HOW TO BOOST SQL SERVER BACKUPS WITH DATA DOMAIN

David Muegge
Principal Systems Engineer
RoundTower Technologies, Inc.

EMC²

Table of Contents

List of Tables

List of Figures

Code Conventions

This article discusses topics relating to T-SQL programming. When names of SQL tables, scripts and procedures are used in headings, text, and descriptions they will be displayed in `monospace font`.

Disclaimer: The views, processes or methodologies published in this article are those of the author. They do not necessarily reflect EMC Corporation's views, processes or methodologies.

Introduction

The relational database management system (RDBMS) is a common and critical software application. Based on technology invented in 1970, RDBMS rose to widespread use in the 1980s and 1990s. It is rare to find a business of any size that does not use some form of relational database.

In enterprise information technology (IT) environments there is generally an infrastructure team responsible for the operation of hardware, such as servers and storage. The team is also responsible for the operating system virtualization software layer. This team has responsibility for the reliability and integrity of the environment, as well as for the protection of the data it stores. Additionally, there are often one or more database administrators (DBAs) who manage the many tasks related to the data stored in relational databases. This team is concerned with the accessibility, availability, and integrity of this data, which typically includes data protection. Since both teams have a shared interest in protection of the enterprise relational databases, this leads to debate on how this important information should be protected.

There are many backup product choices available and the major database vendors, such as Microsoft and Oracle, have native tools for backup and recovery. A frequent debate seen in organizations and across the IT industry is whether to use a traditional backup application or use native tools provided by the database vendor.

EMC provides multiple product options to accomplish backups of relational databases. These products are among the debated approaches to relational database backup. However, EMC Data Domain® is often the preferred choice for a disk-based backup and recovery storage solution regardless of backup software technology. Microsoft SQL Server is a leading RDBMS and is widely deployed in many enterprises today. Data Domain is often used as a backup target for SQL Server.

This article will discuss the common dilemma and approaches around Microsoft SQL Server database backups. It will provide information on how backup administrators and DBAs utilize Data Domain for SQL Server backups and introduce Data Domain Boost for Microsoft Applications (DDBMA) as a solution that can enable both teams to meet their requirements. The article aims to provide the SQL Server DBA with a foundation of knowledge for using DDBMA, including detailed guidance on how the DBA can maximize the usefulness of DDBMA using a freely available open source scripting toolkit. Topics include how to use the scripting toolkit with Transact-SQL query language (T-SQL), SQL Agent jobs, SQL Maintenance Plans, and Data Domain.

SQL Server Database Backup

The Common Dilemma

In many organizations, there is a challenge determining the best way to protect SQL Server databases. There are commonly two teams concerned with protecting SQL Server databases; backup administrators and SQL Server DBAs. Most backup administrators prefer to use traditional backup applications and most SQL Server DBAs prefer to use the native backup tools. The debate between the teams is usually grounded in the lack of understanding each team has of the other's requirements.

The sections below provide information to better understand the point-of-view of each team.

Backup Administrators View of SQL Server Backups and Restores

The backup administrator is generally concerned with a variety of requirements such as:

- Backup and restore reliability
- Backup and restore speed
- Software ease of use
- Media and device support
- Media management capabilities
- Operating system and application support
- Monitoring, notifications, and reporting

Many traditional backup applications provide these features and are the preferred tool for backup administrators. Since SQL Server backups are only one part of the infrastructure they are required to protect, the backup strategy they choose must include all of the above items for multiple systems and applications.

DBAs View of SQL Server Backups and Restores

The SQL Server DBA is generally concerned with protection of the SQL databases, but also concentrates on other processes that use backup and restore operations. The DBA is usually focused on the requirements listed below:

- Backup and restore reliability
- Backup and restore speed
- Software ease of use
- Automation capabilities
    - Integration with other SQL Server processes
    - T-SQL scripting
- Monitoring, notifications, and reporting

While DBAs have some of the same concerns as backup administrators, they view the problem as one best solved by the native SQL Server tools. Often, DBAs view traditional backup applications as unnecessary, unreliable, and difficult to integrate with other database maintenance processes.

The next section will discuss three of the most common approaches to SQL Server data protection.

The Common Approaches

There are various approaches to SQL Server backups. The three approaches seen most often in IT environments will be discussed.

Approach I: Backup Administrator Owns All Backups and Processes

In this approach, a traditional backup application is used to perform all backups and restores of SQL Server databases. The backup administrator manages all backup and restore operations. This method is common in smaller IT environments without dedicated DBAs, or in environments that do not engage in application development or customization. In this case, more complex operations performed by DBAs, such as refreshing development and testing databases, are not required. Figure 1 illustrates how this approach is generally deployed.

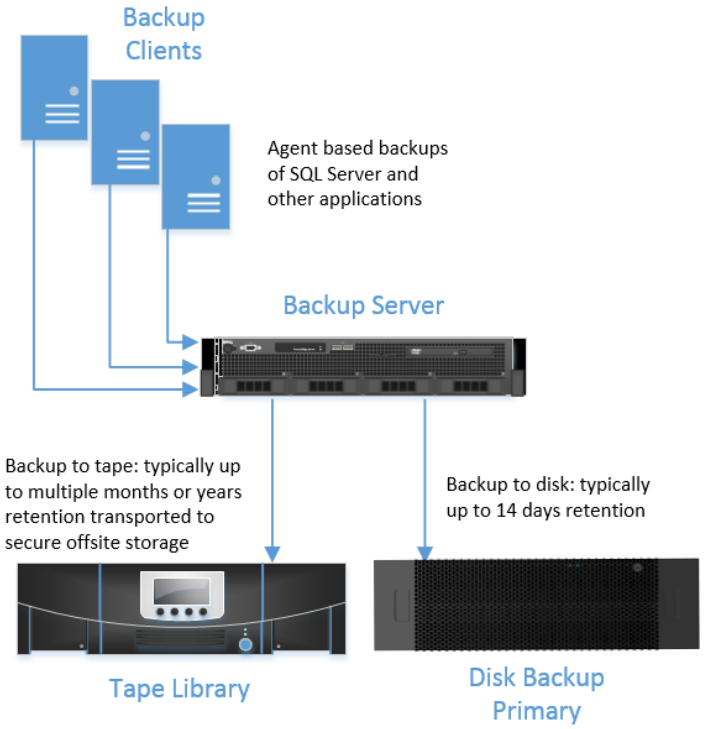


Figure 1: Traditional Backup Application Managed by Backup Administrator

In this scenario only periodic full and incremental backups are required and more complex requirements are not needed. Commonly debated pros and cons of this approach are:

Pros

- Centralized reporting and scheduling of all backups
- Integrated catalog and media management
- Delegated backup and restore functionality via role-based access control
- Deduplicated and compressed backup data
- Widely supported operating systems and applications
- Agents to provide application consistent backups

Cons

- DBAs are not able to use familiar tools
- DBAs will tend to perform additional backups to disk
    - Increases storage space requirements
- DBAs must be concerned with how primary backups are handled when completing operations due to backup chain and scheduling
    - Restoring production to development or testing databases
    - Log shipping to standby servers
- Complexity added due to the necessary integration of backups with maintenance tasks

This approach presents challenges for the DBA as the complexity of the SQL Server environment increases. This generally results in the DBA performing backups outside of the normal procedures, which duplicates data. This practice can also cause issues with the backup chain of full and incremental or differential backups, which can jeopardize the reliability of restores. These situations can result in duplicated effort, inefficient use of resources, reduced reliability, and higher costs for the organization.

Approach II: DBA Owns Database Backups and Processes

In this approach, the SQL Database Administrator owns all backups and processes related to SQL Server databases. This method is common in more complex SQL Server environments where backup processes must be integrated with other SQL Server database maintenance tasks, such as rebuilding indexes and updating database statistics. Figure 2 illustrates how this approach is generally deployed.

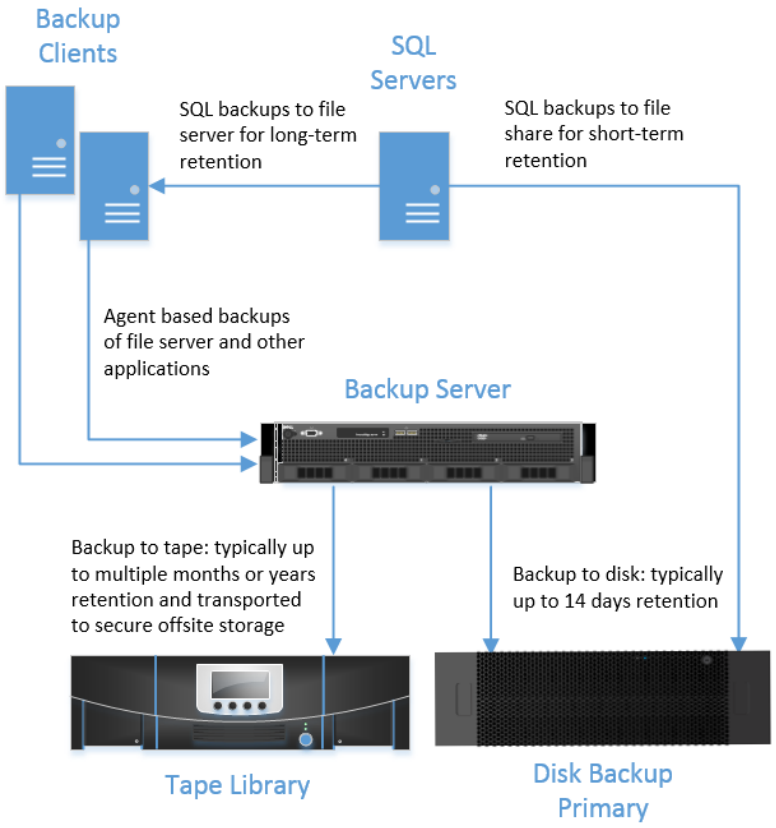


Figure 2: Native SQL Server Backups Managed by Database Administrator

DBAs often script backup and maintenance tasks together in SQL Agent jobs and Maintenance Plans, which provides automation and notification. It is also common for the DBA to place all backup files from SQL Server on a file server to be picked up later by a traditional backup application.

Pros

- DBAs have greater level of control over backup processes
- DBAs can use backups in other processes, such as log shipping for standby and disaster recovery
- DBAs greater flexibility to utilize backup and restore in software development lifecycle
- Integration with other database maintenance processes is increased
- Allows for rapid ad-hoc backup and restore functionality

Cons

- Monitoring and reporting of backups is not centralized
    - Additional work for audits
- Data needs to be backed up twice to write to tape
    - Additional tape backup hardware could be installed in SQL servers, but this adds complexity and is difficult in virtualized environments
- SQL Server native backup tool lacks a backup catalog and media management capabilities

This approach provides DBAs the control and flexibility they desire, but has shortcomings for the infrastructure team. They have less visibility into SQL Server backups and face difficulties with reporting and audits. This method also presents challenges for tape media management, as SQL Server backups are typically only files included in the file server backup where disk-based SQL backups are written.

Approach III: Backup Responsibilities Shared Between Backup Administrators and DBAs

In some cases, responsibilities for SQL Server data protection is shared between two teams and duplication of effort and data is accepted to meet the needs of both teams. While this approach can balance the pros and minimize the cons of the previous two approaches mentioned, there can still be a few challenges.

- Monitoring and reporting is done by multiple systems
- Backup chain can be more difficult to maintain
    - Restores can be more difficult and less reliable
- Complexity is generally increased
- Increased time to complete backups for long-term storage

While the teams will still face challenges with this approach, it is very common to see some form of shared approach is being used in environments where SQL Server is used extensively.

Figure 3 illustrates how this approach is generally deployed.

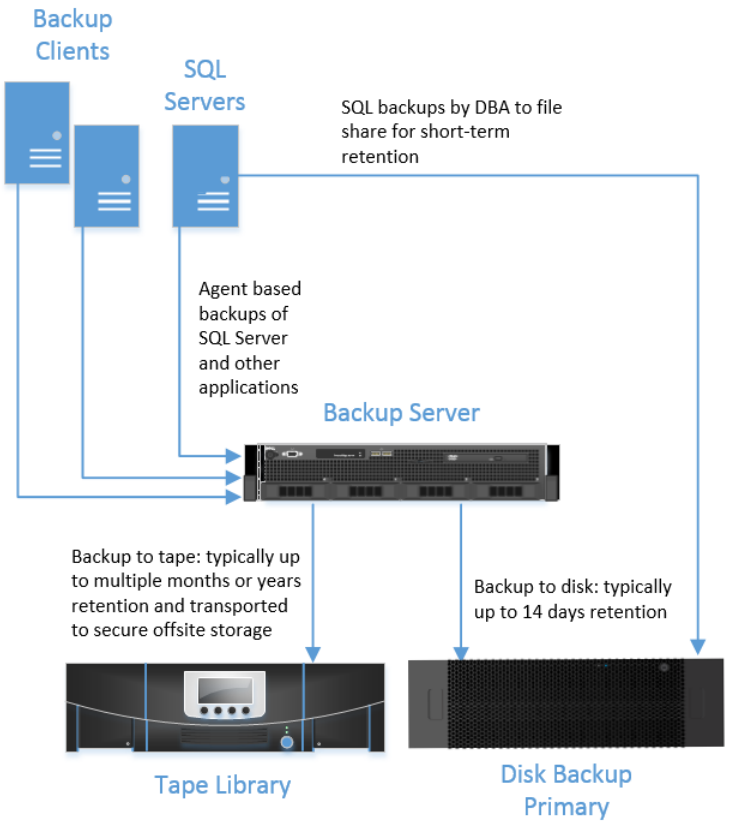


Figure 3: Traditional Backup Application and Native SQL Server Backup with Shared Responsibility

There are ways to use this approach successfully, but good communication between the teams and effective use of various tools is required. An important element in using this approach is to have a thorough understanding of both approaches to SQL Server backup and the tools used by each team. By having this understanding, risks to the SQL Server backup chain can be reduced to maintain restore reliability with minimal duplication of effort and resources. This will also allow both teams to accomplish their respective goals.

The next section will talk about Data Domain; how it is being used in environments today for SQL Server backups, and how it can be used to maximize backup productivity and minimize backup data storage requirements.

Data Domain and SQL Server Backups

Data Domain is a purpose-built backup appliance used by many organizations to store backup data. It has many benefits, including data deduplication and optimized replication. This allows for optimal space utilization in required disk storage and provides an efficient mechanism to move data offsite. Backup data is deduplicated and compressed prior to replication over the network.

The Data Domain appliance can be used as a Common Internet File System (CIFS)/Server Message Block (SMB) or Network File System (NFS) file share. Data Domain can also be integrated into various traditional backup applications. This makes it an excellent option for enterprises to use as a target for a wide range of backup applications.

How Data Domain is Being Used Today for SQL Server

Data Domain is being used in many environments to protect SQL Server data. These environments generally use one of the common approaches discussed in the previous section. One critical factor to consider before choosing an approach is whether the use of tape media is required. While Data Domain can use an archive tier and provide long-term retention of backup data, some organizations prefer to use tape media if long-term retention is required. The preference to use tape media is most commonly due to the portability of tape. Writing SQL Server database data to tape is typically accomplished in one of two ways when Data Domain is used.

The typical backup workflow is similar to that shown in Figure 4 if all backups are being completed with a traditional backup application.

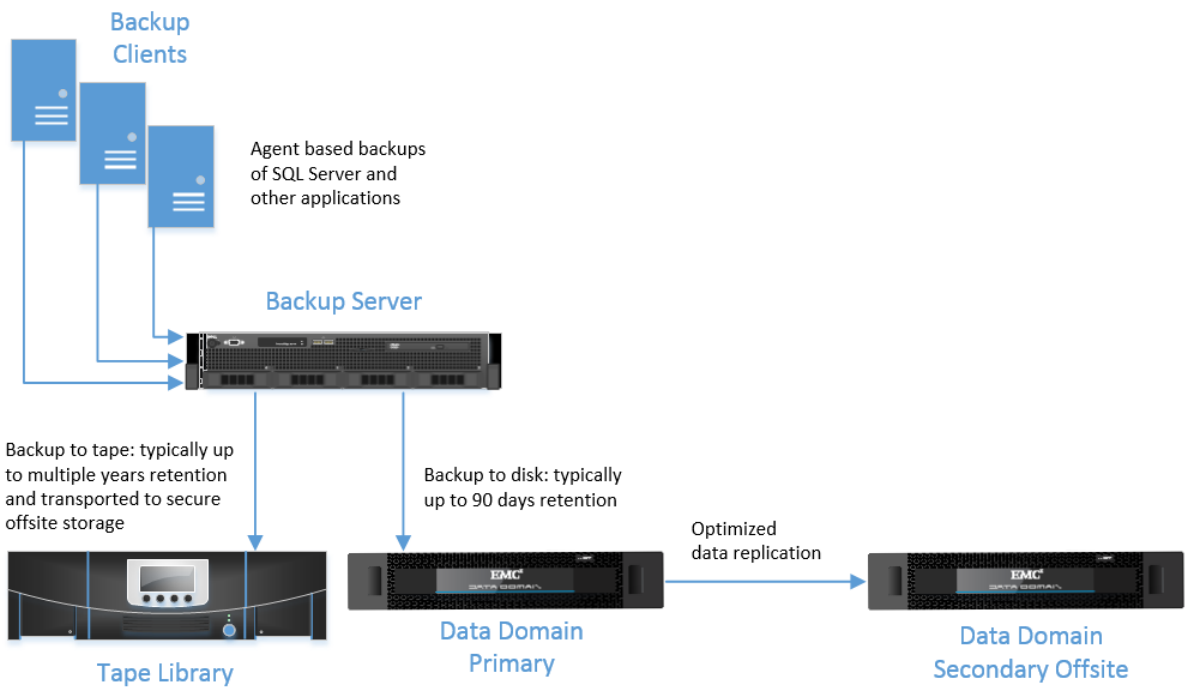


Figure 4: Traditional Backup Application to Data Domain and Tape

This method usually has the most benefit for the backup administrator and greatest drawback for the database administrator as discussed in the previous section.

If SQL Server backups are being completed by the DBA and tape is required for long-term retention, it is common to utilize the Data Domain appliance as a file server for short-term retention backups (typically 30-90 days). The long-term retention backups are sent to a staging area on a different file server to be backed up and sent to tape by a traditional backup application. This backup workflow is generally similar to that shown in Figure 5.

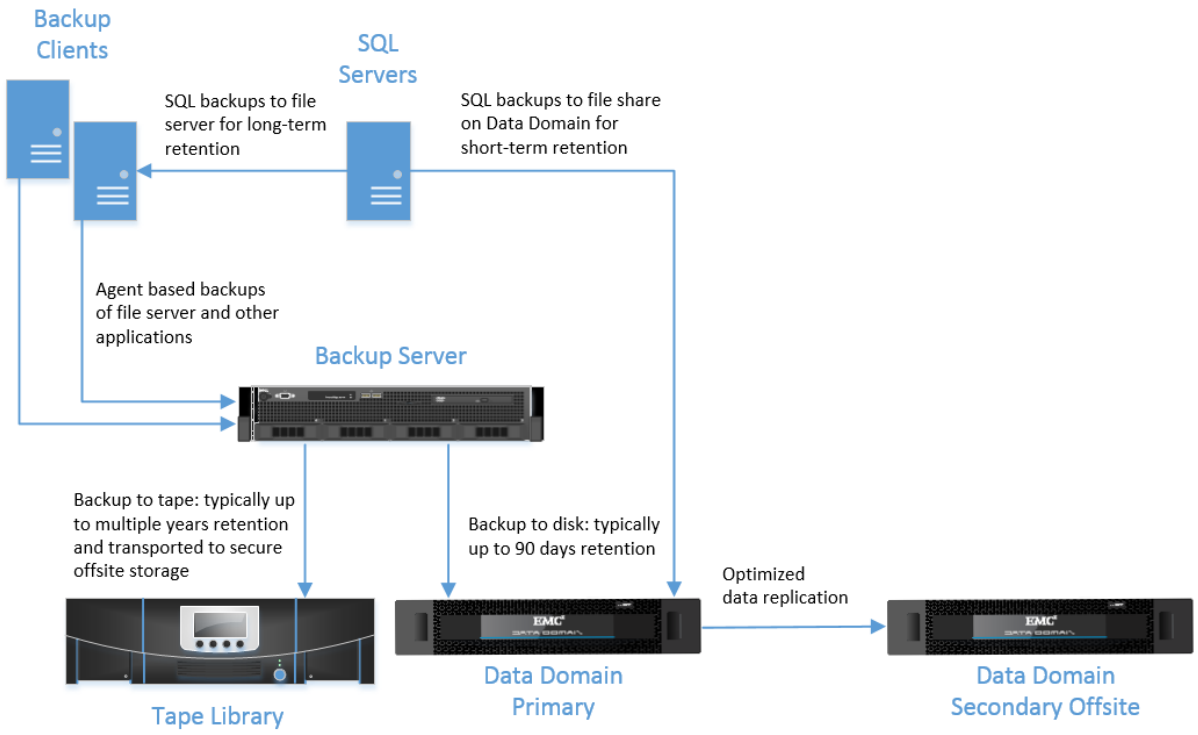


Figure 5: Native SQL Server Backups to Data Domain and Tape

There is another common scenario with SQL Server backups and Data Domain similar to the diagram above, where long-term retention is not required, which allows tape to be completely eliminated from the environment. This practice is becoming much more popular because it simplifies the architecture and eliminates the effort required for media management.

This backup workflow with tape eliminated is generally similar to that shown in Figure 6.

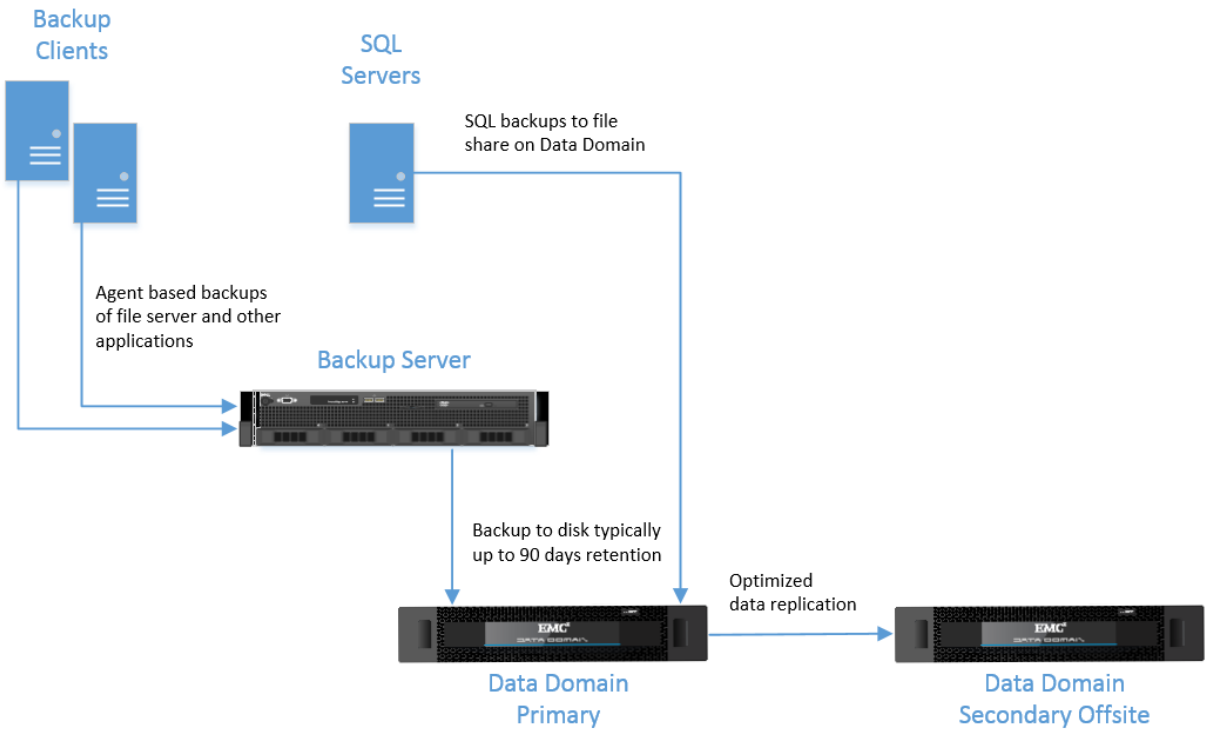


Figure 6: Native SQL Server Backups to Data Domain Only

Current Challenges

Data Domain is an excellent disk-based backup solution, but can still present a couple of challenges when used as a CIFS/SMB target for SQL Server backups.

- **Long-Term Retention**

  When using Data Domain and backing up directly from SQL Server, long-term retention requirements can still be a challenge. There are typically a couple of ways this is handled. One is to pick up SQL backup files from a file server or Data Domain and then write to tape. The challenge with this method is cataloguing and reporting, as SQL backups are only files and are not cataloged as a database backup. Another method is to use a traditional backup application to create a backup to tape. The challenge with this method is to not interfere with the backup chain when doing the majority of backups with native SQL Server tools.

- **SQL Compression**
  When SQL DBAs back up to a network file share, they will typically use compression, especially on larger databases. This reduces the amount of data that is sent over the network, the amount of data stored on disk ,and backup time. Sending compressed backup data to a file share on a Data Domain appliance reduces the effectiveness of Data Domain deduplication. This is why Data Domain deduplicates first and then does compression on the data.

The long-term retention challenge can be overcome by utilizing a command line backup client available for various traditional backup agents. This allows the backups to be controlled by the DBA and automated using SQL Server tools such as SQL Agent Jobs and Maintenance Plans.

The SQL Compression challenge can be overcome by utilizing a relatively new tool from EMC called DD Boost for Microsoft Applications (DDBMA). This tool also offer other benefits discussed in the next section.

Data Domain Boost for Microsoft Applications

EMC released Data Domain Boost for Microsoft Applications in the first half of 2014. This tool has a SQL Server module that is implemented as a SQL Server Management Studio (SSMS) plug-in or used as a command line tool that can be called from SQL Agent jobs or Maintenance Plans. The tool allows SQL Server database data to be deduplicated on the SQL Server before being sent to the Data Domain appliance, reducing the amount of data sent over the network and stored on disk.

Benefits

The benefit of this tool is the ability to optimize the speed of SQL Server backups and reduce the amount of storage space required when using Data Domain as a backup target. This tool also benefits the SQL Server DBA because it puts control of the backups back in their hands, and allows the use of familiar SQL Server tools.

Challenges

The primary challenge with DDBMA is that it is new and detailed information is not yet available on the various ways it can be implemented. The primary documentation for the tool is the administrator's guide, which provides detailed guidance on using the SSMS plug-in. It also provides some information on using the command line tools from the Windows task scheduler to

automate backups. Since this tool is targeted at SQL Server DBAs, they see this as a downside because they are generally not interested in using operating system tools for this purpose. DBAs typically want to use tools such as SQL Server Agent jobs and Maintenance Plans to complete backups. This allows them to automate backups and integrate with other tasks, such as re-indexing databases and updating database statistics.

The tool has some limitations in the initial version, such as the inability to remotely administer and control backups. The SSMS plug-in can only be installed on a full installation of SQL Server and cannot be used to control backups on a remote server. The tool has limited backup catalog functionality, which can only be accessed via the SSMS plug-in. The lack of a catalog that can be queried easily increases difficulty in reporting and identifying specific backups for restore.

The next section will introduce the SQL Server Data Domain Boost Scripting Toolkit[1]. This is a set of freely available open source scripts that provide a foundation for DBAs to integrate DDBMA with SQL Server. This toolkit can help address the challenges of utilizing DDBMA and provide a more complete solution. The toolkit is licensed under the Massachusetts Institute of Technology (MIT) open source license agreement. The goal of the project is to provide SQL Server DBAs and system administrators a good foundation to incorporate the DDBMA module for SQL Server into current backup and maintenance processes.

SQL Server Data Domain Boost Scripting Toolkit

The SQL Server Data Domain Boost Scripting Toolkit provides scripts to create and use T-SQL stored procedures and tables to automate the use of Data Domain Boost for Microsoft Applications. This section will describe the required pre-requisites, installation, and use of the toolkit.

The toolkit can be downloaded from GitHub.

Prerequisite Configuration

The SQL Server Data Domain Boost Scripting Toolkit requires configuration of the following items for use:

- Windows User Account Control must be set to the never notify setting. This is also required for the DDBMA application and is specified in the administrators guide.
- The DDBMA plug-in must be installed on any SQL Server with databases to be backed up or restored.
- The extended stored procedure `xp_cmdshell` must be enabled on any SQL Server with databases to be backed up or restored.
- It is recommended that the SQL Server services be run under a domain account that belongs to the local administrators group. It is also recommended that an administrative database be created on each SQL Server with databases to be backed up or restored. This database acts as a holding place for the stored procedures and table which comprise the toolkit.

Toolkit Installation

Installation of the toolkit consists of running T-SQL scripts to create stored procedures and a table on each SQL Server where backups and/or restores will be completed. The following sections describe each element of the scripting toolkit and its installation script.

Catalog Table

`DDBMASQLCatalog`

This table is used as a catalog for databases backed up on the server. It contains all of the metadata about backups. The table is created using the create script supplied in the toolkit.

`CreateDDBMASQLCatalogTable.sql` – Run this script on the admin database created on each SQL Server involved in backups or restores.

The table has the following structure to hold backup information.

ColumnName	DataType
ClientHostName	varchar(50)
BackupLevel	varchar(10)
BackupSetName	varchar(100)
BackupSetDescription	varchar(255)
BackupDate	datetime
ExpDate	datetime
DDHost	varchar(50)
DDStorageUnit	varchar(50)
DatabaseName	varchar(100)
SQLInstance	varchar(50)

Table 1: DDBMASQLCatalog SQL Table Structure

Stored Procedures

`DDBMABackup`

This stored procedure will use `xp_cmdshell` to execute the DD Boost backup application ("ddbmsqlsv.exe") and back up a database. The stored procedure has parameters that relate to the DDBMA command switches. It uses dynamic SQL to build the backup command with proper options based on parameters and values passed to the procedure. This is similar to the approach SSMS plug-in uses to execute the command.

The procedure will also loop through each line of the backup command output and detect the backup success and error message. It captures the backup time from the output and creates an entry for the backup in the `DDBMASQLCatalog` table upon success. If errors are encountered, it will raise an error that can be detected by SQL server. The procedure is created using the create script supplied in the toolkit.

`CreateDDBMABackupProcedure.sql` – Run this script on the administrative database created on each SQL Server. This file also has information on the procedure parameters and how they correspond to the application command switches.

`DDBMARestore`

This stored procedure will use `xp_cmdshell` to execute the DD Boost restore application ("ddbmsqlrc.exe") and restore a database. The stored procedure has parameters which relate to the DDBMA command switches. It uses dynamic SQL to build the restore command with proper options based on parameters and values passed to the procedure. This is similar to the approach the SSMS plug-in uses to execute the command.

`CreateDDBMARestoreProcedure.sql` – Run this script on the administrative database created on each SQL Server. This file also has information on the procedure parameters and how they correspond to the application command switches.

`DDBMACatalog`

This stored procedure is used to query the `DDBMASQLCatalog` table and to delete entries from the table. The procedure is intended to provide backup reporting capabilities, and provide a way to easily obtain information needed for logic in automated restore routines. One challenge seen with DBAs working on automating restores with DD Boost is locating the correct database version to restore. The "ddbmsqlrc.exe" command uses the backup timestamp as an identifier, so by querying the catalog table the backup time can also be tied to database name, backup name, and backup description. The procedure is created using the create script supplied in the toolkit.

`CreateDDBMACatalogProcedure.sql` – Run this script on the administrative database created on each SQL Server. This file also has information on how the procedure parameters are used.

`DDBMAExpire`

This stored procedure is used to query or delete the expired backup images in a storage unit on a Data Domain appliance. This stored procedure will use `xp_cmdshell` to execute the DD Boost expiration tool application ("ddbmexptool.exe"). The stored procedure has parameters that relate to the DDBMA command switches. It uses dynamic SQL to build the backup command with options based on parameters and values passed to the procedure.

`CreateDDBMAExpireProcedure.sql` – Run this script on the administrative database created on each SQL Server. This file also has information on the procedure parameters and how they correspond to the application command switches.

Toolkit Usage

There are a few common ways most DBAs will utilize the toolkit to perform backups, restores, and related operations. DBAs will generally want to use T-SQL in management studio to perform ad-hoc operations and use T-SQL in SQL Agent jobs for scheduled operations. It is also a common practice to call SQL Agent jobs from Maintenance Plans to perform backups and other database maintenance operations.

The toolkit includes a useful file with examples of various operations. The file name is `ExecuteProceduresExamples.sql` and has examples for calling each procedure with the most commonly used examples. Some of those examples are shown in the next section.

Ad-Hoc Operations

Backup

Figure 7 shows how to perform an ad-hoc full backup of a database.

```
Exec dbo.DDBMABackup @ClientHost = 'SQL2012.ddlab.local',
    @BackupLevel = 'full',
    @BackupSetName = 'ContactsFull',
    @BackupSetDescription = 'Contacts Full Backup',
    @RetentionDays = 7,
    @DDHost = 'dd640-01.ddlab.local',
    @DDBoostUser = 'ddboost',
    @DDStorageUnit = '/Boost-SQL',
    @SQLDatabaseName = 'Contacts'
```

Figure 7: DDBMABackup T-SQL Procedure Full Backup Example

This example does a full backup of the contacts database located on server "SQL2012.ddlab.local" to the "/Boost-SQL" storage unit on the "dd640-01.ddlab.local" Data Domain appliance. It also specifies to retain the backup for a period of 7 days. After the retention period passes the backup image will expire and can then be deleted from the Data Domain system.

Figure 8 shows how to perform an ad-hoc transaction log backup of a database.

```
Exec dbo.DDBMABackup @ClientHost = 'SQL2012.ddlab.local',
    @BackupLevel = 'incr',
    @BackupSetName = 'ContactsLog',
    @BackupSetDescription = 'Contacts Transaction Log Backup',
    @RetentionDays = 7,
    @DDHost = 'dd640-01.ddlab.local',
    @DDBoostUser = 'ddboost',
    @DDStorageUnit = '/Boost-SQL',
    @SQLDatabaseName = 'Contacts'
```

Figure 8: DDBMABackup T-SQL Procedure Transaction Log Backup Example

This example does a transaction log backup of the contacts database located on server "SQL2012.ddlab.local" to the "/Boost-SQL" storage unit on the "dd640-01.ddlab.local" Data Domain system.

It also specifies to retain the backup for a period of 7 days. After the retention period passes, the backup image will expire and can then be deleted from the Data Domain system.

There are additional examples provided in the toolkit for more complex operations like backing up all system databases, all user databases, and specific lists of databases.

Restore

Figure 9 shows how to perform an ad-hoc restore of a database.

```
Exec dbo.DDBMARestore @ClientHost = 'SQL2012.ddlab.local',
    @RecoveryState = 'normal',
    @DDHost = 'dd640-01.ddlab.local',
    @DDBoostUser = 'ddboost',
    @DDStorageUnit = '/Boost-SQL',
    @SQLDatabaseName = 'Contacts',
    @Overwrite = 1
```

Figure 9: DDBMARestore T-SQL Procedure Normal Database Restore Example

This example does a restore from the most recent backup of the contacts database backed up from server “SQL2012.ddlab.local” to the “/Boost-SQL” storage unit on the “dd640-01.ddlab.local” Data Domain system. It will overwrite the existing database, if present.

Figure 10 shows how to perform an ad-hoc restore of a database from a specific backup and leave the database in a state to recover additional transaction logs.

```
Exec dbo.DDBMARestore @ClientHost = 'SQL2012.ddlab.local',
    @BackupTimeStamp = '11/13/2014 22:32:25',
    @RecoveryState = 'norecover',
    @DDHost = 'dd640-01.ddlab.local',
    @DDBoostUser = 'ddboost',
    @DDStorageUnit = '/Boost-SQL',
    @SQLDatabaseName = 'Contacts',
    @Overwrite = 1
```

Figure 10: DDBMARestore T-SQL Procedure “NoRecover” Database Restore Example

This example does a restore from the 11/13/2014 22:32:45 backup of the contacts database backed up from server “SQL2012.ddlab.local” to the “/Boost-SQL” storage unit on the “dd640-01.ddlab.local” Data Domain system. It will overwrite the existing database if present and leave the database in a restoring state. The above examples can be used to restore to another server by executing the procedures on the alternate server. The target server will require that DDBMA and the toolkit are installed locally on that server.

There are additional examples provided in the toolkit for more complex operations like restoring a database to an alternate server. There are also examples for restoring from a replicated copy of the backup from an alternate Data Domain appliance. This scenario could be valuable in a situation where the development and testing environments are in a secondary site with a replicated copy of the database on a Data Domain appliance local to that site.

Catalog

Figure 11 shows how to query the backup catalog table for all backups of the contacts database on the local SQL Server.

```
Exec dbo.DDBMACatalog @DatabaseName = 'Contacts'
```

Figure 11: DDBMACatalog T-SQL Procedure Catalog Query Example

The illustrations in Table 2 show example output of the above procedure.

ClientHostName	BackupLevel	BackupSetName	BackupSetDescription	BackupDate
SQL2012.ddlab.local	incr	ContactsTRN	Contacts Transaction Log Backup	11/14/2014 7:48
SQL2012.ddlab.local	full	ContactsFull	Contacts Full Backup	11/13/2014 10:32
SQL2012.ddlab.local	full	ContactsFull	Contacts Full Backup	11/13/2014 9:32
SQL2012.ddlab.local	full	ContactsFull	Contacts Full Backup	11/11/2014 10:44

ExpDate	DDHost	DDStorageUnit	DatabaseName	SQLInstance
11/21/2014 7:48	dd640-01.ddlab.local	/Boost-SQL	Contacts	MSSQL
11/20/2014 10:32	dd640-01.ddlab.local	/Boost-SQL	Contacts	MSSQL
11/20/2014 9:32	dd640-01.ddlab.local	/Boost-SQL	Contacts	MSSQL
11/18/2014 10:44	dd640-01.ddlab.local	/Boost-SQL	Contacts	MSSQL

Table 2: DDBMACatalog T-SQL Procedure Catalog Query Output Example

Note: The example output has been altered to two tables for more readable display in this article.

To remove these entries from the catalog the same procedure call would be issued with the `@DeleteEntries` parameter set to 1 as shown below.

```
Exec dbo.DDBMACatalog @DatabaseName = 'Contacts' @DeleteEntries = 1
```

Figure 12: DDBMACatalog T-SQL Procedure Catalog Delete Entries Example

Expire

Figure 13 shows how to query the Data Domain appliance storage unit and list or delete expired images.

```
Exec dbo.DDBMAExpire @AppType = 'mssql',
    @ConfigFile = 'C:\dd640-dm.cfg',
    @StartTime = '11/21/2014 00:00:01',
    @EndTime = '11/21/2014 23:59:59',
    @SaveSet = 'ContactsFull',
    @DelExpired = NULL,
    @Clean = NULL,
    @Verbose = NULL
```

Figure 13: DDBMAExpire T-SQL Procedure Expired Images Query Example

The above example will list all images with the “SaveSet/BackupSetName” “ContactsFull” which expired on 11/21/2014.

Example output from the `DDBMAExpire` procedure is shown below.

```
libDDBoost version: major: 2, minor: 6, patch: 2, engineering: 5, build: 430104
client = sql2012.ddlab.local, date&time = 11/14/2014 10:45:40 AM,
client = sql2012.ddlab.local, date&time = 11/14/2014 10:45:42 AM,
client = sql2012.ddlab.local, date&time = 11/14/2014 10:51:08 AM,
client = sql2012.ddlab.local, date&time = 11/14/2014 10:51:10 AM,


size = 2581192 bytes, level = full, retent = 11/21/2014 10:45:39 AM, name = ContactsFull
size = 240 bytes, level = incr, retent = 11/21/2014 10:45:39 AM, name = ContactsFull
size = 2581192 bytes, level = full, retent = 11/21/2014 10:51:07 AM, name = ContactsFull
size = 240 bytes, level = incr, retent = 11/21/2014 10:51:07 AM, name = ContactsFull
```

Figure 14: DDBMAExpire T-SQL Procedure Expired Images Query Output Example

Note: The output has been altered to two tables for better display.

To delete these images from the Data Domain, the same procedure call would be issued with the `@DelExpired` parameter set to 1 as shown below.

```
Exec dbo.DDBMAExpire @AppType = 'mssql',
    @ConfigFile = 'C:\dd640-dm.cfg',
    @StartTime = '11/21/2014 00:00:01',
    @EndTime = '11/21/2014 23:59:59',
    @SaveSet = 'ContactsFull',
    @DelExpired = 1,
    @Clean = NULL,
    @Verbose = NULL
```

Figure 15:* DDBMAExpire *T-SQL Procedure Expired Images Delete Entries Example

The examples are executed on the administrative database located on the server where operations are to be performed.

SQL Agent Jobs

One of the primary drivers for the creation of the SQL Server DD Boost Scripting Toolkit is the ability to automate backup and restore processes via SQL Server Agent jobs. This can be done easily by utilizing the same stored procedure used in ad-hoc operations in the SQL Agent jobs. This will also allow error handling and the use of the SQL Agent notification features. Figure 16 shows calling the `DDBMABackup` procedure from a SQL Agent job step.

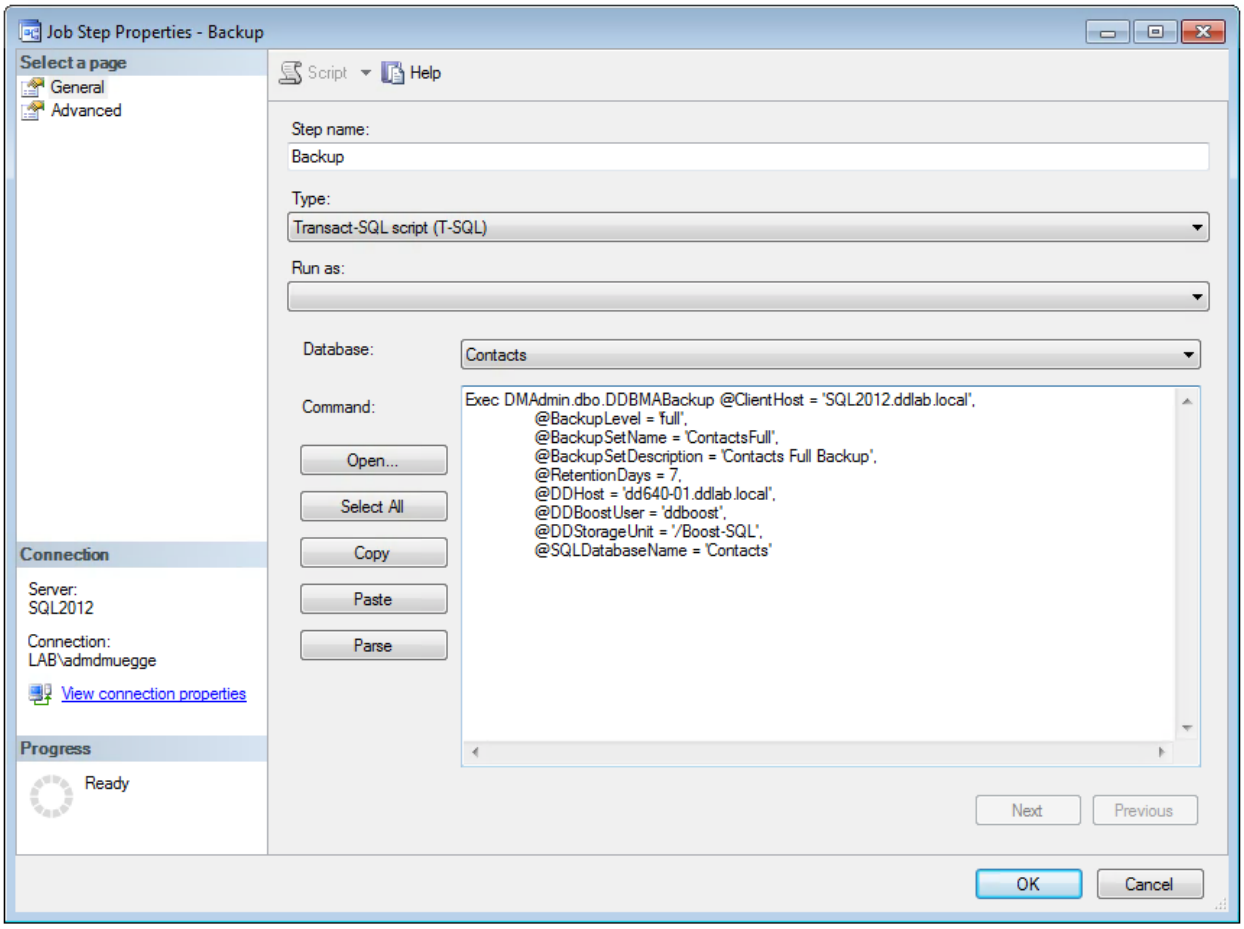


Figure 16: SQL Agent Job Backup Example

To successfully utilize error handling and to capture the DD Boost command output in the job history, the advanced option ‘Include Job Step Output in History’ must be selected. An example of this is shown in Figure 17.

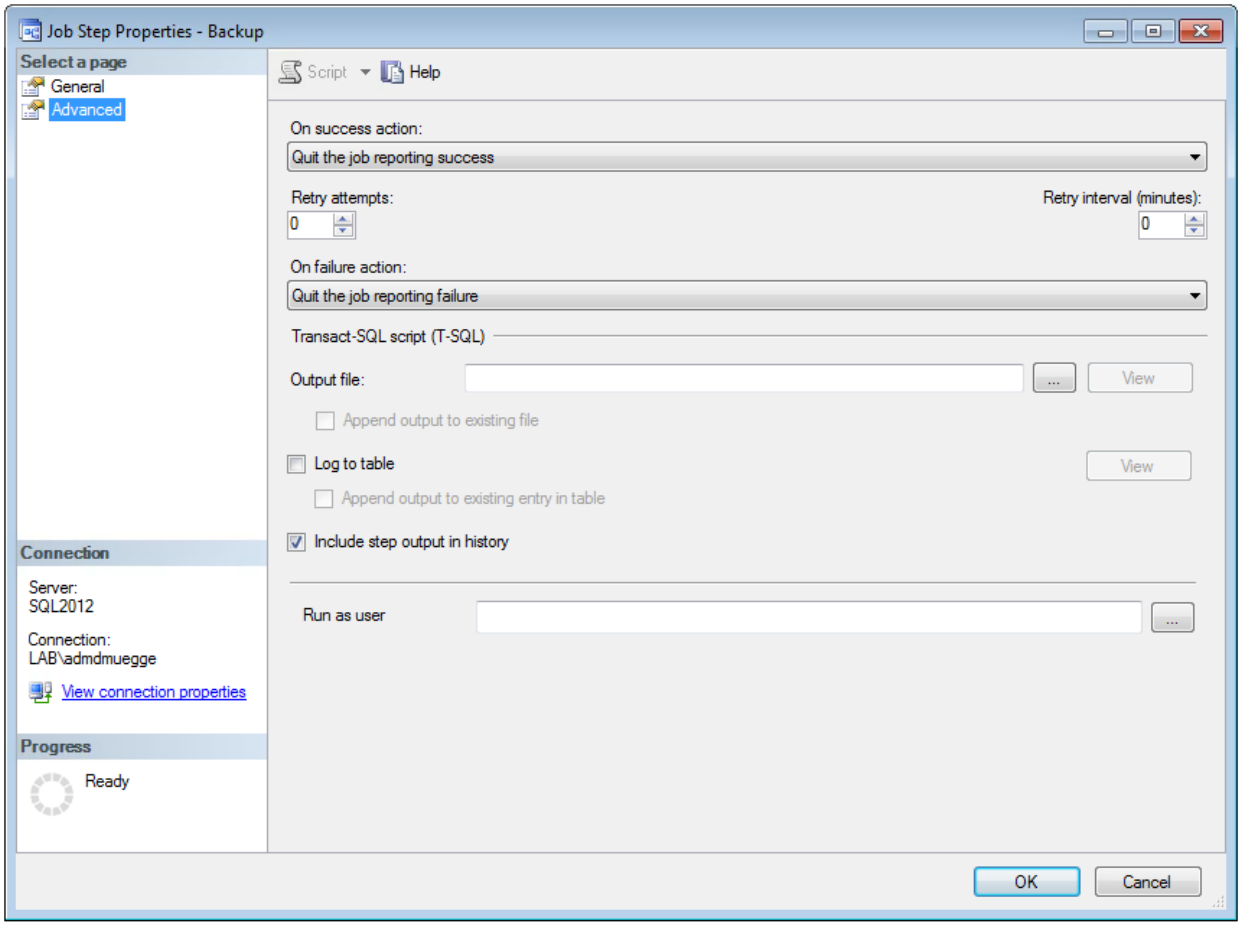


Figure 17: SQL Agent Job Backup Advanced Settings Example

By capturing command output in the history of the job step, the full DD Boost output text is displayed and provides the ability to troubleshoot more effectively. Figure 18 shows job history for the backup job and the detailed output captured.

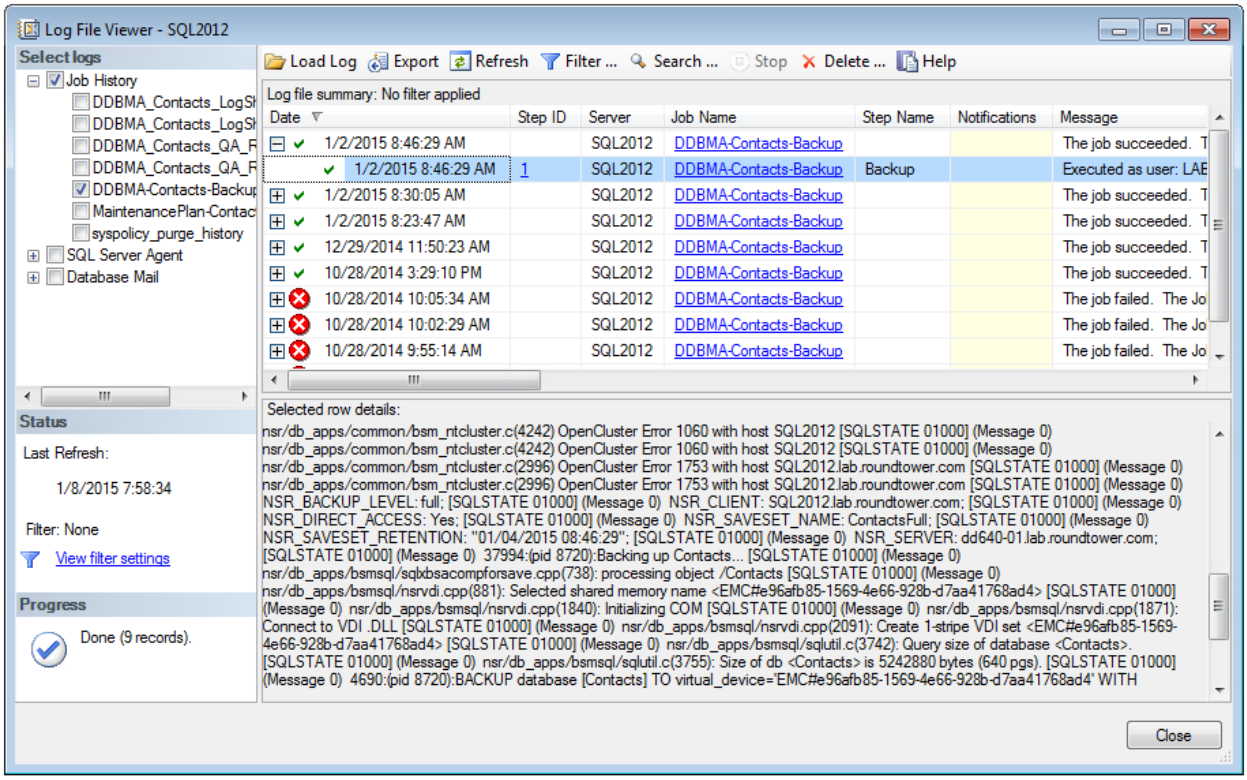


Figure 18: SQL Agent Job Output History Example

Utilizing SQL Agent jobs also allows scenarios such as automated refreshing of development and testing databases or log shipping. This can be done from a single server by utilizing SQL Server linked server functionality combining backup and restore operations in a single job as multiple steps and providing proper workflow and error handling. Figure 19 shows a QA refresh process with two steps. The second restore step is dependent on the backup step completing successfully.

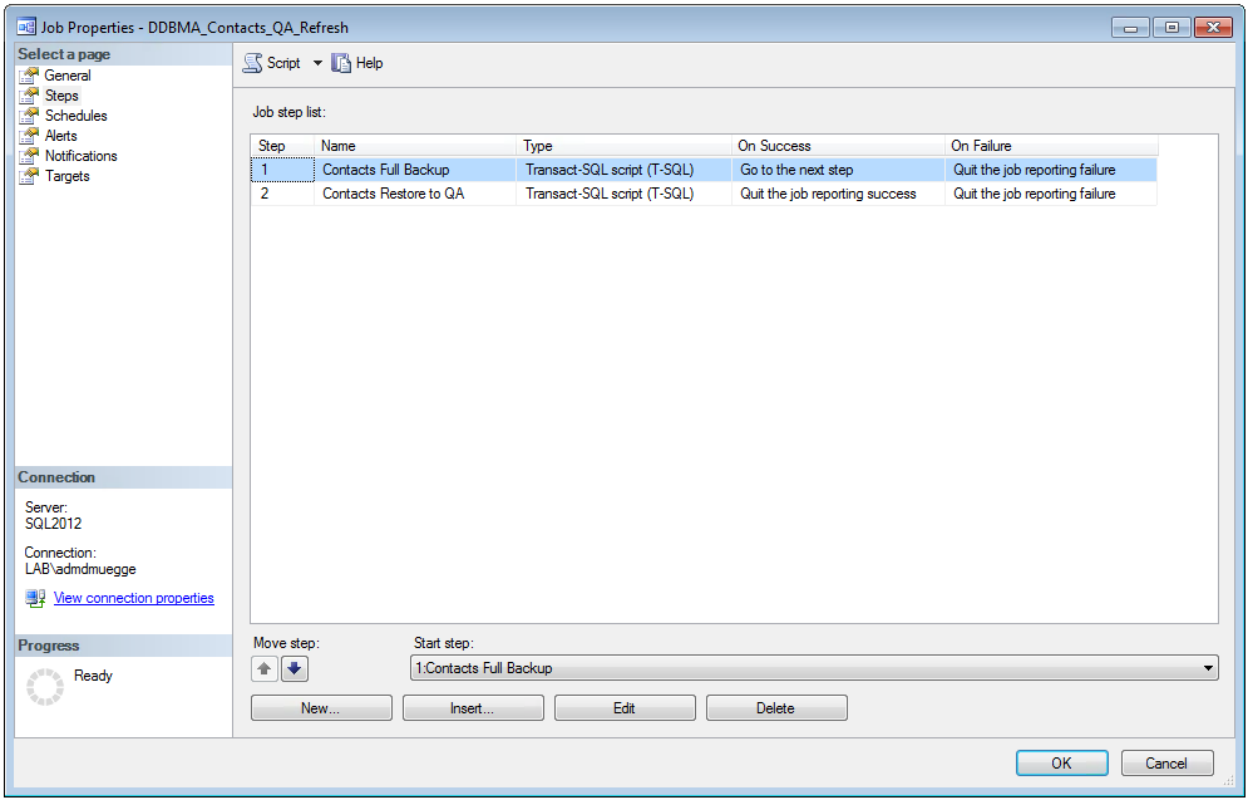


Figure 19: SQL Agent Job Multi-Step Backup and Restore Example

Figure 20 shows the detail of the second step, which calls the `DDBMARestore` procedure on the target server to complete the restore and overwrite the existing database.

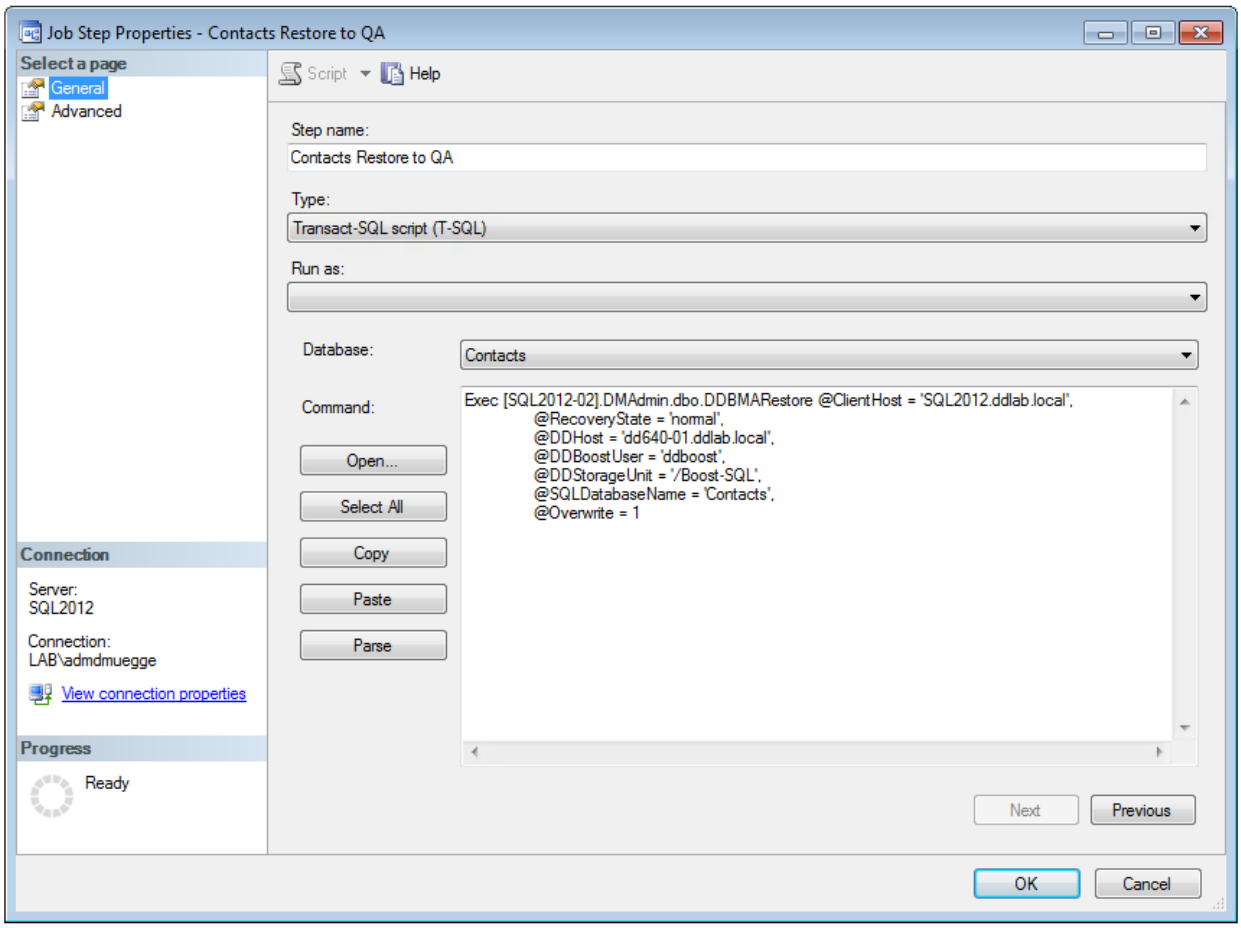
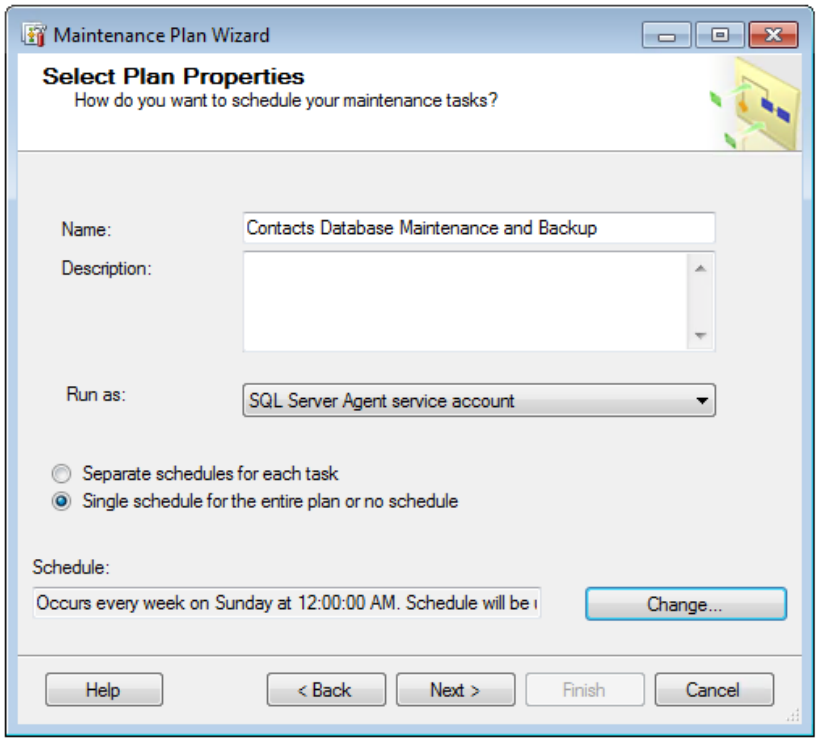


Figure 20: SQL Agent Job Multi-Step Backup and Redirected Restore Example

The above procedure call uses the server name of the target server to run the process on that server, but specifies the source server name in the `@ClientHost` parameter. This combination is what allows the redirected restore to work properly.

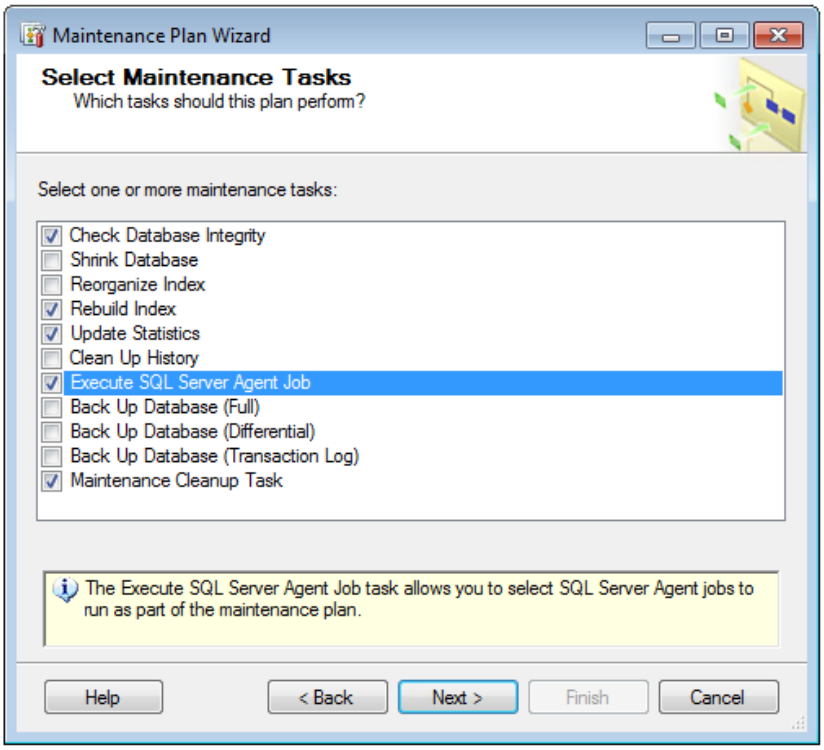
Maintenance Plans

A common method for implementing SQL Server backups and incorporating with other maintenance procedures is the use of Maintenance Plans. By using the Agent Job scenarios explored in the previous section, one can easily utilize DD Boost for SQL Server with Maintenance Plans. Many administrators will configure Maintenance Plans by using the wizard. Examples below will demonstrate this method. The wizard is launched and a name and schedule is defined for the plan.



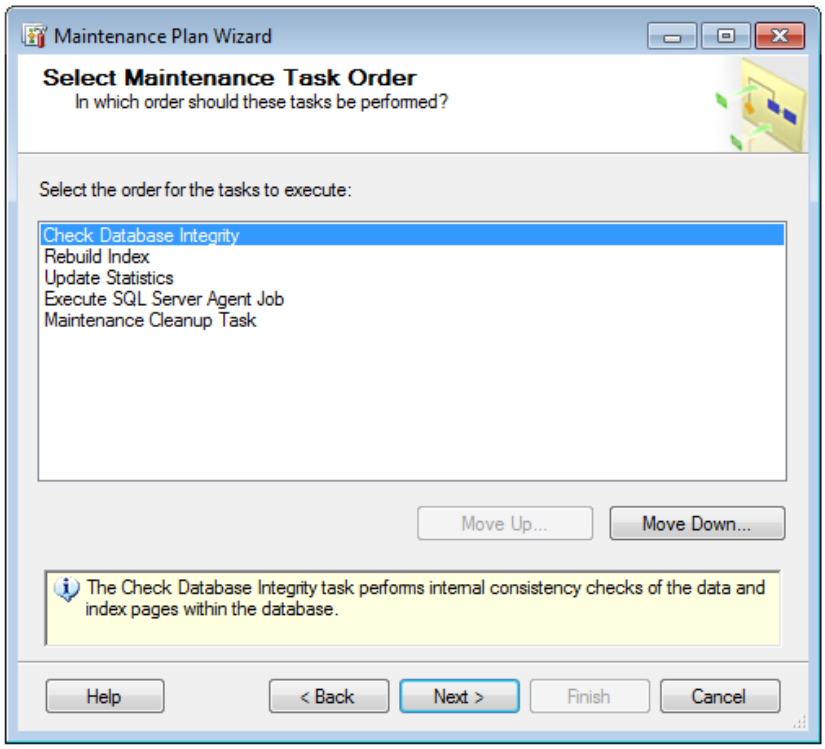
After assigning a name to the plan and configuring a schedule, the next step is to define the tasks to be completed by the plan. Figure 21 shows the available tasks for performing database maintenance and backups.

Figure 21: Maintenance Plan Creation Example



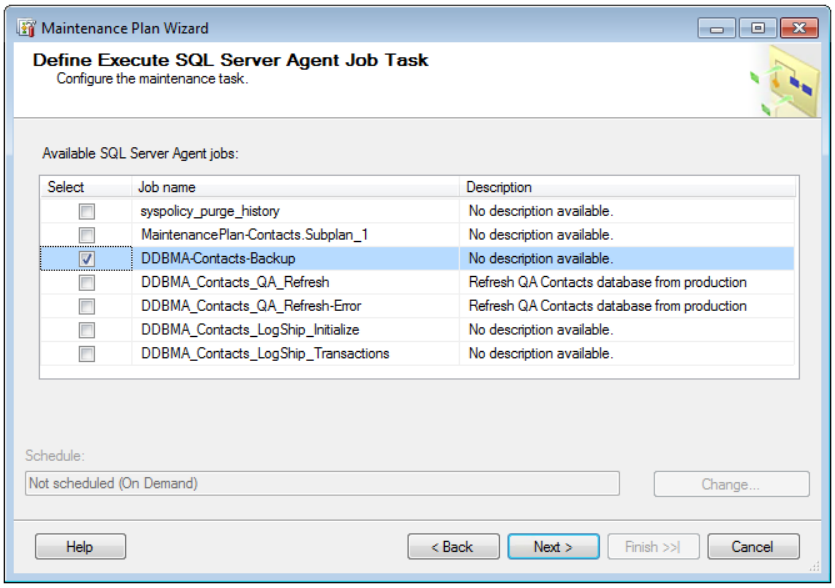
The desired maintenance tasks for the plan are selected from the available choices. When using a SQL Maintenance Plan for backups, typically the 'Back Up Database' task is used, which uses the native SQL Server backup application. To utilize DDBMA and the SQL Server DD Boost Scripting Toolkit; the SQL Server Agent Job task is chosen to execute a previously created SQL Agent job to do the backups.

Figure 22: Maintenance Plan Task Selection Example



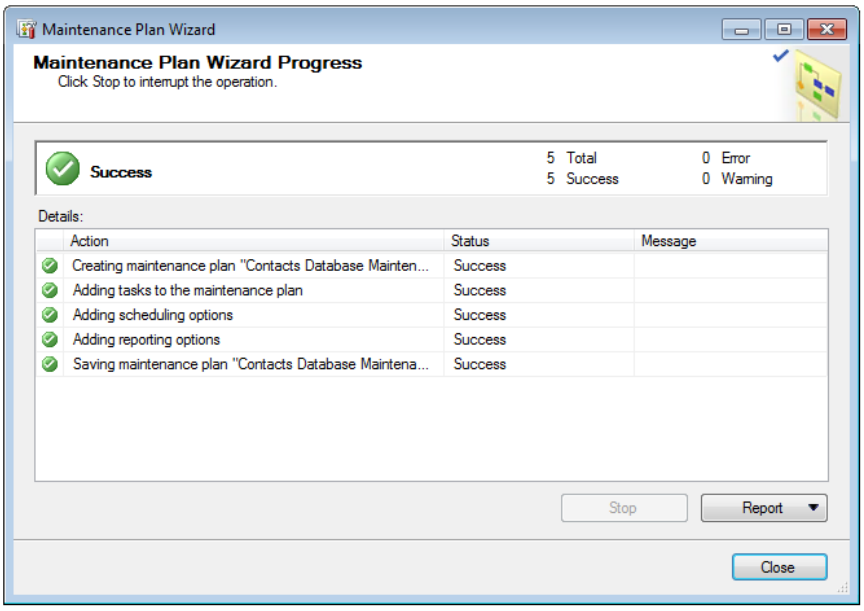
The order in which tasks are to be performed is defined and the parameters of each task is configured. The details of each task configuration are not shown here. The section below will focus on the required options to define the backup agent job.

Figure 23: Maintenance Plan Task Order Example



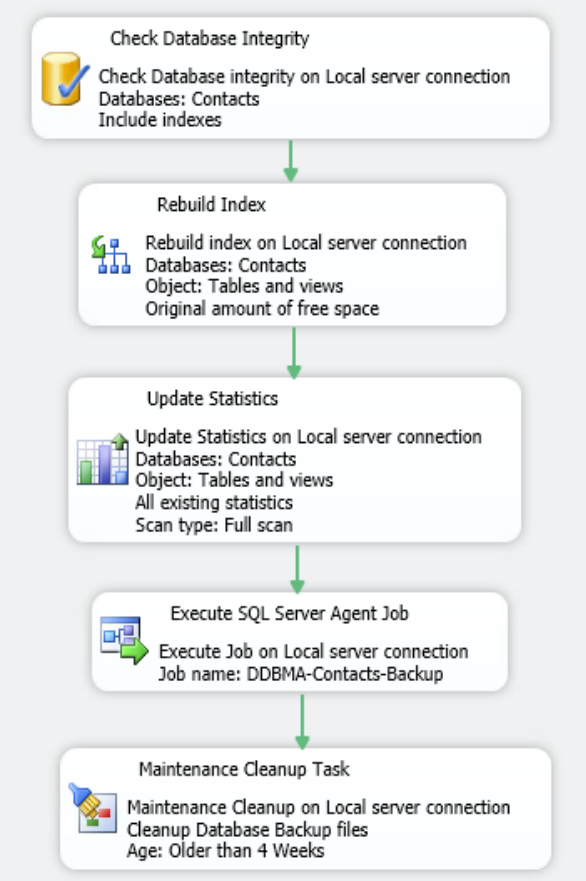
The backup job is configured by selecting the SQL Agent job previously defined to complete a full backup of the contacts database.

Figure 24: Maintenance Plan DDBMA Backup Example



After completing selection of the SQL Server Agent job and configuration of other tasks, the Maintenance Plan wizard shows the final configuration status.

Figure 25: Maintenance Plan Final Configuration Example



Once the Maintenance Plan is created, the properties of the Maintenance Plan show each of the steps in the order of completion. In some cases it may be desirable to modify an existing plan to integrate the DDBMA backups. This can be done by removing the native SQL Backup task and inserting a task to call an Agent job to run the `DDBMABackup` procedure.

Figure 26: Maintenance Plan Step Details Example

Toolkit Considerations

The T-SQL procedures in the toolkit were developed with the largest range of use cases and ease-of-use as primary goals. Using `xp_cmdshell` and the toolkit T-SQL stored procedures, DD Boost backups can be performed using the same method on SQL Servers from version 2000 and above. One consideration for some organizations will be the use of `xp_cmdshell`, which can increase security exposure on SQL Servers. This may not be an issue for many organizations as `xp_cmdshell` is commonly used for various purposes. Risk can also be reduced by structuring user permissions to minimize access to the procedure.

There are some other methods to utilize DD Boost for SQL Server without the use of `xp_cmdshell`. One method, which is available on SQL Server 2012 and above, is the ability to use PowerShell from a SQL Agent job. PowerShell, which can be used in a similar method to T-SQL, can capture output, trap errors, and pass errors to the agent service.

Another consideration is the ability to write backups to tape for long-term storage. When backing up SQL Server using DDBMA, there is not a backup file that can be easily picked up from the Data Domain and sent to tape. This can be overcome by utilizing a standard backup client, such as EMC NetWorker®. A weekly or monthly back up to tape can be done using a SQL Agent job to call the CLI version of the NetWorker Module for Microsoft SQL.

Conclusion

SQL Server database backup options are plentiful and this can lead to debate when defining a backup strategy. If Data Domain is being used in an organization to store backups for SQL Server, DBAs have another choice for SQL Server backups with Data Domain Boost for Microsoft Applications. This product can provide a solution for DBAs that integrates with their existing processes and can also satisfy the requirements of the infrastructure team. The SQL Server Data Domain Boost Scripting Toolkit can help DBAs by giving them a foundation to better utilize DDBMA.

Appendix A: References

Footnotes

1. David Muegge, “SQL Server Data Domain Boost Scripting Toolkit”, Open Source software by David Muegge, last modified January 2, 2015. https://github.com/dmuegge/ddb-sql-toolkit

Bibliography

1. Wikipedia, s.v. “Relational Database Management System”, last modified January 6, 2015, http://en.wikipedia.org/wiki/Relational_database_management_system
2. Donald Feinberg, Merv Adrian, and Nick Heudecker, “2014 Magic Quadrant for Operational Database Management Systems”, Gartner Inc. October 16, 2014. http://www.gartner.com/technology/reprints.do?id=1-1ZRLY7Y&ct=140811&st=sb
3. Mark A. Beyer and Roxane Edjlali, “2014 Magic Quadrant for Data Warehouse Database Management Systems”, Gartner Inc. March 7, 2014. http://www.gartner.com/technology/reprints.do?id=1-1RNK7M1&ct=140310&st=sb
4. Joe Masters Emison, “Information Week Research 2014 State of Database Technology”, UBM LLC. March 2014. http://reports.informationweek.com/cart/index/downloadlink/id/12155
5. Wikipedia, s.v. “Oracle Corporation”, last modified January 14, 2015, http://en.wikipedia.org/wiki/Oracle_Corporation
6. Wikipedia, s.v. “Microsoft SQL Server”, last modified January 14, 2015, http://en.wikipedia.org/wiki/Microsoft_SQL_Server
7. Wikipedia, s.v. “Operating-System-Level-Virtualization”, last modified January 6, 2015, http://en.wikipedia.org/wiki/Operating-system-level_virtualization
8. Pushan Rinnen, Dave Russell, and Jimmie Chang, “2014 Magic Quadrant for Deduplication Backup Target Appliances”, Gartner Inc. July 31, 2014. http://transformation.emc2.at/wp-content/uploads/2014/08/Gartner-MQ-for-Deduplication-Backup-Target-Appliances-2014.pdf
9. “EMC Data Domain Boost for Microsoft Applications Administration Guide”, EMC Corporation. July 9, 2014.
10. “EMC NetWorker Module for Microsoft for SQL VDI User Guide”, EMC Corporation. November 2014

11. Paul S. Randal, “Understanding SQL Server Backups”, Technet Magazine, Microsoft Corporation., accessed January 10, 2015 http://technet.microsoft.com/en-us/magazine/2009.07.sqlbackup.aspx
12. “Backup and Recovery for Microsoft SQL Server Using EMC Data Domain Deduplication Storage Systems Best Practices Planning Guide”, EMC Corporation. February 2012.
13. Brent Ozar, “SQL Server Backup Best Practices”, a blog by Brent Ozar. October 17, 2007. http://www.brentozar.com/archive/2007/10/backing-up-sql-server-my-own-mediocre-practices/
14. “The Business Value of Data Domain Boost”, EMC Corporation. November 2014.
15. Michael K. Campbell, “SQL Server Backups: When More is Less”, SQL Server Pro, Penton Media October 31, 2011. http://sqlmag.com/blog/sql-server-backups-when-more-less
16. David Muegge, “Data Domain Boost for SQL – Deployment Planning Considerations for DBAs”, a blog by David Muegge. August 8, 2014. http://muegge.com/blog/?p=276

About the Author

David Muegge is a Principal Systems Engineer for RoundTower Technologies, Inc. helping customers in Indiana design technology solutions. He has over 20 years of IT experience in many roles, and currently holds the following EMC Proven Professional certifications:

- Technology Architect: VNX Solutions Specialist
- Cloud Architect: Virtualized Infrastructure Specialist
- Technology Architect: Isilon Infrastructure Specialist
- Implementation Engineer: Isilon Specialist

Acknowledgments

Thanks to everyone who helped with input and guidance on this article including; my wife Carolyn; my colleagues at RoundTower; Pamela McMillan, Eduardo Figueredo, Steve Swanson, Scott Sizelove, Kurtis Lindemann, David Hawkins, and Rob Steele; and Michael Oglesby, Jeffrey St. Cyr, and Tom Spreacker of EMC.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” EMC CORPORATION MAKES NO RESPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.