Process of Restoring a Transaction Log



Key Points

You must restore backups in the order in which they were created because the transaction log only contains data changes after the previous full or transaction log backup.

Before you can restore a particular transaction log backup, you must first restore the following previous backups without rolling back uncommitted transactions:

- The full database backup and the last differential backup, if any, taken before the particular transaction log backup.
- All transaction log backups taken after the full database backup or the differential backup, and before the particular transaction log backup.

To apply a transaction log backup, the following requirements must be met:

- You must restore the previous full database backup or differential database.
- You must restore all transaction logs that are created after the full or differential database backup in chronological order. If a transaction log backup in this log chain is lost or damaged, you can restore only the transaction logs before the missing transaction log.
- If you recover a database after restoring one of the intermediate transaction log backups, that is, before the end of the log chain, you cannot restore the database beyond the failure without restarting the complete restore sequence.
- In certain instances, you might have to restore and recover a database to a particular point in time, mark, or LSN before the point of failure. In such instances, you can use a transaction log to restore the database to a specific target recovery point.

For example, the following illustration shows a restore to a recovery point in the middle of a transaction log that was taken at time t9. Changes in the remainder of this backup and the subsequent log backup taken at time t10 are discarded.

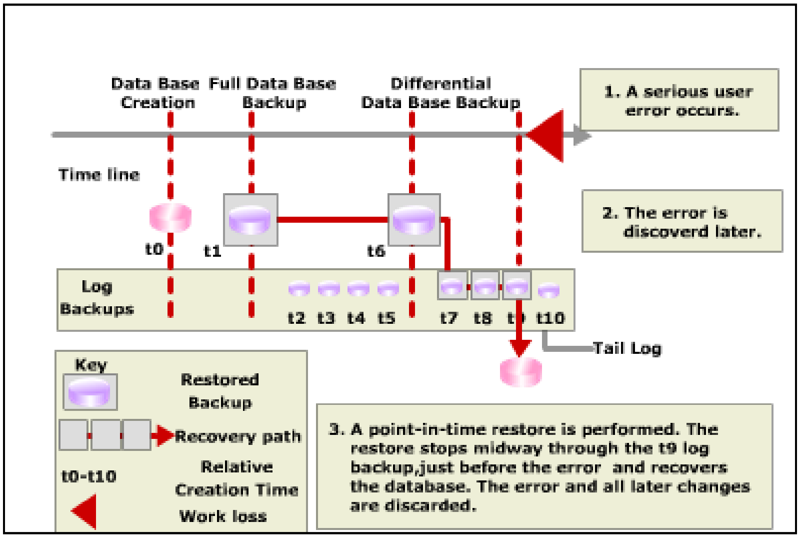


Figure 12.1. Restore to a recovery point

In this case, you can specify the target recovery point by using one of the following:

- A specific point in time within a transaction log
- A named mark that has been inserted into a transaction log record
- An LSN

A specified time or a transaction is always restored from a log backup. Therefore, the target recovery point must be contained in a transaction log backup. To restore a database to a specific point in time or transaction, you need to specify the target recovery point in a STOPAT, STOPATMARK, or STOPBEFOREMARK clause. In every RESTORE LOG statement of the restore sequence, you should specify your target time or transaction in an identical STOPAT, STOPATMARK, or STOPBEFOREMARK clause. When you apply the log backup that contains the recovery point, you can recover only transactions that come before that point.

Restoring a Database by Using Transaction Logs

You can perform the following steps to restore a database by using transaction logs:

1. You must restore a full database backup, whose endpoint is earlier than the target recovery point.
2. When recovery is complete, the time of the database is determined by the recovery point to which you recovered the primary file. Subsequent restores, if any, must have recovery points that are consistent with the database at that time.
3. If a partial restore sequence excludes any FILESTREAM filegroup, point-in-time restore is not supported. You can force the restore sequence to continue. However the FILESTREAM filegroups that are omitted from your RESTORE statement can never be restored.

Before restoring a transaction log, you should:

1. Perform a transaction log backup before performing any recovery actions.
2. Perform full database recovery and transaction log recovery.
3. Restore the transaction log backup until T9.
4. Restore the tail log backup performed in step 1 using point-in-time restore.

Source:

- http://msdn.microsoft.com/en-us/library/ms177446.aspx
- http://msdn.microsoft.com/en-us/library/ms189596.aspx
- http://msdn.microsoft.com/en-us/library/ms190244.aspx

Demonstration: How To Restore a Database by Using File Backups

In this demonstration, you will see how to:

Restore a database by using file backups

Key Points

The steps to restore a database by using file backups are:

1. On the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. In the Microsoft SQL Server Management Studio window, by using the **Connect to Server** dialog box, connect to the default instance of SQL Server Database Engine.
3. In Windows Explorer, browse to the **D:\Demofiles\Mod12** folder, and then double-click the **M12_Demo3.sql** file and then click Execute.
4. In Windows Explorer, browse to the **D:\Demofiles\Mod012** folder, and then double-click the **M12_Demo1.sql** file.

5. In the Microsoft SQL Server Management Studio window, select the **Set the database recovery model to FULL and create a backup of a database** code, and then click **Execute**.
6. In the Microsoft SQL Server Management Studio window, select the **Create a backup of the transaction log** code, and then click **Execute**.
7. In the Microsoft SQL Server Management Studio window, select the **Restore the database backup with the NORECOVERY option** code, and then click **Execute**.
8. In the Microsoft SQL Server Management Studio window, select the **Restore the transaction log backup** code, and then click **Execute**.

Note: Enter the current date, but for the next month. For instance, if today is 2010-04-08, then you should enter 2010-05-08 in the script under RESTORE LOG QuantamCorp.

9. In the Microsoft SQL Server Management Studio window, select the **Restore the database backup with the RECOVERY option** code, and then click **Execute**.

Question: What should you consider before restoring a database?

Question: Why should you perform a tail-log backup before restoring a database?

Demonstration: How To Perform an Online Restore

In this demonstration, you will see how to:

Perform an online restore

Key Points

The steps to perform an online restore are:

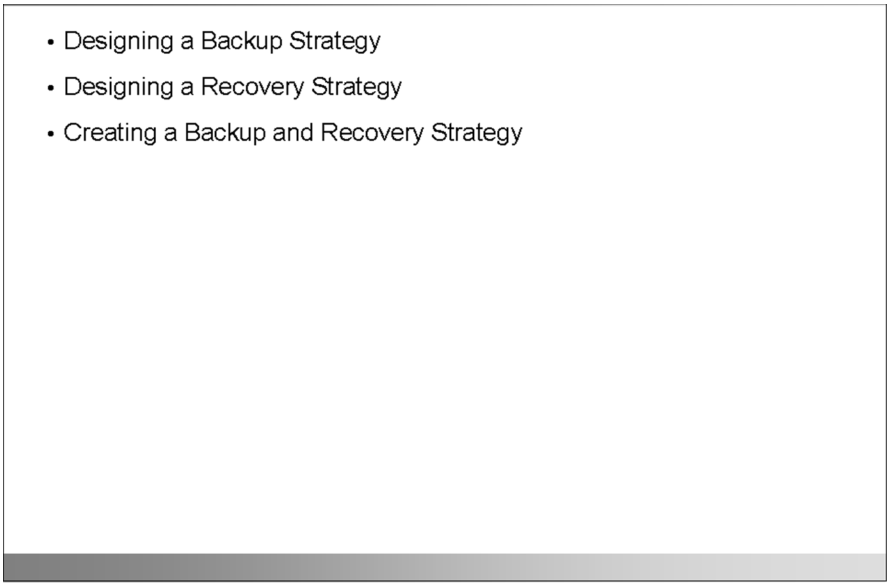
1. In Windows Explorer, browse to the **D:\Demofiles\Mod12** folder, and then double-click the **M12_Demo2.sql** file.
2. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, select the **Add a file and a filegroup to the database and set the filegroup with default settings** code, and then click **Execute**.
3. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, select the **Create a backup of the Primary and the Data filegroups of the database** code, and then click **Execute**.
4. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, select the **Create a transaction log backup with the FORMAT option** code, and then click **Execute**.

5. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, select the **Modify the database content** code, and then click **Execute**.
6. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, select the **Create a tail log backup with the NORECOVERY and the NO_TRUNCATE options** code, and then click **Execute**.
7. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, select the **Restore a primary filegroup of a database with the FILEGROUP, the PARTIAL and the NORECOVERY options** code, and then click **Execute**.
8. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, select the **Restore the transaction log file with the NORECOVERY option** code, and then click **Execute**.
9. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, select the **Restore the tail log file with the RECOVERY option** code, and then click **Execute**.
10. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, select the **Retrieve records from a system table and a user-defined table** code, and then click **Execute**.
11. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, select the **Restore a file and the transaction log file with the NORECOVERY option** code, and then click **Execute**.
12. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, select the **Restore the tail log backup with the RECOVERY option** code, and then click **Execute**.

Question: What is the benefit of performing an online restore?

Question: Which data will users not be able to access during the online restore?

Lesson 3
Designing a Backup and Recovery Strategy



SQL Server enables you to back up and restore your databases. The SQL Server backup and restore component provides an important safeguard for protecting critical data stored in SQL Server databases. A well-planned backup and restore strategy helps protect databases against data loss caused by a variety of failures. You must test your strategy by restoring a set of backups and then recover your database to prepare to respond effectively to a disaster. A copy of data that can be used to restore and recover the data is called a backup. Backups let you restore data after a failure. With good backups, you can recover from many failure scenarios.

In this lesson, you will learn about available backup and recovery options and when they would be best applied. In addition, you will also learn the steps for creating a backup and restore strategy.

Discussion: Designing a Backup Strategy

- What would be the best backup strategy for a database that supports:
  - An airline ticketing system
  - A data warehouse
  - Business reporting that is updated daily
- What would be the impact of performing frequent backups?
- What is the key difference between a strategy that consists of only full backups and a strategy that consists of a full backup with transaction log?
- Why do you need to perform a backup drill?

Key Points

In this discussion, you are presented scenarios in which you are asked to determine the backup strategy for a database. You and your classmates will discuss possible solutions to the scenario.

Question: What would be the best backup strategy for a database that supports:

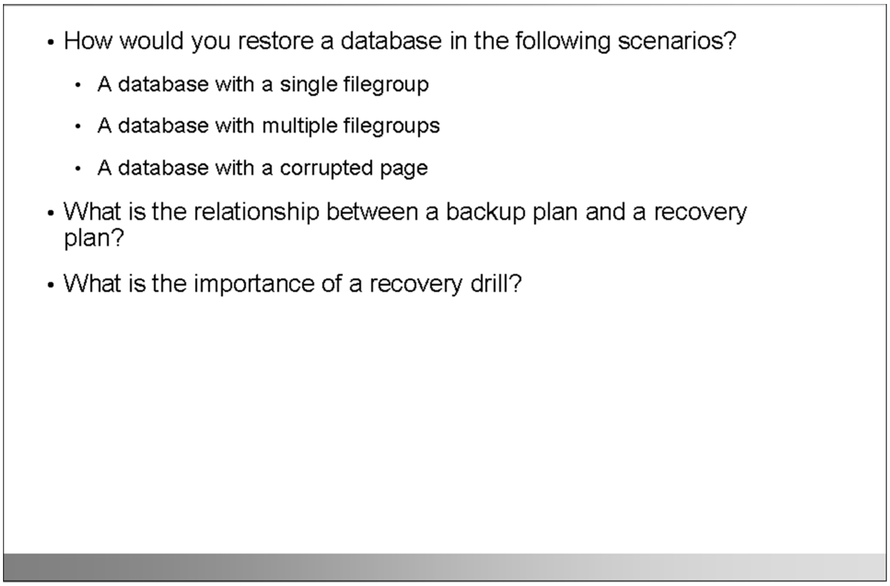
- Airline ticketing system.
- Data Warehouse.
- Business reporting that is updated daily.

Question: What would be the impact of doing too frequent backup?

Question: What is the key difference between a strategy that consists of only full backups and a strategy that consists of a full backup with transaction log?

Question: Why do we need to perform a backup drill?

Discussion: Designing a Recovery Strategy



Key Points

In this discussion, you are presented scenarios in which you are asked to determine the recovery strategy for a database. You and your classmates will discuss possible solutions to the scenario.

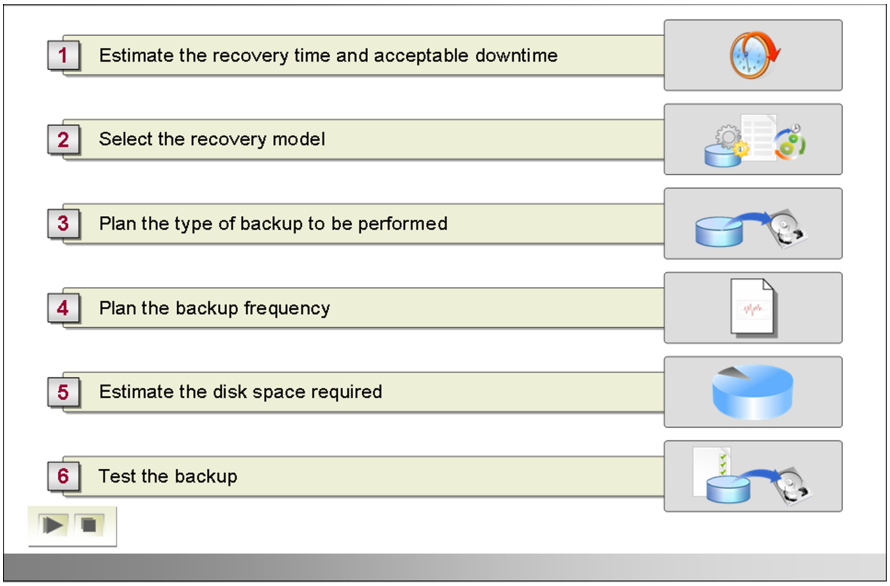
Question: How would you restore a database in the following scenarios?

- A database with a single filegroup
- A database with multiple filegroups
- A database with a page corrupted

Question: What is the relationship between a backup plan and a recovery plan?

Question: What is the importance of a recovery drill?

Creating a Backup and Recovery Strategy



Key Points

The purpose of creating SQL Server backups is to enable the recovery of a damaged database. However, backing up and restoring data must be customized to a particular environment and must work with the available resources. Therefore, you require a backup and restore strategy for reliable recovery. A well-designed backup and restore strategy maximizes data availability and minimizes data loss, while considering your particular business requirements. A backup and restore strategy contains a backup portion and a restore portion. The following are defined by the backup part of the strategy:

- The type and frequency of backups
- The nature and speed of the hardware that is required for them
- How backups are to be tested
- Where and how backup media is to be stored, including security considerations

The restore part of the strategy defines who is responsible for performing restores and how restores should be performed for availability of the database and for minimizing data loss.

The steps for creating a backup and restore strategy include:

1. **Estimate the recovery time and acceptable downtime**. You should determine the amount of downtime that is allowed and use that to come up with the recovery method that will meet the downtime requirements.

2. **Select the recovery model**. Backup and restore operations occur within the context of a recovery model. A recovery model is a database property that controls how the transaction log is managed. Also, the recovery model of a database determines which types of backups and restore scenarios are supported for the database. Typically, a database uses either the simple recovery model or the full recovery model. The full recovery model can be supplemented by switching to the bulk-logged recovery model before bulk operations.

The best choice of recovery model for the database depends on your business requirements. To avoid transaction log management and simplify backup and restore, use the simple recovery model. To minimize work-loss exposure, at the cost of administrative overhead, use the full recovery model.

3. **Plan the type of backup to be performed**. After you have selected a recovery model that meets your business requirements for a specific database, you have to plan and implement a corresponding backup strategy. The optimal backup strategy depends on a variety of factors, of which the following are especially significant:

   - How many hours a day do applications have, to access the database?
   - How frequently are changes and updates likely to occur?

4. **Plan the frequency of backup**. After you have planned the type of backup and if the changes are frequent, consider the following:

   - Under the simple recovery model, consider scheduling differential backups between full database backups.

- Under the full recovery model, schedule frequent log backups.
- Are changes likely to occur in only a small part of the database or in a large part of the database?
- How much disk space will a full database backup require?

5. **Estimate the disk space**. Before you implement a backup and restore strategy, you should estimate how much disk space a full database backup will use. The backup operation copies the data in the database to the backup file. The backup contains only the actual data in the database, and not any unused space. Therefore, the backup is usually smaller than the database itself. You can estimate the size of a full database backup by using the sp_spaceused system-stored procedure.
6. **Test the backup**. You do not have a restore strategy until you have tested your backups. It is very important to thoroughly test your backup strategy for each of your databases by restoring a copy of the database onto a test system. You must test restoring every type of backup that you intend to use.

Question: Why should you create both the backup strategy and recovery strategy together?

Question: Why should you test the backup strategy and recovery strategy?

Source:

- http://msdn.microsoft.com/en-us/library/ms191239.aspx

Lesson 4
Implementing a Repair Strategy

- Repairing System Databases
- Impact of Losing System Databases
- How To Restore a System Database
- Rebuilding System Databases
- Guidelines for Designing a Repair Strategy

SQL Server relies on system databases to store configuration information. If system databases are not accessible or corrupted, some database features are not available, or SQL Server may not even start. If this happens, you should repair the system database so that the services return to normal. You should protect system databases to prevent loss of configurations in the event of failure or corruptions.

In this lesson, you will learn about the impact of losing system databases. You will also learn how to restore a system database and how to rebuild a database. Finally, you will learn about the guidelines for designing a repair strategy.

Repairing System Databases

System Database	Need for Backup	Recovery Model
master	Yes	Simple
model	Yes	User Configurable
msdb	Yes	Simple
Resource	No	Not Applicable
tempdb	No	Not Applicable
Distribution	Yes	Simple

Reapply all hotfixes and service packs after restoring a system database

Key Points

SQL Server maintains a set of system-level databases called system databases, which are essential for the operation of a server instance. You must back up several of the system databases after every significant update. The system databases that you must always back up include msdb, master, and model. If any database uses replication on the server instance, you must also back up the distribution system database. By backing up system databases, you can restore and recover the SQL Server system in case of system failure, such as the loss of a hard disk.

The following table summarizes all the system databases.

System Database	Description	Backups Required	Recovery Model	Comments
master	This database records all the system level information for a SQL Server system.	Yes	Simple	Back up the master database as often as necessary to protect the data sufficiently. Maintain a regular backup schedule, which you can supplement with an additional backup after a substantial update.
model	This database is the template for all databases that are created on the instance of SQL Server.	Yes	User-configurable	Back up the model database only when necessary for your business needs. As a best practice, you should create only full database backups of Model, as required. Backing up the log is unnecessary because Model is small and rarely changes.
msdb	This database is used by SQL Server Agent for scheduling alerts and jobs, and for recording operators. msdb also	Yes	Simple (Default)	You should back up the msdb database whenever it is updated.

System Database	Description	Backups Required	Recovery Model	Comments
	contains history tables, such as the backup and restore history tables.			
Resource (RDB)	This is a read-only database that contains copies of all system objects that ship with Microsoft SQL Server 2005 or SQL Server 2008.	No	Not Applicable	The Resource database resides in the mssqlsystemresource.mdf file, which contains only code. Therefore, SQL Server cannot back up the Resource database.
tempdb	This database is a workspace for holding temporary or intermediate result sets. This database is re-created every time an instance of SQL Server is started. When the server instance is shut down, any data in tempdb is deleted permanently.	No	Simple	You cannot back up the tempdb system database.

System Database	Description	Backups Required	Recovery Model	Comments
Configuring Distribution	This database exists only if the server is configured as a replication Distributor. This database stores metadata and history data for all types of replication, and transactions for transactional replication.	Yes	Simple	None

Note: You can perform a file-based or a disk-based backup on the mssqlsystemresource.mdf file by treating the file as if it were a binary (.exe) file, instead of a database file. However, you cannot use SQL Server restore on the backups. You can only restore a backup copy of mssqlsystemresource.mdf manually, and you must be careful not to overwrite the current Resource database with an out-of-date or potentially unsecured version.

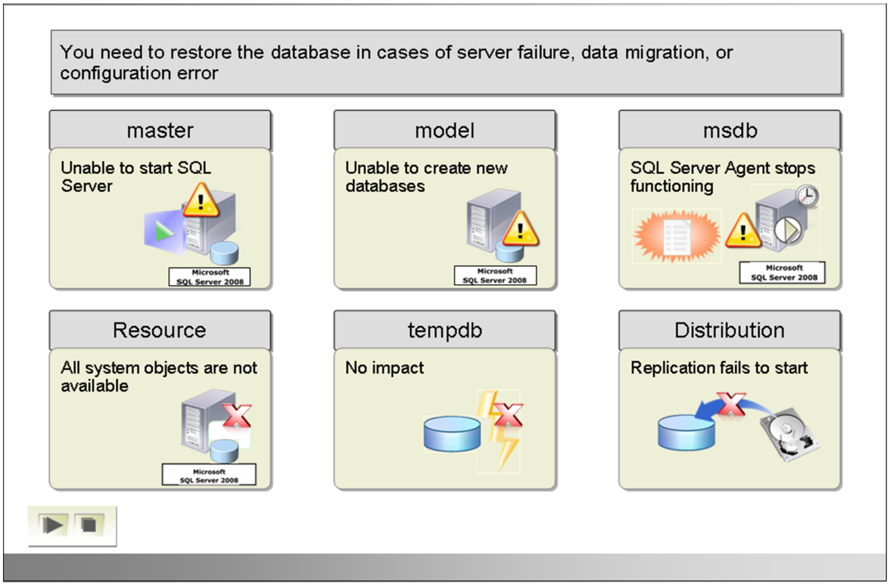
Question: Why should you not back up the tempdb database?

Question: When should you back up the master database and the msdb system database?

Source:

- http://msdn.microsoft.com/en-us/library/ms190190.aspx

Impact of Losing System Databases



Key Points

The scenarios for restoring a system database include:

- **Server failure**. All the servers fail and SQL Server fails to start.
- **Migration**. The databases and SQL server configuration need to be moved to a new server.
- **Configuration error**. An error due to a modification in the SQL Server configuration. If you are unable to resolve the error by rectifying the configuration, restore the system database.
- The impact of losing any of the system databases include:
- **master**. If the master database is lost, SQL Server services will not able to start and all major configurations will be lost. This can cause a serious service interruption.
- **model**. If the model database is lost, SQL Server services will not able to start. This is because SQL Server 2008 will create the tempdb database every time the server starts by using the model database, which is different from the previous version of SQL

Server. Without tempdb SQL Server will not able to start. This can cause a serious service interruption.

- **msdb**. If the msdb database is lost, SQL Server will continue to function, but SQL Server Agent will stop its functioning because the msdb database is used to store the configuration for SQL Server Agent.
- **Resource**. If the Resource database is lost, SQL Server will continue to start but system objects such as dynamic information view will not be available.
- **tempdb**. In the earlier versions of SQL Server 2008, if the tempdb database is lost, SQL Server will fail to start. However, in SQL Server 2008, the tempdb database is not lost because it is created every time the server starts.
- **Distribution**. If the Distribution database is lost, replication will fail, but other aspects of SQL Server will continue to function.

Demonstration: How To Restore a System Database

In this demonstration, you will see how to:

Perform a restore of the master, msdb, model, and tempdb databases

Key Points

The steps to restore the system databases are:

1. On the **Start** menu, point to **Administration Tools**, and then click **Services**. The **Services** window appears.

   Note: SQL Server Management Studio application should not be running.

2. In the **Services** window, stop all SQL Server services.
3. In Windows Explorer, browse to **C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA**.
4. Rename the **master.mdf** database file to **master_old.ndf**, the **MSDBData.mdf** database to **MSDBData_old.ndf**, and the **model.mdf** database to **model_old.ndf**.
5. Rename the **mastlog.ldf** database file to **mastlog_old.ldf**, the **MSDBLog.ldf** database to **MSDBLog_old.ldf**, and the **modellog.ldf** database to **modellog_old.ldf**.

6. In the **Services** window, right-click the **SQL Server (MSSQLSERVER)** service, and then click **Start** to verify that you are unable to start the **SQL Server (MSSQLSERVER)** service. The Services message box appears.

7. In the Services message box, click **OK**.

8. On the Start menu, point to **Administrative Tools**, and then click **Event Viewer**. The Event Viewer window appears.

9. In the Event Viewer window, in the navigation pane, under **Event Viewer**, expand **Windows Logs**, and then click **Application**. Verify the error that occurs with ID 17113 with the following error:

Error 2(The system cannot find the file specified.) occurred while opening file 'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\master.mdf' to obtain configuration information at startup. An invalid startup option might have caused the error. Verify your startup options, and correct or remove them if necessary.

10. On the Start menu, click **Run**, type **cmd**, and then click **OK**. The Command Prompt window appears.

11. At the command prompt, run the following commands.

```
cd\
```

```
cd C:\Program Files\Microsoft SQL Server\100\Setup
   Bootstrap\Release
```

```
setup /ACTION=REBUILDDATABASE /QUIET
   /INSTANCENAME=MSSQLSERVER
   /SQLSYSADMINACCOUNTS=Administrator /SAPWD=Pa$$w0rd
```

12. In Windows Explorer, browse to **C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Log**, and then double-click the **summary.txt** file.

13. In the **summary.txt** file, verify the following result:

Final Result: Passed

14. Close the Command Prompt window.

15. On the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, click **Configuration Tools**, and then click **SQL Server Configuration Manager**. The SQL Server Configuration Manager window appears.
16. In the **SQL Server Configuration Manager** window, click SQL Server Services.
17. In the **SQL Server Configuration Manager** window, in the work pane, right-click **SQL Server (MSSQLSERVER)**, and then click **Properties**. The **SQL Server (MSSQLSERVER) Properties** dialog box appears.
18. In the **SQL Server (MSSQLSERVER) Properties** dialog box, on the **Advanced** tab, in the **Startup Parameters** box, type **–m;** in front of the existing startup options, and then click **OK**.
19. In the **Warning** box, click **OK**.
20. In the **SQL Server Configuration Manager** window, right-click **SQL Server (MSSQLSERVER)**, and then click **Start**.
21. Ignore the warning message that is displayed. Ensure that the **SQL Server (MSSQLSERVER)** service is running.

Note: Before starting this task, ensure that only SQL Server (MSSQLSERVER) service is started. Please stop other SQL services, if they are running.

Note: If an error message appears, click **OK** and ensure that the **SQL Server (MSSQLSERVER)** service is running. If the **SQL Server (MSSQLSERVER)** service is not started, wait for few minutes and then start the **SQL Server (MSSQLSERVER)** service.

22. On the **Start** menu, click **Run**. The **Run** dialog box appears.
23. In the **Open** box of the **Run** dialog box, type **cmd**, and then click **OK**. The Command Prompt window appears.
24. At the command prompt, run the **sqlcmd** command.
25. At the sqlcmd prompt, run the following T-SQL commands.

```
RESTORE DATABASE master FROM DISK = 'D:\Data\master.bak'
   WITH REPLACE;
```

```
GO
```

26. Close the Command Prompt window.
27. In the **SQL Server Configuration Manager** window, in the work pane, right-click **SQL Server (MSSQLSERVER)** and then click **Properties**. The **SQL Server (MSSQLSERVER) Properties** dialog box appears.

Note: Ensure that the **SQL Server (MSSQLSERVER)** service is running. If the service is stopped, then start the **SQL Server (MSSQLSERVER)** service.

28. In the **SQL Server (MSSQLSERVER) Properties** dialog box, on the **Advanced** tab, in the **Startup Parameters** box, type **–T3608;** in front of the existing startup options, and then click **OK**.
29. In the **Warning** box, click **OK**.

Note: If a warning message appears, click **OK** to ignore the message. Ensure that the **SQL Server (MSSQLSERVER)** service is running.

Note: If an error message appears, click **OK** and ensure that the **SQL Server (MSSQLSERVER)** service is running. If the **SQL Server (MSSQLSERVER)** service is not started, wait for few minutes and then start the **SQL Server (MSSQLSERVER)** service.

30. Ignore the warning message that is displayed. Ensure that the **SQL Server (MSSQLSERVER)** service is running.
31. On the Start menu, click **Run**, type **cmd**, and then click **OK**. The Command Prompt window appears.
32. At the command prompt, run **sqlcmd**.
33. At the SQL command prompt, run the following commands.

```
RESTORE DATABASE model FROM DISK = 'D:\Data\model.bak'
   WITH REPLACE;

GO
```

34. In the **SQL Server Configuration Manager** window, in the work pane, right-click **SQL Server (MSSQLSERVER)** and then click **Properties**. The **SQL Server (MSSQLSERVER) Properties** dialog box appears
    Properties. The **SQL Server (MSSQLSERVER) Properties** dialog box appears

35. In the **SQL Server (MSSQLSERVER) Properties** dialog box, on the **Advanced** tab, in the **Startup Parameters** box, remove **–T3608;** from the existing startup options, and then click **OK**.

36. In the **Warning** box, click **OK**.

Note: If an error message appears, click **OK** and ensure that the **SQL Server (MSSQLSERVER)** service is running. If the **SQL Server (MSSQLSERVER)** service is not started, wait for few minutes and then start the **SQL Server (MSSQLSERVER)** service.

37. At the sqlcmd prompt, run the following commands.

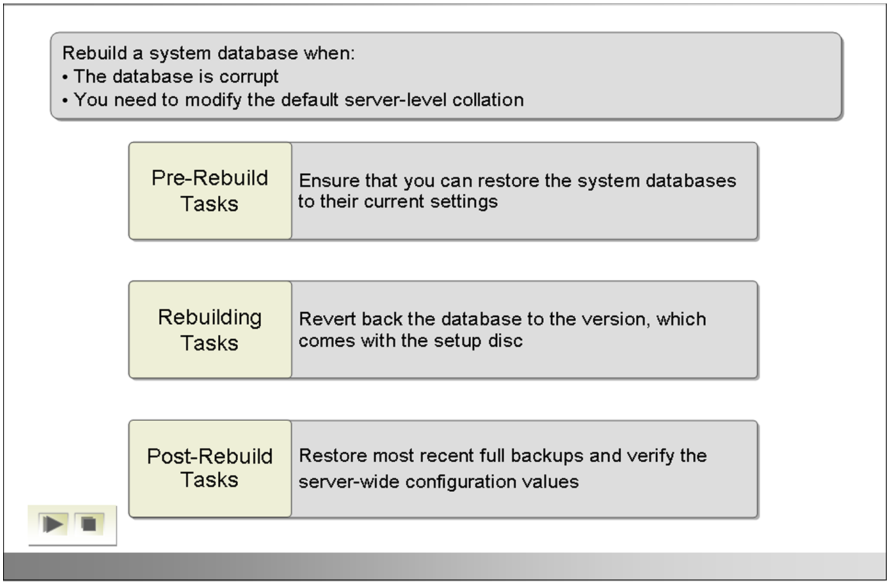
```
RESTORE DATABASE msdb FROM DISK = 'D:\Data\msdb.bak' WITH
   REPLACE;

GO
```

Question: Do you need to restore the tempdb database?

Question: Why should you put the SQL Server services into the master database recovery mode for restoring the master database?

Rebuilding System Databases



Key Points

When you rebuild the master, model, msdb, and tempdb system databases, the databases are dropped and re-created in their original location. This is different from using restore, which helps to revert the database to the state where backup was performed. Rebuild system database is to revert the database to the version, which comes with the Setup disc. If you specify a new collation in the rebuild statement, the system databases are created using that collation setting. Any user modifications to these databases are lost. For example, you may have user-defined objects in the master database, scheduled jobs in msdb, or changes to the default database settings in the model database.

Pre-Rebuild Tasks

Perform the following tasks before you rebuild the system databases to ensure that you can restore the system databases to their current settings:

1. Record all server-wide configuration values.
2. Record all service packs and hotfixes applied to the instance of SQL Server and the current collation.
3. Record the current location of all data and log files for the system databases.
4. Locate the current backup of the master, model, and msdb databases.
5. If the instance of SQL Server is configured as a replication Distributor, locate the current backup of the distribution database.
6. Ensure you have appropriate permissions to rebuild the system databases.
7. Verify that copies of the master, model, msdb data, and log template files exist on the local server.

Rebuilding Task

You cannot specify the system databases to be rebuilt. For clustered instances, you must perform the rebuilding procedure on the active node. The procedure does not rebuild the resource database. You can rebuild a system database by using the following procedure:

1. Insert the SQL Server 2008 installation media into the disk drive, or change the folder to the location of the setup.exe file on the local server.
2. At the command prompt, type the following command.

   ```
   Setup /QUIET /ACTION=REBUILDDATABASE
      /INSTANCENAME=InstanceName
      /SQLSYSADMINACCOUNTS=accounts /[ SAPWD=
      StrongPassword ] [ /SQLCOLLATION=CollationName]
   ```

   If you are using the Windows Vista® operating system with User Account Control (UAC) enabled, you need to run the command prompt as an Administrator.

3. After the setup has completed rebuilding the system databases, verify that the system database is rebuilt successfully by examining the Summary.txt log file.

4. After rebuilding the system databases, all the system databases will be restored to the same state as a new SQL Server installation. You may need to restore the msdb database from a backup or reapply all configuration changes.

Post-Rebuild Tasks

After rebuilding the database, you may need to perform the following additional tasks:

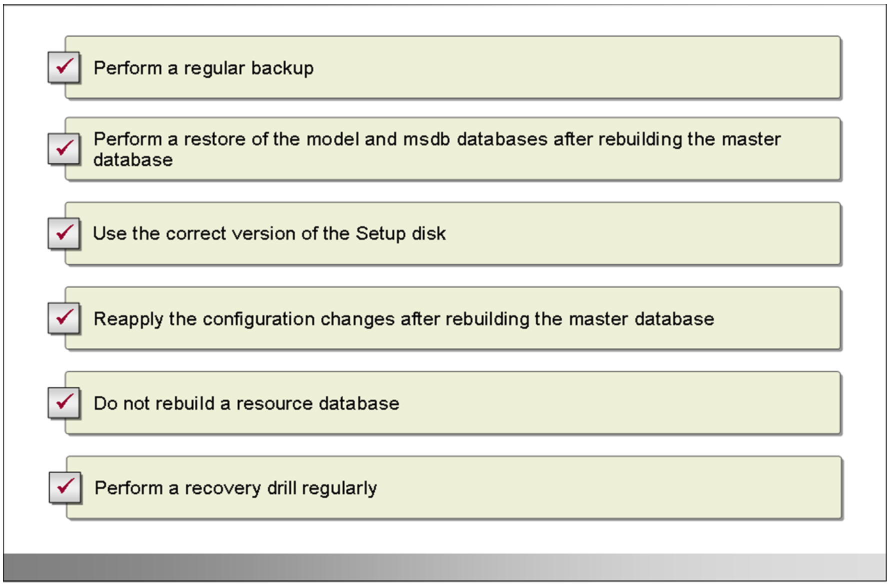
1. Apply the latest service pack and any applicable hotfixes.
2. Restore your most recent full backups of the **master**, **model**, and **msdb** databases.
3. Re-create any missing entries if a backup is not available or if the restored backup is not current. For example, re-create all missing entries for your user databases, backup devices, SQL Server logons, and endpoints. The best way to re-create entries is to run the original scripts that created them.
4. If the instance of SQL Server is configured as a replication Distributor, you must restore the distribution database.
5. Move the system databases to the locations you recorded previously.
6. Verify that the server-wide configuration values match the values you recorded previously.

Question: Which databases are rebuilt every time when you rebuild the system databases?

Source:

- http://technet.microsoft.com/en-us/library/dd207003.aspx

Guidelines for Designing a Repairing Strategy



Key Points

While designing a repairing strategy, you should consider the following guidelines:

- **Perform a regular backup**. Perform a regular backup on the master, model, and msdb databases. Performing a regular backup will help ensure that the database is updated.
- **Perform a restore of the model and msdb database after rebuilding the master database**. Rebuilding the master database will put the default version of msdb and model back to SQL Server, to bring the configuration back.
- **Use the correct version of the Setup disk**. Use the correct version of the Setup disk to perform the rebuild action to avoid incorrect data in the rebuild database.

- **Reapply the configuration changes after rebuilding the master database**. Restoring or rebuilding the master database may restore the configuration back to default. Therefore, you must reapply all server configuration changes to SQL Server.
- **Do not rebuild a Resource database**. Repairing the resource database when it is corrupted helps overwrite the corrupted resources database with the correct one, and restores the database. Do not rebuild a Resource database instead use the setup utility to repair the resource database.
- **Perform a recovery drill regularly**. Performing the recovery drill regularly helps to ensure that the restore steps are working properly. Also get the staff familiar with the recovery steps.

Lab 12: Designing a Backup and Recovery Strategy for SQL Server 2008

- Exercise 1: Designing a Backup and Recovery Solution (Discussion)
- Exercise 2: Backing Up a Database
- Exercise 3: Restoring a Database by Using Online Operations
- Exercise 4: Restoring a System Database

Logon Information

Virtual machine	50400A-NYC-SQL1
User name	Administrator
Password	Pa$$w0rd

Estimated time: 45 minutes

Objectives:

After completing this lab, you will be able to:

- Restore a backup and recovery solution.
- Back up a database.
- Restore a database by using online operations.
- Restore a system database.

Scenario

QuantamCorp has deployed first two SQL Server 2008 servers in their organization and will continue to deploy more to replace their existing installation of SQL Server 2005. SQL Server 2008 will be the primary database engine used in their organization. Recently, a technical audit on SQL Server 2008 deployment is performed. The technical audit report states that the backup and recovery plan of all system must be tested annually. You have decided to perform a backup and recovery on a test lab to test if the backup and recovery steps are functioning properly.

- Identify the different backup types supported in SQL Server2008.
- Determine the relationship between backup and restore strategy.
- Perform a backup of database.
- Perform a restore of database.
- Perform a piecemeal restore of database.
- Perform a restore of system database.
- Perform a rebuild of system database.

Exercise 1: Designing a Backup and Recovery Solution (Discussion)

Scenario

You are required to create a backup and a recovery plan for a Customer Relationship Management (CRM) application that is used extensively during the working hours. You should not allow a full database backup process to be performed during the working hours. This may lead to performance related issues. Also as per the business requirements you should be able to restore the database to a consistent stage after a failure. At minimum, you need to restore all data that is updated two hours prior to any failure. Corporate policy also defines you can only perform major system backup and changes between 12:00 and 16:00. A recent study shows a full backup of the CRM application database takes around two hours. The SQL Server production environment has SQL Server 2008 Enterprise Edition.

Based on the above scenario, answer the following questions:

1. What backup strategy could you adopt for the CRM application database, assuming that you are using a Full Recovery Model? What schedule would you use for each backup type?
2. A system failure occurs on a Thursday at 10:30. This is caused because of a failure in the RAID drive that is used to store the database files and the log files. However, all your database backup files are not stored locally. Hence, you can access the database backup for your application database. The hardware team has repaired the RAID drive. The RAID drive is now functioning properly. You are now required to recover the system. How should you perform the restore?
3. Your have created a backup plan that includes a full database backup and a transaction log backup. A failure in the RAID drive that is used to store the database and transaction log files has lead to a system failure. The application team wants to know why you did not consider performing a tail log backup before the restore. What would you tell the application team?
4. Your application team is unable to access a particular table. You use DBCC CHECKDB to verify the status of the database. You then realize that there are some integrity issues with a data page. What would you do to restore access to all the database tables?
5. When you are testing the environment for your application, your application database server fails. The master and msdb databases get corrupt. And all the data is lost. What will be the impact of this data loss?
6. To lessen the impact of the master and msdb data loss, on the SQL Server and the SQL Server Agent services, what should you do?

Exercise 2: Backing Up a Database

Scenario

Your CRM applications requires a backup strategy in which you do not have to back up the complete database but you have to perform a partial backup due to the size of the database.

Also, you need to create a database backup plan that backs up both user and system databases so that you can restore the database partially and minimize the time required for backup.

The main tasks for this exercise are as follows:

1. Create and configure a test database.
2. Back up the system and the user databases.

Task 1: Create and configure a test database.

1. Open the Microsoft SQL Server Management Studio window.
2. In the Connect to Server dialog box, type the following to connect to Microsoft SQL Server Management Studio:
    - Server type: **Database Engine**
    - Server name: **NYC-SQL1**
    - Authentication: **Windows Authentication**
3. Using the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a Transact SQL (T-SQL) script to create a test database **Mod12**, with the following structure.

File name	Settings
Mod12_Primary.mdf	Current size: 10 Maximum size: 50 Filegrowth: 5
Mod12_Log.ldf	Current size: 10 Maximum size: 50 Filegrowth: 5

4. Add three filegroups to the **Mod12** database: **HistorialData**, **PresentData**, and **TableData**.
5. Add the following files to the filegroup and set **TableData** as the primary filegroup.

File name	File path	Filegroup	Settings
Table_Data01	D:\Data\Mod12_Data01.ndf	TableData	Current size = 10 Maximum size = 50 Filegrowth = 5
			Current size =

File name	File path	Filegroup	Settings
Table_Data02	D:\Data\Mod12_Data02.ndf	TableData	10 Maximum size = 50 Filegrowth = 5
History_Data01	D:\Data\Mod12_History01.ndf	HistorialData	Current size = 10 Maximum size = 50 Filegrowth = 5
History_Data02	D:\Data\Mod12_History02.ndf	HistorialData	Current size = 10, Maximum size = 50 Filegrowth = 5
Present_Data01	D:\Data\Mod12_Present01.ndf	PresentData	Current size = 10, Maximum size = 50 Filegrowth = 5
Present_Data02	D:\Data\Mod12_Present02.ndf	PresentData	Current size = 10, Maximum size = 50 Filegrowth = 5

6. Add a secondary log file named **Mod12_Log02** to the **Mod12** database with the following settings:
   - File path: **D:\data\Mod12_Log02.ldf**
   - Current size: **10**
   - Maximum size: **50**
   - Filegrowth: **5**

7. Create a table named **[dbo].[CallDetails]** in the database with the following fields:
   - call_date: **datetime**
   - caller: **varchar(20)**
   - duration: **int**
   - additional_info: **nchar(4000)**
   - Allow only the **duration** and **additional_info** fields to have NULL values.

In addition, create a constraint named **PK_CallDetails** on the primary key **call_date**.

Task 2: Back up the system and the user databases.

1. Using the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a T-SQL script to backup the **master** database at the **D:\Data\master.bak** path.
2. In the Query Editor pane, run a T-SQL script to backup the **msdb** database at the **D:\Data\msdb.bak** path.
3. In the Query Editor pane, run a T-SQL script to backup the **model** database at the **D:\Data\model.bak** path.
4. In the Query Editor pane, run a T-SQL script to backup the **Mod12** database at the **D:\Data\Mod12.bak** path.

Results: After completing this exercise, you should have created and configured a test database and backed up the system and the user databases.

Exercise 3: Restoring a Database by Using Online Operations

Scenario

After you create the backup plan, you want to test run the backup plan to ensure it's working properly. In the test run, you would perform database backup operations to cover recovery of corrupted data page and a corrupted database file. The main tasks for this exercise are as follows:

1. Attach a database that contains a corrupt page.
2. Perform an online restore of a data page.
3. Verify if the data page is successfully restored.
4. Attach a database that contains a corrupt file.
5. Perform an online restore of a filegroup.
6. Verify if the data file is successfully restored.

Task 1: Attach a database that contains a corrupt page.

1. Using the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a T-SQL script to create a database named **Mod12_Restore**. Attach **D:\Data\Mod12_Restore.mdf** and **D:\Data\Mod12_Restore_Log.ldf** to the **Mod12_Restore** database.
2. In the Query Editor pane, run a T-SQL script to view all records of the **CallDetails** table of the **Mod12_Restore** database.
3. Verify the error that occurs with the following error. Also, note the page number of the error.

Msg 824, Level 24, State 2, Line 1

SQL Server detected a logical consistency-based I/O error: incorrect checksum (expected: 0xe8c32716; actual: 0xe8c3d8bc). It occurred during a read of page (1:171) in database ID 15 at offset 0x00000000156000 in file 'D:\Data\Mod12_Restore.mdf'. Additional messages in the SQL Server error log or system event log may provide more detail. This is a severe error condition that threatens database integrity and must be corrected immediately. Complete a full database consistency check (DBCC CHECKDB). This error can be caused by many factors; for more information, see SQL Server Books Online.

4. Check all object allocation and structural integrity of the **Mod12_Restore** database.
5. Verify the error that occurs with the following error:

Msg 8928, Level 16, State 1, Line 1

Object ID 2105058535, index ID 1, partition ID 72057594038976512, alloc unit ID 72057594043236352 (type In-row data): Page (1:171) could not be processed. See other errors for details.

Msg 8939, Level 16, State 98, Line 1

Table error: Object ID 2105058535, index ID 1, partition ID 72057594038976512, alloc unit ID 72057594043236352 (type In-row data), page (1:171). Test (IS_OFF (BUF_IOERR, pBUF->bstat)) failed. Values are 12716041 and -4

Task 2: Perform an online restore of a data page.

1. Using the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a T-SQL script to perform an online restore of the **1:171** page of the **Mod12_Restore** database.
2. In the Query Editor pane, run a T-SQL script to create a backup log for the Mod12_Restore database at the **D:\Data\Mod12_Restore_log.bak** path.
3. In the Query Editor pane, run a T-SQL script to restore the backup log for the **Mod12_Restore** database located at **D:\Data\Mod12_Restore_log.bak**.

Task 3: Verify if the data page is successfully restored.

1. Using the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a T-SQL script to view all records of the **CallDetails** table of the **Mod12_Restore** database.
2. Check all object allocation and structural integrity of the **Mod12_Restore** database.

Task 4: Attach a database that contains a corrupt file.

1. Using the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a T-SQL script to create a master database named **Mod12_File**. Attach the **D:\Data\Mod12_File_Primary.mdf**, the **D:\Data\Mod12_File_Data.ndf**, and the **D:\Data\Mod12_File_Log.ldf** files to the **Mod12_File** database.

2. In the Query Editor pane, run a T-SQL script to view all records of the **CallDetails** table of the **Mod12_File** database.

3. Verify the error that occurs with the following error:

Msg 824, Level 24, State 2, Line 1

SQL Server detected a logical consistency-based I/O error: incorrect checksum (expected: 0x68c6a712; actual: 0x68c61cb8). It occurred during a read of page (3:8) in database ID 16 at offset 0x00000000010000 in file 'D:\Data\Mod12_File_Data.ndf'. Additional messages in the SQL Server error log or system event log may provide more detail. This is a severe error condition that threatens database integrity and must be corrected immediately. Complete a full database consistency check (DBCC CHECKDB). This error can be caused by many factors; for more information, see SQL Server Books Online.

4. Check all object allocation and structural integrity of the **Mod12_File** database.

5. Verify the error that occurs with the following error:

Msg 8928, Level 16, State 1, Line 1

Object ID 2105058535, index ID 1, partition ID 72057594038976512, alloc unit ID 72057594043236352 (type In-row data): Page (1:171) could not be processed. See other errors for details.

Msg 8939, Level 16, State 98, Line 1

Table error: Object ID 2105058535, index ID 1, partition ID 72057594038976512, alloc unit ID 72057594043236352 (type In-row data), page (3:8). Test (IS_OFF (BUF_IOERR, pBUF->bstat)) failed. Values are 12716041 and -4.

Task 5: Perform an online restore of a filegroup.

1. Using the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a T-SQL script to perform an online restore of the **HistorialData** filegroup attached to the **Mod12_File** database, located at **D:\Data\mod12_file.bak**.

2. In the Query Editor pane, run a T-SQL script to create a backup log for the **Mod12_File** database.

3. In the Query Editor pane, run a T-SQL script to restore the log backup from the **D:\Data\Mod12_File_log.bak** path.

Task 6: Verify if the data file is successfully restored.

1. Using the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a T-SQL script to view all records of the **CallDetails** table of the **Mod12_File** database.
2. Check all object allocation and structural integrity of the **Mod12_File** database.
3. Close the Microsoft SQL Server Management Studio window without saving the any query.

Results: After completing this exercise, you should have attached a database that contains a corrupt page, performed an online restore of a data page, verified if the data page is successfully restored, attached a database that contains a corrupt file, performed an online restore of a filegroup, and verified if the data file is successfully restored.

Exercise 4: Restoring a System Database

Scenario

Recently, data on a system database which resides on another SQL Server instance was corrupted and your team members were unable to restore the system database because they couldn't locate the system database backup.

You checked and ensured that the backup of the system databases on the SQL Server exists. However, before restoring the database, you want to perform a test recovery by using the recent system database backup to ensure that they are usable.

The main tasks for this exercise are as follows:

1. Simulate a system database failure.
2. Rebuild the system databases by using SQL 2008 setup.
3. Configure SQL Server to start in the single user mode.
4. Restore the master database.
5. Restore the msdb database.

Task 1: Simulate a system database failure.

1. Open the **Services** window to manage services in your computer.
2. Using the **Services** window, stop all the SQL Server services.
3. Rename the **master.mdf** database file to **master_old.ndf**, the **MSDBdata.mdf** database to **MSDBdata_old.ndf**, and the **model.mdf** database to **model_old.ndf**.
4. Rename the **mastlog.ldf** database file to **mastlog_old.ldf**, the **MSDBlog.ldf** database to **MSDBblog_old.ldf**, and the **modellog.ldf** database to **modellog_old.ldf**.
5. Verify that you cannot start the **SQL Server (MSSQLSERVER)** service by using the **Services** window, to attempt to start the service.

Task 2: Rebuild the system databases by using SQL 2008 setup.

1. Start the command prompt.
2. Using the command prompt, run the SQL 2008 setup located at the **C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Release** path to rebuild the system databases with the **Administrator** account, and the **Pa$$w0rd** password.
3. Open the **summary.txt** file located at **C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Log** to verify the following result:

Final Result: Passed

4. Close the Command Prompt window.

Task 3: Configure SQL Server to start in the single user mode.

1. Open the SQL Server Configuration Manager window.
2. Using the SQL Server Configuration Manager window, start the **SQL Server Services** window.
3. Using the **SQL Server Services** dialog box, set the startup parameter to **–m;** to configure SQL Server to start in the single user mode.
4. Start the **SQL Server (MSSQLSERVER)** service.

5. Ignore the warning message that is displayed. Ensure that the **SQL Server (MSSQLSERVER)** service is running.

Task 4: Restore the master database.

1. Open the command prompt window.
2. Start the sqlcmd prompt.
3. At the sqlcmd prompt, run T-SQL commands to restore the **master** database at the **D:\Data\master.bak** path.

Task 5: Restore the msdb database.

1. Using the **SQL Server Configuration Manager** window, start the **SQL Server (MSSQLSERVER) Properties** dialog box.
2. Using the **SQL Server (MSSQLSERVER) Properties** dialog box, remove **–m;** before the existing startup options.
3. Start the **SQL Server (MSSQLSERVER)** service.
4. Ignore the warning message that is displayed. Ensure that the **SQL Server (MSSQLSERVER)** service is running.
5. Open the command prompt window, and start the sqlcmd prompt.
6. At the sqlcmd prompt, run a T-SQL code to restore the msdb database at the **D:\Data\msdb.bak** path.

Results: After completing this exercise, you should have simulated a system database failure, rebuilt system databases by using SQL 2008 setup, configured SQL Server to start in the single user mode, restored the master database, configured SQL Server to start in the normal mode, and restored the msdb database.

Module Review and Takeaways

- Review Questions
- Considerations for Designing a Backup and Recovery Strategy for SQL Server 2008

Review Questions

1. You are planning to create a backup and recovery plan for your reporting server which contains information uploaded from another application database every night. What recovery model should this database use?
2. You are planning to create a backup and recovery plan for your reporting server which contains information uploaded from another application database every night. What backup strategy should this database use?
3. You are planning a backup strategy for an application database for a ticketing application. What recovery model should this database use?
4. You have an application which consists of multiple databases. Recently, one of the databases got corrupted and you performed a restore of the failed database. You find that the application is not functioning properly until all the databases are restored from the last full backup. The application provides a message that this

is expected as the application would make changes to all the databases. How can you prevent the need to restore all the databases to a closer point?

5. Your colleague informs you that the tempdb database is corrupted in one of the SQL Servers. What should you do to recover the tempdb?

6. You are now recovering a failed database. You need to restore all configuration changes made to the database before it failed. How should you perform the restore?

Considerations for Designing a Backup and Recovery Strategy for SQL Server 2008

When designing a backup and restore strategy, consider the following:

- Always include system databases as part of the backup plan to ensure that the application databases are associated with the correct version of system databases.
- Use simple recovery model for databases with infrequent data changes.
- Use full recovery model for databases that require a restore at any point of time
- Use marked transactions if you need to ensure the backup is consistent across multiple databases
- Use Enterprise Edition for online backup and restore support.
- Perform a backup of system databases every time a configuration change is made. Backup only the master, model, and msdb databases.
- Practice the recovery procedure to ensure that the defined procedure works
- Store backup of system databases off-site to provide additional protection.

Lab 1: Designing an Administrative Solution for SQL Server 2008

Exercise 1: Designing an Administrative Solution (Discussion)

Question 1

Your management has asked you to prepare a project plan for implementing these changes. What tasks would you include in the project plan?

Answer 1

The project plan depends on the organization, but the following tasks must be there:

- Gather business requirements and confirm that the solution addresses the problems.
- Create test lab for verifying the change.
- Develop the deployment scripts.
- Apply changes in test lab and verify impact to other systems.
- Conduct user acceptance test and get user's consent to deploy the changes.
- Deploy the changes.
- Perform post roll-out verification test.
- Monitor if the changes cause other issues in production.

Question 2

What feature should you use to ensure consistency and uniformity in implementing configuration changes?

Answer 2

You can use script action in SQL Server Management Studio (SSMS) to quickly generate the change scripts instead of developing the scripts from the beginning.

Question 3

How can you execute a script once, and then apply the changes to all servers? The suggested method must require minimal effort for managing all SQL Servers.

Answer 3

You should consider using SSMS with centralized management features, such as registering all servers in your SSMS console and use SQL script to deploy the changes.

Question 4

How can you configure your SQL Server environment to minimize the overtime that your colleagues need to work?

Answer 4

You should consider using SQL Server Agent to schedule and perform the daily tasks. You can use Transact-SQL (T-SQL) or a PowerShell script to perform these tasks. You can use SQL Server Agent to create a complex schedule that will repeat the tasks. This will relieve you and your team from performing these tasks manually.

Question 5

How can you reduce the possibility of human errors? The suggested method must support both configuration for SQL and Windows.

Answer 5

You should consider creating a SQL PowerShell script to perform all tasks. By using SQL PowerShell scripts, you can ensure that a task is performed in the same manner on all systems. This will also help you reduce the possibilities of human error. Moreover, you can also use SQL PowerShell scripts to repeat the tasks on multiple servers, so the time required to perform the tasks is reduced. In addition, you can create a SQL PowerShell script that will repeat a task on all the required servers. SQL PowerShell also provides support for performing operations on remote computers.

Exercise 2: Using SQL PowerShell to Automate SQL Server Configuration

To start the virtual machine, perform the following steps:

1. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn On**.
2. Connect to the **50400A-NYC-SQL1** virtual machine by using Virtual Machine Remote Control Client (VMRC).
3. In the 50400A-NYC-SQL1 virtual machine, on the **Remote Control** menu, point to **Special Keys**, and then click **Send Ctrl+Alt+Delete**.
4. Log on to 50400A-NYC-SQL1, with the user name, **Administrator**, and the password, **Pa$$w0rd**, and then click **Forward**.

Task 1: Create a test database by using SQL PowerShell.

1. On NYC-SQL1, on the **Start** menu, click **Run**, type **cmd**, and then click **OK**. The Administrator: C:\Windows\system 32\cmd.exe window appears.
2. At the command prompt, type **sqlps**, and then press Enter.
3. At the SQL Server PowerShell prompt, run **cd SQL\localhost\default\databases**.
4. At the database prompt, run **Get-ChildItem**.
5. At the database prompt, type each of the following commands, and then press Enter.

```
$NewDB=New-Object
Microsoft.SqlServer.Management.Smo.Database

$NewDB.Parent=(Get-Item ..)

$NewDB.Name="PS_Test"

$NewDB.Create()
```

Task 2: Configure the database compatibility level.

1. At the database prompt, run the **cd ps_test** command.
2. At the ps_test prompt, run the following commands.

```
$MyDB=Get-Item .

$MyDB.CompatibilityLevel

$MyDB.CompatibilityLevel="Version80"

$MyDB.Alter()
```

Task 3: Verify the database configuration.

1. At the ps_test prompt, run the following command.

```
$MyDB.CompatibilityLevel
```

Task 4: Drop the test database.

1. At the ps_test prompt, run the following commands.

```
$MyDB.Drop()

cd ..

Get-ChildItem
```

2. Close the Administrator: C:\Windows\system 32\cmd.exe – sqlps window.

Exercise 3: Managing SQL Server through Central Management Server

Task 1: Create a Central Management Server

1. On NYC-SQL1, on the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. To log on to SQL Server Management Studio, in the **Connect to Server** dialog box, type the following:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. In SQL Server Management Studio, on the **View** menu, click **Registered Servers**.
5. In Registered Servers, expand **Database Engine**, expand **Local Server Groups**.
6. In Registered Servers, right-click **Local Server Groups**, and then click **New Server Group**. Type **Development Servers** as Group name, and then click **OK**.
7. In Registered Servers, right-click **Development Servers**, and then click **New Server Registration**.
8. In the **New Server Registration** dialog box, type NYC-SQL/INSTANCE1 as Server name and click **Save**.
9. In the **New Server Registration** dialog box, specify NYC-SQL/INSTANCE2 as Server name and click **Save**.

Task 2: Execute multiple Server Query

1. Under **Local Server Groups**, double-click **nyc-sql1**, and then click **New Query**.
2. In Query Editor, type and execute a Transact-SQL statement

```
USE master

GO

SELECT * FROM sysdatabases;

GO
```

Task 3: Change the Result Display Format for Multi-Server Query

1. In Microsoft SQL Server Management Studio, on the **Tools** menu, click **Options**.
2. Expand **Query Results**, expand **SQL Server**, and then click **Multiserver Results**.
3. On the **Multiserver Results** page, click the box against the **Merge results** option, select False in the drop-down list, click the **Add server name to the results**, select **False** from the drop-down list and then click **OK**.
4. In Query Editor, type and execute a Transact-SQL statement

```
USE master

GO

SELECT * FROM sysdatabases;

GO
```

Exercise 4: Creating SQL Server PowerShell Scripts

Task 1: Add a SQL Server PowerShell script to create a SQL Server Agent job.

1. On the **Start** menu, point to **All Programs**, click **Accessories**, and then click **Notepad**. The Notepad window appears.
2. In the Notepad window, type the following SQL Server PowerShell script.

```
$server=Get-Item \SQL\localhost\default

$job=New-Object
Microsoft.SqlServer.Management.Smo.Agent.Job

$Job.parent=$server.JobServer

$job.Name="Backup System Database job"

$job.Create()
```

Task 2: Add the SQL Server PowerShell script to create a SQL Server Agent job step.

1. In the Notepad window, type the following SQL Server PowerShell script at the end of the existing script.

```
$masterstep=New-Object
Microsoft.SqlServer.Management.Smo.Agent.JobStep

$masterstep.parent=$job

$masterstep.Name="backup master"

$masterstep.SubSystem="TransactSQL"

$masterstep.Command="BACKUP DATABASE master TO
DISK='D:\Labfiles\Mod01\master.bak'"

$masterstep.OnSuccessAction = "GoToNextStep"

$masterstep.Create()


$modelstep=New-Object
Microsoft.SqlServer.Management.Smo.Agent.JobStep
```

```
$modelstep.parent=$job
    $modelstep.Name="backup model"
    $modelstep.SubSystem="TransactSQL"
    $modelstep.Command="BACKUP DATABASE model TO
    DISK='D:\Labfiles\Mod01\model.bak'"
    $modelstep.OnSuccessAction = "GoToNextStep"
    $modelstep.Create()
```

Task 3: Save the SQL PowerShell script.

1. In the Notepad window, click **File**, and then click **Save**.
2. In the **File name** box, type **D:\Labfiles\Mod01\create_job.ps1**, and then click **Save**.
3. Close **create_job.ps1-Notepad**.

Task 4: Start SQL PowerShell and run the script.

1. On the Start menu, click **Run**, type **cmd**, and then click **OK**. The Administrator: C:\Windows\system 32\cmd.exe window appears.
2. At the Administrator: C:\Windows\system 32\cmd.exe, type **sqlps**, and then press Enter.
3. At the SQL Powershell prompt, run the **D:\Labfiles\Mod01\create_job.ps1**.

Task 5: Verify the SQL Server Agent configuration.

1. In the **Microsoft SQL Server Management Studio** window, under **NYC-SQL1 (SQL Server 10.0.1600 – NYC-SQL1\Administrator)**, expand **SQL Server Agent**, and then expand **Jobs**.
2. Right-click **Backup System Database job**, and then click **Properties**. The **Job Properties – Backup System Database job** dialog box appears.
3. In the **Job Properties – Backup System Database job** dialog box, under **Select a page**, click **Steps**.
4. Verify the following two steps for the Backup System Database job step list:

- backup master
- backup model

5. Close the **Job Properties – Backup System Database job** dialog box.
6. Close the Microsoft SQL Server Management Studio window.
7. In the Microsoft SQL Server Management Studio message box, click **No**.
8. Close the Administrator: C:\Windows\system 32\cmd.exe - sqlps window.

Task 6: Close the virtual machine and discard undo disks.

1. Close the Virtual Machine Remote Control window.
2. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn off machine and discard changes** and then click **OK**.

Lab 2: Deploying SQL Server 2008

Exercise 1: Planning a SQL Server Upgrade (Discussion)

Question 1

Which upgrade strategy should you consider while upgrading SQL Servers in the Account department?

Answer 1

You should consider performing a side-by-side upgrade because the servers that are used to run SQL Server 7.0 will not fulfill the minimum system requirements of SQL Server 2008. Thus, an in-place upgrade is not feasible. You can also perform server consolidation along with the upgrade. The new server will be more powerful than the existing server. It will also be able to process the request that initially required four servers to process.

Question 2

What should you consider while planning the upgrade strategy for the customer services team?

Answer 2

You need to ensure that the applications used by the custom services team run on SQL Server 2008. You also need to set up a test lab to perform a trial upgrade or a trial migration. You should then run each of the applications against the SQL Server test lab. If the application is not working properly, you may want to adjust the compatibility level configuration of the database to resolve the issue. If the issue still exists, you may need to consider revising your approach to migrate the existing application. For example, you can upgrade the application to a version that supports SQL Server 2008. Ensure that all the applications are working properly before you start the installation. You should also test the applications if you are planning to implement server consolidation because this may impact the upgrade strategy.

Question 3

In which order should you upgrade the departments?

Answer 3

Answers will vary. However, the suggested order is: IT, HR, Account, Sales, and finally Customer Services. This list of departments depends on the risk associated with the upgrade. It is ordered by the number of servers required to upgrade and its impact on business. As a common practice, IT is usually selected as the pilot when planning the upgrade because it has minimum impact on business. This list may vary depending on the importance of the LOB applications running in each department and the impact on business when performing the upgrade.

Question 4

How can you ensure minimum risk and downtime during upgrade?

Answer 4

You should always perform side-by-side upgrade that allows fallback if the upgrade fails. Also, the upgrade should be scheduled during non-working hours when there are minimum changes from users. When planning the upgrade, you may consider doing a phased upgrade, instead of performing the upgrade for all servers, simultaneously, because performing a single upgrade for all servers may increase the risk associated with the upgrade.

Lab 3: Designing the Physical Structure of SQL Server 2008

Exercise 1: Planning the Physical Structure of SQL Server 2008 (Discussion)

Question 1

How would you design the physical structure of the database to provide optimal query performance?

Answer 1

The physical structure design of the database can vary. A suggested solution can be to create multiple filegroups in the database, with each filegroup dedicated to store a particular type of data. For example, you can have one filegroup for a customer table, one filegroup for a product table, and one filegroup for a table storing details of calls. With multiple filegroups, you can configure the physical file storage for each filegroup that allows distribution of disk input/output (I/O) across multiple drives between different types of data. You can then associate multiple files with each filegroup. As a best practice, place each file on a separate drive. This provides distribution of load across multiple drives for better I/O performance.

Question 2

How can you resolve the concerns of the application team related to the performance when querying the call details table that contains a large volume of data?

Answer 2

You should consider adding a partition to the table. The partitioning function should be based on a column that helps to distribute I/O across multiple disks for a majority of queries, such as the column storing the call dates. You can consider setting up the partition based on the date and time of the call, and store the data on separate filegroups based on the call date.

In the partitioning setup, you should have at least three partitions. One partition for storing recent calls, such as recent months; one partition for storing not so recent calls, such as calls since last 12 months; and another one for storing historical calls, such as calls prior to the last 12 months. These partitions help you ensure that the application performs well during saving and querying of new records. This is because the database files used to store recent calls are of a small size.

Question 3

What would you suggest to the team to help improve performance of the queries that access the call log table?

Answer 3

To improve query performance, you should consider enabling full-text index. Full-text index provides better performance as compared to the LIKE operator because full-text index has dedicated indexes that are designed to perform text-based queries. With full-text index, SQL Server will use the full-text index to perform the search, instead of full-table scan, which the LIKE operator would use.

Question 4

The application team wants to prevent users from searching for call information based on some restricted words. What would you suggest?

Answer 4

You should create a stoplist and associate it with the full-text index created on the call details table. A Stoplist can be used to prevent full-text searches with certain words. The rows with these words will not have entries created in full-text index, and thus, will not be searched by the full-text query.

Question 5

After you have configured your filegroups, what should you add to the database documentation that is available to the application design team when they are working on the table design?

Answer 5

You must document the filegroup design and partitioning schemes and mention that they need to move all tables to the respective filegroup. You should also change the default filegroup of the database to prevent the application team from accidentally creating tables on the PRIMARY filegroup, which is usually reserved for system tables.

Exercise 2: Implementing the Physical Structure of a Database

To start the virtual machine, perform the following steps:

1. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn On**.
2. Connect to the **50400A-NYC-SQL1** virtual machine by using Virtual Machine Remote Control Client (VMRC).
3. In the 50400A-NYC-SQL1 virtual machine, on the **Remote Control** menu, point to **Special Keys**, and then click **Send Ctrl+Alt+Delete**.
4. Log on to 50400A-NYC-SQL1, with the user name, **Administrator**, and the password, **Pa$$w0rd**, and then click **Forward**.

Task 1: Create a test database.

1. On NYC-SQL1, on the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. To log on to SQL Server Management Studio, in the **Connect to Server** dialog box, type the following:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect.**
4. On the **Start** menu, point to **All Programs**, click **Accessories**, and then click **Windows Explorer**.
5. In Windows Explorer, browse to the **D:\Labfiles\Mod03** folder, and then double-click the **E2_T1.sql** file.

6. In the Microsoft SQL Server Management Studio window, click **Execute**.
7. In Windows Explorer, browse to the **D:\Data** folder and verify the database file and the transaction log file, **Mod03_Primary.mdf** and **Mod03_Log.ldf**.

Task 2: Add filegroups to the database.

1. In Windows Explorer, in the **D:\Labfiles\Mod03** folder, double-click the **E2_T2.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 3: Modify the physical file structure of the database.

1. In Windows Explorer, in the **D:\Labfiles\Mod03** folder, double-click the **E2_T3.sql** file.
2. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Add two files to the TableData filegroup** code, and then click **Execute**.
3. In Windows Explorer, browse to the **D:\Data** folder and verify the additional database files, **Mod03_Data01.ndf** and **Mod03_Data02.ndf**.
4. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Add two files to the HistoryData filegroup** code, and then click **Execute**.
5. In Windows Explorer, in the **D:\Data** folder, verify the additional database files, **Mod03_History01.ndf** and **Mod03_History02.ndf**.
6. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Add two files to the PresentData filegroup** code, and then click **Execute**.
7. In Windows Explorer, in the **D:\Data** folder, verify the additional database files, **Mod03_Present01.ndf** and **Mod03_Present02.ndf**.
8. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Modify File Group** code, and then click **Execute**.

9. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Add a log file to the database** code, and then click **Execute**.
10. In Windows Explorer, in the **D:\Data** folder, verify the additional log file, **Mod03_Log02.ldf**.

Task 4: Verify the physical file structure of the database.

1. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
sp_helpdb 'Mod03'
```

2. Click **Execute**.
3. Verify that the files listed are as shown in the following table.

Name	Field	Filename	Filegroup	Size	Maxsize	Growth	Usage
Primary_Data	1	D:\data\Mod03_Primary.mdf	PRIMARY	10,240 KB	51,200 KB	5,120 KB	Data only
Primary_Log	2	D:\data\Mod03_Log.ldf	NULL	10,240 KB	51,200 KB	5,120 KB	Log only
Table_Data01	3	D:\data\Mod03_Data01.mdf	TableData	10,240 KB	51,200 KB	5,120 KB	Data only
Table_Data02	4	D:\data\Mod03_ Data02.mdf	TableData	10,240 KB	51,200 KB	5,120 KB	Data only
History_Data01	5	D:\data\Mod03_History01.mdf	Historial Data	10,240 KB	51,200 KB	5,120 KB	Data only
History_Data02	6	D:\data\Mod03_ History02.mdf	Historial Data	10,240 KB	51,200 KB	5,120 KB	Data only
Present_Data	7	D:\data\Mod03_Present01.	Present Data	10,240 KB	51,200 KB	5,120 KB	Data only

01		mdf					
Present_Data02	8	D:\data\Mod03_ Present02.mdf	Present Data	10,240 KB	51,200 KB	5,120 KB	Data only
Mod03_Log02	9	D:\data\Mod03_Log02.ldf	NULL	10,240 KB	51,200 KB	5,120 KB	Log only

Exercise 3: Configuring Partitions

Task 1: Create a partition function and a partition scheme.

1. In the Microsoft SQL Server Management Studio window, in the database list, click **Mod03**.
2. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
CREATE PARTITION FUNCTION pfCallDetails (datetime)

AS RANGE RIGHT FOR VALUES ('20060101','20090101')
```

3. Click **Execute**.
4. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
CREATE PARTITION SCHEME psCallDetails

AS PARTITION pfCallDetails

TO (HistorialData,PresentData,TableData)
```

5. Click **Execute**.

Task 2: Create a test table by using the partition scheme.

1. In Windows Explorer, in the **D:\Labfiles\Mod03** folder, double-click the **E3_T2.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 3: Verify the partition scheme.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer, click **Refresh**, and then expand **Databases**.
2. Right-click **Mod03**, point to **Reports**, next point to **Standard Reports**, and then click **Disk Usage**. The Disk Usage pane appears.
3. In the Disk Usage pane, expand **Disk Space Used by Data Files**, and note the **Space Used** column data.
4. In Windows Explorer, in the **D:\Labfiles\Mod03** folder, double-click the **E3_T3.sql** file.

5. In the Microsoft SQL Server Management Studio window, click **Execute**.
6. In the Disk Usage pane, click **Refresh**.
7. In the Disk Usage pane, expand **Disk Space Used by Data Files** and note the **Space Used** column data.
8. Close the Disk Usage report.

Exercise 4: Implementing Database Compression

Task 1: Estimate the database compression savings when it is enabled.

1. In the **Microsoft SQL Server Management Studio** window, click **Mod03** from the database list.
2. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

```
EXEC sp_estimate_data_compression_savings 'dbo',
'CallDetails', NULL, NULL, 'ROW' ;
```

3. Click **Execute**. Note the result.
4. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

```
EXEC sp_estimate_data_compression_savings 'dbo',
'CallDetails', NULL, NULL, 'PAGE';
```

5. Click **Execute**. Note the result.

Task 2: Enable row compression and page compression.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **Databases**, right-click **Mod03**, point to **Reports**, then point to **Standard Reports**, and then click **Disk Usage by Table**. Note the values returned.
2. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

```
SELECT * FROM CallDetails
```

3. Click **Execute**. Note the result and execution time.
4. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

```
ALTER TABLE CallDetails

REBUILD WITH (DATA_COMPRESSION = ROW);
```

5. Click **Execute**.

6. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

   ```
   ALTER INDEX IX_CallDetailscaller
   ON CallDetails
   REBUILD WITH ( DATA_COMPRESSION = PAGE ) ;
   ```

7. Click **Execute**.
8. In the **Microsoft SQL Server Management Studio** window, on the **Disk Usage by ... PM – NYC-SQL1** tab, click **Refresh**. Note the updated values.
9. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

   ```
   SELECT * FROM CallDetails
   ```

10. Click **Execute**. Note the result and the execution time of the query.

Exercise 5: Configuring FILESTREAM Support in SQL Server 2008

Task 1: Enable FILESTREAM support.

1. On the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, click **Configuration Tools**, and then click **SQL Server Configuration Manager**. The Sql Server Configuration Manager window appears.
2. In the Sql Server Configuration Manager window, in the console pane, click **SQL Server Services**.
3. In the result pane, right-click **SQL Server (MSSQLSERVER)**, and then click **Properties**. The **SQL Server (MSSQLSERVER) Properties** dialog box appears.
4. In the **SQL Server (MSSQLSERVER) Properties** dialog box, on the **FILESTREAM** tab, ensure that the **Enable FILESTREAM for Transact-SQL access** check box is selected.
5. Ensure that the **Enable FILESTREAM for file I/O streaming access** check box is selected. In the **Windows share name** box, ensure that **MSSQLSERVER** appears.
6. Ensure that the **Allow remote clients to have streaming access to FILESTREAM data** check box is selected, and click **OK**.
7. Close the Sql Server Configuration Manager window.

Task 2: Change the FILESTREAM access level by using Microsoft SQL Server Management Studio.

1. In the **Microsoft SQL Server Management Studio** window, click **New Query**. The Query Editor pane appears.
2. In the Query Editor pane, type the following T-SQL code.

```
EXEC sp_configure filestream_access_level, 2

RECONFIGURE
```

3. Click **Execute**.

Task 3: Create a database with FILESTREAM support.

1. In Windows Explorer, browse to the **D:\Labfiles\Mod03** folder, and then double-click the **E4_T3.sql** file.
2. In the **Microsoft SQL Server Management Studio** window, click **Execute**.

Task 4: Create a table with FILESTREAM support.

1. In Windows Explorer, browse to the **D:\Labfiles\Mod03** folder, and then double-click the **E4_T4.sql** file.
2. In the **Microsoft SQL Server Management Studio** window, click **Execute**.

Task 5: Verify FILESTREAM configuration.

1. In Windows Explorer, browse to the **D:\Labfiles\Mod03** folder, and then double-click the **E4_T5.sql** file.
2. In the **Microsoft SQL Server Management Studio** window, click **Execute**.
3. Close all open windows.

Task 6: Close the virtual machine and discard undo disks.

1. Close the Virtual Machine Remote Control window.
2. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn off machine and discard changes** and then click **OK**.

Lab 4: Designing a Strategy for Maintaining a Database in SQL Server 2008

Exercise 1: Designing a Database Maintenance Strategy (Discussion)

Question 1

What should you suggest to the application team to resolve the performance issue due to the impact of loading a large amount of data into the database?

Answer 1

Queries run against out-dated statistics cause performance issues. You should consider enabling the autocreate and autoupdate statistics options in the database. To help increase performance of application during updates, you should enable the asynchronous update of statistics. Alternatively, you should update the statistics manually after loading large volumes of data.

Question 2

What would you suggest the application team for modifying the table schema?

Answer 2

You should create a clustered index on the table performing poorly. Also create a nonclustered index with included columns to cover the most commonly used queries. This would help to increase the performance of queries. Because the query will search for information within the XML data, suggest the developer to create an index on the XML column as well.

Question 3

The application has team raised a concern about the reduced database performance while loading data due to the impact of creating an index. What would you suggest them?

Answer 3

You should consider disabling the index before loading the data and rebuild the index only after the data is completed loaded. This helps to ensure that no index operations are performed while loading data and also increases the performance during loading of data.

Question 4

After reviewing the table schema you realize that it has many fields created with the nchar(4000) data type. However, the nchar(4000) data type stores only 10-200 characters. What would you suggest the application team to help them optimize their disk usage? In addition, the application team wants to improve the overall query performance without modifying the database schema and application?

Answer 4

You should consider implementing database compression. Database compression helps to reduce the overall disk consumption. Database compression also helps to speed up the read process by reducing the number of blocks required to read. Database compression is configured at the database level. It does not affect the schema of the application database. Therefore, you do not need to modify the application while configuring database compression.

Exercise 2: Maintaining Statistics

To start the virtual machine, perform the following steps:

5. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn On**.
6. Connect to the **50400A-NYC-SQL1** virtual machine by using Virtual Machine Remote Control Client (VMRC).
7. In the 50400A-NYC-SQL1 virtual machine, on the **Remote Control** menu, point to **Special Keys**, and then click **Send Ctrl+Alt+Delete**.
8. Log on to 50400A-NYC-SQL1, with the user name, **Administrator**, and the password, **Pa$$w0rd**, and then click **Forward**.

Task 1: Create a test database.

1. On NYC-SQL1, on the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. In the **Connect to Server** dialog box, type the following details to log on to Microsoft SQL Server Management Studio:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. On the **Start** menu, point to **All Programs**, point to **Accessories**, and then click **Windows Explorer**.
5. In Windows Explorer, browse to the **D:\Labfiles\Mod04** folder, and then double-click the **E2_T1.sql** file.
6. In the Microsoft SQL Server Management Studio window, click **Execute**.

7. In Windows Explorer, browse to the **D:\Data** folder and verify the database file and the transaction log files.

Task 2: Create a test table and add test records to the table.

1. In Microsoft SQL Server Management Studio window, in the database list, click **Mod04**.
2. In Windows Explorer, in the **D:\Labfiles\Mod04** folder, double-click the **E2_T2.sql** file.
3. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create a test table** code, and then click **Execute**.
4. In the Query Editor pane, select the **Insert records in the test table** code, and then click **Execute**.

Task 3: Run a query with updated statistics.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and type the following T-SQL code.

   ```
   sp_updatestats
   ```

2. Click **Execute**.
3. In the Microsoft SQL Server Management Studio window, on the Query menu, click **Include Actual Execution Plan**.
4. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

   ```
   USE Mod04

   GO

   SELECT * FROM CallDetails WHERE duration BETWEEN
   10 AND 30
   ```

5. Click **Execute**.
6. In the Query Editor pane, on the **Execution plan** tab, note the execution plan.

Task 4: Disable auto-create and auto-update statistics.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and type the following T-SQL code.

```
ALTER DATABASE Mod04
    SET AUTO_CREATE_STATISTICS OFF WITH NO_WAIT;
ALTER DATABASE Mod04
    SET AUTO_UPDATE_STATISTICS OFF WITH NO_WAIT;
```

2. Click **Execute**.

Task 5: Modify the data in the table.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and type the following T-SQL code.

```
DELETE FROM CallDetails WHERE call_date>'2000-1-1'
```

2. Click **Execute**.
3. In Windows Explorer, in the **D:\Labfiles\Mod04** folder, double-click the **E2_T5.sql** file.
4. In the Microsoft SQL Server Management Studio window, click **Execute**.
5. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

```
DELETE FROM CallDetails WHERE call_date<'2004-1-1'
```

6. Click **Execute**.

Task 6: Run the query with outdated statistics.

1. In the Microsoft SQL Server Management Studio window, on the **Query** menu, click **Include Actual Execution Plan**.
2. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and type the following T-SQL code.

```
SELECT * FROM CallDetails WHERE duration BETWEEN
10 AND 20 AND YEAR(call_date)>2005
```

3. Click **Execute**.

4. In the Query Editor pane, on the **Execution plan** tab, note that the execution plan is displayed.
5. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

   ```
   DBCC SHOW_STATISTICS (calldetails,PK_CallDetails)
   ```

6. Click **Execute**. Note that the statistics of the deleted data still exist.

Task 7: Manually update statistics.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and type the following T-SQL code.

   ```
   sp_updatestats
   ```

2. Click **Execute**.
3. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

   ```
   DBCC SHOW_STATISTICS (calldetails,PK_CallDetails)
   ```

4. Click **Execute**.

Task 8: Enable auto-create and auto-update statistics.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and type the following T-SQL code.

   ```
   ALTER DATABASE Mod04
       SET AUTO_CREATE_STATISTICS ON WITH NO_WAIT;
   ALTER DATABASE Mod04
       SET AUTO_UPDATE_STATISTICS ON WITH NO_WAIT;
   ```

2. Click **Execute**.

Exercise 3: Maintaining Indexes

Task 1: Create a test table and add test records to the table.

1. In Windows Explorer, in the **D:\Labfiles\Mod04** folder, double-click the **E3_T1.sql** file.
2. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create a test table** code, and then click **Execute**.
3. In the Query Editor pane, select the **Insert records in the test table** code, and then click **Execute**.

Task 2: Run a test query without index.

1. In the Microsoft SQL Server Management Studio window, on the **Query** menu, click **Include Actual Execution Plan**.
2. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

```
SELECT * FROM CallDetailsHistory WHERE
source_countrycode BETWEEN 60 AND 100 AND
dest_countrycode BETWEEN 100 AND 300
```

3. Click **Execute**.
4. In the Query Editor pane, on the **Execution plan** tab, note the use of table scan.

Task 3: Create a clustered index.

1. In Windows Explorer, in the **D:\Labfiles\Mod04** folder, double-click the **E3_T3.sql** file.
2. In the Microsoft SQL Server Management Studio window, on the **Query** menu, click **Include Actual Execution Plan**.
3. In the Microsoft SQL Server Management Studio window, click **Execute**.
4. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

```
SELECT * FROM CallDetailsHistory WHERE
source_countrycode BETWEEN 60 AND 100 AND
dest_countrycode BETWEEN 100 AND 300
```

5. Click **Execute**.
6. In the Query Editor pane, on the **Execution plan** tab, note that clustered index scan is used.

Task 4: Create a nonclustered index with included column.

1. In Windows Explorer, in the **D:\Labfiles\Mod04** folder, double-click the **E3_T4.sql** file
2. In the Microsoft SQL Server Management Studio window, on the **Query** menu, click **Include Actual Execution Plan**.
3. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create a nonclustered index** code, and then click **Execute**.
4. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

```
SELECT source_countrycode,dest_countrycode FROM
CallDetailsHistory WHERE source_countrycode = 100
```

5. Click **Execute**.
6. In the Query Editor pane, on the **Execution plan** tab, note the use of clustered and nonclustered index.
7. Close all the Query Editor panes without saving changes.
8. In Windows Explorer, in the **D:\Labfiles\Mod04** folder, double-click the **E3_T4.sql** file.
9. In the Microsoft SQL Server Management Studio window, in the database list, click **Mod04**.
10. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Delete the nonclustered index** code, and then click **Execute**.
11. In the Microsoft SQL Server Management Studio window, enable the display of Actual Execution Plan.
12. In the Query Editor pane, delete the existing code, and type the following T-SQL code.

```
SELECT source_countrycode,dest_countrycode FROM
CallDetailsHistory WHERE source_countrycode = 100
```

13. Click **Execute**.
14. In the Query Editor pane, on the **Execution plan** tab, note that only nonclustered index is used.

Task 5: Create an XML index.

1. In Windows Explorer, in the **D:\Labfiles\Mod04** folder, double-click the **E3_T5.sql** file.
2. In the Microsoft SQL Server Management Studio window, enable the display of Actual Execution Plan.
3. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Select the value stored in the /call_info/source and /call_info/dest rows of the additional info column in the CallDetailsHistory table by using an XML query function to return only records with /call_info/@to attribute value set to 1234** code, and then click **Execute**.
4. In the Query Editor pane, on the **Execution plan** tab, note the use of the table value function query.
5. In the Query Editor pane, select the **Create a primary XML index and a secondary XML index for PATH, VALUE, and PROPERTY** code, and then click **Execute**.
6. In the Query Editor pane, select the **Select the value stored in the /call_info/source and /call_info/dest rows of the additional info column in the CallDetailsHistory table by using an XML query function to return only records with /call_info/@to attribute value set to 1234** code, and then click **Execute**.
7. In the Query Editor pane, on the **Execution Plan** tab, note the use of clustered XML index scan.

8. Exercise 4: Implementing Full-Text Indexing

Task 1: Create a test table and add records to the table.

2. In Windows Explorer, in the **D:\Labfiles\Mod04** folder, double-click the **E4_T1.sql** file.
3. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create a test table** code, and then click **Execute**.
4. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Add records to the test table** code, and then click **Execute**.

Task 2: Modify the database file structure for full-text index.

1. In the Microsoft SQL Server Management Studio window, in the database list, click **Mod04**.
2. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
ALTER DATABASE Mod04

ADD FILEGROUP FullTextData

GO
```

3. Click **Execute**.
4. In Windows Explorer, in the **D:\Labfiles\Mod04** folder, double-click the **E4_T2.sql** file.
5. In the Microsoft SQL Server Management Studio window, click **Execute**.
6. In the Windows Explorer, browse to the **D:\Data** folder and verify the additional database file that is created.

Task 3: Create a full-text catalog and a full-text index.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
CREATE FULLTEXT CATALOG Mod04_Catalog AS DEFAULT
```

2. Click **Execute**.

3. In Windows Explorer, in the **D:\Labfiles\Mod04** folder, double-click the **E4_T3.sql** file.
4. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 4: Verify the full-text index setup.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
SELECT * FROM CallDetails_FT

WHERE CONTAINS( additional_info,'call')
```

2. Click **Execute**. Make a note of the rows returned.

Task 5: Manage a full-text index stoplist.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
CREATE FULLTEXT STOPLIST CallDetailsStopList FROM
   SYSTEM STOPLIST;
```

2. Click **Execute**.
3. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
ALTER FULLTEXT STOPLIST CallDetailsStopList ADD 'call'
   LANGUAGE 'English';
```

4. Click **Execute**.
5. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
ALTER FULLTEXT INDEX ON dbo.CallDetails_FT SET STOPLIST
   CallDetailsStopList
```

6. Click **Execute**.

Task 6: Verify the full-text index setup.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

   ```
   SELECT * FROM CallDetails_FT

   WHERE CONTAINS( additional_info,'call')
   ```

2. Click **Execute**.

Task 7: Close the virtual machine and discard undo disks.

1. Close the Virtual Machine Remote Control window.
2. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn off machine and discard changes** and then click **OK**.

Lab 5: Designing Solutions for Managing SQL Server 2008

Exercise 1: Designing a Management Strategy for SQL Server 2008 (Discussion)

Question 1

What should you do to ensure that all changes in the database tables are logged without changing the existing table schema and applications?

Answer 1

To log changes in the database, you should implement change data capture (CDC). By using CDC, you can perform data capture at the server side. CDC monitors changes to the database or tables. SQL Server Agent helps to detect these changes. The logged data can be stored in a dedicated table for a long time. CDC works at the server side and requires no change in the application. The application can continue performing its task as usual. CDC will then monitor the changes and insert a log record in the log table.

Question 2

Your management needs an immediate solution to prevent the system failure from occurring again. In addition, the existing application should not be updated. What solution should you propose?

Answer 2

You should suggest to the senior management to implement Resource Governor and configure all the resources that are used by the developers. You should configure the resources so that they are stored in a separate resources group. In addition, you should assign lower resource priority to the resource group. Configure the maximum resources that are used by the developer to a lower value, such as 50%. This will help ensure that enough system resources are assigned to the production system.

Question 3

The security consultant wants to prevent developers from making unauthorized changes to the database schema. What would you suggest to resolve this concern?

Answer 3

You should suggest that the security consultant implement a DDL trigger to detect database schema changes. Database schema changes can include changes such as a user dropping an application table. You should roll back any changes made by the user. Also include logic in the DDL trigger to log the event in the log table and send an e-mail notification about the event occurrence.

Question 4

How would you ensure that all databases are created in a standardized way?

Answer 4

You should configure SQL Server policies and apply a policy constraint to ensure database objects are created as per the development standard. For example, all table names need to begin with tbl as the prefix. You should configure the policy to perform a real-time check. You should also schedule the policy to evaluate the databases against the policies created as per the company standards.

Exercise 2: Configuring Policies by Using Policy-Based Management

To start the virtual machine, perform the following steps:

1. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn On**.
2. Connect to the **50400A-NYC-SQL1** virtual machine by using Virtual Machine Remote Control Client (VMRC).
3. In the 50400A-NYC-SQL1 virtual machine, on the **Remote Control** menu, point to **Special Keys**, and then click **Send Ctrl+Alt+Delete**.
4. Log on to 50400A-NYC-SQL1, with the user name, **Administrator**, and the password, **Pa$$w0rd**, and then click **Forward**.

Task 1: Create a test database.

1. On NYC-SQL1, on the **Start** menu, point to **All Programs**, point to **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. To log on to SQL Server Management Studio, in the **Connect to Server** dialog box, type the following:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. On the **Start** menu, point to **All Programs**, point to **Accessories**, and then click **Windows Explorer**.
5. In Windows Explorer, browse to the **D:\Labfiles\Mod05** folder, and then double-click the **E2_T1.sql** file.
6. In the Microsoft SQL Server Management Studio window, click

Execute.

Task 2: Create a condition.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **Management**, expand **Policy Management**, right-click **Conditions**, and then click **New Condition**. The **Create New Condition** dialog box appears.
2. In the **Create New Condition** dialog box, in the **Name** box, type **Mod05 Tables**.
3. In the **Facet** list, click **Multipart Name**.
4. In the **Expression** area, for the first row, in the **Field** list, click **@Name**, and in the **Operator** list, click **LIKE**.
5. In the **Expression** area, click the ellipses button besides the **Value** list. The **Advanced Edit** dialog box appears.
6. In the **Advanced Edit** dialog box, type **'tbl%'** to force all table names to start with the letters **tbl**. Click **OK** to close the Advanced Edit dialog box.
7. On the **Description** page, type **Database table names must begin with tbl**, and then click **OK** to create the condition.

Task 3: Create a policy category and subscribe it to a database.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **Management**, right-click **Policy Management**, and then click **Manage Categories**. The **Manage Policy Categories** dialog box appears.
2. In the **Manage Policy Categories** dialog box, in the **Categories** area, for the second row, in the **Name** box, type **Mod05 Category**.
3. Clear the **Mandate Database Subscriptions** check box, and then click **OK**.
4. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **Databases**, right-click **Mod05**, point to **Policies**, and then click **Categories**. The **Categories** dialog box appears.

5. In the **Categories** dialog box, select the **Subscribed** check box for **Mod05 Category**, and then click **OK**.

Task 4: Create a policy within the policy category.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, right-click **Policies**, and then click **New Policy**. The **Create New Policy –** dialog box appears.
2. In the **Create New Policy –** dialog box, in the **Name** box, type **Mod05 Policy**.
3. In the **Check condition** list, under **Multipart Name**, click **Mod05 Tables**.
4. In the **Against targets** area, select the **Every Table in Every Database** check box to apply the policy.
5. Below the **Every Table** check box, in the **Every Database** area, expand **Every**, and then click **New condition**. The **Create New Condition** dialog box appears.
6. In the **Create New Condition** dialog box, in the **Name** box, type **Mod05 Database**.
7. In the **Expression** area, for the first row, in the **Field** list, click **@Name**.
8. In the **Expression** area, click the ellipses button besides the **Value** list. The **Advanced Edit** dialog box appears.
9. In the **Advanced Edit** dialog box, type **'Mod05'** and then click **OK.**
10. In the **Create New Condition — Mod05 Database** dialog box, click **OK**.
11. In the **Evaluation Mode** list, click **On change: prevent**. Ensure that the **Server restriction** box shows **None**. Click **OK** to start the policy creation process.

Note: Ensure that the **Mod05 Policy** is enabled. If **Mod05 Policy** is disabled, in the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **Policies** under **Policy Management**. Then right click **Mod05 Policy**. Then Select **Enable**.

12. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **Policies**, and right-click **Mod05 Policy**, and then click **Properties**. The **Open Policy – Mod05 Policy** dialog box appears.
Policy, and then click **Properties**. The **Open Policy – Mod05 Policy** dialog box appears.
13. In the **Open Policy – Mod05 Policy** dialog box, on the **Description** tab, in the **Category** list, click **Mod05 Category**, and then click **OK**.

Task 5: Test the policy.

1. In Windows Explorer, in the **D:\Labfiles\Mod05** folder, double-click the **E2_T5.sql** file.
2. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Test the effect of the policy on the CallDetails table** code, and then click **Execute**.

Note: The policy prevents the table from being created and returns a message that provides the name of the policy.

3. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Test the effect of the policy on the tb1CallDetails table** code, and then click **Execute**, and note the table that is created.

Task 6: Delete the test database.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
USE master

DROP DATABASE Mod05
```

2. Click **Execute**.
3. In the Object Explorer pane, under **Management**, under **Policies**, right-click **Mod05 Policy**, and then click **Disable**.

Exercise 3: Governing Resources Consumption

Task 1: Create resource pools.

1. In Windows Explorer, in the **D:\Labfiles\Mod05** folder, double-click the **E3_T1.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 2: Create a workload group.

1. In Windows Explorer, in the **D:\Labfiles\Mod05** folder, double-click the **E3_T2.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 3: Enable Resource Governor.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
ALTER RESOURCE GOVERNOR RECONFIGURE

GO
```

2. Click **Execute**.

Task 4: Create and register a classifier function.

1. In Windows Explorer, in the **D:\Labfiles\Mod05** folder, double-click the **E3_T4.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.
3. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
ALTER RESOURCE GOVERNOR with (CLASSIFIER_FUNCTION =
   dbo.fnGroupClassifier)

ALTER RESOURCE GOVERNOR RECONFIGURE

GO
```

4. Click **Execute**.

Task 5: Verify Resource Governor.

1. In Windows Explorer, in the **D:\Labfiles\Mod05** folder, double-click the **E3_T5.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.
3. On the Start menu, click **Run**, type **cmd**, and then click **OK**. The Command Prompt window appears.
4. At the command prompt, run the following command.

   ```
   sqlcmd –E
   ```

5. At the sqlcmd prompt, run the following command.

   ```
   SELECT * FROM sys.tables;

   GO
   ```

Task 6: Delete Resource Governor.

1. In Windows Explorer, in the **D:\Labfiles\Mod05** folder, double-click the **E3_T6.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Note: Before you generate the ALTER RESOURCE GOVERNOR RECONFIGURE statement, verify that there are no active requests in groupAdhoc. If there are active requests, ALTER RESOURCE GOVERNOR will fail. To avoid this issue, you should wait until all the sessions from the workload group have disconnected; you can also explicitly stop sessions in the workload group by using the KILL command; or you can restart the server. The workload group will not be re-created. Alternatively, if you have issued the DROP WORKLOAD GROUP statement, but decide that you do not want to explicitly stop sessions to apply the change, you can re-create the group by using the same name that it had before you issued the DROP statement. Then, move the group to the original resource pool.

Exercise 4: Implementing DDL Triggers

Task 1: Create a test database.

1. In Windows Explorer, in the **D:\Labfiles\Mod05** folder, double-click the **E4_T1.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 2: Create a log table.

1. In Windows Explorer, in the **D:\Labfiles\Mod05** folder, double-click the **E4_T2.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 3: Create and test a DDL trigger.

1. In Windows Explorer, in the **D:\Labfiles\Mod05** folder, double-click the **E4_T3.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.
3. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
CREATE TABLE TestTable (col_key int)

DROP TABLE TestTable ;

GO
```

4. Click **Execute**.
5. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
SELECT * FROM DDL_Log
```

6. Click **Execute**.
7. Close all the Query Editor panes without saving.

Task 4: Delete the test database.

1. In the Microsoft SQL Server Management Studio window, click **New Query**.
2. In the Query Editor pane, type the following T-SQL code.

```
USE master

DROP DATABASE Mod05
```

2. Click **Execute**.

Exercise 5: Implementing CDC

Task 1: Create a test database.

1. In Windows Explorer, in the **D:\Labfiles\Mod05** folder, double-click the **E5_T1.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 2: Create a test table.

1. In Windows Explorer, in the **D:\Labfiles\Mod05** folder, double-click the **E5_T2.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 3: Enable CDC for the test table.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
EXEC sys.sp_cdc_enable_db

GO

EXEC sys.sp_cdc_enable_table N'dbo',
   N'TestTable',DEFAULT,DEFAULT, 1

GO
```

2. Click **Execute**.

Task 4: Verify the CDC configuration.

1. In Windows Explorer, in the **D:\Labfiles\Mod05** folder, double-click the **E5_T4.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.
3. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
SELECT * FROM cdc.fn_cdc_get_all_changes_dbo_TestTable
```

```
(sys.fn_cdc_get_min_lsn('dbo_TestTable'),
   sys.fn_cdc_get_max_lsn(),N'all')
```

4. Click **Execute**. Note that there are four rows returned and each row corresponds to a change that was implemented previously.

5. In the Microsoft SQL Server Management Studio window, in the **Object Explorer**, Click **Connect** and then Select **Integration Services**. The **Connect to Server** dialog box appears.

6. To log on to SQL Server Management Studio, in the **Connect to Server** dialog box, type the following:

    1. Server type: **Integration Services**

    2. Server name: **NYC-SQL1**

    3. Authentication: **Windows Authentication**

7. In Object Explorer, navigate to **NYC-SQL (Integration Services 10.0.1600 – NYC-SQL1\Administrator)\Stored Packages\MSDB**, under **MSDB**, expand **Data Collector**.

8. In Object Explorer, view the **TSQLQueryUpLoad** package.

Task 5: Disable the CDC configuration.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
EXEC sys.sp_cdc_disable_db

GO
```

2. Click **Execute**.

Task 6: Close the virtual machine and discard undo disks.

1. Close the Virtual Machine Remote Control window.

2. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn off machine and discard changes** and then click **OK**.

Lab 6: Automating the Database Management Strategy for SQL Server 2008

Exercise 1: Planning an Automated Database Management Strategy (Discussion)

Question 1

Which strategy would you propose to reduce the operating cost and the administrative effort in managing backup for your SQL Server?

Answer 1

Propose a strategy to use SQL Server Agent and create a job to perform a backup of the system databases and schedule the backup job to perform at regular intervals.

Question 2

Which strategy should you suggest to automate the monitoring of system events?

Answer 2

You should suggest using SQL Server Agent to monitor events by defining alerts. This will help to create event responses, such as sending an email message, or triggering another SQL Server Agent Job to perform further actions or troubleshoot issues with events.

Question 3

If you want to create an automated response or log events centrally when an error occurs on any of the managed servers, would you consider using event forwarding or event monitoring to monitor events on the servers?

Answer 3

You should configure event notifications on monitored servers. Then create a stored procedure on the consolidated monitoring server to write the event information to a database. You should configure the Service Broker route event notification messages from the system that is monitored to route the events to the consolidated database.

Question 4

When should you consider using event notification over event monitoring?

Answer 4

You should consider using event notification when a user performs an operation at the SQL Server level or the database level. For example, use event notification to monitor the action when a user creates a table in a database. You should use event monitoring to monitor errors at the SQL Server level. For instance, use event monitoring when a transaction log is full.

Question 5

Why should you use scripts rather than performing tasks manually?

Answer 5

You should use scripts to reduce possible human errors that may occur when you perform a task manually. Other support staff members such as operators can use scripts created in advance to perform administrative functions, if they have the rights. For example, an operator can perform a task such as backing up of a transaction log. The operator only needs to run a PowerShell file that does not require any knowledge of a transaction log backup.

Moreover, you can use another program such as Microsoft System Center Operations Manager (SCOM) to run the script.

Exercise 2: Using SQL Server Agent

To start the virtual machine, perform the following steps:

1. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn On**.
2. Connect to the **50400A-NYC-SQL1** virtual machine by using Virtual Machine Remote Control Client (VMRC).
3. In the 50400A-NYC-SQL1 virtual machine, on the **Remote Control** menu, point to **Special Keys**, and then click **Send Ctrl+Alt+Delete**.
4. Log on to 50400A-NYC-SQL1, with the user name, **Administrator**, and the password, **Pa$$w0rd**, and then click **Forward**.

Task 1: Start SQL Server Agent.

1. On NYC-SQL1, on the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, click **Configuration Tools**, and then click **SQL Server Configuration Manager**. The Sql Server Configuration Manager window appears.
2. In the SQL Server Configuration Manager window, in the navigation pane, under **SQL Server Configuration Manager (Local)**, click **SQL Server Services**.
3. In the work pane, right-click **SQL Server Agent (MSSQLSERVER)**, and then click **Properties**. The **SQL Server Agent (MSSQLSERVER) Properties** dialog box appears.
4. In the **SQL Server Agent (MSSQLSERVER) Properties** dialog box, on the Service tab, in the **Startup Mode** list, ensure **Automatic** is selected, and then click **OK**.
5. Close Sql Server Configuration Manager.

Task 2: Create a job schedule.

1. On NYC-SQL1, on the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. In the **Connect to Server** dialog box, type the following detailsto log on to Microsoft SQL Server Management Studio:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **SQL Server Agent**, right-click **Jobs**, and then click **Manage Schedules**. The **Manage Schedules** dialog box appears.
5. In the **Manage Schedules** dialog box, in the work pane, click **New**. The **New Job Schedule** dialog box appears.
6. In the **New Job Schedule** dialog box, in the **Name** box, type **Daily 12:00AM**. Ensure that the **schedule type** is **Recurring**.
7. Under **Frequency**, in the **Occurs** list, click **Daily**.
8. In the **Recurs every** box, type or ensure that **1** is selected in the **day(s)** box to schedule a job to occur once in a day.
9. In the **Occurs once at** box, type or select **12:00:00AM**, and then click **OK**.
10. In the **Manage Schedules** dialog box, click **OK**.

Task 3: Create a SQL Server Agent job.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **NYC-SQL1(SQL Server 10.0.1600-NYC-SQL1\Administrator)**,under **SQL Server Agent**, right-click **Jobs**, and click **New Job**. The **New Job** dialog box appears.
2. In the **New Job** dialog box, on the **General** page, in the **Name** box, type **Backup System Databases**.

3. On the **Steps** page, click **New**. The **New Job Step** dialog box appears.
4. In the **New Job Step** dialog box, in the **Step name** box, type **Backup Master**. Ensure that the **Transact-SQL Script (T-SQL)** is selected in the **Type** list.
5. In the **Command** box, type the following command.

```
BACKUP DATABASE master TO DISK='D:\Data\master.bak'
```

Note: You should click **Parse** to verify your syntax. If your syntax is correct, the message "The command was successfully parsed". However, if an error occurs, you need to correct the syntax to move ahead.

6. In the Select a page pane, click the **Advanced** page to set job step options.
7. Click the **ellipses (...)** button next to the **Run as User** field. The **Select User** dialog box appears.
8. In the **Select User** dialog box, click **Browse**. The **Browse for Objects** dialog box appears.
9. In the Browse for Objects dialog box, select the **[dbo]** check box, and then click **OK**.
10. In the **Select User** dialog box, click **OK**.
11. In the **New Job Step** dialog box, click **OK**.
12. In the **New Job** dialog box, click **OK**.

Note: The **dbo** user is a member of the sysadmin role.

Task 4: Assign a schedule to a job.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **SQL Server Agent**, expand **Jobs**, right-click **Backup System Databases**, and then click **Properties**. The **Job Properties – Backup System Databases** dialog box appears.

2. In the **Job Properties – Backup System Databases** dialog box, in the navigation page, click **Schedules**, and then click **Pick**. The **Pick Schedule for Job – Backup System Databases** dialog box appears.
Pick. The **Pick Schedule for Job – Backup System Databases** dialog box appears.

3. In the **Pick Schedule for Job – Backup System Databases** dialog box, ensure the **Daily 12:00AM** Name is selected, and then click **OK**.

4. In the **Job Properties – Backup System Databases** dialog box, double-click the **Daily 12:00AM** schedule. The Job Schedule Properties - daily 12:00AM page appears. Verify that Start date is set on or before the current date. If it is not, set the date to current, and then click **OK**.

5. In the Job Properties – Backup System Databases window, click the **OK** button.

Task 5: Test the job.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **Jobs**, right-click **Backup System Databases**, and then click **Start Job at step.** The Start Jobs –NYC-SQL1 message box appears.

2. In the **Start Jobs – NYC-SQL1** message box, click **Close**.

3. In Windows Explorer, browse to the **D:\Data** folder and verify that the master.bak file exists.

Exercise 3: Creating Scripts to Automate Administrative Tasks

Task 1: Create a VBScript for creating a test database.

1. On the Start menu, point to **All Programs**, click **Accessories**, and then click **Notepad**. The Notepad window appears.
2. Type the following script in notepad.

```
Dim oServer

Dim oDatabase

Dim oDBFileData

Dim oLogFile


Set oServer = CreateObject("SQLDMO.SQLServer")


Set oDatabase =CreateObject("SQLDMO.Database")

Set oDBFileData =CreateObject("SQLDMO.DBFile")

Set oLogFile=CreateObject("SQLDMO.LogFile")


oDatabase.Name = "Mod06"


oDBFileData.Name = "Mod06_Data01"

oDBFileData.PhysicalName = "d:\Data\Mod06_Data01.mdf"

oDBFileData.PrimaryFile = True


oDBFileData.FileGrowthType = SQLDMOGrowth_MB

oDBFileData.FileGrowth = 10


oDatabase.FileGroups("PRIMARY").DBFiles.Add oDBFileData
```

```
oLogFile.Name = "Mod06_Log01"
oLogFile.PhysicalName = "D:\Data\Mod06_Log01.ldf"
oDatabase.TransactionLog.LogFiles.Add oLogFile

oServer.LoginSecure = True
oServer.Connect "."
oServer.Databases.Add oDatabase
```

3. Click **File**, then click **Save**.
4. In the **Save As** dialog box, in the **File name** box, type **D:\Data\create_db.vbs**, and then click **Save**.
5. On the **Start** menu, click **Run**, type **cmd**, and then click **OK**. The command prompt window appears.
6. At the command prompt, type **cscript D:\Data\create_db.vbs**, and then press enter.

Task 2: Create a VBScript to modify the database property.

1. In the Notepad window, click **File**, and then click **New**.
2. Type the following script in notepad.

```
Dim oServer
Dim oDatabase
Dim oDBFile1
Dim oDBFile2
Dim oLogFile
Dim oFileGroup

Set oServer = CreateObject("SQLDMO.SQLServer")

Set oFileGroup =CreateObject("SQLDMO.FileGroup")
Set oDatabase =CreateObject("SQLDMO.Database")
```

```
Set oDBFile1 =CreateObject("SQLDMO.DBFile")
Set oDBFile2 =CreateObject("SQLDMO.DBFile")
Set oLogFile=CreateObject("SQLDMO.LogFile")

oServer.LoginSecure = True
oServer.Connect "."

Set oDatabase = oServer.Databases("Mod06")

oFileGroup.Name = "Mod06_LookupFG"
oDatabase.FileGroups.Add oFileGroup


oDBFile1.Name = "Mod06_Lookup01"
oDBFile1.PhysicalName = "d:\Data\Mod06_Lookup01.ndf"
oDBFile1.Size = 10
oDBFile1.FileGrowthType = SQLDMOGrowth_MB
oDBFile1.FileGrowth = 10

oDatabase.FileGroups("Mod06_LookupFG").DBFiles.Add
oDBFile1

oDBFile2.Name = "Mod06_Lookup02"
oDBFile2.PhysicalName = "d:\Data\Mod06_Lookup02.ndf"
oDBFile2.Size = 10
oDBFile2.FileGrowthType = SQLDMOGrowth_MB
oDBFile2.FileGrowth = 10
```

```
oDatabase.FileGroups("Mod06_LookupFG").DBFiles.Add
oDBFile2
```

2. Click **File**, then click **Save**.
3. In the **Save As** dialog box, in the **File name** box, type **D:\Data\add_db.vbs**, and then click **Save**.
4. At the command prompt, type **cscript D:\Data\add_db.vbs**, and then press enter.
5. In Windows Explorer, browse to the **D:\Data** folder and verify that two database files exist.

Task 3: Create a VBScript to query WMI information.

1. In the Notepad window, click **File**, and then click **New**.

```
strComputer = "."

Set objWMIService = GetObject("winmgmts:" _

    & "{impersonationLevel=impersonate}!\\" &
strComputer & "\root\cimv2")


Set colServiceList = objWMIService.ExecQuery _

    ("Select * from Win32_Service where DisplayName =
'SQL Server Browser'")


For Each objService in colServiceList

    errReturnCode = objService.Change( , , , ,
"Automatic")

Next


For Each objService in colServiceList

    errReturn = objService.StartService()

Next
```

2. Click **File**, then click **Save**.
3. In the **Save As** dialog box, in the **File name** box, type

D:\Data\sql_service.vbs, and then click **Save**.

4. At the command prompt, type **cscript D:\Data\sql_service.vbs**, and then press enter.
5. On the **Start** menu, click **Administrative Tools**, and then click **Services**. Verify that the SQL Server Browser service is started and the startup type is set to automatic.
6. In the Services window, click the **Close** button.
7. In the Notepad window, click the **Close** button.

Exercise 4: Enabling Event Notification

Task 1: Configure Service Broker objects for Event Notification.

1. In Windows Explorer, browse to the **D:\Labfiles\Mod06** folder, and then double-click the **E4_T1.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 2: Enable Event Notification.

1. In Windows Explorer, browse to the **D:\Labfiles\Mod06** folder, and then double-click the **E4_T2.sql** file.
2. In the Microsoft SQL Server Management Studio window, in the **Database** list, select **Mod06**, and then click **Execute**.

Task 3: Test Event Notification.

1. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
CREATE DATABASE [Mod06_Event]

GO

DROP DATABASE [Mod06_Event]

GO
```

2. Click **Execute**.
3. At the command prompt, run the **sqlcmd –E** command.
4. At the sqlcmd prompt, run the following command.

```
SELECT * FROM sys.tables;

GO

exit
```

Note: Close all query windows before executing E4_T3.sql file.

5. In Windows Explorer, browse to the **D:\Labfiles\Mod06** folder, and then double-click the **E4_T3.sql** file.

6. In the Microsoft SQL Server Management Studio window, in the **Database** list, select **Mod06**, and then click **Execute**.
7. In the Query Editor pane, in the **Results** tab, click the first XML message to display the information that is captured.

Exercise 5: Setting Up SQL Server Agent to Monitor Events

Task 1: Create a SQL Server Agent job.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **SQL Server Agent**, right-click **Jobs**, and then click **New Job**. The **New Job** dialog box appears.
2. In the **New Job** dialog box, on the **General** page, in the **Name** box, type **Alert Response Job**.
3. On the **Steps** page, click **New**. The **New Job Step** dialog box appears.
4. In the **New Job Step** dialog box, in the **Step name** box, type **Execute SQL**. Ensure that **Transact-SQL Script (T-SQL)** is selected in the **Type** list.
5. In the **Command** box, type the following command.

```
SELECT @@SERVERNAME
```

Note: You should click **Parse** to verify your syntax. If your syntax is correct, the message "The command was successfully parsed." Parse succeeded" appears. However, if an error occurs, you need to correct the syntax to move ahead.

6. In the **New Job Step** dialog box, click **OK**.
7. In the **New Job** dialog box, click **OK**.

Task 2: Create an alert.

1. In the **Microsoft SQL Server Management Studio** window, in the Object Explorer pane, under **SQL Server Agent**, right-click **Alerts**, and then click **New Alert**. The **New Alert** dialog box appears.
2. In the **New Alert** dialog box, in the **Name** box, type **Alert for Severity 15**.
3. In the **Type** list, click **SQL Server event alert**.
4. In the **Severity** list, click **015 – Syntax Error in SQL Statements**, and then click **OK**.

Task 3: Create a response for an alert.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **SQL Server Agent**, expand **Alerts**, right-click **Alert for Severity 15**, and then click **Properties**. The **'Alert for Severity 15' alert properties** dialog box appears.
2. In the **'Alert for Severity 15' alert properties** dialog box, in the navigation page, click **Response**.
3. In the work pane, select the **Execute job** check box. In the **Execute job** list, click **Alert Response Job ([Uncategorized (Local)])**, and then click **OK**.

Task 4: Verify SQL Server Agent Monitoring.

1. In the **Microsoft SQL Server Management Studio** window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
RAISERROR (N'This is message %s %d.',15, 1, N'SQL
Syntax',1) WITH LOG
```

2. Click **Execute**. Verify the message displayed with the following message.

Msg 50000, Level 15, State 1, Line 1

This is message SQL Syntax 1.

3. In the Object Explorer pane, under **Jobs**, right-click **Alert Response Job**, and click **View History**. The Job History window appears. Note that the job is run once.
4. Close the Job History window.
5. Close the Command Prompt window.
6. Close the Microsoft SQL Server Management studio window.
7. Close the Windows Explorer window.

Task 5: Close the virtual machine and discard undo disks.

1. Close the Virtual Machine Remote Control window.
2. On the **Virtual Server Administration** Web site, click

50400A-NYC-SQL1, and then click **Turn off machine and discard changes** and then click **OK**.

Lab 7: Designing a Strategy for Securing SQL Server 2008

Exercise 1: Designing a Strategy for Database Security (Discussion)

Question 1

How can you ensure that only authorized people have access to the data stored on a SQL Server?

Answer 1

You should consider creating an access matrix. You should then specify the type of users that have access to SQL Server and the database. You can then implement the changes according to the matrix that you have defined. You should grant minimum permissions to authorized users.

Question 2

At which level should you apply access control, the database-level, the server-level, or both levels?

Answer 2

You should apply access control on both levels, the server level and the database level. Use access control at the server level for restricting user access to SQL Server. Use access control at the database level to restrict actions of users who have access to the information stored in a database table.

Question 3

Your organization has more than 100 tables in a few of its databases. One of your colleagues has raised a concern that he does not want to grant individual permissions on each table. How would you resolve his concern?

Answer 3

You can use a database role to get permissions on tables and add users to the role. However, in SQL Server 2008, you should use a schema to separate tables into different groups. You should then assign permissions at the schema level. By using database roles and schemas, you can reduce the administrative effort in managing table permissions.

Question 4

How will you ensure that an unauthorized user cannot view any sensitive information? The user should not be able to view the data even after opening the database by using Microsoft SQL Server Management Studio (SSMS). In addition, the user should not be allowed to view the data even after mounting the database file obtained from the server.

Answer 4

You should consider implementing cell-level encryption to avoid storing credit card information in the table as clear text. This will help you ensure that only applications with appropriate keys or certificates would have access to the data. These applications can decrypt the content by using the respective key or certificate. You can also use transparent data encryption for encrypting the data written in the database file. By using this encryption, you can ensure that an unauthorized user cannot read the database content even after mounting the database file.

Question 5

How would you implement SQL Server to protect the database keys?

Answer 5

You should store all database keys off-site. You should ensure that there are no backup copies of the key that is on the server and is used to decrypt the content. If a key exits on the server, any user can use this key to decrypt the content. You should consider using Extensible Key Management (EKM) to store the keys on a hardware device for better data security.

Exercise 2: Configuring Security in SQL Server.

To start the virtual machine, perform the following steps:

5. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn On**.
6. Connect to the **50400A-NYC-SQL1** virtual machine by using Virtual Machine Remote Control Client (VMRC).
7. In the 50400A-NYC-SQL1 virtual machine, on the **Remote Control** menu, point to **Special Keys**, and then click **Send Ctrl+Alt+Delete**.
8. Log on to 50400A-NYC-SQL1, with the user name, **Administrator**, and the password, **Pa$$w0rd**, and then click **Forward**.

Task 1: Enable SQL authentication.

1. On NYC-SQL1, on the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. To log on to SQL Server Management Studio, in the **Connect to Server** dialog box, type the following:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, right-click **NYC-SQL1(SQL Server 10.0.1600 – NYC-SQL1\Administrator)**, and then click **Properties**. The **Server Properties – NYC-SQL1** dialog box appears.
5. In the navigation pane, click **Security**. The Security page appears.

6. In the work pane, under **Server authentication**, select the **SQL Server and Windows Authentication mode** option button, and then click **OK**. The Microsoft SQL Server Management Studio message box appears.
7. In the Microsoft SQL Server Management Studio message box, click **OK**.
8. In the Object Explorer pane, right-click **NYC-SQL1 (SQL Server 10.0.1600 – NYC-SQL1\Administrator)**, and then click **Restart**.
9. In the **Microsoft SQL Server Management Studio** message box, click **Yes** to restart the service.
10. In the **Microsoft SQL Server Management Studio** message box, click **Yes** to stop the service.

Note: It will take some time for the NYC-SQL1 instance to restart. Wait until the service restarts.

Task 2: Create a test database.

1. In Windows Explorer, browse to the **D:\Labfiles\Mod07** folder, and then double-click the **E2_T2.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 3: Create SQL Server logon.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **NYC-SQL1(SQL Server 10.0.1600 – NYC-SQL1\Administrator)**, right-click the **Security** folder, point to **New**, and then click **Login**. The **Login – New** dialog box appears.
2. In the **Login – New** dialog box, on the **General** page, in the **Login name** box, type **peter**.
3. Click **SQL Server authentication**.
4. In the **Password** box, type **Pa$$w0rd**. Then, in the **Confirm password** box, retype **Pa$$w0rd**.
5. Ensure that the following check boxes are selected:

- **Enforce password policy**
- **Enforce password expiration**

6. Clear the **User must change password at next login** check box.
7. In the **Default database** list, click **Mod07**, and then click **OK**.
8. In the Microsoft SQL Sever Management Studio window, right-click the **Security** folder, point to **New**, and then click **Login**. The **Login – New** dialog box appears.
9. In the **Login – New** dialog box, on the **General** page, in the **Login name** box, type **michael**.
10. Click **SQL Server authentication**.
11. In the **Password** box, type **Pa$$w0rd**. Then, in the **Confirm password** box, retype **Pa$$w0rd**.
12. Ensure that the following check boxes are selected:
    - **Enforce password policy**
    - **Enforce password expiration**
13. Clear the **User must change password at next login** check box.
14. In the **Default database** list, click **Mod07**, and then click **OK**.

Task 4: Create a database user.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **Databases**.
2. Expand **Mod07**, right-click **Security**, point to **New**, and then click **User**. The **Database User - New** dialog box appears.
3. In the **Database User - New** dialog box, on the **General** page, in the **User name** box, type **peter**.
4. Click the **ellipses (...)** button next to **Login name**. The **Select Login** dialog box appears.
5. The **Select Login** dialog box, click **Browse**. The **Browse for Objects** dialog box appears.

6. In the **Browse for Objects** dialog box, select the **[peter]** check box, and then click **OK**.
7. In the **Select Login** dialog box, click **OK**.
8. In the **Database User - New** dialog box, click **OK**.
9. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **Mod07**, right-click **Security**, point to **New**, and then click **User**. The **Database User - New** dialog box appears.
10. In the **Database User - New** dialog box, on the **General** page, in the **User name** box, type **michael**.
11. Click the **ellipses (...)** button next to **Login name**. The **Select Login** dialog box appears.
12. In the **Select Login** dialog box, click **Browse**. The **Browse for Objects** dialog box appears.
13. In the **Browse for Objects** dialog box, select the **[michael]** check box, and then click **OK**.
14. In the **Select Login** dialog box, click **OK**.
15. In the **Database User - New** dialog box, click **OK**.

Task 5: Create schemas.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under the **Mod07** database, right-click **Security**, point to **New**, and then click **Schema**. The **Schema - New** dialog box appears.
2. In the **Schema - New** dialog box, on the **General** page, in the **Schema name** box, type **Calls**.
3. In the **Schema owner** box, type **dbo**, and then click **OK**.
4. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under the **Mod07** database, right-click **Security**, point to **New**, and then click **Schema**. The **Schema - New** dialog box appears.
5. In the **Schema - New** dialog box, on the **General** page, in the **Schema name** box, type **Customers**.
6. In the **Schema owner** box, type **dbo**, and then click **OK**.

Task 6: Create test tables.

1. In Windows Explorer, in the **D:\Labfiles\Mod07** folder, double-click the **E2_T6.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 7: Grant a permission on the test table.

1. In Windows Explorer, in the **D:\Labfiles\Mod07** folder, double-click the **E2_T7.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 8: Test the permission settings.

1. On the **Start** menu, click **Run**, type **cmd**, and then click **OK**. The Administartor:C:\Windows\system32\cmd.exe window appears.
2. At the Administartor:C:\Windows\system32\cmd.exe, run the following command.

```
sqlcmd -U peter -P Pa$$w0rd
```

3. At the sqlcmd prompt, run the following command.

```
SELECT * FROM Calls.CallDetails;

GO
```

Note: No error message will occur.

4. At the sqlcmd prompt, run the following command.

```
SELECT * FROM Customers.CustomerInfo;

GO
```

Note: No error message will occur.

5. Close the SQLCMD.

6. On the **Start** menu, click **Run**, type **cmd**, and then click **OK**. The Administrator:C:\Windows\system32\cmd.exe window appears.
7. At the Administrator:C:\Windows\system32\cmd.exe window, run the following command.

   ```
   sqlcmd -U michael -P Pa$$w0rd
   ```

8. At the sqlcmd prompt, run the following command.

   ```
   SELECT * FROM Calls.CallDetails;
   GO
   ```

Note: No error message will occur.

9. At the SQLCMD prompt, run the following command.

   ```
   SELECT * FROM Customers.CustomerInfo;
   GO
   ```

Note: An error message is displayed.

Exercise 3: Enabling Database Encryption

Task 1: Create a test table.

1. In Windows Explorer, browse to the **D:\Labfiles\Mod07** folder, and then double-click the **E3_T1.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.
3. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
INSERT INTO TestTable (id,content,create_date) VALUES
     (1,'Test',GETDATE())

INSERT INTO TestTable (id,content,create_date) VALUES
     (2,'Test',GETDATE())
```

4. Click **Execute**.

Task 2: Create a database master key.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
Use Mod07

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'P@ssw0rd'
```

2. Click **Execute**.

Task 3: Create a certificate.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
CREATE CERTIFICATE Mod07_Cert  WITH SUBJECT = 'Mod07 Cert';

GO
```

2. Click **Execute**.

Task 4: Create a symmetric key.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
CREATE SYMMETRIC KEY Mod07_Key

WITH ALGORITHM = AES_256

ENCRYPTION BY CERTIFICATE Mod07_Cert;

GO
```

2. Click **Execute**.

Task 5: Enable encryption on the test table.

1. In Windows Explorer, browse to the **D:\Labfiles\Mod07** folder, and then double-click the **E3_T5.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 6: Verify configuration.

1. In Windows Explorer, browse to the **D:\Labfiles\Mod07** folder, and then double-click the **E3_T6.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 7: Create a server certificate.

1. In Windows Explorer, browse to the **D:\Labfiles\Mod07** folder, and then double-click the **E3_T7.sql** file.

2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 8: Create a database encryption key.

1. In Windows Explorer, browse to the **D:\Labfiles\Mod07** folder, and then double-click the **E3_T8.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**. Ignore the warning message that appears.

Task 9: Enable transparent database encryption.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
ALTER DATABASE Mod07

SET ENCRYPTION ON

GO
```

2. Click **Execute**.

Task 10: Close the virtual machine and discard undo disks.

1. Close the Virtual Machine Remote Control window.
2. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn off machine and discard changes** and then click **OK**.

Lab 8: Designing a Strategy for Monitoring SQL Server 2008

Exercise 1: Designing a Data Collection Architecture (Discussion)

Question 1

How would you configure the data collection architecture?

Answer 1

You should configure one management data warehouse and have all SQL Server report the collected information back to the data management warehouse located in the headquarters. This helps you to perform analysis based on the information collected from all SQL Server 2008 servers in your organization.

Question 2

How would you configure data collection such that the data upload process will have minimum impact to business?

Answer 2

You should configure the data collection to perform in every 15mins interval and upload the data to the management data warehouse at off-peak hours. In addition, you should have a dedicated server that hosts the management data warehouse. This will help you minimize the impact of using data collection and ensures that enough information is collected for your analysis. Moreover, ensure that the server used to run the management data warehouse has enough disk space to store the information uploaded by other servers.

Question 3

Your colleague wants to know the difference between using Data Collection in SQL Server and the Data Collector Set in Windows. How would you respond to help him understand the difference?

Answer 3

A Data Collector Set in Windows collects values reported by performance counters. Whereas, the data collection feature in SQL allows the administrator to collect information from SQL. For example, you can use data collection to collect information that you need from a dynamic management view at a regular time. All collected data are then store on the management data warehouse. This helps you to create custom reports by using the collected data with familiar technologies, like SQL Server Reporting Services (SSRS).

Exercise 2: Monitoring SQL Server Instance

To start the virtual machine, perform the following steps:

1. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn On**.
2. Connect to the **50400A-NYC-SQL1** virtual machine by using Virtual Machine Remote Control Client (VMRC).
3. In the 50400A-NYC-SQL1 virtual machine, on the **Remote Control** menu, point to **Special Keys**, and then click **Send Ctrl+Alt+Delete**.
4. Log on to 50400A-NYC-SQL1, with the user name, **Administrator**, and the password, **Pa$$w0rd**, and then click **Forward**.

Task 1: Start and configure the Reliability and Performance Monitor tool.

1. On NYC-SQL1, on the **Start** menu, point to **Administrative Tools**, and then click **Reliability and Performance Monitor**. The Reliability and Performance Monitor window appears.
2. In the **Reliability and Performance Monitor** window, in the navigation pane, under **Monitoring Tools**, and then click **Performance Monitor**.
3. In the result pane, click the **add** button. The **Add Counters** dialog box appears.
4. In the **Add Counters** dialog box, in the **Available counters** list for **<Local computer>**, expand **PhysicalDisk**, click **% Disk Time**, hold Ctrl, and then click **Avg. Disk Queue Length**. Then click **Add**.
5. In the **Available counters** list for **<Local computer>**, expand **SQLServer:Buffer Manager**, and click **Page writes/sec**, hold Ctrl, click **Buffer cache hit ratio**, and then click **Total Pages**. Then click **Add**.
6. In the **Available counters** list for **<Local computer>**, expand

Processor, and click **% Privileged Time**, hold Ctrl, and then click **% User Time**. Then click **Add**.

7. In the **Available counters** list for **<Local computer>**, expand **System**, and click **Processor Queue Length**. Then click **Add**.
8. In the **Available counters** list for **<Local computer>**, expand **Memory**, click **Available Bytes**, hold Ctrl, and then click **Pages/sec**. Then click **Add**.
9. In the **Available counters** list for **<Local computer>**, expand **Process**, and click **Working Set**, and then click **Add**.
10. In the **Available counters** list for **<Local computer>**, expand **SQLServer:Memory Manager**, and click **Total Server Memory (KB)**, and then click **Add**.
11. Click **OK** to close the Add Counters dialog box.

Task 2: Generate SQL workload.

1. On the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. To log on to SQL Server Management Studio, in the **Connect to Server** dialog box, type the following:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. On the **Start** menu, point to **All Programs**, point to **Accessories**, and then click **Windows Explorer**.
5. In Windows Explorer, browse to the **D:\Labfiles\Mod08** folder, and then double-click the **E2_T2.sql** file.
6. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 3: Monitor the result.

1. In the **Reliability and Performance Monitor** window, monitor the change in values reported to represent the SQL workload.

Exercise 3: Implementing Tracing in SQL Server 2008

Task 1: Enable SQL trace.

1. In Windows Explorer, in the **D:\Labfiles\Mod08** folder, double-click the **E3_T1.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 2: Generate workload on the database.

1. In Windows Explorer, in the **D:\Labfiles\Mod08** folder, double-click the **E3_T2.sql** file.
2. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Clear all data from the CallDetails table and insert new records in the table**, and then click **Execute**.
3. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Generate a table schema to test the deadlock**, and then click **Execute**.
4. On the **Start** menu, click **Run**, type **cmd**, and then click **OK**. The Command Prompt window appears.
5. At the command prompt, type **sqlcmd**, and then press enter.
6. At the sqlcmd prompt, run the following command.

```
BEGIN TRAN

UPDATE tempdb.dbo.DLTable SET col1 = 1
```

7. At the sqlcmd prompt, run the following command.

```
BEGIN TRAN

UPDATE tempdb.dbo.DLLock SET col1 = 1

UPDATE tempdb.dbo.DLTable SET col1 = 1
```

8. At the sqlcmd prompt, run the following command.

```
UPDATE tempdb.dbo.DLLock SET col1 = 1
```

9. At the sqlcmd prompt, run **EXIT**.

Task 3: Stop SQL trace.

1. In Windows Explorer, in the **D:\Labfiles\Mod08** folder, double-click the **E3_T3.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 4: Analyze the captured trace.

1. On the Start menu, point to **All Programs**, click **Microsoft SQL Server 2008**, click **Performance Tools**, and then click **SQL Server Profiler**. The SQL Server Profiler window appears.
2. In the SQL Server Profiler window, on the File menu, point to **Open**, and then click **Trace File**. The **Open File** dialog box appears.
3. In the **Open File** dialog box, in the **File name** box, type **D:\Data\Mod08_Trace.trc**, and then click **Open**.
4. Monitor the recorded SQL statements and close SQL Server Profiler.

Exercise 4: Using DTA

Task 1: Use DTA to analyze the captured trace.

1. On the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, click **Performance Tools**, and then click **Database Engine Tuning Advisor**. The Database Engine Tuning Advisor window appears.
2. In the **Connect to Server** dialog box, logon to Microsoft SQL Server 2008, with the following information:
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. In the work pane, on the **General** tab, under **Workload**, click the **Browse** button. The **Select Workload File** dialog box appears.
5. In the **Select Workload File** dialog box, browse to **D:\Data\Mod08_Trace.trc**, and then click **Open**.
6. In the Database Engine Tuning Advisor window, on the **General** tab, under **Select databases and tables to tune**, select the **Mod08** check box.
7. On the **Tuning Options** tab, under **Limit tuning time**, in the **Stop at** box, type the stop time such that it is about one hour ahead of the current time.
8. Under **Physical Design Structure (PDS) to keep in database**, click **Keep clustered indexes only**.
9. Click **Start Analysis** and let the tuning process complete.

Task 2: Analyze the result.

1. In the Database Engine Tuning Advisor window, on the **Recommendations** tab, observe Index Recommendations and Partition Recommendations.
2. On the **Reports** tab, under **Tuning Reports**, in the **Select report** list, click **index details report (current)**.

3. Under **Tuning Reports**, in the **Select report** list, click **index detail report (recommended)**.
4. Close Database Engine Tuning Advisor.

Exercise 5: Monitoring Performance by Using Data Collection

Task 1: Configure Management Data Warehouse.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **Management**, right-click **Data Collection**, and then click **Configure Management Data Warehouse**. The Configure Management Data Warehouse Wizard appears.
2. On the **Welcome to the Configure Management Data Warehouse Wizard** page, click **Next**.
3. On the **Select configuration task** page, ensure that the **Create or upgrade a management data warehouse** option is selected, and then click **Next**.
4. On the **Configure Management Data Warehouse Storage** page, click **New**. The **New Database** dialog box appears.
5. In the **New Database** dialog box, in the **Database name** box, type **MDW**, and then click **OK**.
6. In the **Configure Management Data Warehouse** Wizard, click **Next**.
7. On the **Map Logins and Users** page, under **Users mapped to this login**, select the **NT AUTHORITY\SYSTEM** check box.
8. Under **Database role membership for: MDW**, select the **mdw_admin** check box, and then click **Next**.
9. On the **Complete the Wizard** page, click **Finish**.
10. On the **Configure Data Collection Wizard Progress** page, click **Close**.

Task 2: Enable Data Collection.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **Management**, right-click **Data Collection**, and then click **Configure Management Data Warehouse**. The Configure Management Data Warehouse Wizard appears.

2. On the **Welcome to the Configure Management Data Warehouse Wizard** page, click **Next**.
3. On the **Select configuration task** page, select the **Set up data collection** check box, and then click **Next**.
4. On the **Configure Management Data Warehouse Storage** page, click the **ellipsis** button besides Server name. The **Connect to Server** dialog box appears.
5. In the Connect to Server dialog box, logon to Microsoft SQL Server Management Studio, with the following credentials:
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
6. Click **Connect**.
7. In the **Database name** list, click **MDW**, and then click **Next**.
8. On the **Complete the Wizard** page, click **Finish**.
9. On the **Configure Data Collection Wizard Progress** page, click **Close**.

Task 3: Manually trigger the Data Collection process.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **Data Collection**, expand **System Data Collection Sets**, and right-click **Disk Usage**, and then click **Collect and Upload Now.** The Collect and upload Data Collection Sets – SQL2008 message box appears**.**
2. In the **Collect and upload Data Collection Sets – NYC-SQL1** message box, after the data collection process is complete, click **Close**.
3. In the Microsoft SQL Server Management Studio window, in the Object Explorer under **Data Collection**, right-click **Query Statistics**, and then click **Collect and Upload Now**.
4. In the **Collect and upload Data Collection Sets – NYC-SQL1** message box, after the data collection process is complete, click **Close**.

5. In the Microsoft SQL Server Management Studio window, in the Object Explorer under **Data Collection**, right-click **Server Activity**, and then click **Collect and Upload Now**.
Activity, and then click **Collect and Upload Now**.

6. In the **Collect and upload Data Collection Sets – NYC-SQL1** message box, after the data collection process is complete, click **Close**.

Task 4: Generate Data Collection reports.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, right-click **Data Collection**, point to **Reports**, click **Management Data Warehouse**, and then click **Server Activity History**. The Server Activity History report is displayed in the work pane.

2. In the Object Explorer pane, right-click **Data Collection**, point to **Reports**, click **Management Data Warehouse**, and then click **Disk Usage Summary**. The Disk Usage Collection Set report is displayed in the work pane.

3. In the Object Explorer pane, right-click **Data Collection**, point to **Reports**, click **Management Data Warehouse**, and then click **Query Statistics History**. The Query Statistics History report is displayed in the work pane.

4. Close the Microsoft SQL Server Management Studio window.

5. Close the Reliability and Performance Monitor window.

6. Close the Windows Explorer window.

7. Close the Command Prompt window.

Task 5: Close the virtual machine and discard undo disks.

1. Close the Virtual Machine Remote Control window.

2. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn off machine and discard changes** and then click **OK**.

Lab 9: Designing a Strategy for Content Distribution in SQL Server 2008

Exercise 1: Selecting a Content Distribution Technology (Discussion)

Question 1

What would you suggest to help resolve the performance issue related to report generation without modifying the report and the application code?

Answer 1

You should consider using scalable shared databases that allows the databases to be stored on separate systems. You should ensure that the performance of the server will not get affected when a user attempts to generate reports by using the data stored on the reporting server. The server is used to host the production data.

Question 2

How would you resolve the need of the application team to access data stored on multiple databases?

Answer 2

You should consider using distributed queries with a linked server. This solution allows you to connect and use information from multiple database servers. However, it does not require a large amount of data to be downloaded to the client application for further processing. This download of less data allows reports to use the ability to filter and combine data with SQL statements.

Question 3

What would you propose to help improve the ability to troubleshoot an SSIS package execution?

Answer 3

The application developers should modify the SSIS package to enable logging. They should log entries only during SSIS package run time, and avoid logging entries any other time. However, prior to logging entries, the application developers need to enable support for logging entries during run time.

Exercise 2: Performing a Remote Query

To start the virtual machine, perform the following steps:

1. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn On**.
2. Connect to the **50400A-NYC-SQL1** virtual machine by using Virtual Machine Remote Control Client (VMRC).
3. In the 50400A-NYC-SQL1 virtual machine, on the **Remote Control** menu, point to **Special Keys**, and then click **Send Ctrl+Alt+Delete**.
4. Log on to 50400A-NYC-SQL1, with the user name, **Administrator**, and the password, **Pa$$w0rd**, and then click **Forward**.

Task 1: Create a test database on the NYC-SQL1 instance.

1. On NYC-SQL1, on the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. In the **Connect to Server** dialog box, use the following credentials to log on to Microsoft SQL Server Management Studio:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. On the **Start** menu, point to **All Programs**, point to **Accessories**, and then click **Windows Explorer**.
5. In Windows Explorer, browse to the **D:\Labfiles\Mod09** folder, and then double-click the **E2_T1.sql** file.
6. In the Microsoft SQL Server Management Studio window, click **Execute**. The table is created with a warning message.

Task 2: Create a test database on the NYC-SQL1\DEVELOPMENT instance.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, click **Connect**, and then click **Database Engine**. The Connect to Server dialog box appears.
2. In the **Connect to Server** dialog box, use the following credentials to log on to Microsoft SQL Server Management Studio:
   - Server name: **NYC-SQL1\DEVELOPMENT**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. In Windows Explorer, in the **D:\Labfiles\Mod09** folder, double-click the **E2_T2.sql** file.
5. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 3: Set up a linked server.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, click **Connect**, and then click **Database Engine**. The **Connect to Server** dialog box appears.
2. In the **Connect to Server** dialog box, use the following credentials to log on to Microsoft SQL Server Management Studio:
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. In Windows Explorer, in the **D:\Labfiles\Mod09** folder, double-click the **E2_T3.sql** file.
5. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 4: Verify the linked server setup.

1. In Windows Explorer, in the **D:\Labfiles\Mod09** folder, double-click the **E2_T4.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.

Exercise 3: Monitoring an SSIS Package

Task 1: Enable logging in an SSIS Package.

1. On the Start menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Business Intelligence Development Studio.** The Start Page – Microsoft Visual Studio (Administrator) window appears.
2. In the Start Page – Microsoft Visual Studio (Administrator) window, on the **File** menu, point to **Open**, and then click **File**. The **Open File** dialog box appears.
3. In the **Open File** dialog box, in the **File name** box, type **D:\Data\Import Product Info.dtsx**, and then click **Open**.
4. Right-click anywhere on the **Control Flow** tab, and click **Logging**. The Configure SSIS Logs: Import Product Info window appears.
5. In the Configure SSIS Logs: Import Product Info window, in the Containers pane, ensure that the **Import Product Info** check box is selected.
6. In the work pane, on the **Providers and Logs** tab, in the **Provider type** list, click **SSIS log provider for Text files**, and then click the **Add** button.
7. On the **Details** tab, ensure that the **Events** check box is selected.
8. On the **Providers and Logs** tab, under **Select the logs to use for the container**, under the **Configuration** field, click the second row, and then click **New Connection**. The **File Connection Manager Editor** dialog box appears.
9. In the **File Connection Manager Editor** dialog box, in the **Usage type** list, click **Create file**.
10. Click the **Browse** button. The **Select File** dialog box appears.
11. In the **Select File** dialog box, in the **File name** box, type **D:\Data\log.txt**, and then click **Open**.
12. In the **File Connection Manager Editor** dialog box, click **OK**.

13. In the Configure SSIS Logs: Import Product Info window, click **OK**.
14. Click **Save Import Product Info.dtsx**, and close the **Import Product Info.dtsx** [Design] – Microsoft Visual Studio (Administrator) window.

Task 2: Test the SSIS package.

1. In Windows Explorer, browse to D:\Data, and double-click **Import Product Info.dtsx**. The Execute Package Utility window appears.
2. In the Execute Package Utility window, in the navigation pane, click **Logging**.
3. On the **Logging** page, under **Log providers**, in the **Log Provider** list, click **SSIS log provider for Text files**.
4. In the **Configuration String** list, click **log.txt**.
5. Click **Execute**. The **Package Execution Progress** dialog box appears.
6. In the **Package Execution Progress** dialog box, click **Close**.
7. In the **Execute Package Utility** window, click **Close**.
8. In Windows Explorer, browse to D:\data, and then double-click the **log.txt** file to verify its content.
9. Close the Microsoft SQL Server Management Studio window.
10. Close the Windows Explorer window.

Task 3: Close the virtual machine and discard undo disks.

1. Close the Virtual Machine Remote Control window.
2. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn off machine and discard changes** and then click **OK**.

Lab 10: Designing a Strategy for Replication in SQL Server 2008

Exercise 1: Designing a Replication Strategy (Discussion)

Question 1

Which strategy would you propose to address the performance issue with the application?

Answer 1

You should suggest using replication to duplicate the database on a server located at each office. This would reduce the volume of data that the application would load from the server located in corporate headquarters. Replication would also eliminate the extra load on wide area network (WAN).

Question 2

How would you replicate the App_Lookup database?

Answer 2

You should use the snapshot replication. Schedule the replication to perform only after updating the data in the App_Lookup database. The snapshot replication then sends the updated database to each database server located at each regional office. You should then allow your applications to load the required data from App_Lookup located in the local database.

Question 3

How would you replicate the App_Data database?

Answer 3

You should use the peer-to-peer transaction replication to replicate the App_Data database. By using the peer-to-peer replication, you can minimize the time required for replicating data on all servers. This replication will also help you ensure that the changes are replicated across all servers. Replication topologies other than peer-to-peer replication would take a longer time to update all the servers.

You can also use the merge replication. However, the using merge replication over transactional replication has one drawback. The drawback is that merge replication cannot leverage peer-to-peer replication that allows updates the data across all servers in a short time.

Exercise 2: Setting Up Snapshot Replication

To start the virtual machine, perform the following steps:

1. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn On**.
2. Connect to the **50400A-NYC-SQL1** virtual machine by using Virtual Machine Remote Control Client (VMRC).
3. In the 50400A-NYC-SQL1 virtual machine, on the **Remote Control** menu, point to **Special Keys**, and then click **Send Ctrl+Alt+Delete**.
4. Log on to 50400A-NYC-SQL1, with the user name, **Administrator**, and the password, **Pa$$w0rd**, and then click **Forward**.

Task 1: Create a test database.

1. On NYC-SQL1, on the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**.
2. To log on to SQL Server Management Studio, in the **Connect to Server** dialog box, type the following:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. In Windows Explorer, browse to the **D:\Labfiles\Mod10** folder, and then double-click the **E2_T1.sql** file.
5. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 2: Create a snapshot publication.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **NYC-SQL1 (SQL Server 10.0.1600**

– NYC-SQL1\Administrator), expand the **Replication** folder, right-click **Local Publications**, and then click **New Publication**. The New Publication Wizard launches.

2. On the **New Publication Wizard** page, click **Next**.
3. On the **Distributor** page, ensure that the **' NYC-SQL1' will act as its own Distributor; SQL Server will create a distribution database and log** check box is selected, and then click **Next**.
4. On the **Snapshot Folder** page, in the **Snapshot folder** box, type **\\NYC-SQL1\Repl**, and then click **Next**.
5. On the **Publication Database** page, in the **Databases** list, ensure **Mod10** is selected, and then click **Next**.
6. On the **Publication Type** page, ensure **Snapshot publication** is selected, and then click **Next**.
7. On the **Articles** page, select the **Tables** check box, and click **Next**.
8. On the **Filter Table Rows** page, click **Next**.
9. On the **Snapshot Agent** page, select the **Create a snapshot immediately and keep the snapshot available to initialize subscriptions** check box, and then click **Next**.
10. On the **Agent Security** page, click **Security Settings**. The **Snapshot Agent Security** dialog box appears.
11. In the **Snapshot Agent Security** dialog box, in the **Process account** box, type **NYC-SQL1\Administrator**.
12. In the **Password** box, type **Pa$$w0rd**.
13. In the **Confirm Password**, type **Pa$$w0rd**, and click **OK**.
14. In the New Publication Wizard, on the **Agent Security** page, click **Next**.
15. On the **Wizard Actions** page, click **Next**.
16. On the **Complete the Wizard** page, in the **Publication name** box, type **Mod10 Snapshot Pub**, and then click **Finish**. Wait until the snapshot publication is created.
17. On the **Creating Publication** page, click **Close**.

Task 3: Create subscriptions.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under the **Replication** folder, expand **Local Publications**, right-click **[Mod10]: Mod10 Snapshot Pub,** and then click **New Subscriptions**. . The New Subscription Wizard launches.
2. On the **New Subscription Wizard** page, click **Next**.
3. On the **Publication** page, click **Next**.
4. On the Distribution Agent Location page, ensure that the **Run all agents at the Distributor, NYC-SQL1 (push subscriptions)** option is selected, and then click **Next**.
5. On the **Subscribers** page, in the **Add Subscriber** list, click **Add SQL Server Subscriber**. The **Connect to Server** dialog box appears.
6. In the **Connect to Server** dialog box, logon to another SQL instance with the following credentials:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1\DEVELOPMENT**
   - Authentication: **Windows Authentication**
7. Click **Connect**.
8. In the **New Subscription Wizard**, on the **Subscribers** page, in the **Subscription Database** list for the **NYC-SQL1\DEVELOPMENT** subscriber, click **<New database...>**. The **New Database** window appears.
9. On the **General** page, in the work pane, in the **Database name** box, type **Mod10**, and then click **OK**.
10. In the **New Subscription Wizard**, on the **Subscribers** page, click **Next**.
11. On the **Distribution Agent Security** page, click the **ellipsis** button, in the **NYC-SQL1\DEVELOPMENT** row. The **Distribution Agent Security** dialog box appears.
12. In the **Distribution Agent Security** dialog box, in the **Process account** box, type **NYC-SQL1\Administrator**.
13. In the **Password** box, type **Pa$$w0rd**.

14. In the **Confirm Password**, type **Pa$$w0rd**, and click **OK**.
15. In the New Subscription Wizard, on the **Distribution Agent Security** page, click **Next**.
16. On the **Synchronization Schedule** page, click **Next**.
17. On the **Initialize Subscriptions** page, click **Next**.
18. On the **Wizard Actions** page, click **Next**.
19. On the **Complete the Wizard** page, click **Finish**.
20. On the **Creating Subcription(s)** page, click **Close**.

Task 4: Verify the snapshot replication.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under the **Replication** folder, expand **Local Publications**, right-click **[Mod10]: Mod10 Snapshot Pub,** and then click **Launch Replication Monitor**. The Replication Monitor window appears.
2. In the **Replication Monitor** window, expand **NYC-SQL1**, click **[Mod10]: Mod10 Snapshot Pub**, and verify that the snapshot is running.
3. Close the Replication Monitor window.
4. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, click **Connect**, and then click **Database Engine**. The **Connect to Server** dialog box appears.
5. In the **Connect to Server** dialog box, logon to another SQL instance with the following credentials:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1\DEVELOPMENT**
   - Authentication: **Windows Authentication**
6. Click **Connect**.
7. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, expand **NYC-SQL1**, expand **Databases**, expand **Mod10**, expand **Tables**, right-click **dbo.CallDetails**, and then click **Select Top 1000 Rows**.

8. Verify the contents of the dbo.CallDetails table.
9. Close the Microsoft SQL Server Management Studio window.
10. In the **Microsoft SQL Server Management Studio** dialog box, click **No**.

Exercise 3: Setting Up Peer-to-Peer Replication

Task 1: Create a test database.

1. On the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio message box appears.
2. In the Microsoft SQL Server Management Studio message box, click **No**. The **Connect to Server** dialog box appears.
3. To log on to SQL Server Management Studio, in the **Connect to Server** dialog box, type the following:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
4. Click **Connect**.
5. In Windows Explorer, in the **D:\Labfiles\Mod10** folder, double-click the **E3_T1.sql** file.
6. In the Microsoft SQL Server Management Studio window, click **Execute**.

Task 2: Initialize a database schema.

1. In the Microsoft SQL Server Management Studio window, expand **NYC-SQL1**, expand **Databases**, right-click **Mod10_P2P**, point to **Tasks**, and then click **Copy Database**. The Copy Database Wizard appears.
2. On the **Welcome to the Copy Database Wizard** page, click **Next**.
3. On the **Select a Source Server** page, click **Next**.
4. On the **Select a Destination Server** page, in the **Destination server** box, type **NYC-SQL1\DEVELOPMENT**, and then click **Next**.
5. On the **Select the Transfer Method** page, click **Use the SQL Management Object method**, and then click **Next**.

6. On the **Select Databases** page, ensure that the check box in the Copy column next to **Mod10_P2P** is selected, and then click **Next**.
7. On the **Configure Destination Database (1 of 1)** page, click **Next**.
8. On the **Select Server Objects** page, click **Next**.
9. On the **Configure the Package** page, click **Next**.
10. On the **Schedule the Package** page, click **Next** to run the copy task immediately.
11. On the **Complete the Wizard** page, click **Finish**.
12. On the **Performing operation** page, click **Close**.
13. In the Microsoft SQL Server Management Studio window, under **NYC-SQL1**, under **Databases**, right-click **Mod10_P2P**, point to **Tasks**, and then click **Copy Database**. The Copy Database Wizard appears.
14. On the **Welcome to the Copy Database Wizard** page, click **Next**.
15. On the **Select a Source Server** page, click **Next**.
16. On the **Select a Destination Server** page, in the **Destination server** box, type **NYC-SQL1\INSTANCE3**, and then click **Next**.
17. On the **Select the Transfer Method** page, click **Use the SQL Management Object method**, and then click **Next**.
18. On the **Select Databases** page, ensure that the check box in the Copy column next to **Mod10_P2P** is selected, and then click **Next**.
19. On the **Configure Destination Database (1 of 1)** page, click **Next**.
20. On the **Select Server Objects** page, click **Next**.
21. On the **Configure the Package** page, click **Next**.
22. On the **Schedule the Package** page, click **Next** to run the copy task immediately.
23. On the **Complete the Wizard** page, click **Finish**.
24. On the **Performing operation** page, click **Close**.

Task 3: Create a publication.

1. In the Microsoft SQL Server Management Studio window, under **NYC-SQL1**, expand **Replication**, right-click **Local Publications**, and then click **New Publication**. The **New Publication Wizard** launches.
2. On the **New Publication Wizard** page, click **Next**.
3. On the **Publication Database** page, in the **Databases** list, click **Mod10_P2P**, and then click **Next**.
4. On the **Publication Type** page, click **Transactional publication**, and then click **Next**.
5. On the **Articles** page, select the **Tables** check box, and click **Next**.
6. On the **Filter Table Rows** page, click **Next**.
7. On the **Snapshot Agent** page, select the **Create a snapshot immediately and keep the snapshot available to initialize subscriptions** check box, and then click **Next**.
8. On the **Agent Security** page, click **Security Settings**. The **Snapshot Agent Security** dialog box appears.
9. In the **Snapshot Agent Security** dialog box, in the **Process account** box, type **NYC-SQL1\Administrator**.
10. In the **Password** box, type **Pa$$w0rd**.
11. In the **Confirm Password**, type **Pa$$w0rd**, and click **OK**.
12. In the New Publication Wizard, on the **Agent Security** page, click **Next**.
13. On the **Wizard Actions** page, click **Next**.
14. On the **Complete the Wizard** page, in the **Publication name** box, type **Mod10_P2P Trans Pub**, and then click **Finish**.
15. On the **Creating Publication** page, click **Close**.

Task 4: Configure Distribution Agent.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, click **Connect**, and then click **Database Engine**. The **Connect to Server** dialog box appears.

2. In the **Connect to Server** dialog box, logon to another SQL instance with the following credentials:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1\DEVELOPMENT**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **NYC-SQL1\DEVELOPMENT (SQL Server 10.0.1600 – NYC-SQL1\Administrator)**, expand **Replication**, right-click **Replication**, and then click **Configure Distribution**. The Configure Distribution Wizard launches.
5. On the **Configure Distribution Wizard** page, click **Next**.
6. On the **Distributor** page, click **Next**.
7. On the **Snapshot Folder** page, in the **Snapshot folder** box, type **\\NYC-SQL1\Repl_DEVELOPMENT**, and then click **Next**.
8. On the **Distribution Database** page, click **Next**.
9. On the **Publishers** page, click **Next**.
10. On the **Wizard Actions** page, click **Next**.
11. On the **Complete the Wizard** page, click **Finish**.
12. On the **Configuring** page, click **Close**.
13. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, click **Connect**, and then click **Database Engine**. The **Connect to Server** dialog box appears.
14. In the **Connect to Server** dialog box, logon to another SQL instance with the following credentials:
    - Server type: **Database Engine**
    - Server name: **NYC-SQL1\INSTANCE3**
    - Authentication: **Windows Authentication**
15. Click **Connect**.
16. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, under **NYC-SQL1\INSTANCE3 (SQL Server 10.0.1600 – NYC-SQL1\Administrator)**, expand

Replication, right-click **Replication**, and then click **Configure Distribution**. The Configure Distribution Wizard launches.

17. On the **Configure Distribution Wizard** page, click **Next**.
18. On the **Distributor** page, click **Next**.
19. On the **Snapshot Folder** page, in the **Snapshot folder** box, type **\\NYC-SQL1\Repl_INSTANCE3**, and then click **Next**.
20. On the **Distribution Database** page, click **Next**.
21. On the **Publishers** page, click **Next**.
22. On the **Wizard Actions** page, click **Next**.
23. On the **Complete the Wizard** page, click **Finish**.
24. On the **Configuring** page, click **Close**.

Task 5: Enable peer-to-peer replication.

1. In the Microsoft SQL Server Management Studio window, for the **NYC-SQL1** instance, under **Local Publications**, right-click **[Mod10_P2P]: Mod10_P2P Trans Pub**, and then click **Properties**. The **Publication Properties – Mod10_P2P Trans Pub** dialog box appears.
2. In the **Publication Properties – Mod10_P2P Trans Pub** dialog box, on the **Subscription Options** page, in the work pane, under **Peer-to-Peer Replication**, in the property list for **Allow peer-to-peer subscriptions**, click **True**, and then click **OK**.

Task 6: Configure Peer-To-Peer topology.

1. In the Microsoft SQL Server Management Studio window, for the **NYC-SQL1** instance, under **Local Publications**, right-click **[Mod10_P2P]: Mod10_P2P Trans Pub**, and then click **Configure Peer-To-Peer Topology**. The Configure Peer-To-Peer Topology Wizard launches.
2. On the **Configure Peer-To-Peer Topology Wizard** page, click **Next**.
3. On the **Publication** page, ensure **Mod10_P2P Trans Pub** is selected, and then click **Next**.

4. On the **Configure Topology** page, right-click anywhere in the white area, and then click **Add a New Peer Node**. The **Connect to Server** dialog box appears.
5. In the **Connect to Server** dialog box, logon to another SQL instance with the following credentials:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1\DEVELOPMENT**
   - Authentication: **Windows Authentication**
6. Click **Connect**. The **Add a New Peer Node** dialog box appears.
7. In the **Add a New Peer Node** dialog box, in the **Select Database** list, click **Mod10_P2P**.
8. In the **Peer Originator ID** combo box, type **3**, and then click **OK**.
9. In the **Configure Peer-To-Peer Topology Wizard** dialog box, right-click **NYC-SQL1**, click **Connect to ALL Displayed Nodes**, and then click **Next**.
10. On the **Log Reader Agent Security** page, click the **ellipsis** button besides the NYC-SQL1\DEVELOPMENT server. The **Log Reader Agent Security** dialog box appears.
11. In the **Log Reader Agent Security** dialog box, in the **Process account** box, type **NYC-SQL1\Administrator**.
12. In the **Password** box, type **Pa$$w0rd**.
13. In the **Confirm Password** box, type **Pa$$w0rd**, and then click **OK**.
14. On the **Log Reader Agent Security** page, click **Next**.
15. On the **Distribution Agent Security** page, click the **ellipsis** button, against **NYC-SQL1**. The **Distribution Agent Security** dialog box appears.
16. In the **Distribution Agent Security** dialog box, in the **Process account** box, type **NYC-SQL1\Adminsitrator**.
17. In the **Password** box, type **Pa$$w0rd**.
18. In the **Confirm Password** box, type **Pa$$w0rd**, and then click **OK**.

19. On the **Distribution Agent Security** page, click the **ellipsis** button, besides the **NYC-SQL1\DEVELOPMENT**. The **Distribution Agent Security** dialog box appears.
20. In the **Distribution Agent Security** dialog box, in the **Process account** box, type **NYC-SQL1\Administrator**.
21. In the **Password** box, type **Pa$$w0rd**.
22. In the **Confirm Password** box, type **Pa$$w0rd**, and then click **OK**.
23. On the **Distribution Agent Security** page, click **Next**.
24. On the **New Peer Initialization** page, ensure if the **I created the peer database manually, or I restored a backup of the original publication database which has not been changed since the backup was taken** option is selected, and then click **Next**.
25. On the **Complete the Wizard** page, click **Finish**.
26. On the Building the Peer-To-Peer Topology page, click **Close**.
27. In the Microsoft SQL Server Management Studio window, for the **NYC-SQL1** instance, under **Local Publications**, right-click **[Mod10_P2P]: Mod10_P2P Trans Pub**, and then click **Configure Peer-To-Peer Topology**. The Configure Peer-To-Peer Topology Wizard launches.
28. On the **Configure Peer-To-Peer Topology Wizard** page, click **Next**.
29. On the **Publication** page, click **Mod10_P2P Trans Pub**, and then click **Next**.
30. On the **Configure Topology** page, right-click anywhere in the white area, and then click **Add a New Peer Node**. The **Connect to Server** dialog box appears.
31. In the **Connect to Server** dialog box, logon to another SQL instance with the following credentials:
    - Server type: **Database Engine**
    - Server name: **NYC-SQL1\INSTANCE3**
    - Authentication: **Windows Authentication**

32. Click **Connect**. The **Add a New Peer Node** dialog box appears.
33. In the **Add a New Peer Node** dialog box, in the **Select Database** list, click **Mod10_P2P**.
34. In the **Peer Originator ID** combo box, type **4**, and then click **OK**.
35. In the **Configure Peer-To-Peer Topology Wizard** dialog box, right-click NYC-SQL1, click **Connect to ALL Displayed Nodes**, if **Connect to Server** dialog box appears, click **Cancel**, and then click **Next**.
36. On the **Log Reader Agent Security** page, click the **ellipsis** button besides the NYC-SQL1\INSTANCE3 server. The **Log Reader Agent Security** dialog box appears.
37. In the **Log Reader Agent Security** dialog box, in the **Process account** box, type **NYC-SQL1\Administrator**.
38. In the **Password** box, type **Pa$$w0rd**.
39. In the **Confirm Password** box, type **Pa$$w0rd**, and then click **OK**.
40. On the **Log Reader Agent Security** page, click **Next**.
41. On the **Distribution Agent Security** page, click the **ellipsis** button, against **NYC-SQL1**. The **Distribution Agent Security** dialog box appears.
42. In the **Distribution Agent Security** dialog box, in the **Process account** box, type **NYC-SQL1\Adminsitrator**.
43. In the **Password** box, type **Pa$$w0rd**.
44. In the **Confirm Password** box, type **Pa$$w0rd**, and then click **OK**.
45. On the **Distribution Agent Security** page, click the **ellipsis** button, besides the **NYC-SQL1\INSTANCE3** server. The **Distribution Agent Security** dialog box appears.
46. In the **Distribution Agent Security** dialog box, in the **Process account** box, type **NYC-SQL1\Administrator**.
47. In the **Password** box, type **Pa$$w0rd**.
48. In the **Confirm Password** box, type **Pa$$w0rd**, and then click **OK**.

49. On the **Distribution Agent Security** page, click **Next**.
50. On the **New Peer Initialization** page, ensure if the **I created the peer database manually, or I restored a backup of the original publication database which has not been changed since the backup was taken** option is selected, and then click **Next**.
51. On the **Complete the Wizard** page, click **Finish**.
52. On the **Building the Peer-To-Peer Topology** page, click **Close**.

Task 7: Verify the replication setup.

1. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, for the NYC-SQL1 instance, under **Databases**, right-click **Mod10_P2P**, and then click **New Query**.
2. In Windows Explorer, in the **D:\Labfiles\Mod10** folder, double-click the **E3_T7.sql** file.
3. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Verify the replication setup in NYC-SQL1**, and then click **Execute**.
4. In the Object Explorer pane, in the **NYC-SQL1\DEVELOPMENT** instance, right-click **Databases**, and then click **Refresh**.
5. Expand **Databases**, expand **Mod10_P2P**, expand **Tables**, right-click **dbo.CallDetails**, and then click **Select Top 1000 Rows**.
6. Verify if the data inserted is replicated to the NYC-SQL1\DEVELOPMENT instance.
7. In the Object Explorer pane, for the NYC-SQL1\INSTANCE3 instance, expand **Databases**, expand **Mod10_P2P**, expand **Tables**, right-click **dbo.CallDetails**, and then click **Select Top 1000 Rows**.
8. Verify if the data inserted is replicated to the NYC-SQL1\INSTANCE3 instance.
9. In the Microsoft SQL Server Management Studio window, in the Object Explorer pane, for the NYC-SQL1\DEVELOPMENT instance, under **Databases**, right-click **Mod10_P2P**, and then click **New Query**.

10. In Windows Explorer, in the **D:\Labfiles\Mod10** folder, double-click the **E3_T7.sql** file.
11. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Verify the replication setup in NYC-SQL1\DEVELOPMENT**, and then click **Execute**.
12. In the Object Explorer pane, for the NYC-SQL1 instance, under **Databases**, expand **Mod10_P2P**, expand **Tables**, right-click **dbo.CallDetails**, and then click **Select Top 1000 Rows**.
13. Verify if the data inserted is replicated to the NYC-SQL1 instance.
14. In the Object Explorer pane, for the NYC-SQL1\INSTANCE3 instance, under **Databases**, under **Mod10_P2P**, under **Tables**, right-click **dbo.CallDetails**, and then click **Select Top 1000 Rows**.
15. Verify if the data inserted is replicated to the NYC-SQL1\INSTANCE3 instance.

Task 8: Close the virtual machine and discard undo disks.

1. Close the Virtual Machine Remote Control window.
2. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn off machine and discard changes** and then click **OK**.

Lab 11: Designing a High-Availability Solution for SQL Server 2008

Exercise 1: Designing a High-Availability Strategy (Discussion)

Question 1

What would you suggest to the application team to meet their availability needs for their application databases in the company's corporate headquarter?

Answer 1

You should suggest using failover clustering. Failover clustering helps to address the need for high-availability by using a single site. To remove all single point of failure, you should consider using storage area network (SAN). You should use SAN as a storage that will remove the single point of failure for a single quorum disk.

Question 2

What would you suggest to meet the requirements for setting up a disaster recovery (DR) site?

Answer 2

You should consider using database mirroring for replicating database content from the production site to the DR site. Switching of active and passive server roles is difficult to perform by using replication. Hence, database mirroring is preferred over replication.

Question 3

You application team wants to roll out a small-scale application. The application will provide the same level of availability with no budget required to implement failover clustering. How would you meet this need of the application team?

Answer 3

You should consider using database mirroring in high-availability mode with the witness server role. With this configuration, you can ensure automatic switchover between servers in case of any server fails. You can also use high availability mode support to ensure integrity of data between servers.

Question 4

Which mode should you consider to use for the database mirroring session between the production site and the DR site?

Answer 4

Consider using the high-performance mode for the database mirroring session. This mode will help you ensure that all changes in the database are initially committed only to the primary site server. This mode ensures optimal application performance. These changes would then be replicated to the DR site asynchronously. The only disadvantage in using this mode is loss of data during failover clustering. A data loss may occur if the network connection between the production site and the DR site is slow.

Exercise 2: Implementing Database Mirroring

To start the virtual machine, perform the following steps:

1. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn On**.
2. Connect to the **50400A-NYC-SQL1** virtual machine by using Virtual Machine Remote Control Client (VMRC).
3. In the 50400A-NYC-SQL1 virtual machine, on the **Remote Control** menu, point to **Special Keys**, and then click **Send Ctrl+Alt+Delete**.
4. Log on to 50400A-NYC-SQL1, with the user name, **Administrator**, and the password, **Pa$$w0rd**, and then click **Forward**.

Task 1: Configure NYC-SQL1 for outbound connections.

1. On NYC-SQL1, on the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio window appears.
2. To log on to Microsoft SQL Server Management Studio, in the **Connect to Server** dialog box, type the following:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. On the **Start** menu, point to **All Programs**, click **Accessories**, and then click **Windows Explorer**.
5. In Windows Explorer, browse to the **D:\Labfiles\Mod11** folder, and then double-click the **E2_T1.sql** file.

6. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create an encryption key** code, and then click **Execute**.
7. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create a certificate** code, and then click **Execute**.
8. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create an endpoint** code, and then click **Execute**.
9. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Backup the certificate** code, and then click **Execute**.

Task 2: Configure NYC-SQL1\DEVELOPMENT for outbound connections.

1. In the Microsoft SQL Server Studio window, in the Object Explorer pane, click **Connect**, and then click **Database Engine**.
2. To log on to SQL Server Management Studio, in the **Connect to Server** dialog box, type the following:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1\DEVELOPMENT**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. In Windows Explorer, browse to the **D:\Labfiles\Mod11** folder, and then double-click the **E2_T2.sql** file.
5. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create an encryption key** code, and then click **Execute**.
6. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create a certificate** code, and then click **Execute**.

7. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create an endpoint** code, and then click **Execute**.
8. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Backup the certificate** code, and then click **Execute**.

Task 3: Configure NYC-SQL1\INSTANCE3 for outbound connections.

1. In the Microsoft SQL Server Studio window, in the Object Explorer pane, click **Connect**, and then click **Database Engine**.
2. To log on to SQL Server Management Studio, in the **Connect to Server** dialog box, type the following:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1\INSTANCE3**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. In Windows Explorer, browse to the **D:\Labfiles\Mod11** folder, and then double-click the **E2_T3.sql** file.
5. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create an encryption key** code, and then click **Execute**.
6. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create a certificate** code, and then click **Execute**.
7. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create an endpoint** code, and then click **Execute**.
8. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Backup the certificate** code, and then click **Execute**.

Task 4: Configure NYC-SQL1 for inbound connections.

1. In the Microsoft SQL Server Studio window, in the Object Explorer pane, click the **NYC-SQL1 (SQL Server 10.0.1600 – NYC-SQL1\Administrator)** database node.

2. On the **File** menu, point to **Open**, and then click **File**. The **Open File** dialog box appears.
3. In the **Open File** dialog box, browse to the **D:\Labfiles\Mod11** folder, click the **E2_T4.sql** file, and then click **Open**.
4. In the Query Editor pane, select the **Create a login account and add a user to the account** code, and then click **Execute**.
5. In the Query Editor pane, select the **Add certificates to the user account** code, and then click **Execute**.
6. In the Query Editor pane, select the **Grant the CONNECT permission to the endpoint** code, and then click **Execute**.

Task 5: Configure NYC-SQL1\DEVELOPMENT for inbound connections.

1. In the Microsoft SQL Server Studio window, in the Object Explorer pane, click the **NYC-SQL1\DEVELOPMENT (SQL Server 10.0.1600 – NYC-SQL1\Administrator)** database node.
2. On the **File** menu, point to **Open**, and then click **File**. The **Open File** dialog box appears.
3. In the **Open File** dialog box, browse to the **D:\Labfiles\Mod11** folder, click the **E2_T5.sql** file, and then click **Open**.
4. In the Query Editor pane, select the **Create a login account and add a user to the account** code, and then click **Execute**.
5. In the Query Editor pane, select the **Add certificates to the user account** code, and then click **Execute**.
6. In the Query Editor pane, select the **Grant the CONNECT permission to the endpoint** code, and then click **Execute**.

Task 6: Configure NYC-SQL1\INSTANCE3 for database mirroring.

1. In the Microsoft SQL Server Studio window, in the Object Explorer pane, click the **NYC-SQL1\INSTANCE3 (SQL Server 10.0.1600 – NYC-SQL1\Administrator)** database node.
2. On the **File** menu, point to **Open**, and then click **File**. The **Open File** dialog box appears.

3. In the **Open File** dialog box, browse to the **D:\Labfiles\Mod11** folder, click the **E2_T6.sql** file, and then click **Open**.
4. In the Query Editor pane, select the **Create a login account and add a user to the account** code, and then click **Execute**.
5. In the Query Editor pane, select the **Add certificates to the user account** code, and then click **Execute**.
6. In the Query Editor pane, select the **Grant the CONNECT permission to the endpoint** code, and then click **Execute**.

Task 7: Create a test database and a test table and add records to the table.

1. In the Microsoft SQL Server Studio window, in the Object Explorer pane, click the **NYC-SQL1 (SQL Server 10.0.1600 – NYC-SQL1\Administrator)** database node.
2. On the **File** menu, point to **Open**, and then click **File**. The **Open File** dialog box appears.
3. In the **Open File** dialog box, browse to the **D:\Labfiles\Mod11** folder, click the **E2_T7.sql** file, and then click **Open**.
4. In the **Microsoft SQL Server Studio** window, click **Execute**. The table is created with a warning message.

Task 8: Configure NYC-SQL1 for database mirroring.

1. In the Microsoft SQL Server Studio window, in the Object Explorer pane, click the **NYC-SQL1 (SQL Server 10.0.1600 – NYC-SQL1\Administrator)** database node.
2. On the **File** menu, point to **Open**, and then click **File**. The **Open File** dialog box appears.
3. In the **Open File** dialog box, browse to the **D:\Labfiles\Mod11** folder, click the **E2_T8.sql** file, and then click **Open**.
4. In the Query Editor pane, select the **Set recovery mode of the database to FULL** code, and then click **Execute**.
5. In the Query Editor pane, select the **Backup the database and the log file** code, and then click **Execute**.

Task 9: Configure NYC-SQL1\DEVELOPMENT for database mirroring.

1. In the Microsoft SQL Server Studio window, in the Object Explorer pane, click the **NYC-SQL1\DEVELOPMENT (SQL Server 10.0.1600 – NYC-SQL1\Administrator)** database node.
2. On the **File** menu, point to **Open**, and then click **File**. The **Open File** dialog box appears.
3. In the **Open File** dialog box, browse to the **D:\Labfiles\Mod11** folder, click the **E2_T9.sql** file, and then click **Open**.
4. In the **Microsoft SQL Server Management Studio** window, click **Execute**.

Task 10: Enable database mirroring.

1. In the Microsoft SQL Server Studio window, in the Query Editor pane for the NYC-SQL1\DEVELOPMENT instance, delete the existing code, and then type the following T-SQL code.

```
USE master

GO

ALTER DATABASE Mod11 SET PARTNER = 'TCP://nyc-
   sql1:8081';

GO
```

2. Click **Execute**.
3. In the Microsoft SQL Server Studio window, in the Object Explorer pane, right-click the **NYC-SQL1** database node, and then click **New Query**. A new Query Editor pane appears.
4. In the Query Editor pane for the NYC-SQL1 instance, type the following T-SQL code.

```
USE master

GO

ALTER DATABASE Mod11 SET PARTNER = 'TCP://nyc-
   sql1:8082';

GO
```

5. Click **Execute**.

6. In the Query Editor pane for the NYC-SQL1 instance, delete the existing code, and then type the following T-SQL code.

```
USE master

GO

ALTER DATABASE Mod11 SET WITNESS = 'TCP://nyc-
   sql1:8083';

GO
```

7. Click **Execute**.

Task 11: Perform an automatic and manual switch over.

1. In the Microsoft SQL Server Studio window, in the Query Editor pane for the NYC-SQL1 instance, delete the existing code, and then type the following T-SQL code.

```
USE master

GO

ALTER DATABASE Mod11 SET PARTNER FAILOVER;

GO
```

2. Click **Execute**.
3. In the Microsoft SQL Server Studio window, in the Object Explorer pane, under **NYC-SQL1 (SQL Server 10.0.1600 – NYC-SQL1\Administrator)**, expand **Databases**. Ensure that Mod11 is included in the database list and verify if it's functioning.
4. In the Object Explorer pane, under **NYC-SQL1\DEVELOPMENT (SQL Server 10.0.1600 – NYC-SQL1\Administrator)**, expand **Databases**. Ensure that Mod11 is listed and verify if it is disabled and involved in database mirroring session.

Task 12: Close the virtual machine and discard undo disks.

1. Close the Virtual Machine Remote Control window.
2. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn off machine and discard changes** and then click **OK**.

Lab 12: Designing a Backup and Recovery Strategy for SQL Server 2008

Exercise 1: Designing a Backup and Recovery Solution (Discussion)

Question 1

What backup strategy could you adopt for the CRM application database, assuming that you are using a Full Recovery Model? What schedule would you use for each backup type?

Answer 1

Consider creating a backup plan as follows.

Backup type	Frequency
Full backup	Perform a daily backup at 00:00.
Transaction log backup	Perform a transaction log backup anytime between six in the morning to 00:00. Consider performing a backup in every two hours.

Question 2

A system failure occurs on a Thursday at 10:30. This is caused because of a failure in the RAID drive that is used to store the database files and log files. However, all your database backup files are not stored locally. Hence, you can access the database backup for your application database. The hardware team has repaired the RAID drive. The RAID drive is now functioning properly. You are now required to recover the system. How should you perform the restore?

Answer 2

If you have scheduled to perform a full database backup everyday at 00:00, you need to restore the full backup since Thursday 00:00. You can then restore all the transaction log backups that are performed during six to ten in the morning sequentially.

Question 3

Your have created a backup plan that includes a full database backup and a transaction log backup. A failure in the RAID drive that is used to store the database and the transaction log files has lead to a system failure. The application team wants to know why you did not consider performing a tail log backup before the restore. What would you tell the application team?

Answer 3

A tail-log backup is only possible if the database is accessible or the database log files are undamaged. You should perform a tail-log backup when the database is online. In the current scenario, the disk drive has completely failed. Hence, the database file will no longer be available. This will make the tail-log backup impossible to perform.

Question 4

Your application team is unable to access a particular table. You use DBCC CHECKDB to verify the status of the database. You then realize that there are some integrity issues with a data page. What would you do to restore access to all the database tables?

Answer 4

You should perform an online restore of the data page by using the full backup. Ensure that you use the full backup that was performed the previous day. You should then perform a transaction log backup for the tail log. After performing the backup, restore all transaction log backups. Ensure that you restore all the backups that were performed since the full backup on the previous day.

Question 5

When you are testing the environment for your application, your application database server fails. The master and msdb databases get corrupt. And all the data is lost. What will be the impact of this data loss?

Answer 5

SQL Server would not start if the master database is not available. SQL Server Agent would not start if the msdb database is not available.

Question 6

To lessen the impact of the master and msdb data loss, on the SQL Server and the SQL Server Agent services, what should you do?

Answer 6

Perform regular backups for your system database. For instance, you can perform a backup every Saturday evening for all your system databases. However, you should perform a backup after you have modified your system configuration. This will help you ensure that even your recent data is backed up.

Exercise 2: Backing Up a Database

To start the virtual machine, perform the following steps:

1. On the **Virtual Server Administration** Web site, click **50400A-NYC-SQL1**, and then click **Turn On**.
2. Connect to the **50400A-NYC-SQL1** virtual machine by using Virtual Machine Remote Control Client (VMRC).
3. In the 50400A-NYC-SQL1 virtual machine, on the **Remote Control** menu, point to **Special Keys**, and then click **Send Ctrl+Alt+Delete**.
4. Log on to 50400A-NYC-SQL1, with the user name, **Administrator**, and the password, **Pa$$w0rd**, and then click **Forward**.

Task 1: Create and configure a test database.

1. On NYC-SQL1, on the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**.
2. To log on to SQL Server Management Studio, in the **Connect to Server** dialog box, type the following:
   - Server type: **Database Engine**
   - Server name: **NYC-SQL1**
   - Authentication: **Windows Authentication**
3. Click **Connect**.
4. In Windows Explorer, browse to the **D:\Labfiles\Mod12** folder, and then double-click the **E2_T1.sql** file.
5. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create a database** code, and then click **Execute**.

6. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Add three filegroups** code, and then click **Execute**.
7. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Add files to the filegroups** code, and then click **Execute**.
8. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Add a secondary log file to the database** code, and then click **Execute**.
9. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, select the **Create a table and add a constraint on the table** code, and then click **Execute**.

Task 2: Back up the system and the user databases.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
USE master;

BACKUP DATABASE master TO DISK = 'D:\Data\master.bak'
```

2. Click **Execute**.
3. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
USE master;

BACKUP DATABASE msdb TO DISK = 'D:\Data\msdb.bak'
```

4. Click **Execute**.
5. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
USE master;
```

```
BACKUP DATABASE model TO DISK = 'D:\Data\model.bak'
```

6. Click **Execute**.
7. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
USE master;

BACKUP DATABASE Mod12 TO DISK = 'D:\Data\Mod12.bak'
```

8. Click **Execute**.

Exercise 3: Restoring a Database by Using Online Operations

Task 1: Attach a database that contains a corrupt page.

1. In Windows Explorer, in the **D:\Labfiles\Mod12** folder, double-click the **E3_T1.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.
3. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
USE Mod12_Restore

GO

SELECT * FROM CallDetails
```

4. Click **Execute**.
5. Verify the error that occurs with the following error. Also, note the page number of the data page in which error occurs.

Msg 824, Level 24, State 2, Line 1

SQL Server detected a logical consistency-based I/O error: incorrect checksum (expected: 0xe8c32716; actual: 0xe8c3d8bc). It occurred during a read of page (1:171) in database ID 11 at offset 0x00000000156000 in file 'D:\Data\Mod12_Restore.mdf'. Additional messages in the SQL Server error log or system event log may provide more detail. This is a severe error condition that threatens database integrity and must be corrected immediately. Complete a full database consistency check (DBCC CHECKDB). This error can be caused by many factors; for more information, see SQL Server Books Online.

6. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
USE master
```

```
GO
DBCC CHECKDB ('Mod12_Restore')
```

7. Click **Execute**.

8. Verify the error that occurs with the following error.

Msg 8928, Level 16, State 1, Line 1

Object ID 2105058535, index ID 1, partition ID 72057594038976512, alloc unit ID 72057594043236352 (type In-row data): Page (1:171) could not be processed. See other errors for details.

Msg 8939, Level 16, State 98, Line 1

Table error: Object ID 2105058535, index ID 1, partition ID 72057594038976512, alloc unit ID 72057594043236352 (type In-row data), page (1:171). Test (IS_OFF (BUF_IOERR, pBUF->bstat)) failed. Values are 12716041 and -4.

Task 2: Perform an online restore of a data page.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
RESTORE DATABASE Mod12_Restore PAGE='1:171'
   FROM DISK='D:\Data\mod12_restore.bak'
   WITH NORECOVERY;
```

2. Click **Execute**.

3. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
BACKUP LOG Mod12_Restore
   TO DISK = 'D:\Data\Mod12_Restore_log.bak'
   WITH INIT, NO_TRUNCATE
```

4. Click **Execute**.

5. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
RESTORE LOG Mod12_Restore FROM
   DISK='D:\Data\Mod12_Restore_log.bak'

   WITH RECOVERY;
```

6. Click **Execute**.

Task 3: Verify if the data page is successfully restored.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
USE Mod12_Restore

GO

SELECT * FROM CallDetails
```

2. Click **Execute**. Note that no error will occur.
3. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
USE master

GO

DBCC CHECKDB ('Mod12_Restore')
```

4. Click **Execute**. Note that no error will occur.

Task 4: Attach a database that contains a corrupt file.

1. In Windows Explorer, in the **D:\Labfiles\Mod12** folder, double-click the **E3_T4.sql** file.
2. In the Microsoft SQL Server Management Studio window, click **Execute**.
3. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
USE Mod12_File
GO
SELECT * FROM CallDetails
```

4. Click **Execute**.
5. Verify the error that occurs with the following error. Also, note the page number of the data page in which error occurs.

Msg 824, Level 24, State 2, Line 1

SQL Server detected a logical consistency-based I/O error: incorrect checksum (expected: 0x68c6a712; actual: 0x68c61cb8). It occurred during a read of page (3:8) in database ID 12 at offset 0x00000000010000 in file 'D:\Data\Mod12_File_Data.ndf'. Additional messages in the SQL Server error log or system event log may provide more detail. This is a severe error condition that threatens database integrity and must be corrected immediately. Complete a full database consistency check (DBCC CHECKDB). This error can be caused by many factors; for more information, see SQL Server Books Online.

6. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
USE master
GO
DBCC CHECKDB ('Mod12_File')
```

7. Click **Execute**.
8. Verify the error that occurs with the following error.

Msg 8928, Level 16, State 1, Line 1

Object ID 2105058535, index ID 1, partition ID 72057594038976512, alloc unit ID 72057594043236352 (type In-row data): Page (3:8) could not be processed. See other errors for details.

Msg 8939, Level 16, State 98, Line 1

Table error: Object ID 2105058535, index ID 1, partition ID 72057594038976512, alloc unit ID 72057594043236352 (type In-row data), page (3:8). Test (IS_OFF (BUF_IOERR, pBUF->bstat)) failed. Values are 12716041 and -4.

Task 5: Perform an online restore of a filegroup.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
RESTORE DATABASE Mod12_File FILEGROUP='TableData'
   FROM DISK='D:\Data\mod12_file.bak'
   WITH NORECOVERY;
```

2. Click **Execute**.
3. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
BACKUP LOG Mod12_File
   TO DISK = 'D:\Data\Mod12_File_log.bak'
   WITH INIT, NO_TRUNCATE
```

4. Click **Execute**.
5. In the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
RESTORE LOG Mod12_File FROM
   DISK='D:\Data\Mod12_File_log.bak'
   WITH RECOVERY;
```

6. Click **Execute**.

Task 6: Verify if the data file is successfully restored.

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, delete the existing code, and then type the following T-SQL code.

```
USE Mod12_File
GO
SELECT * FROM CallDetails
```

2. Click **Execute**. Note the no error occurs.
3. In Query Editor, enter the following Transact-SQL code:

```
USE master
GO
DBCC CHECKDB ('Mod12_File')
```

4. Click **Execute**. Note that no error appears.
5. In the Microsoft SQL Server Management Studio window, click the **Close** button.
6. In the Microsoft SQL Server Management Studio message box, click **No** to close the window without saving the Query.

Exercise 4: Restoring a System Database

Task 1: Simulate a system database failure.

1. On the **Start** menu, point to **Administration Tools**, and then click **Services**. The Services window appears.
2. In the Services window, stop all SQL Server services.
3. In Windows Explorer, browse to **C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA**.

Note: Before starting next step, ensure that all SQL services are stopped.

4. Rename the **master.mdf** database file to **master_old.mdf**, the **MSDBData.mdf** database to **MSDBData_old.mdf**, and the **model.mdf** database to **model_old.mdf**.
5. Rename the **mastlog.ldf** database file to **mastlog_old.ldf**, the **MSDBLog.ldf** database to **MSDBLog_old.ldf**, and the **modellog.ldf** database to **modellog_old.ldf**.
6. In the Services window, right-click the **SQL Server (MSSQLSERVER)** service, click **Start** to verify that you are unable to start the **SQL Server (MSSQLSERVER)** service and then click **OK**.
7. On the **Start** menu, point to **Administrative Tools**, and then click **Event Viewer**. The Event Viewer window appears.
8. In the Event Viewer window, in the navigation pane, under **Event Viewer (Local)**, expand **Windows Logs**, and then click **Application**. Verify the error that occurs with ID 17113 with the following error:

Error 2(The system cannot find the file specified.) occurred while opening file 'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\master.m

df' to obtain configuration information at startup. An invalid startup option might have caused the error. Verify your startup options, and correct or remove them if necessary.

Task 2: Rebuild the system databases by using SQL 2008 setup.

1. On the **Start** menu, click **Run**, type **cmd**, and then click **OK**. The Command Prompt window appears.
2. At the command prompt, run the following commands.

```
cd\

cd C:\Program Files\Microsoft SQL Server\100\Setup
   Bootstrap\Release

setup /ACTION=REBUILDDATABASE /QUIET
   /INSTANCENAME=MSSQLSERVER
   /SQLSYSADMINACCOUNTS=Administrator /SAPWD=Pa$$w0rd
```

3. In Windows Explorer, browse to **C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Log**, and then double-click the **Summary.txt** file.
4. In the **Summary.txt** file, verify the following result:

Final result: Passed

5. Close the Command Prompt window.

Task 3: Configure SQL Server to start in the single user mode.

1. On the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, click **Configuration Tools**, and then click **SQL Server Configuration Manager**. The Sql Server Configuration Manager window appears.
2. In the Sql Server Configuration Manager window, click **SQL Server Services**.
3. In the Sql Server Configuration Manager window, in the work pane, right-click **SQL Server (MSSQLSERVER)**, and then click

Properties. The **SQL Server (MSSQLSERVER) Properties** dialog box appears.

4. In the **SQL Server (MSSQLSERVER) Properties** dialog box, on the **Advanced** tab, in the **Startup Parameters** box, type **–m;** in front of the existing startup options, and then click **OK**.
5. In the **Warning** box, click **OK**.

Note: If a warning message appears, click **OK** to ignore the message. Ensure that the **SQL Server (MSSQLSERVER)** service is running.

Note: If an error message appears, click **OK** and ensure that the **SQL Server (MSSQLSERVER)** service is running. If the **SQL Server (MSSQLSERVER)** service is not started, wait for few minutes and then start the **SQL Server (MSSQLSERVER)** service.

Task 4: Restore the master database.

Note: Before starting this task, ensure that only SQL Server (MSSQLSERVER) service is started. Please stop other SQL services, if they are running.

1. On the **Start** menu, click **Run**, type **cmd**, and then click **OK**. The Command Prompt window appears.
2. At the command prompt, run the **sqlcmd** command.
3. At the sqlcmd prompt, run the following T-SQL commands.

```
RESTORE DATABASE master FROM DISK =
   'D:\Data\master.bak' WITH REPLACE;

GO
```

Task 5: Restore the msdb database.

1. In the SQL Server Configuration Manager window, in the work pane, right-click **SQL Server (MSSQLSERVER)** and then click

Properties. The **SQL Server (MSSQLSERVER) Properties** dialog box appears.

Note: Ensure that the **SQL Server (MSSQLSERVER)** service is running. If the service is stopped, then start the **SQL Server (MSSQLSERVER)** service.

2. In the **SQL Server (MSSQLSERVER) Properties** dialog box, on the **Advanced** tab, in the **Startup Parameters** box, remove **–m;** from the existing startup options, and then click **OK**.
3. In the **Warning** box, click **OK**.

Note: If a warning message appears, click **OK** to ignore the message. Ensure that the **SQL Server (MSSQLSERVER)** service is running.

Note: If an error message appears, click **OK** and ensure that the **SQL Server (MSSQLSERVER)** service is running. If the **SQL Server (MSSQLSERVER)** service is not started, wait for few minutes and then start the **SQL Server (MSSQLSERVER)** service.

4. At the sqlcmd prompt, run the following commands.

```
RESTORE DATABASE msdb FROM DISK = 'D:\Data\msdb.bak'
   WITH REPLACE;

GO
```

5. Close the Command Prompt window.
6. Close the Sql Server Configuration Manager window.
7. Close the Event Viewer window.
8. Close the Services window.
9. Close the Log window.

Task 6: Close the virtual machine and discard undo disks.

1. Close the Virtual Machine Remote Control window.
2. On the Virtual Server Administration Web site, click **50400A-NYC-SQL1**, and then click **Turn off machine and discard changes** and then click **OK**.

Exercise 3: Designing a Failover Cluster

Task 1: Determine the design of an SQL failover cluster.

Question 1

Because of limited number of resources, you only have two Windows 2008 servers available to use as the cluster node. Your team member wants to know the pre-requisites for setting up the failover cluster. What would you suggest?

Answer 1

To setup the failover cluster, at minimum you need to prepare two new IPs, one for cluster and one for use with SQL cluster, a new computer name for SQL Server cluster and a shared storage.

Question 2

How would you setup the shared folder as a quorum to reduce the chances of occurrence of a single point of system failure?

Answer 2

You should consider using Node and Disk Majority as a quorum mode. In this quorum mode, nodes and a shared disk get votes. This configuration allows a loss of half the nodes, providing the disk witness is available, or over half the nodes are available without the disk witness being available.

Question 3

Initially, you were planning to use your company's storage area network (SAN) as a shared storage and a quorum. Your storage team has now informed you that the capacity of the SAN is full and it will take some time to expand its capacity. However, you must setup the cluster and release it to the application team to use before storage team can expand the storage capacity. What are the alternatives that you can consider?

Answer 3

You can consider using the Node and File Share Majority quorum mode instead of the Node and Disk Majority quorum mode. The Node and File Share Majority mode is essentially the same as the Node and Disk Majority mode, except that the shared disk in the Node and Disk Majority mode is replaced with a file-share witness. This removes the need for shared disk as quorum. You can consider an external storage that supports connecting to two servers at the same time, as a replacement for shared disks. For eg, you can consider an external storage like a NAS server that supports iSCSI or an external SCSI disk array.

Question 4

Your team is planning to have a real-time copy of the data available at the backup site to increase the availability of the database. However, the data stored on the backup site must always be available. Also, the data would be used by the Internet application in case the production server fails or the primary site fails. What would you propose to achieve this?

Answer 4

You can consider setting up a geographically dispersed cluster. But this would create a dependency on the shared storage that is being replicated to the backup site. If the SAN that is used does not support replication, you should consider using database mirroring. This database mirroring should replicate the database from SQL cluster to the standalone database server at the backup site.

Module 2: Deploying SQL Server 2008

Demonstration: How To Check System Requirements by Using the Upgrade Advisor

In this demonstration, you will see how to:

Check if the server meets the minimum requirements by using Upgrade Advisor

Key Points

The steps to check if the server meets the minimum requirements by using Upgrade Advisor are:

1. On the Start menu, click **My Company**, browse to **D:\SQL2008**, and then double-click **setup**. The SQL Server Installation Center window appears.
2. In the SQL Server 2008 Installation Center window, on the **Planning** page, click **Install Upgrade Advisor**. The Microsoft SQL Server 2008 Upgrade Advisor Setup wizard launches.
3. On the **Welcome to the Installation Wizard for Microsoft SQL Server 2008 Upgrade Advisor** page, click **Next**.
4. On the **License Agreement** page, click **I accept the terms in the license agreement**, and then click **Next**.

5. On the **Registration Information** page, click **Next**.
6. On the **Feature Selection** page, click **Next**.
7. On the **Ready to Modify the Program** page, click **Install**.
8. On the **Completing the Microsoft SQL Server 2008 Upgrade Advisor installation** page, click **Finish** to complete the installation.
9. On the Start menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server 2008 Upgrade Advisor**. The Microsoft SQL Server 2008 Upgrade Advisor window appears.
10. In the Microsoft SQL Server 2008 Upgrade Advisor window, click **Launch Upgrade Advisor Analysis Wizard**. The Microsoft SQL Server 2008 Upgrade Advisor Analysis Wizard launches.
11. On the **Welcome to the Upgrade Advisor for Microsoft SQL Server 2008** page, click **Next**.
12. On the **SQL Server Components** page, verify the server name, and then select the **SQL Server** check box, and then click **Next**.
13. On the **Connection Parameters** page, connect to your default instance.
14. On the **SQL Server Parameters** page, verify if the **All Database** check box is selected, and then click **Next**.
15. On the **Confirm Upgrade Advisor Settings** page, click **Run** to start the upgrade advisor.

Question: What is the role of Upgrade Advisor?

Question: Do you need to install SQL server 2008 before running Upgrade Advisor?

Lab 2: Deploying SQL Server 2008

Exercise 2: Preparing for the Upgrade

Scenario

You have a SQL Server 2005 installed in your environment and you need to upgrade it to SQL Server 2008 due to the requirements from the application team. You plan to use SQL Server Upgrade Advisor to check if the server is compatible with SQL Server 2008 before upgrade.

The main tasks for this exercise are as follows:

1. Start SQL setup.
2. Install SQL Server 2008 Upgrade Advisor.
3. Run SQL Server 2008 Upgrade Advisor.
4. View SQL Server 2008 Upgrade Advisor report.

Note: In this lab, the name of the SQL Server instance is MSSQLSERVER.

Task 1: Start SQL setup.

High-level steps

1. In the Windows Explorer, browse to the **D:\SQL2008** folder, and run the **setup** file and install:
   - .NET Framework 3.5 SP1
   - hotfix KB942288
2. Restart the machine.

Detailed steps

1. On the **Start** menu, click **My Computer**, browse to **D:\SQL2008**, and then double-click **setup**. The Microsoft SQL Setup 2008 Setup message box appears.

2. In the **Microsoft SQL Setup 2008 Setup** message box, click **OK** to install .NET Framework 3.5 SP1.
3. In the **Microsoft .NET Framework 3.5 SP1 Setup** dialog box, click **I have read and ACCEPT the terms of the License Agreement**, and then click **Install**.
4. Click **Exit** to close the .NET Framework 3.5 SP1 installer.
5. In the **Windows Update Standalone Installer** message box, click **OK** to install hotfix KB942288.
6. In the **Download and Install Updates** dialog box, click **Restart Now**.

Task 2: Install SQL Server 2008 Upgrade Advisor.

High-level steps

1. In Windows Explorer, browse to the **D:\SQL2008** folder, run the **setup** file and start the **Microsoft SQL Server 2008 Upgrade Advisor Setup** wizard.
2. Using the **Microsoft SQL Server 2008 Upgrade Advisor Setup** wizard, install SQL Server 2008 Upgrade Advisor.

Detailed steps

1. On the **Start** menu, click **My Computer**, browse to **D:\SQL2008**, and then double-click **setup**. The SQL Server Installation Center window appears.
2. In the SQL Server 2008 Installation Center window, on the **Planning** page, click **Install Upgrade Advisor**. The Microsoft SQL Server 2008 Upgrade Advisor Setup wizard launches.
3. On the **Welcome to the Installation Wizard for Microsoft SQL Server 2008 Upgrade Advisor** page, click **Next**.
4. On the **License Agreement** page, click **I accept the terms in the license agreement**, and then click **Next**.
5. On the **Registration Information** page, click **Next**.

6. On the **Feature Selection** page, click **Next**.
7. On the **Ready to Modify the Program** page, click **Install**.
8. On the **Completing the Microsoft SQL Server 2008 Upgrade Advisor installation** page, click **Finish** to complete the installation.

Task 3: Run SQL Server 2008 Upgrade Advisor.

High-level steps

1. Open the Microsoft SQL Server 2008 Upgrade Advisor window.
2. Using the Microsoft SQL Server 2008 Upgrade Advisor window, start the Microsoft SQL Server 2008 Upgrade Advisor Analysis Wizard.
3. Using the Microsoft SQL Server 2008 Upgrade Advisor Analysis Wizard, run SQL Server 2008 Upgrade Advisor, with the following details:
   - Instance name: MSSQLSERVER
   - Authentication: Windows Authentication

Detailed steps

1. On the **Start** menu, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server 2008 Upgrade Advisor**. The Microsoft SQL Server 2008 Upgrade Advisor window appears.
2. In the Microsoft SQL Server 2008 Upgrade Advisor window, click **Launch Upgrade Advisor Analysis Wizard**. The Microsoft SQL Server 2008 Upgrade Advisor Analysis Wizard launches.
3. On the **Welcome to the Upgrade Advisor for Microsoft SQL Server 2008** page, click Next.
4. On the **SQL Server Components** page, verify that the server name, and then select the **SQL Server** check box, and then click **Next**.

5. On the **Connection Parameters** page, verify the following settings, and then click **Next**:
   - Instance name: **MSSQLSERVER**
   - Authentication: **Windows Authentication**
6. On the **SQL Server Parameters** page, verify if the **All Database** check box is selected, and then click **Next**.
7. On the **Confirm Upgrade Advisor Settings** page, click **Run** to start the upgrade advisor.

Task 4: View SQL Server 2008 Upgrade Advisor report.

High-level steps

1. Open the Microsoft SQL Server 2008 Upgrade Advisor Report Viewer window and review the help window to upgrade a Database Server.

> **Results**: After completing this exercise, you should have installed SQL Server 2008 Upgrade Advisor and finished analyzing an existing SQL Server installation for upgrade with Microsoft SQL Server 2008 Upgrade Advisor.

Detailed steps

1. In the Microsoft SQL Server 2008 Upgrade Advisor Analysis Wizard, on the Upgrade Advisor Progress page, click **Launch Report**. The Microsoft SQL Server 2008 Upgrade Advisor Report Viewer window appears.
2. In the Microsoft SQL Server 2008 Upgrade Advisor Report Viewer window, under **Database Server**, expand all entries.
3. Under the first entry, click the **Tell me more about this issue and how to resolve it** link. The SQL Server 2008 Upgrade Advisor window appears.
4. In the SQL Server 2008 Upgrade Advisor window, review the help information displayed, and then close the SQL Server 2008 Upgrade Advisor window.

5. Close the Microsoft SQL Server 2008 Upgrade Advisor Report Viewer window.
6. In the Microsoft SQL Server 2008 Upgrade Advisor Analysis Wizard, click **Close**.

Exercise 3: Upgrading to SQL Server 2008

Scenario

After checking the server for compatibility, you confirm that all applications on your server are compatible with SQL Server 2008. You decide to perform an in-place upgrade because of the limited hardware available while will not be enough to perform side-by-side upgrade.

The main task for this exercise is to perform in-place upgrade.

Task 1: Perform in-place upgrade.

High-level steps

1. In Windows Explorer, browse to the **D:\SQL2008** folder, and open the **SQL Server Installation Center** window.
2. Using the **SQL Server 2008 Installation Center** window**,** perform in-place upgrade for the MSSQLSERVER instance with the **Full-text Upgrade** as **Rebuild**.

Result: After completing this exercise, you should have performed an in-place upgrade.

Detailed steps

1. In Windows Explorer, in the **D:\SQL2008** folder, double-click **setup**. The SQL Server Installation Center window appears.
2. In the SQL Server 2008 Installation Center window, on the **Installation** page, click **Upgrade from SQL Server 2000 or SQL Server 2005**. The **SQL Server 2008 Setup** dialog box appears.
3. In the **SQL Server 2008 Setup** dialog box, click **OK**.
4. In the **SQL Server 2008 Setup** dialog box, on the **Product Key** page, click **Enter the product key**.
5. In the **Enter the product key** box, type the product key, and then click **Next**.

6. On the **License Terms** page, select the **I accept the license terms.** check box, and then click **Next**.
7. On the **Setup Support Files** page, click **Install** to start the upgrade process.
8. In the Upgrade to SQL Server 2008 wizard, on the **Setup Support Rules** page, click **Next**.
9. On the **Select Instance** page, verify if **MSSQLSERVER** is clicked in the **Instance to upgrade** list, and then click **Next** to continue.
10. On the **Select Features** page, click **Next**.
11. On the **Instance Configuration** page, click **Next**.
12. On the **Disk Space Requirements** page, click **Next**.
13. On the **Server Configuration** page, click **Next**.
14. On the **Full-text Upgrade** page, click **Rebuild**, and then click **Next**.
15. On the **Error and Usage Reporting** page, click **Next**.
16. On the **Upgrade Rules** page, click **Next**.
17. On the **Ready to Upgrade** page, click **Upgrade**.
18. On the **Upgrade Progress** page, click **Next**.
19. On the **Complete** page, click **Close**.
20. Close the SQL Server 2008 Installation Center window.

Exercise 4: Performing Post-Upgrade Tasks

Scenario

You have completed the upgrade and need to configure the upgraded SQL Server to ensure the SQL Server works with your applications. You also need to check if it provides optimum performance.

The main tasks for this exercise are as follows:

1. Register SQL Server in Microsoft SQL Server Management Studio.
2. Update the SQL Server statistics.
3. Configure the firewall.

Task 1: Register SQL Server in Microsoft SQL Server Management Studio.

1. On the Start menu, click **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**. The Microsoft SQL Server Management Studio message box appears.
2. In the **Microsoft SQL Server Management Studio** message box, click **Yes**.
3. In the **Connect to Server** dialog box, enter the following credentials to log on to Microsoft SQL Server Management Studio:
   - Server type
   - Server name
   - Authentication
4. Click **Connect**.

Task 2: Update the SQL Server statistics.

High-level steps

1. In the Microsoft SQL Server Management Studio window, in the Query Editor pane, run a query to update SQL Server statistics.

Detailed steps

1. In the **Microsoft SQL Server Management Studio** window, click **New Query** to view the Query Editor pane.
2. In the Query Editor pane, type the following Transact-SQL (T-SQL) code.

```
sp_updatestats
```

3. Click **Execute**.
4. In the Microsoft SQL Server Studio window, click the **Close** button. The Microsoft SQL Server Management Studio message box appears.
5. In the **Microsoft SQL Server Management Studio** message box, click **No** to exit without saving the query.

Task 3: Configure the firewall.

High-level steps

1. Using Control Panel, open the Windows Firewall window.
2. Using the Windows Firewall window, add the sqlservr program to the **Allow a program through Windows Firewall** list.

Results: After completing this exercise, you should have successfully registered SQL Server in Microsoft SQL Server Management Studio, updated SQL Server statistics, and configured the Windows firewall to allow remote servers and clients to connect to SQL Server.

Detailed steps

1. On the Start menu, click **Control Panel**. The Control Panel window appears.
2. In the Control Panel window, double-click **Windows Firewall**. The Windows Firewall window appears.
3. In the Windows Firewall window, click **Allow a program through Windows Firewall**. The **Windows Firewall Settings** dialog box appears.

4. In the **Windows Firewall Settings** dialog box, on the **Exceptions** tab, click **Add program**. The **Add a Program** dialog box appears.
5. In the **Add a Program** dialog box, click **Browse**. The **Browse** dialog box appears.
6. In the **Browse** dialog box, browse to the C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Binn folder, click **sqlservr.exe**, and then click **Open**.
7. In the **Add a Program** dialog box, click **OK**.
8. In the **Windows Firewall Settings** dialog box, ensure that the **sqlservr** check box is selected, and then click **OK**.

Task 4: Close the virtual machine and discard undo disks.

1. For the virtual machine that is running, close the Virtual Machine Remote Control window.
2. In the Close box, click Turn off machine and discard changes, and then click OK.

Lab 11: Designing a High-Availability Solution for SQL Server 2008

Exercise 3: Designing a Failover Cluster

Scenario

Your application requires high-availability support to reduce the possibility of being unable to support mission critical applications. Your team has decided to implement failover cluster to provide the level of availability required by the application. You are now assigned the task to design this failover cluster.

The main task for this exercise is to determine the design of an SQL failover cluster.

Note: In this lab, the names of the SQL Server instances are NYC-SQL1 and MSSQLSERVER, and the name of the failover cluster node is NYC-FC.Woodgrove.com

Note: Before performing the following steps, you need to perform some preparatory tasks. Refer the Trainer Preparation Guide of this course for how to perform the preparatory tasks.

Task 2: Verify the setup for SQL Server failover cluster.

High-level steps

1. In the Microsoft SQL Server Management Studio window, by using the **Connect to Server** dialog box, connect to the **Cluster** server.
2. In the **Cluster** server instance, by using the Query Editor, run a T-SQL code to display all records from the **Production. Products** table.
3. Using the Connect to Server dialog box, try connecting to the **NYC-SQL1** server instance. Note that you are unable to connect to the **NYC-SQL1** server instance.
4. Open the Failover Cluster Management window.

5. Using the Failover Cluster Management Studio window, change the state of the SQL Server service in the SQL Server (MSSQLServer) instance to offline.
6. In the Query Editor pane, run a code to display all records from the **Production. Products** table. Note that the query execution fails.
7. Using the Failover Cluster Management Studio window, change the state of the SQL Server service in the SQL Server (MSSQLServer) instance to online.
8. In the Query Editor pane, run a code to display all records from the **Production. Products** table. Note that the query is executed successfully.

Detailed steps

1. In the Microsoft SQL Server Studio window, in the Object Explorer pane, in the Object Explorer pane, click **Connect**, and then click **Database Engine**. The **Connect to Server** dialog box appears.
2. In the **Connect to Server** dialog box, in the **Server name** box, type or select **CLUSTER**, and then click **Connect**.
3. In the Object Explorer pane, right-click the **CLUSTER** database node, and then click **New Query**.
4. In the Microsoft SQL Server Management Studio window, in the toolbar, in the database list, select the database.
5. In the Query Editor pane, type the following code, and then click **Execute**.

   SELECT * FROM <*table name*>

6. In the Object Explorer pane, click **Connect**, and then click **Database Engine**. The **Connect to Server** dialog box appears.
7. In the **Connect to Server** dialog box, in the **Server name** box, type or select **NYC-SQL1**, and then click **Connect**. The **Cannot connect to NYC-SQL1** error message appears.
8. In the **Connect to Server** message box, click **OK**.

9. In the **Connect to Server** dialog box, click **Cancel**.

Note: Connecting to the NYC-SQL1 server, helps you to verify that the SQL Server Cluster is accessible only by the SQL Cluster name. The SQL Server Cluster is not accessible by its Computer Name or IP Address.

10. On the **Start** menu, point to **Administrative Tools**, and then click **Failover Cluster Management**. The Failover Cluster Management window appears.
11. In the Failover Cluster Management window, in the navigation pane, expand the **Failover Cluster Management** node, then expand **NYC-FC.Woodgrovebank.com**, next expand **Services and Applications**, and then click **SQL Server (MSSQLSERVER)**. A summary of SQL Server (MSSQLServer) appears in the work pane.
12. In the work pane, in the **Resources and Applications** area, under **Other Resources**, right-click **SQL Server**, and then click **Take this resource offline**.
13. In the **Please confirm action** warning box, click **Take SQL Server offline**. Wait until the Offline status is displayed.
14. In the **Microsoft SQL Server Management Studio** window, click **Execute** to run the same code again to verify that the query execution fails.
15. In the Failover Cluster Management window, in the work pane, in the **Resources and Applications** area, under **Other Resources,** right-click **SQL Server**, and then click **Bring this resource online**. Wait until the Online status is displayed.
16. In the **Microsoft SQL Server Management Studio** window, click **Execute** to run the code again to verify that the code is executed successfully.

Task 3: Verify SQL Server Cluster configuration.

High-level steps

1. Using the Failover Cluster Management window, for the SQL Server (MSSQLServer) instance, open the **SQL Server Properties** dialog box.
2. Using the **SQL Server Properties** dialog box, by using the **Dependencies** tab, add a new row and specify the IP address as a dependency.
3. Using the **SQL Server Properties** dialog box, by using the **Policies** tab, change the maximum restarts in specified period value to **5**.
4. Using the Failover Cluster Management window, simulate failure of the SQL Server.
5. In the Failover Cluster Management window, verify that the SQL Server, **NYC-SQL1**, is listed as a node.

Results: After completing this exercise, you should have determined the design of an SQL failover cluster and verified the setup and configuration of the failover cluster.

Detailed steps

1. In the Failover Cluster Management window, in the work area for the SQL Server (MSSQLServer) service, in the **Resources and Applications** area, under **Other Resources**, right-click **SQL Server**, and then click **Properties**. The **SQL Server Properties** dialog box appears.
2. In the **SQL Server Properties** dialog box, on the **Dependencies** tab, under **Specify the resources that must be brought online before this resource can be brought online**, click in the row specifying **Click here to add a dependency**, and then in the **Resource** list, click the IP address.
3. On the **Policies** tab, in the **Response to resource failure** area, under **If resource fails, attempt restart on current node**, in the **Maximum restarts in the specified period** box, type or select **5**, and then click **OK**.
4. In the Failover Cluster Management window, in the work pane for the SQL Server (MSSQLServer) service, in the **Resources and Applications** area, under **Other Resources**, right-click **SQL**

Server, point to **More Actions**, and then click **Simulate failure of this resource**.

5. In the **Please confirm action** box, click **Simulate Failure of SQL Server**, and then wait until the SQL Server resource displays the online status.
6. Under **Other Resources**, right-click **SQL Server**, and then click **Properties**. The **SQL Server Properties** dialog box appears.
7. In the **SQL Server Properties** dialog box, on the **General** tab, ensure that **SQL Server** is listed in the **Resource Name** box, and then click **OK**.
8. In the Failover Cluster Management window, in the navigation pane, under the NYC-FC.Woodgrove.com node, click **Nodes**. Ensure that NYC-SQl1 is listed in the work pane.