Chapter No.1
"Getting Started with SQL Server 2008 R2"
In this package, you will find:
A Biography of the author of the book
A preview chapter from the book, Chapter NO.1 "Getting Started with SQL Server 2008 R2"
A synopsis of the book’s content
Information on where to buy this book

About the Author
Satya Shyam K Jayanty is an Independent Consultant working as Principal Architect & Director for D Bi A Solutions Limited based in Europe. He has consulted in a wide range of industries, including the stock exchange, insurance, tele-communications, financial, retail, and manufacturing sectors, among others. Satya's SQL specialty is performance tuning, data platform review, and designing high availability database solutions.

He graduated with a Master's degree in Economics and Commerce from Osmania University, Hyderabad (India) and earned an MBA—Systems from the University of West England.

For More Information:
In 1992, Satya started his career in the IT industry as a computer operator and assistant programmer, and then in 1996 transformed his career as a SYBASE DBA. He has been working with SQL Server (beginning from version 4.2) for more than 15 years and is an accomplished Microsoft SQL Server MVP since 2006. He is a regular speaker and SME volunteer at major technology conferences such as Microsoft Tech-Ed (Europe, India, and North America), SQL PASS (Europe and North America), SQL Bits—UK, and manages the Scottish Area SQL Server user group based in Scotland. He is also a moderator in most web-based SQL Server forums (Microsoft Technet and www.sqlserver-performance.com), a writer, a contributing editor, and blogs at www.sqlserver-qa.net, www.sqlserver-performance.com, and www.beyondrelational.com.

For More Information:
Microsoft SQL Server 2008 R2 Administration Cookbook

Microsoft SQL Server is a powerful relational database engine, widely used to store and maintain data in Enterprises of various levels—be they small, medium, or large. SQL Server delivers a rich set of integrated services and reporting features that enable you to do more with your data such as query, search, synchronize, report, and analyze. SQL Server 2008 R2 accelerates the capabilities to scale database operations and is a highly scalable data platform.

This practical cookbook will show you the advanced administration techniques for managing and administering a scalable and high-performance SQL Server 2008 R2 system. It contains over 70 practical, task-based, and immediately useable recipes covering a wide range of advanced administration techniques for administering a high-performance SQL Server 2008 R2 system and solving specific administration problems. It shows how the SQL Server 2008 R2 system can be taken further.

Packed with reusable, real-world recipes, the book starts by providing an overview of the server and specialized editions of SQL Server 2008 R2 technologies, features, and solutions. Next, it covers database administration and management principles with real-world examples. The book then dives deep into topics such as administration of core Business Intelligence Services, management of core SQL Server 2008 R2 technologies, availability, security and programmability, implementing new manageability features and practices, and maintenance and monitoring, among others.

What This Book Covers

Chapter 1, Getting Started with SQL Server 2008 R2, begins with SQL Server 2008 R2 version's new features and enhancements such as master data services, data-tier applications, and adding the service pack features using Slipstream technology. We will run through the designing best practice approach in upgrading to SQL Server 2008 R2 and take advantage of federated servers enhancements.

Chapter 2, Administrating the Core Database Engine, covers the new SQL Server 2008 R2 feature utility administration and enhancements within DBA tasks such as availability, security, performance monitoring, tuning, and automated administration practices. The recipes also emphasize the usage of SQL Server Agent scheduled jobs for ETL and replication processes.

Chapter 3, Managing the Core Database Engine, enables the user to produce a resilient data platform, which is possible with new features of SQL Server 2008 R2 such as Utility Control point, multi-server management, and implementing central management feature

For More Information:
enhancements. The recipes cover the key best practices that combine a streamline of services for deployment, migration, and management of data platform.

Chapter 4, Administering core Business Intelligence Services, emphasizes on the process and procedures in preparing, installing, and implementing scale-out deployment of core business intelligence services—Analysis Services (AS), Reporting Services (RS), and Integration Services (IS), and finally key practices in managing the ETL process efficiently.

Chapter 5, Managing Core SQL Server 2008 R2 Technologies, dives into the implementation phases of SQL Server 2008 R2 new features and essential steps in planning and implementing self-service BI services, SQLAzure connectivity, StreamInsight technologies, and deploying the master data services feature. These recipes involve the planning, design, and implementation of features that are important to the management of the core technologies.

Chapter 6, Availability and Programmability Enhancements, is the biggest among all the chapters, and it covers the key SQL Server 2008 R2 availability enhancements within failover clustering, database mirroring, log shipping, and replication. The recipes also highlight the programmability enhancements such as compressed storage feature, data partitioning solutions, sparse columns, spatial data storage methods, and management of data-tier applications.

Chapter 7, Implementing New Manageability Features and Practices, spotlights the new manageability features and practices such as auditing, security, compression, change tracking, policy-based management, transparent data encryption, implementing best practices analyzer, and PowerShell for SQL Server features.

Chapter 8, Maintenance and Monitoring, is based on the typical day-to-day tasks of a DBA. It will go through the aspects of maintenance and monitoring the best practices with real world examples. The recipes will help the DBAs to adopt best methods with available technologies to keep up the database functionality at peak performance.

Chapter 9, Troubleshooting, covers the brainier task of administration which helps the users to solve the problems proactively, and this chapter spotlights the essential chores of troubleshooting. The recipes begin with monitoring methods and various available features such as filtered indexes, table hints, and query hints, and perform parallel query processing methods by taking advantage of hardware and system resources.

For More Information:
Getting Started with SQL Server 2008 R2

In this chapter, we will cover:

- Adding SQL Server R2 Service Pack features using Slipstream Technology
- Designing a best practice approach in upgrading to SQL Server 2008 R2
- Working with Data-Tier applications
- Designing and adopting SQL Server 2008 R2 solutions
- Designing applications to use federated servers

Introduction

Microsoft SQL Server 2008 has opened up a new dimension within data platforms and SQL Server 2008 R2 has been developed on the areas of core Database Platform and rich Business Intelligence. On the core database environment, SQL Server 2008 R2 advances the new enhancements as a primary goal of scalability and availability for highly transactional applications on enterprise-wide networks. On Business Intelligence platforms, the new features that are elevated include Master Data Management (MDM), StreamInsight, PowerPivot for Excel 2010, and Report Builder 3.0. The SQL Server 2008 R2 Installation Center includes system configuration checker rules to ensure the deployment and installation completes successfully. Further, the SQL Server setup support files will help to reduce the software footprint for installation of multiple SQL instances.

For More Information:
This chapter begins with SQL Server 2008 R2 version's new features and enhancements, and adding the service pack features using Slipstream technology. Then an explanation towards how best the master data services can help in designing and adopting key solutions, working with data-tier applications to integrate development into deployment, and an explanation of how best the federated servers enhancement can help to design highly scalable applications for data platforms.

Adding SQL Server 2008 R2 Service Pack features using Slipstream technology

The success of any project relies upon the simpler methods of implementation and a process to reduce the complexity in testing to ensure a successful outcome. This can be applied directly to the process of SQL Server 2008 R2 installation that involves some downtime, such as the reboot of servers. This is where the Slipstream process allows other changes to the databases or database server. This method offers the extension of flexibility to upgrade the process as an easier part, if there are minimal changes to only those required for the upgrade process. The following recipe is prepared to enable you to get to know Slipstream.

Slipstream is the process of combining all the latest patch packages into the initial installation. The major advantage of this process is time, and the capability to include all the setup files along with service pack and hotfixes. The single-click deployment of Slipstream helps us to merge the original source media with updates in memory and then install the update files to enable multiple deployments of SQL Server 2008 R2.

Getting Ready

In order to begin adding features of SQL Server using Slipstream, you need to ensure you have the following in place:

- **.NET Framework 3.5 Service Pack 1**: It helps improvements in the area of data platform, such as ADO.NET Entity Framework, ADO.NET data services, and support for new features of SQL Server 2008 version onwards.
  

- **Windows Installer 4.5**: It helps the application installation and configuration service for Windows, which works as an embedded chainer to add packages to a multiple package transaction. The major advantage of this feature enables an update to add or change custom action, so that the custom action is called when an update is uninstalled.

For More Information:


- **SQL Server setup support files**: It installs SQL Server Native Client that contains SQL OLEDB provider and SQL ODBC driver as a native dynamic link library (DLL) supporting applications using native code APIs to SQL Server.

### How to do it...

Slipstream is a built-in ability of the Windows operating system and since the release of SQL Server 2008 Service Pack 1, it is included.

The best practice is to use Slipstream Service Pack as an independent process for Service pack installation, Cumulative Update patching, and Hotfix patching. The key step to Slipstream success is to ensure the following steps are succeeded:

1. The prerequisite steps (mentioned in the earlier sections) are completed.
2. In case of multiple language instances of SQL Server, we need to ensure that we download the correct service pack language from http://www.microsoft.com/downloads/en/ that suits the instance.
3. The Service Pack files are independent to each platform to download, such as X86 for 32-bit, X64 for 64-bit, and IA64 for Itanium platform.

To perform the Slipstream Service Pack process, you need to complete the following steps:

1. Create two folders on the local server: SQL2K8R2_FullSP and SQL2K8R2SP.
2. Obtain the original SQL Server 2008 R2 setup source media and copy to SQL2K8R2_FULLSP folder.
3. Download the Service Pack1 from Microsoft Downloads site to save in SQL2K8R2SP folder, as per the platform architecture:
   - SQLServer2008SP1-KB968369-IA64-ENU.exe
   - SQLServer2008SP1-KB968369-x64-ENU.exe
   - SQLServer2008SP1-KB968369-x86-ENU.exe

For More Information:
4. Extract the package file using Windows Explorer or using a command prompt operation, as shown in the following screenshot:

5. In case the platform consists of multiple SQL instances with different architectures, for instance SQL Server 2008 R2 Enterprise Edition 64-bit as a default instance and SQL Server 2008 R2 Standard Edition as a named instance, then make sure you download the relevant architecture file http://www.microsoft.com/downloads/en/ as stated previously and extract to relevant folders.

6. This is the first checkpoint to proceed further and the key to ensuring the original setup media is updated correctly.

7. Copy the executable and localized resource file from the extracted location to the original source media location using robocopy utility, which is available from Windows Server 2008 onwards:

For More Information:
8. Copy all the files except the module program file that is executed by various programs and applications in Windows operating systems.

9. It is important to ensure the correct architecture files are copied, such X64 and X86 related files.

10. In addition to the initial checkpoint, this additional checkpoint is required in order to ensure the correct path is specified that will be picked up by Slipstream during the setup of SQL Server 2008 R2 and Service Pack installation.

11. The defaultsetup.ini is the key to guide the Slipstream process to install the RTM version and Service Pack files. The file can be located within the SQL2K8R2_FullSP folder as per the architecture.

12. From Windows Explorer, go to the SQL2K8R2_FullSP folder and open the defaultsetup.ini file to add the correct path for the PCUSOURCE parameter.

13. The file can be located from the SQL Server setup folder location for the processor, for instance, the 32-bit platform the file is available from \servername\directory\SQL Server 2008 R2\X86 folder.

For More Information:
14. The previous screenshot represents the file existence within the server, to ensure that the matching SQL Server Product ID (license key) is supplied.

15. There is more attached to the process if the file does not exist, there is no harm to the Slipstream process, the file can be created at the original folder defined in the following steps.

16. It is essential that the license key (product ID) and PCUSource information is included as follows:

```plaintext
;SQLSERVER2008 Configuration File
[SQLSERVER2008]
PID="??"
PCUSOURCE=??
```

17. Now, the PCUSOURCE value should consist of the full path of Service pack files that are copied during the initial step, the entry should be as follows:

```
add PCUSOURCE="{Full path}\PCU".
```

18. The full path must include the absolute path to the PCU folder, for instance, if the setup files exist in local folder the path must be as follows:

```
<drivename>\SQLServer2008R2_FullSP
```

19. If that folder is shared out, then the full path must be:

```
\\MyServer\SQLServer2008_FullSP1
```

20. The final step of this Slipstream process is to execute the `setup.exe` from SQL2K8R2_FullSP folder.

**How it works...**

The Slipstream steps and installation process are a two-fold movement. Slipstream uses the **Remote Installation Services (RIS)** technology of Windows Server services to allow configuration management to be automated. The RIS process is capable of downloading the required files or images from the specific path to complete the installation process.

The SQL Server 2008 R2 setup runs a pre-check before preceding the installation. The **System Configuration Check (SCC)** application scans the computer where the SQL Server will be installed. The SCC checks for a set of conditions that prevent a successful installation of SQL Server services.

Before the setup starts the SQL Server installation wizard, the SCC executes as a background process and retrieves the status of each item. It then compares the result with the required conditions and provides guidance for the removal of blocking issues.

---

**For More Information:**

The SQL Server Setup validates your computer configuration using a System Configuration Checker (SCC) before the Setup operation completes using a set of check-parameters that will help to resolve the blocking issues. The sample list of check-parameters is as follows:

<table>
<thead>
<tr>
<th>Check Item</th>
<th>Description</th>
<th>User action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RebootRequiredCheck</td>
<td>Checks if a pending computer restart is required. A pending restart can cause a Setup to fail.</td>
<td>A computer restart is required. You must restart this computer before installing SQL Server.</td>
</tr>
<tr>
<td>OsVersionCheck</td>
<td>Checks if the computer meets the minimum operating system version requirements.</td>
<td>Ensure that the operating system version meets the minimum requirements for this product.</td>
</tr>
<tr>
<td>ThreadHasAdminPrivilegeCheck</td>
<td>Checks if the account running SQL Server Setup has administrator rights on the computer.</td>
<td>Ensure that the account that is running SQL Server Setup has administrative rights on the computer.</td>
</tr>
<tr>
<td>WmiServiceStateCheck</td>
<td>Checks if the WMI service has been started and is running on the computer.</td>
<td>Ensure that the Windows Management Instrumentation (WMI) service is running.</td>
</tr>
<tr>
<td>FacetPowerShellCheck</td>
<td>Checks if Windows PowerShell is installed. Windows PowerShell is a pre-requisite of SQL Server 2008 R2 Express with Advanced Services.</td>
<td>For installations of Microsoft SQL Server 2008 Express with Advanced Services, ensure that Windows PowerShell is installed.</td>
</tr>
<tr>
<td>FacetWOW64PlatformCheck</td>
<td>Determines if the Setup program is supported on the operating system platform.</td>
<td>This rule will block unsupported installations on the 64-bit platform.</td>
</tr>
<tr>
<td>SqlUnsupportedProductBlocker</td>
<td>Checks whether SQL Server 7.0 or SQL Server 7.0 OLAP Services is installed.</td>
<td>Remove SQL Server 7.0 or install SQL Server 2008 R2 on a different computer.</td>
</tr>
</tbody>
</table>

For More Information:
The following are some of the additional checks that SCC performs to determine if the SQL Server editions in an in-place upgrade path are valid:

- Checks the system databases for features that are not supported in the SQL Server edition to which you are upgrading
- Checks that neither SQL Server 7.0 nor SQL Server 7.0 OLAP Services is installed on the server

SQL Server 2008 or higher versions are not supported on the server that has SQL Server 7.0.

- Checks all user databases for features that are not supported by the SQL Server edition
- Checks if the SQL Server service can be restarted
- Checks that the SQL Server service is not set to **Disabled**
- Checks if the selected instance of SQL Server meets the upgrade matrix requirements
- Checks if SQL Server Analysis Services is being upgraded to a valid edition
- SCC checks if the edition of the selected instance of SQL Server is supported for 'Allowable Upgrade Paths'

**There's more...**

As the prerequisite process of Slipstream is completed, we need to ensure that the installation of SQL Server 2008 R2, Service Pack, and Hotfixes patches are applied with the setup steps. To confirm the workflow process is followed correctly from the folder `SQL2K8R2_FullSP`, double-click on `setup.exe` file to continue the installation of RTM version, Service Pack, and required hotfix patches.

While continuing the setup at the **Installation Rules** screen, the SCC rule checks for **Update Setup Media Language Compatibility** value, which should be **passed** in order to proceed, as shown in the following screenshot:

---

**For More Information:**

If you have failed to see the update setup media language rule, then the same information can be obtained once the installation process is completed. The complete steps and final result of setup are logged as a text file under the folder: `C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Log`. The log file is saved as `Summary_<MachineName>_Date_Time.txt`, for example, 'Summary_DBiA-SSQA_20100708_200214.txt'.

Designing a best practice approach to upgrading to SQL Server 2008 R2

The upgrade is the most important aspect of the SQL Server 2008 R2 platform management. To prepare for an upgrade, begin by collecting information about the effects of the upgrade and the risks it might involve. When you identify the risks upfront, you can determine how to lessen and manage them throughout the upgrade process.
Upgrade scenarios will be as complex as your underlying applications and instances of SQL Server. Some scenarios within your environment might be simple, other scenarios may prove complex. For instance, the existing data platform is hosted with high availability components such as failover clustering, database mirroring, and replication. Start to plan by analyzing upgrade requirements, including reviewing upgrade strategies, understanding SQL Server 2008 R2 hardware and software requirements, and discovering any blocking problems caused by backward-compatibility issues.

This recipe introduces you to the methods that need to be followed when you design an SQL Server upgrade process. It will also present the best practices scenario for pre-upgrade, during the upgrade, and post-upgrade tasks that are involved within the upgrade of current SQL Server instance to SQL Server 2008 R2 version.

**Getting Ready**

The upgrade is not restricted to databases. It is important for the upgrade project to consider the various tools, components, and services of SQL Server 2008 R2 and non-database components, such as:

- SQL Server Management Studio (SSMS)
- Business Intelligence Development Studio (BIDS)
- SQL Server Reporting Services (SSRS)
- SQL Server Analysis Services (SSAS)
- Data Mining
- Linked Server configuration
- Log Shipping servers
- Database Mirroring pair
- SQL Server Replication
- SQL Server Agent jobs
- DTS Packages
- SQL Server Integration Services (SSIS)
- Microsoft Desktop Engine (MSDE) or SQL Server Express edition

Further to the list of tools, components, and services, you need to include technical issues (if any) and decisions that are involved in an upgrade to SQL Server 2008 R2, in addition to recommendations for planning and deploying an upgrade. The upgrade processes include upgrade strategies (pre/during/post), test, and rollback considerations and upgrade tools.

For More Information:
The windows architecture upgrade is also essential such as an upgrade from a 32-bit to a 64-bit platform. Additionally, the upgrade from a Standalone server to Microsoft Clustering services is required:

- **Upgrading from SQL Server 2005**: Run the upgrade from the command prompt on each failover cluster node, or by using the Setup UI to upgrade each cluster node. If Full-text search and Replication features do not exist on the instance being upgraded, then they will be installed automatically with no option to omit them.

- **Upgrading from SQL Server 2000**: This is similar to upgrading from SQL Server 2005. You can run the upgrade from the command prompt on each failover cluster node, or by using the Setup UI to upgrade each cluster node. It's supported for 32-bit scenarios only. Failover cluster upgrades from SQL Server 2000 (64-bit) are not supported.

Before you install SQL Server 2008 R2, we should also run the SQL Server Best Practices Analyzer (BPA) against your current legacy instances of SQL Server. If bad or questionable practices exist, you can then address them before the upgrade, moving the fixes through test and into production. The BPA tool installation is a straight-forward process and this diagnostic tool performs the following functions:

- Gathers information about a server and installed SQL Server instances
- Determines if the configurations are set in accordance with the best recommended practices
- Produces a report on all configurations that indicates settings that differ from recommendations
- The report consists of any potential problems in the installed instance of SQL Server and recommends solutions to potential problems

No configuration settings or data is changed when you execute this tool.

**How to do it...**

The following steps are classified as the best practices approach in preparing the environment for an upgrade:

1. To obtain the relevant Best Practices Analyzer (BPA) tool that suits your legacy instances of SQL Server, refer to the following links:

For More Information:  

2. To install SQL Server 2008 R2 BPA tool the following tools are required:
   - PowerShell V2.0: review the requirements and download it from http://support.microsoft.com/kb/968929.


4. Upgrade strategies include two fundamental methods, they are: 'side-by-side' and 'in-place'. It's worth mentioning about additional conventional methods of upgrade, such as using Copy Database Wizard and manual Schema rebuild method with scripts.
   - Using the side-by-side upgrade requires the user to move all or some data from an instance of SQL server 2000, or SQL Server 2005, or SQL Server 2008 to a separate instance of SQL Server 2008 R2. The variations in this strategy include, one-server—the new instance exists on the same server as target instance); and two-servers—the new instance exists on a different server than the target instance.
   - Using the in-place upgrade will involve a direct upgrade of the previous version of SQL Server to SQL Server 2008 R2, where the older instance is replaced.
   - Using Copy Database Wizard to upgrade an SQL Server 2000 or SQL Server 2005 database, offers the advantage for the database to be available immediately, which is then upgraded automatically.
   - Using the Schema rebuild method with scripts is a manual operation, which requires individual script files to create database, tables, logins, users, and scheduled jobs. Additionally, these external components also require scripting, such as SSIS packages, Linked Server information, and database maintenance plans.

5. Documentation is the key to a successful upgrade. Everyone should work as part of a team. The planning process will begin with a document that stands as a communication to involve all the stakeholders and teams to complete the data platform upgrade process.

6. In case of any specific requirement from Business teams within the upgrade process, that information must be documented in an Upgrade document along with their contact information.

For More Information:
7. Using the documentation acts as a base to execute the upgrade during the deployment phase. The plan should be as detailed as possible, and you should store the resulting document or documents by using some form of change control, such as a source control system. In the rest of this section, we will detail these steps.

8. Finally, within that planning documentation, include the upgrade requirements in addition to the rationale for choosing an upgrade strategy (refer step 3) for each instance or class of instances. Use the rest of the plan to detail remaining issues.

9. Detail the steps required for taking the systems offline for a period of time and bringing them back online.

10. Upgrade the Advisor tool:
   - Upgrade Advisor is available in the \X86 (or x64, ia64)\redist\Upgrade Advisor folder of the SQL Server installation media.

Within this feature pack page choose the appropriate file that suits to the environment, X86, X64, or IA64.

11. Run the SQL Server 2008 Upgrade Advisor to determine potential blocking issues:
   - Deprecated features
   - Discontinued features
   - Breaking changes
   - Behavior changes

It analyzes objects and code within legacy instances to produce reports that detail upgrade issues, if there are any, organized by SQL Server component.

The resulting reports show detected issues and provide guidance about how to fix the issues, or work around them. The reports are stored on disk, and we can review them by using Upgrade Advisor or export them to Microsoft Excel for further analysis.

In addition to analyzing data and database objects, Upgrade Advisor can analyze Transact-SQL scripts and SQL Server Profiler/SQL Trace traces. Upgrade Advisor examines SQL code for syntax that is no longer valid in SQL Server 2008 R2.

For More Information:
Whether you choose an 'in-place' upgrade or a 'side-by-side' upgrade, you can still run Upgrade Advisor on your legacy systems. We can run Upgrade Advisor from a local or remote server. To execute from a Command Prompt window, we require a configuration file name as an input parameter as follows:

```
C:\Program Files\Microsoft SQL Server 2008 R2 Upgrade Advisor\UpgradeAdvisorWizardCmd.exe" -ConfigFile "C:\Documents and Settings\SSQA.net\My Documents\SQLServer 2008 R2 Upgrade Advisor Reports\MyServer\Config.xml"
```

### How it works...

As we discussed, the best practices approach for a server instance or database upgrade process, it is essential to understand how the process works when you adopt any two upgrade choices that are available in SQL Server 2008 R2 and their characteristics.

- **In-place upgrade**: Using the SQL Server 2008 Setup program to directly upgrade an instance of SQL Server 2000, or SQL Server 2005 to SQL Server 2008 results in the older instance of SQL Server being replaced. The number of servers used in 'In-place' upgrade is 1, which implies that all the steps within an upgrade are performed on the same server.

- **Side-by-side upgrade**: Using steps to move all or some data from an instance of SQL Server 2000 or SQL Server 2005 to a separate instance of SQL Server 2008. Inside the side-by-side upgrade strategy, we have two variations of how upgrade is processed:
  - **One server**: The new instance exists on the same server as the target instance
  - **Two servers**: The new instance exists on a different server than the target instance

Characteristics of an In-Place Upgrade vs. a Side-by-Side Upgrade are as follows:

<table>
<thead>
<tr>
<th>Process</th>
<th>In-Place Upgrade</th>
<th>Side-by-Side Upgrade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of resulting instances</td>
<td>One only</td>
<td>Two</td>
</tr>
<tr>
<td>Number of physical servers involved</td>
<td>One</td>
<td>One or more</td>
</tr>
<tr>
<td>Data file transfer</td>
<td>Automatic</td>
<td>Manual</td>
</tr>
<tr>
<td>SQL Server instance configuration</td>
<td>Automatic</td>
<td>Manual</td>
</tr>
<tr>
<td>Supporting tool</td>
<td>SQL Server Setup</td>
<td>Several data transfer methods</td>
</tr>
</tbody>
</table>
The important process of the upgrade is a collection of database objects required from the previous versions of SQL Server. During the upgrade process, Transact-SQL code objects are essentially passive. Whether the process of an 'in-place' upgrade or a 'side-by-side' upgrade is chosen, the end result will be the same as far as your Transact-SQL code is concerned.

If in case any external scripts required on the user database are not associated within Stored Procedures or Functions, then they will remain unchanged by a direct upgrade process. You must apply these scripts manually to ensure that the post-upgrade tasks for the user databases can be classified as completed.

Based upon the new server or existing server upgrade, it is a best practice to move any Transact-SQL external scripts to a new server, or correct references within your database to those scripts.

The Post Upgrade process is also equally important and it is easy to analyze how the new SQL Server 2008 R2 instance performs compared with your original SQL Server 2000, SQL Server 2005, or SQL Server 2008 instance. Download the RML Utilities for SQL Server from http://go.microsoft.com/fwlink/?LinkId=133157 under the Microsoft download site. These utilities stand as a suite of tools for load testing, workload replay, and performance analysis.

As soon as you have completed the upgrade tasks, you need to perform two important steps to ensure that the initial process of SQL Server upgrade is accepted. The two steps are:

1. Integrate the new SQL Server instance into the application and database server environment.
2. Application testing process, such as: Change connectivity settings to the new server, if a side-by-side upgrade is chosen.
3. Change authentication mode of upgraded SQL Server instance.
   - **Linked servers**: The current system might depend on linked server relationships and definitions that must be applied for an upgrade. The application might fail, if those linked servers are not defined and tested correctly.
   - **Logins**: All the required logins and users with relevant privileges on the database must be applied for an upgrade of databases. By using Transfer Logins task from SSIS, the logins can be transferred between a source and a destination SQL Server instance.
   - **Scheduled jobs**: The routine administrative tasks are coupled as jobs that are executed using SQL Server Agent service, which are stored in msdb system database, which may not be part of the usual upgrade method. In such cases, it is essential to script all the scheduled jobs on the source server and execute them on the destination server.

For More Information:
Getting Started with SQL Server 2008 R2

- **Imports and exports:** The legacy database system might receive data imports and can become the source of data exports. These imports and exports might use DTS, converted to SSIS, or use other tools. You have to isolate these requirements and make sure of the resulting upgraded instance's correct participation.

- **Components referring to older SQL Server versions:** If the user is selectively transitioning legacy SQL Server instances, make sure that the resulting instance of SQL Server 2008 has components that can still connect successfully to the older SQL Server versions.

- **Drivers required for changing to a 64-bit version of SQL Server:** These required drivers might include drivers for accessing other database systems and mainframes from a 64-bit server.

- **Patches, hotfixes, and cumulative updates:** After you upgrade to SQL Server 2008 from another edition of SQL Server, you must reapply any hotfix or service pack updates to the upgraded SQL Server instance. The process of Slipstream helps here to avoid spending time installing hotfixes or service pack patch upgrades.

4. Determine whether the upgrade was successful by using the following methods:
   - Upgrade to support SQL Server 2008.
   - Change connectivity settings.
   - Change authentication mode.
   - Accept the upgrade, and how it will make the "go/no-go" decision:

5. Verify tests to ensure applications using the upgraded database servers run as expected and required.

6. If available, enlist the support of the QA team to develop appropriate acceptance tests.

7. Determine exactly when and how a rollback to the legacy SQL Server might be required.

8. Test the rollback plan.

If the application connectivity tests are successful, then the next step is to test the reporting section. For instance, if the installation includes custom report items, assemblies, or extensions, you must re-deploy the custom components. If you deployed and used any custom extensions, or custom assemblies for reports with SSRS 2000 or SSRS 2005, you need to redeploy the extensions or assemblies for use with SSRS 2008 R2.

These two steps need not necessarily be sequential. For example, you might apply some acceptance criteria immediately to obtain a go/no-go decision. This could then be followed by integrating the new instance and applying the remaining set of acceptance tests.

For More Information:
Update statistics include:

- Rebuild indexes on user database tables
- Re-organizing indexes if it is a very large database
- Rebuild cubes
- Reconfigure log shipping
- Database mirroring
- Test a failover cluster
- Verify that SQL Server Agent jobs run correctly

**Backward Compatibility—Deprecate & Discontinued features**

Backward compatibility with earlier versions of SQL Server was a high priority from SQL Server 2008 version onwards. So in most cases, applications will behave as in the past. There are several features from SQL Server 2008 version onwards that are marked for removal in the next version of SQL Server.

After you upgrade, you should remove the usage of these features from existing applications and avoid them in new development work. Several features from earlier versions of the SQL Server Database Engine are not supported in SQL Server 2008 and SQL Server 2008 R2, so you must use replacement features for these.

With an in-place upgrade, the upgrade process handles all aspects of the upgrade, automatically by upgrading the metadata for each database found in SSAS 2005. However, for SQL Server 2000, the upgrade process will not automatically reprocess the upgraded databases. Each database must be fully processed after the upgrade to ensure users can access the data that is contained in each database.

If there is a frequent upgrade failure, then the easiest resolution is to reinstall SSAS 2000. Restore the installation to the state before the upgrade process was started. This ensures that all the data and configuration information that is needed to restore the existing installation is available.

The manual transfer of database objects needs to be performed with utmost care due to the nature of all database objects (inclusive within database and exclusive of database). The outer objects of SQL Server are also important for databases, such as the objects that you must transfer, which include the following:

- Data files
- Database objects
- SSAS cubes
- Configuration settings
- Security settings
- SQL Server Agent jobs
- SSIS packages

For More Information:

There's more.

The following Upgrade checklist will be a useful resource while upgrading the SQL Server platform:

### Planning for Upgrade

<table>
<thead>
<tr>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Obtain all the necessary leads pertaining to the SQL Server data platform (hardware), web and application environment (software) to document high-level and detailed information.</td>
</tr>
<tr>
<td>2. Highlight all the important resources required for pre-upgrade, during-upgrade, and post-upgrade tasks.</td>
</tr>
<tr>
<td>3. Highlight all the required application testing and business escalation resources.</td>
</tr>
<tr>
<td>4. Document implementation steps including the back-out plan tasks.</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Hardware and Software requirements for SQL Server 2008 R2.</td>
</tr>
<tr>
<td>2. Hardware considerations—existing and new.</td>
</tr>
<tr>
<td>3. Application compatibility—essential for third-party applications.</td>
</tr>
<tr>
<td>4. Ready with required environment for thorough testing at pre-upgrade and post-upgrade implementation.</td>
</tr>
</tbody>
</table>

### Strategic Decision

<table>
<thead>
<tr>
<th>Strategic Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Determine whether In-place or Side-by-side upgrade method is being used.</td>
</tr>
<tr>
<td>2. In case of failover clustering and/or database mirroring environment:</td>
</tr>
<tr>
<td>- Establish backup and recovery plan.</td>
</tr>
<tr>
<td>- Establish roll-back and roll-forward plan.</td>
</tr>
</tbody>
</table>

### Pre-Upgrade

<table>
<thead>
<tr>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In case of side-by-side upgrade, ensure that the required databases, jobs, logins, packages, and linked-server information are restored.</td>
</tr>
<tr>
<td>2. Ensure no errors are reported on environment (OS, SQL Server, and application).</td>
</tr>
<tr>
<td>3. Download and execute SQL Server 2008 R2 Upgrade Advisor.</td>
</tr>
<tr>
<td>5. Perform basic connectivity testing from the application.</td>
</tr>
</tbody>
</table>

For More Information:

### The Upgrade

<table>
<thead>
<tr>
<th>Documentation</th>
<th>1. Ensure to list out all the steps that will be implemented, a screenshot of each screen is an ideal concept to perform the upgrade by any user.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. Make sure to include all the checkpoints that will control the GO or NO-GO decision.</td>
</tr>
<tr>
<td></td>
<td>3. Every checkpoint should insist on the point of referring to relevant log files including SQL Server error logs.</td>
</tr>
<tr>
<td></td>
<td>4. Action a roll-forward plan by troubleshooting the issue wherever possible.</td>
</tr>
<tr>
<td></td>
<td>5. In case of serious issue of environment, invoke the no-go step to implement the back-out plan.</td>
</tr>
<tr>
<td>Environment</td>
<td>1. Obtain necessary passwords from safe.</td>
</tr>
<tr>
<td></td>
<td>2. Perform a health check on OS and SQL Server.</td>
</tr>
<tr>
<td></td>
<td>3. Perform the upgrade using SQL Server installation center tool.</td>
</tr>
<tr>
<td></td>
<td>4. Perform the system and user database backups, once the upgrade is completed successfully.</td>
</tr>
</tbody>
</table>

### Post-Upgrade

<table>
<thead>
<tr>
<th>Acceptance</th>
<th>1. Determine the application acceptance by performing end-to-end testing.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. Integrate the new instance to connect the live application.</td>
</tr>
<tr>
<td></td>
<td>3. Apply required security patches and hotfixes.</td>
</tr>
<tr>
<td></td>
<td>5. Review event viewer and SQL server error logs.</td>
</tr>
<tr>
<td></td>
<td>6. Perform</td>
</tr>
<tr>
<td>Documentation</td>
<td>1. Document any new steps and modify the existing steps wherever necessary.</td>
</tr>
<tr>
<td></td>
<td>2. Perform failover and failback tasks to ensure Disaster Recovery provision is not compromised.</td>
</tr>
</tbody>
</table>
Working with Data-Tier applications

Data-Tier Applications (DAC) is the newest addition in SQL Server 2008 R2 data management and Visual Studio 2010 development system. The DAC helps users to develop, deploy, and manage the data-tier portion of the applications in a flexible manner and more efficiently than ever before. The key output of DAC is automation and facilitation of database objects and SQL Server instance level objects within the lifecycle of database systems.

Database objects that fall under this category include:

- Object permissions
- Role membership (mappings between users and database roles)
- Extended properties
- Statistics
- Diagrams
- Plan guides

SQL Server instance-level objects in this category include:

- Linked servers
- SQL jobs
- Certificates
- Endpoints
- Credential objects
- User-defined policies and conditions
- Role membership (mappings between logins and server roles)

This recipe gives an insight of the DAC process, how to install it, and use it to deploy within the network.

Getting ready

The self-contained unit of deployment is the first step for the DAC process that enables the Developers and Database Administrators (DBA) to package the SQL Server objects, such as database and instance objects into a single-entity called Data-Tier Applications (DAC) package.

For More Information:
The official representation of DAC is represented by Microsoft as follows:

For the ease of understanding the Production deployment and Development extract process, it is represented with numbers—P1 for production and D1 for Development.

The build process of the DAC package can be accomplished using Data-Tier application project system in Visual Studio application. The data management process of the DAC package can be accomplished by using SQL Server Management Studio (SSMS). The DAC deployment installs a new database on the instance, creates the database objects, and creates the logins associated with the users of the database. If a previous version of the DAC is already available, then the DAC package can be used to upgrade the existing DAC instance to a newer version.

The following are few of the best practice items provided for DAC package deployment and extraction that are directed for Independent Software Vendors (ISV) and Internet (Web hosting) Service Providers (ISP):

- The database should contain up to a few gigabytes of data (for example, up to 10 GB).
- There are certain limitations and support restrictions for data-tier applications, such as the common database objects. They are:
  - Objects marked for deprecation, including defaults, rules, and numbered stored procedures

For More Information:
CLR objects and data types (such as Spatial, Geography, Geometry, Hierarchy ID data types, SQL assemblies, CLR stored procedures, and functions)
- User-defined aggregates and user-defined CLR types
- Partition schemes and partition functions
- XML schema collections, XML indexes, and spatial indexes
- Service broker objects
- FileStream columns
- Symmetric keys, asymmetric keys, and certificates
- DDL triggers
- Application roles
- Full-text catalog objects
- Extended stored procedures
- Encrypted objects (for example, encrypted stored procedures, views, functions, and triggers)
- Objects containing cross-database dependencies and linked server references
- Extended properties
- Linked Servers
- Synonyms

The Data-Tier Application is wrapped with schema and metadata.
- The schema consists of tables, views, constraints, stored procedures, users, logins, and file group
- The metadata consists of management policies and failover policies that map a database having a direct relationship with the relevant .dacpac definition file

There are no specific hardware or software requirements for data-tier applications. However, to manage the data-tier applications through the SQL Server Utility, users must consider the limitations imposed by the utility and managed instances.

Though the data-tier applications can be used to work with existing databases, or implement them as new releases, we need to know how to create data-tier applications in existing systems to register each database as a data-tier application.

For More Information:
The build process of the DAC package can be accomplished by using Data-Tier application project system in Visual Studio application.

The data management process of the DAC package can be accomplished by using SQL Server Management Studio (SSMS). The DAC deployment installs a new database on the instance, creates the database objects, and creates the logins associated with the users of the database.

If a previous version of the DAC is already available, the DAC package can be used to upgrade the existing DAC instance to a newer version.

How to do it...

The Extract Data-tier application process creates a DAC package from a database. To extract a DAC package, perform the following steps:

1. Register the instance of SQL Server. In the Object Explorer pane, click on Connect and follow the instructions in the Connect to Server dialog box.
2. In Object Explorer, select the node of the instance from which you want to extract a data-tier application and expand its Databases node.
3. Select the database for the extraction. Right-click the databases node, point to Tasks, and then click Extract Data-tier Application to launch the Extract Data-tier Application Wizard.
4. On the Set Properties page (shown in the next screenshot), review or change the properties and then click on Next. These properties are displayed in Visual Studio and in SQL Server Management Studio and are used as follows:
   - **Application Name**: It identifies the application. For example, if a database called FinanceDB serves the Finance application, the application name should be set to Finance. The application name is used when a data-tier application is upgraded. In order to upgrade a DAC V1 to a DAC V2, the application names of V1 and V2 must be identical.
   - **Version**: It's the version of the data-tier application. By default, the version number is 1.0.0.0.
   - **Description**: Optional.

For More Information:

Getting Started with SQL Server 2008 R2

- **Save to DAC package file**: It specifies the filename and path for the DAC package file. This file must end with the .dacpac extension.

![Set Properties](image)

5. This is where the retrieval of database objects and relevant logins with checks of DAC package can be created with a validation process.

6. The validation process ensures that all database objects are supported by a DAC, and all dependent objects are available in the database. For example, if a view depends on a table and the latter was dropped from the database, a DAC cannot be extracted.

7. When the validation completes, all issues and errors that prevent the creation of a DAC package appear in the summary tree (refer to the next screenshot). If there are no issues with the validation report, then click **Next** to create and save the DAC package file.

For More Information:

8. On the Build Package page, review the status of building the DAC package file. When the build is finished, click Finish to close the wizard. Finally, verify that a new DAC package file was created in the path you specified in step 4.

9. The extracted DAC package file can now be imported to a Data-tier Application project in Visual Studio. In addition, the DAC package can also be deployed to create a new data-tier application on an instance of SQL Server, or to upgrade an older version of an existing data-tier application.

In an ISV-based environment, if the referred .dacpac file is received from a third-party or an unknown source, then it is essential to review the file contents that contain the pre and post schema change scripts. The default practice must include verification of the package file contents with an anti-virus scanner.

For More Information:
How it works...

There are three key components of a data-tier application project initiation—SQL Server objects, a server selection policy, and custom scripts (sometimes referred to as additional or extra files).

In order to get started, define the schema of an application. Developers can import a DAC package into the project system, or they can select an instance of SQL Server and import objects from an existing database. Then, users can create and modify database objects and SQL Server instance objects that are associated with the database.

Programming users can utilize advanced code editing features (such as IntelliSense technology, auto-completion, code snippets, and the Transact-SQL compiler) to write Transact-SQL code for stored procedures, functions, users, and logins.

After adding tables to the data-tier application projects, developers can create a DAC package and deploy the project to a target instance of SQL Server to test the release. For the control deployments, a server selection policy enables developers to set prerequisites and conditions on the target instance of SQL Server where the data-tier application is deployed. This process helps to dictate prerequisite conditions on target instances.

The server selection policy helps developers to express and dictate deployment intents. Each time a DAC package is deployed, the policy is evaluated against the target to ensure that the instance meets the application requirements.

Custom deployment actions and data manipulation operations are also available in the data-tier application project through custom scripts. Users can provide Transact-SQL scripts and include them in the DAC package. While the system does not automatically execute or reason over custom scripts, users can manually execute these scripts during DAC deployment or upgrade.

The unpack functionality is provided to let users view the content of a DAC package and generate the Transact-SQL script that corresponds to the deployment or upgrade of a data-tier application. The best practice is to use the unpack feature before you deploy or upgrade a DAC package in production, to review the code and compare or identify changes.

There's more...

In order to simplify the management of SQL Server environments, the health of DAC resources (for example, CPU, and disk space) across multiple computers and instances can be viewed in a central interface called the Utility Explorer in SQL Server Management Studio.

The Utility Explorer displays aggregated data and information, enabling DBAs to easily obtain utilization reports and statistics for their SQL Server installations. Furthermore, in order to customize CPU and disk space performance reports, DBAs can use Utility Health Policies to tune and control views showing the DAC resource consumption.

For More Information:
Designing and adopting SQL Server 2008 R2 solutions

The design of a solution needs guidance and planning, that has a series of collection consisting of information such as infrastructure planning, design, and implementation process which leads the sequence of core decision points. Designing the SQL Server 2008 R2 is no different to the key steps mentioned precisely. It also provides a means of validation to design and deploy the key business decisions to ensure that the solution meets the near-to-clear requirements for both technology and business stakeholders.

The design and adoption program must include technologists, such as Architects, to have an operational grasp of the technology to make the best use of new features. Also, the business team involvement is a high-level need to ensure that they understand the technology decisions being made for development and support. This can affect the business.

Getting ready

SQL Server 2008 R2 contains new features within the Business Intelligence stack such as PowerPivot, StreamInsight, and Master Data Services. A series of technology adoption program needs to be designed from an IT and Business perspective. A step beyond the traditional technology provision and SQL Server 2008 R2, consists of a series of best practices to implement the real-world BI solutions to leverage the best use of technologies.

The adoption programs should be designed to help users understand how the new SQL Server features interact with established technologies such as SQL Server Analysis Services, or SQL Server Integration Services, or SQL Server Reporting Services. In this recipe, we will go through the required process in designing and adopting the SQL Server 2008 R2 solutions.

The guidance must include the activities of planning a design, understanding new features, and implementing them effectively. In this case, SQL Server 2008 R2 new features. The critical design elements of planning are project scope determination, roles of people (such as dividing the technology and business decision-maker levels and infrastructure adoption of SQL Server 2008 R2 technologies), core database engine, integration services, analysis services, reporting services, and master data services.

Each user and stakeholder's needs should be classified taking the following considerations into account:

- Organizations that are implementing SQL Server for a specific application
- Organizations that are implementing SQL Server to provide a business intelligence platform for managers and decision makers to conduct data integration, reporting, and analysis

For More Information:

Getting Started with SQL Server 2008 R2

- Organizations that are implementing SQL Server as a service that is expected to be available to multiple business units
- Organizations that are upgrading an existing SQL Server implementation by migrating to new hardware

How to do it...

The following set of processes is involved in designing and adopting specific SQL Server 2008 R2 solutions:

1. In a typical application fetching data from a database server approach, the SQL Server infrastructure is designed to support a specific application.
2. This approach involves gathering the requirements about the application, determining which SQL Server roles will be required, and optimizing the server design for those specific applications.
3. Another approach is to design Database as a Service that makes the easy availability of new features for general platform usage.
4. Now the important aspect of design: Before proceeding to the implementation of database infrastructure, the aspect of storage and performance requirements will need to be gathered.
5. So the core service for storing, processing, and securing data is catered through the database engine that supports both relational (OLTP) and analytical (OLAP) methods for desktop requirements to data warehouse databases.
6. If the application being implemented requires one or more OLTP databases, then the same process of collection of performance requirement must be repeated.

Other services that may be selected in later tasks could determine the need for the Database Engine, such as:

- **Reporting Services**: It requires access to SQL Server-based database server to store metadata; this metadata database can be on the same server as Reporting Services, or on another database server. Integration Services can store packages in the msdb database—which is the system database—or on the filesystem.
- **Storage location**: It will be determined by the outcome of the decision of whether or not a database may be required for Integration Services. The inner process of Integration Services are required, such as:
  - Merge data from heterogeneous data sources
  - Populate data warehouses and data marts
  - Cleanse and standardize data
  - Automate administrative functions and data loading

For More Information:
For Data Warehouse, SQL Server Analysis Services (SSAS) supports OLAP (online analytical processing) and data mining functionalities. This allows a database administrator to design and create multidimensional structures that contain data aggregated from other data sources, such as relational databases.

Analysis Services may be needed if the organization needs to rapidly access reports with varying degrees of granularity (for example, yearly totals, monthly totals, quarterly totals, and individual transactions) and requires that the yearly totals appear just as quickly as the daily totals.

An online transaction processing (OLTP) system would have to add up these values, while an SSAS system, in many cases, will already have the answers pre-calculated.

SQL Server Reporting Services (SSRS) delivers enterprise reporting functionality for creating reports that gather content from a variety of data sources, publishing the reports in various formats, and centrally managing their security and subscriptions. Reports can be delivered on a scheduled basis to users, accessed through a web interface from SharePoint, or from custom applications including Windows Forms applications that call through web services.

Reporting Services can be used to generate reports on OLTP databases, SSAS cubes, data warehouses, data marts, or third-party data sources such as flat files, Oracle databases, or web services.

**How it works...**

There are potential reasons why a database might not be located within an existing instance:

- **Regulatory requirements**: Corporate policies or government regulations may require separate storage for certain types of data or for certain parts of the organization.

- **Memory isolation**: There may be a database application with a memory leak. It's therefore desirable to place it in its own instance and cap the memory available to this instance to protect other databases on the server from being affected.

- **Fault tolerance**: Clustering is on the instance level; thus, it will be necessary to place a database in an instance with similar fault-tolerance requirements.

- **Authentication**: Authentication is set at the instance level to be either an SQL Server authentication or a Windows authentication. The new database's authentication requirements need to be compatible with the instance's authentication setting.

- **Security concerns**: Total database encryption can be used to add security to a database. The service master key is for all databases in an instance. To minimize exposure, if a key is compromised, it may be desirable to place the database in a separate instance.

- **Support requirements**: The vendor or developer may require that the database runs in a separate instance.

**For More Information:**

Two factors that may change the number of servers running SQL Server are:

- Whether or not scaling out will be implemented
- Whether fault tolerance will be implemented at the database level or instance level

Scaling out is the process of distributing an SQL Server load across multiple servers; thus, the decision to scale out will increase the number of servers required. Multiple scale-out options are available, and this section will provide a brief overview. Scale-out options require design changes that occur at the database level and are outside the scope of this guide. It is expected that any decisions relative to scaling out should be made in conjunction with the database administrator.

Note that a new server, by definition, is a new instance.

There's more...

The detailed information on the installation and configuration process is discussed in Chapter 5, Management of core SQL Server 2008 R2 technologies recipes:

- Designing Data Integration Solution for Master Data Management
- Designing and Deploying framework to use Master Data Services
- Troubleshooting SQL Server Master Data Services

Designing applications to use federated servers

The high level of performance is an immediate requirement in current application database systems. In this scenario, to manage an application as 'high-available' and establish a load balancing of processing load for each task across multiple servers is called Federated Servers. Adding a federation of database servers to a highly-available system offers the flexibility to manage the resources for mission-critical processes efficiently.

Getting ready

The application system that is always available (high-availability) is a key ingredient for the success of every enterprise which establishes a federation of database servers, thereby distributing the processing load across a group of servers by horizontally partitioning the data in an SQL Server database. These servers are managed independently, but cooperate to process requests on the database.

For More Information:
The Federated Server is made up of two systems; a 'program'—to access data and a 'database structure'—to store the data. Similarly, to achieve the high levels of performance required by a client-server application or a website on a 24/7 basis, then a multi-tier system is needed that can typically balance the processing load for each tier across multiple servers.

By design, SQL Server features can share the database processing load across a group of servers by horizontally partitioning the data in a database. These servers are managed independently, but cooperate to process the database requests from the applications; for example, a cooperative group of servers is called a federation.

Designing and implementing structured storage within the database—which is called partition—is the first step to planning a federated database design. To accomplish the concept of database partition, we should create the underlying objects, such as partition function and partition scheme. The partition function handles the mapping of the rows of a table to the set of partitions based on certain column values, which are called partitioning columns. A partition scheme handles the mapping of each partition specified by a partition function to a file group.

The important choice for partitioning column values can be determined by the extent to which the data is grouped logically. Grouping by date is an adequate approach for managing subsets of data. For example, the SalesHistory and SalesHistoryArchive tables are partitioned by the TransactionDate field. Each partition consists of one month, which enables the SalesHistory table to maintain the year's worth of transactions as current and the SalesHistoryArchive table to maintain the data older than one year. By partitioning the tables in this way, the database engine can offer the scalability to transfer the old data quickly and efficiently.

### How to do it...

Once we have decided the partition function and partition scheme, the implementation of the partitioned table or index is the key step. The steps for creating a partitioned table or index using Transact-SQL are as follows:

1. Create partition function (rules) that specify how a table or index is partitioned:

   ```sql
   CREATE PARTITION FUNCTION partition_function_name(input_parameter_type)
   AS RANGE [ LEFT | RIGHT ]
   FOR VALUES ( [boundary_value [ ,...n ] ] )
   [ ; ]
   ``

2. From the previous syntax, the key value is RANGE, you have to choose between RANGE RIGHT and RANGE LEFT. CREATE PARTITION FUNCTION [dbo].[PF_SalesMonths] (datetime) AS RANGE RIGHT FOR VALUES ('20110201','20110301','20110401','20110501','20110601','20110701','20110801','20110901','20111001','20111101','20111201');
3. The following list represents how an underlying table would be partitioned:

<table>
<thead>
<tr>
<th>Partition</th>
<th>1</th>
<th>2</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SalesDate</td>
<td>&lt;February 1, 2011</td>
<td>SalesDate&gt;= February 1, 2011 AND SalesDate&lt;March 1, 2011</td>
<td>SalesDate&gt;= December 1, 2011</td>
</tr>
</tbody>
</table>

If no RANGE option is mentioned, then RANGE LEFT is the default.

4. Create a partition scheme that specifies the placement of partitions of a partition function on file groups:

   CREATE PARTITION SCHEME PS_SalesRange AS PARTITION PF_SalesMonths TO ([PRIMARY], [FEBRUARY], [MARCH], [APRIL], [MAY], [JUNE], [JULY], [AUGUST], [SEPTEMBER], [OCTOBER], [NOVEMBER], [DECEMBER])

5. The scheme defines the function of where to put the partition data. This step enables the partition of the SalesHistory table based on the Date by month range.

6. Create a table or index using the defined partition scheme.

   CREATE TABLE SalesHistoryPartitionedTable (
   TransactionID int PRIMARY KEY,
   SalesDate DATETIME,
   ProductInt
   ) ON PS_SalesRange(ID)

These steps can be achieved using SQL Server Management studio:

For More Information:
The partitioned table approach is a common solution. The pattern of data and the relationship of such tables are important to route the federation.

To accomplish this task, it is essential to develop a list of the SQL statements that will be executed by the application during typical processing periods. The process of developing a list can be accomplished as follows:

- Differentiate the list into `SELECT`, `UPDATE`, `INSERT`, and `DELETE` categories
- Order the list in each category by the frequency of execution
- If the SQL statements reference the stored procedures, then use the base `SELECT`, `INSERT`, `UPDATE`, and `DELETE` statements from the stored procedure
- If you are partitioning an existing SQL Server database, you can use SQL Server Profiler to obtain such a list

The frequency of such SQL statements can be determined to sustain a reasonable estimate for an OLTP system where distributed partitioned views work best. Such systems are characterized by having individual SQL statements that retrieve relatively small amounts of data when compared to the types of queries in a decision support, or an OLAP system.

When each SQL statement references a small amount of data, just studying the frequency of each statement yields a reasonable approximation of the data traffic in the system.

However, many systems have some groups of SQL statements that reference a lot of data. You may want to take the additional step of weighing these queries to reflect their larger data requirements.

The routing rules must be able to define which member server can most effectively process each SQL statement. They must establish a relationship between the context of the input by the user and the member server that contains the bulk of the data required to complete the statement.

The applications should be able to take a piece of data entered by the user, and match it against the routing rules to determine which member server should process the SQL statement.

High-level performance cannot be achieved simply by deploying the Federated Server process; the database design and table normalization is essential to keep up the performance and scalability.

This is referred to as collocating the SQL statement with the data required by the statement, it is the key to design a partitioning scheme to be clear about what data belongs to each member table. The partitioning scheme can be designed as follows:

- The original table is replaced with several smaller member tables
- Each member table will have the same number of columns matching as the original table

For More Information:
Each column will have the same attributes as the column in the original table where the attributes, such as data type and collation must be matched

How it works...

- The process of federated servers is implemented with the design architecture of the User/Business/Data services tier.
- The initial tier of User Services is a set of thin clients that focus on managing the application user interface.
- The User Services tier calls the next tier to perform the business functions needed to support user requests.
- So the Business Services Tier consists of a set of 'COM+' components that encapsulate the business logic of the organization. The Business Services tier uses the next tier for any permanent data storage that has to be performed.
- The final stage of storage is the Data Services Tiers, such as SQL Server databases, which can store data in a permanent medium. This is also referred to as persisting the data.

For More Information:
Where to buy this book


Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.