Tech Note 975
Defragmenting MES Databases

Topic#: 002800
Created: August 2013

Introduction

MES (SQLServer) databases need to be regularly defragmented. Indexes including the primary key will need to be defragmented on a regular basis due to inserts and updates on the table.

Without defragmentation, the query performance degrades over time.

Indexes may need to be rebuilt or reorganized depending on the level of fragmentation. Typically we want to reorganize the index if its fragmentation is over 10% and less than 30%. Any indexes with over 30% fragmentation will require to be Rebuilt. If tables have less than 1000 rows, the table is too small to be concerned with index fragmentation.

Wonderware generally recommends running the defragmentation on a weekly basis, during periods of low demand. If a Primary Key index needs to be rebuilt, then it could cause a table lock and a production transaction could fail. If you are a 24x7 plant then you should find time once a quarter where you can run the procedure. If you never defrag, your system performance will become progressively slower, potentially causing locks that will cause transactions to fail. For this reason, we recommend you schedule a job to run and deal with the potential failed transactions.

The standard Microsoft SQLServer maintenance plans have an **Index Rebuild** and **Index Reorganize** job. The problem with these jobs is that they do not check the fragmentation level to determine if the jobs are needed or not.

This *Tech Note* provides detailed guidelines and options for defragmenting and maintaining your production databases.

Note: This operation is highly recommended for any Microsoft SQL Database used by Wonderware Products.

Application Versions

- Factelligence 3.5
- MES 4.x
- Microsoft SQLServer 2005, 2008 and 2012 (including while running with compatibility mode 80)

Buiding the Defrag Tables

To determine the fragmentation level, Wonderware Technical Support created the following scripts:

- **deFrag_CreateTables**
- **ww_spDefrag**

Download the scripts (provided as .txt files).

Creating the Tables

1. Run the script **defrag_CreateTables.sql**. This script creates the following tables to track the defrag process:

- **Temp_Frag_log**
- **Temp_Frag_Data**

2. Run the **ww_spDefrag.sql** script. This script creates the **ww_spDefrag** stored procedure that will do the defragmentation.
3. Schedule a job (or maintenance plan) to defrag the database.

You can use the following options:

Option A

To schedule it to run whenever the job runs, use the following T-SQL:

```
EXEC ww_spDefrag @p_Schedule=NULL,@p_DayNo=NULL,@p_RunHour=NULL
```

Option B

To schedule the specific time to run, for example every Sunday at 2AM:

```
EXEC ww_spDefrag @p_Schedule='Weekly',@p_DayNo=7,@p_RunHour=2
```

This could be useful if you have data transfer job the runs regularly and you don’t want the defrag to interfere with it. In this case, you would add a step to the data transfer job to do the defrag. The stored procedure would run every period (say 10 minutes) but only defrag when schedule (Sunday at 2AM).

Option C

To update more than just the current db (this examples does the MESDB and the ArchiveDB) at the default time Sunday at 2AM:

```
EXEC ww_spDefrag @p_db_names='MESDB,ArchiveDB'
```

Parameter Details

The following table contains details on the parameters for the Stored Procedure.

ww_spDefrag Parameters

Parameter	**Default**	**Description**
@p_Schedule	**Weekly**	Daily, Weekly, Monthly, Quarterly or Null to defrag when run.
@p_DayNo	**7**	CSV of day weekday:1=Sun,2=Mon,3=Tue,4=Wed,5=Thur,6=Fri,7=Sat). This value is the SQLServer weekday value.
@p_RunHour	**2**	The hour of the day it should run at (2 for 2AM).
@p_db_names	**Null**	Null/empty local db, Supports CSV for dbname like MESDB,ArchiveDB.
@p_ReorganizeLevel	**10**	Percent frag to force a reoganization.
@p_RebuildLevel	**30**	Percent frag to force a rebuild.
@p_mode	**UPDATE**	TEST or UPDATE
@p_KeepDays	**365**	Keeps the last 365 days of log_data records.
@p_min_row_Count	**1000**	Tables with less then 1000 rows will not be updated (default is 1000).
@p_FillFactor	**Null**	Index property, Null=not used, 1 to 100 valid, default Null.
@p_PAD_INDEX	**0**	Index property, 0=Off, 1=On, set to 1 when using FillFactor.
@p_Stat_noComp	**0**	Index property, 0=Off, 1=On, should keep off.
@p_Allow_Row_Lock	**1**	Index property, 0=Off, 1=On, default is 1.
@p_Allow_Page_lock	**1**	Index property, 0=Off, 1=On, default is 1.
@p_Sort_in_TempDB	**1**	Index property, 0=Off, 1=On, when sorting in tempdb more space is required to rebuild.
@p_OnLine	**0**	Index property, 0=Off, 1=On, only supported for Enterprise edition.
@p_Reorg_Page_Lock	**Fix**	Fix or Skip. If Fix, for reorg index will be altered to allow page locks (else it fails).
@p_Debug	**0**	0=only errors,1=only alters,2=all.

Creating a Maintenance Plan

The following table explains how this Stored Procedure can be used as part of a complete maintenance plan. In this example, the plant has reduced activity Sunday morning at 2AM. This scenario uses Factelligence 3.5, where the archiving is done via a stored procedure that needs to be scheduled.

Note: You can change the name of the Database according to your application and version.

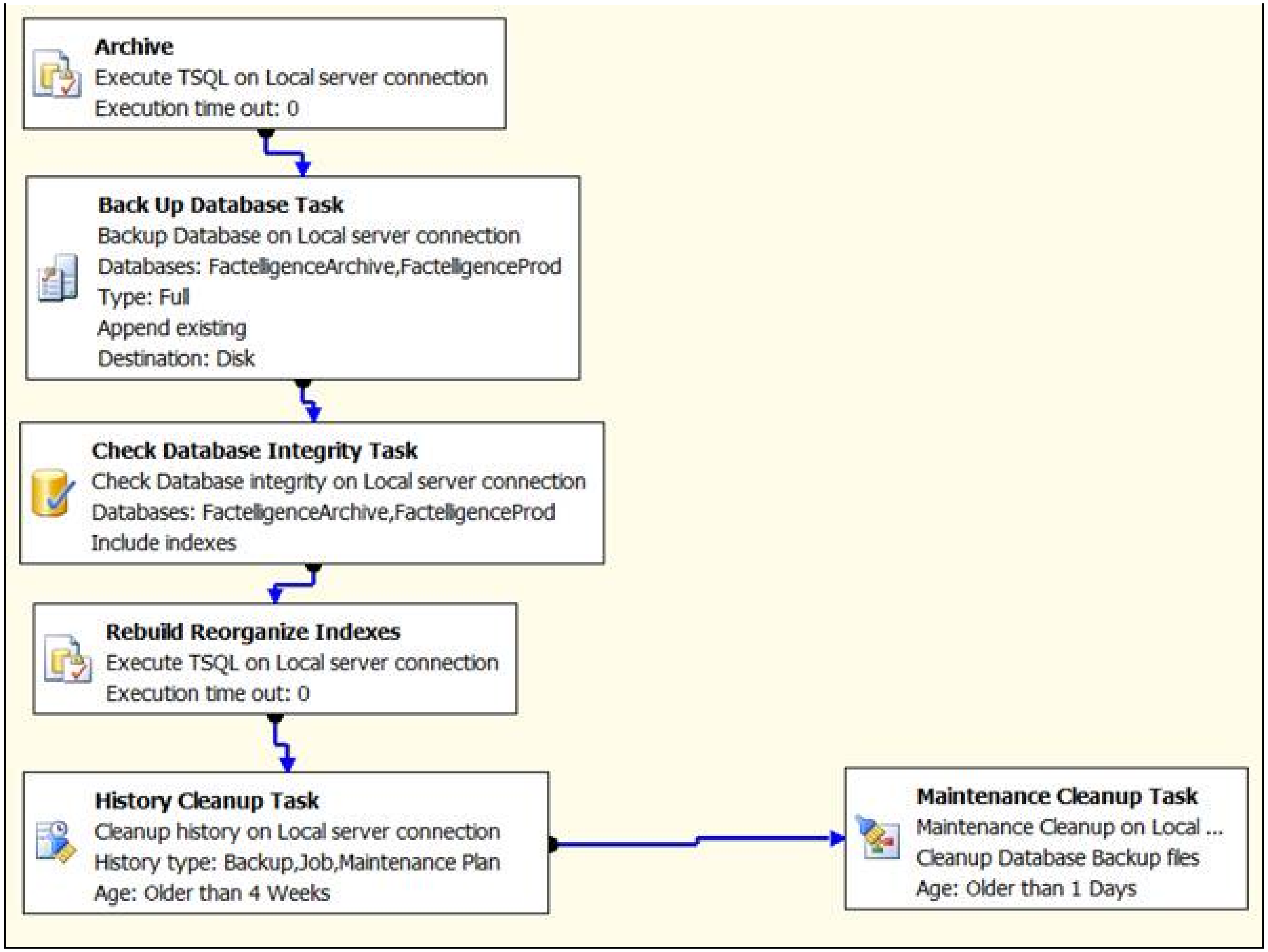
Name	**ArchiveBackupDefrag**		
Schedule	Daily at 1:00 a.m.		
Step 1	Execute T-SQL Statement Task	**Name**:	**Archive**
		T-SQL:	`EXEC` `FactelligenceProd.dbo.sp_Archive_Wo`
Step 2	Backup Database Task	**Name**:	**Backup Database Task**
		Type:	Full
		Databases:	• FactillegenceArchive • FactelligenceProd
		Folder:	<Specify your backup folder>
		Options:	• Create a backup file for every database • Verify Backup integrity
Step 3	Check Database Integrity Task	**Name**:	Test Database Integrity Task
		Databases:	• FactelligenceArchive • FactelligenceProd
		Options:	Include Indexes
Step 4	Execute T-SQL Statement Task	**Name**:	Rebuild Reorganize Indexes

		T-SQL:	`EXEC`
			`FactelligenceProd.dbo.ww_spDefrag.ww_spDefrag`
			`@p_Schedule='Weekly'`
			`,@p_DayNo=7`
			`,@p_RunHour=NULL`

```
,@p_Debug=1
```

Step 5	History Cleanup Task	Age: Older than 4 weeks	
Step 6	Maintenance Cleanup Task	**Options:**	Backup files
			Search folders and delete files based on extension
		Folder:	Same as in Backup Database Task
		Extensions:	bak
		File Age:	Older than 1 day

In SQLServer, this Maintenance Plan looks like Figure 1 (below).



Archive
Execute TSQL on Local server connection
Execution time out: 0

Back Up Database Task
Backup Database on Local server connection
Databases: FactelligenceArchive,FactelligenceProd
Type: Full
Append existing
Destination: Disk

Check Database Integrity Task
Check Database integrity on Local server connection
Databases: FactelligenceArchive,FactelligenceProd
Include indexes

Rebuild Reorganize Indexes
Execute TSQL on Local server connection
Execution time out: 0

History Cleanup Task
Cleanup history on Local server connection
History type: Backup,Job,Maintenance Plan
Age: Older than 4 Weeks

Maintenance Cleanup Task
Maintenance Cleanup on Local ...
Cleanup Database Backup files
Age: Older than 1 Days

Figure 1: SQL Server Maintenance Plan

Make sure the connection between tasks is **Completion** so that even if it fails the next task is done (Figure 2 below).

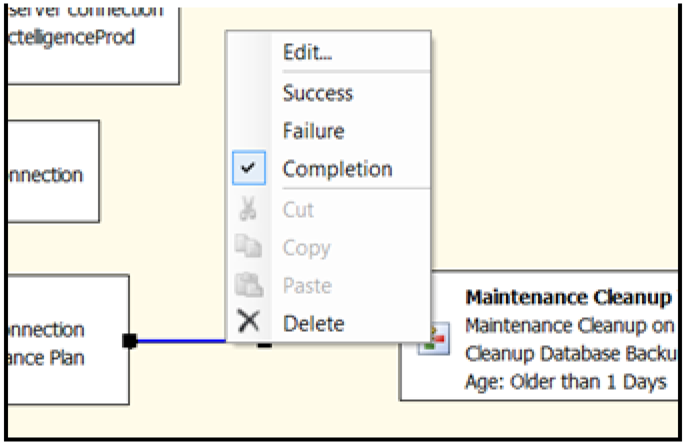


FIGURE 2: TASK CONNECTION IS SET TO COMPLETION

D. McKenna

Tech Notes are published occasionally by Wonderware Technical Support. Publisher: Invensys Systems, Inc., 26561 Rancho Parkway South, Lake Forest, CA 92630. There is also technical information on our software products at **Wonderware Technical Support.**

For technical support questions, send an e-mail to **wwsupport@invensys.com**.

Back to top