dbWarden - A Free SQL Server Monitoring Package

Stevie-Rounds, 2013-04-12

Overview

This past year my wife and I have been creating (compiling, writing, rewriting, etc) a set of tools that allows us to efficiently monitor our SQL Server 2005 and beyond databases without spending money on a monitoring tool. I work at a company that has a very tight budget and I wanted a way for the database to tell me if something was wrong, or about to go wrong. I also wanted something that was created using SQL and easy to deploy to a new instance of SQL Server.

We decided we wanted to start monitoring things like;

- File growth (TempDB and LDF)
- Performance statistics (CPU, connections and memory)
- Database blocking
- Long running queries
- Long running SQL jobs

We used SQL Jobs and stored procedures to capture data and send emails or text alerts based on metrics that we defined. We created a couple of tables to store the configuration of these alerts. For example, I need to know when there is blocking that lasts longer than 20 seconds. My wife didn't need to know about any blocking until well after that. You can also configure which databases you want to be excluded from being monitored or just excluded from specific alerting.

In addition to the monitoring and alerts, we wanted to create a comprehensive daily report for checking the overall health of a database server. It had to provide a good summary view of the server and what happened in the last 24 hours. We found a daily health report created via a stored procedure, that guest columnist Ritesh Medhe had published on SQLServerCentral.com. It dynamically generates an HTML blob and delivers the results via email. http://www.sqlservercentral.com/articles/Automating+SQL+Server+Health+Checks/

68910/. It was a great idea and we decided to expand on it by adding more details to existing sections in his Health Check and adding sections of our own.

We've collected everything in a self-contained script that we could run on a new server. It creates a database and all the stored procedures, tables, SQL Jobs, etc. to install our monitoring solution. I was getting tired of calling it "our monitoring solution" and we decided to name it. DBWarden was born!

DBWarden relies on SQL Agent, DBMail and the use of Operators.

There is a data dictionary to help explain the usage of the tables. We included a set of objects used to create the data dictionary. These data dictionary objects were created by a columnist on SQLServerCentral.com, David Poole in Oiling the gears for the data dictionary. It's a very easy way to get all the extended properties in place.

Installation and Setup

DBWarden can be installed by simply running the SQL script on your database server. There is a header in the file that contains the objects included in the script and brief install instructions. The script can be found on our Sourceforge page http://sourceforge.net/projects/dbwarden/.

Before running the script, you will need to find/replace the phrase CHANGEME with relevant data. There are two areas you will find this. The first section is creating the DBMail Operators. There is an operator for email and one for text. These operators will be used later in the script to populate the configuration tables. The other section is the update to the DatabaseSettings table. You can list the databases that you don't want to be part of the monitoring.

**This includes Schema change tracking. If you do NOT want schema change tracking on a database, make sure you add the database name and set SchemaTracking to 0. Otherwise, the database trigger and schemechangelog table WILL be created in that database.*

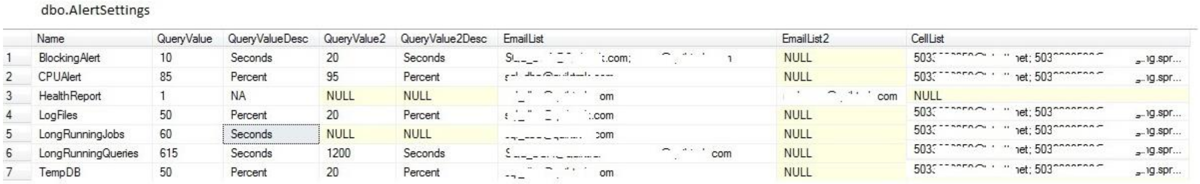
You also have the ability to easily rename the database name where DBWarden will reside. Right after the first section that creates the operators is the create database statement. The default is [dba]. Use Find/Replace to change "[dba]" to "[newDatabaseName]".

Now you are ready to run the script. It will create the database and all objects needed, including setting up the extended properties.

Configuration for Alerting

There are two configuration tables, AlertSettings and DatabaseSettings. Alert settings is used for individual alert settings such as blocking and controlling the thresholds for long running queries and jobs, log file growth, TempDB growth and CPU usage. There are two settings for each alert, one for email and one for text. The “QueryValue” column and “QueryValuedDesc” column are the configurations for the email alerts. The “EmailList” column is used for the primary recipients and “EmailList2” is used for the carbon copied recipients of the email alerts. The “QueryValue2” column and “QueryValuedDesc2” column are they configurations for the text alerts. The “CellList” column is used for the recipients of the text alerts.

There is also an entry for the Health report to determine who to email the results to, although the health report can’t be texted so those settings will not be used for this alert. Here is a screenshot of the table contents with default settings followed by a description of those settings. These settings can be modified in the DBWarden script for the insert to this table or via a table update after the inserts are made.



dbo.AlertSettings

	Name	QueryValue	QueryValueDesc	QueryValue2	QueryValue2Desc	EmailList	EmailList2	CellList
1	BlockingAlert	10	Seconds	20	Seconds	[redacted]	NULL	[redacted]
2	CPUAlert	85	Percent	95	Percent	[redacted]	NULL	[redacted]
3	HealthReport	1	NA	NULL	NULL	[redacted]	[redacted]	NULL
4	LogFiles	50	Percent	20	Percent	[redacted]	NULL	[redacted]
5	LongRunningJobs	60	Seconds	NULL	NULL	[redacted]	NULL	[redacted]
6	LongRunningQueries	615	Seconds	1200	Seconds	[redacted]	NULL	[redacted]
7	TempDB	50	Percent	20	Percent	[redacted]	NULL	[redacted]

Each of the items are explained below.

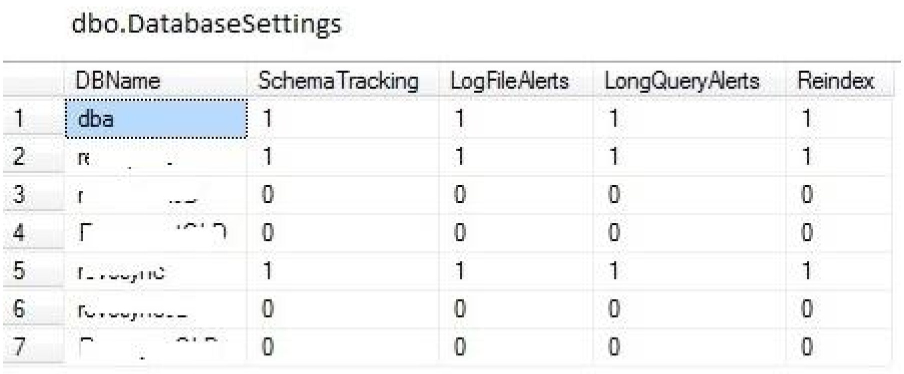
- Blocking alert – email if blocking occurs for 10 seconds, start sending text alerts after 20 seconds of blocking
- CPU alert – When the CPU percentage of SQL Server rises above 85%, it will send an email. A text alert is sent at 95%.   This alert is sent via the job dba_BlockingAlert.
- Health Report – The health report will be sent to the email in EmailList. The Health Report is sent via the job dba_HealthReport.
- Log file growth – When an LDF is more than 50% used, send an email. If the LDF grows and has 20% space left. This alert is sent via the job dba_CheckFiles.
- Long running jobs – 60 seconds – Only report long running jobs that run longer than 60 seconds (possibly indicating it’s about to grow), it sends a text alert. This alert is sent via the job dba_LongRunningJobs.

- Long running queries – Send an email when queries run longer than 612 seconds and text alerts after 1200 seconds. This alert is sent via the job dba_LongRunningQueries.
- TempDB growth – same as above, 50% for email notification, 20% for text alert. This alert is sent via the job dba_CheckFiles.

The DatabaseSettings table is for turning monitoring options on/off for a given database. 0 is Off and 1 is On. All user databases are monitored by default. When SchemaTracking is On a database trigger called dbo.tr_DDL_SchemaChangeLog and table called dbo.SchemaChangeLog are created on that database. Each time a schema modification is made the trigger puts a record into SchemaChangeLog.

This data is reported in the Health Report via the job dba_HealthReport, detailed below. SchemaTracking should not be on for replicated database and should be tested for other databases. We have found that some third party tools will stop being able to upgrade and perform certain operations due to the database trigger. LogFileAlerts is used by the job dba_CheckFiles detailed below. If this is On for a database, it's ldf is monitored for growth. The LongQueryAlerts flag is used by the job dba_LongRunningQueries to know which databases to collect long running query information for.

There is also a flag in the DatabaseSettings table to stop Reindexing. This is a work in progress feature. We created a wrapper procedure for Ola Hallengren's reindex/defrag script to look at this table to determine if a database should be skipped during reindex/defrag. We have not included the wrapper procedure in the DBWarden script as of yet.



dbo.DatabaseSettings

	DBName	SchemaTracking	LogFileAlerts	LongQueryAlerts	Reindex
1	dba	1	1	1	1
2	[redacted]	1	1	1	1
3	[redacted]	0	0	0	0
4	[redacted]	0	0	0	0
5	[redacted]	1	1	1	1
6	[redacted]	0	0	0	0
7	[redacted]	0	0	0	0

SQL Jobs for Monitoring and Alerting

By default, all SQL Jobs will be disabled so you can modify the job schedules if you do not like the defaults. Enable them to start collecting data and send alerts. Here is a description of the jobs and the default schedules;

- dba_BlockingAlert – Runs every 5 seconds. This will run usp_CheckBlocking and will store data collected into the BlockingHistory table. There is a trigger on this table that will send the alert. The trigger is used because of a bug experienced previously with inaccurate reporting.
- dba_CheckFiles (LDF and TempDB) – Runs every 1 hour. This will run usp_CheckFiles to determine if TempDB or any LDF is growing. It utilizes the stored procedure usp_FileStats to collect data into the FileStatsHistory table.
- dba_CPUAlert – Runs every 5 minutes. This will run usp_CPUProcessAlert to determine if the CPU thresholds have been met. It utilizes the stored procedure usp_CPUStats to collect data into the CPUStatsHistory table.
- dba_HealthReport – Runs every day at 6am. The details of this health report are in a section below. You can have the report run less frequently than daily, but certain sections listed as “Last 24 hours” will only report on information from the last 24 hours. This will run rpt_HealthReport with the following parameters

  § @Recipients – Primary recipients

  § @CC – Carbon Copy recipients

  § @InsertFlag – To control whether the report is stored in the HealthReport table.

  § @IncludePerfStats – To include Performance statistics; Buffer Hit Cache and CPU graphs. You must have the dba_CPUAlert, dba_MemoryUsageStats, and dba_PerfStats jobs running to collect data in order to turn this on.

  § @EmailFlag – If you don’t want to email the report (for future use when we create a UI for retrieving past Health Reports)

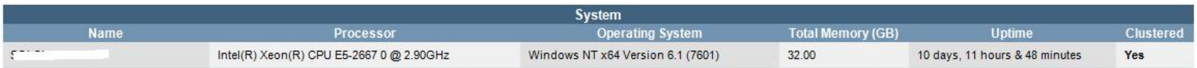
- dba_LongRunningJobs – Runs every 1 hour, This will run usp_LongRunningJobs to determine if and SQL Jobs are currently running over their average runtime. It utilizes the stored procedure usp_JobStats to collect data into the JobStatsHistory table.

- dba_LongRunningQueries – Runs every 5 minutes. This will run usp_LongRunningQueries to determine if any queries on a database are running too long. It also stores data into the QueryHistory table.
- dba_MemoryUsageStats – Runs every 15 minutes. This will run usp_MemoryUsageStats which gathers information using dm_os_sys_info and inserts this data into the MemoryUsageHistory table.
- dba_PerfStats – runs every 5 minutes. This will run usp_PerfStats which gathers information using dm_os_performance_counters and collects data into the PerfStatsHistory table.

Health Report

Here are the components included in our health report. Ritesh Medhe's Health Check included sections that we used for Job Status, Databases, Disks, and SQL Server Database Backup Stats.

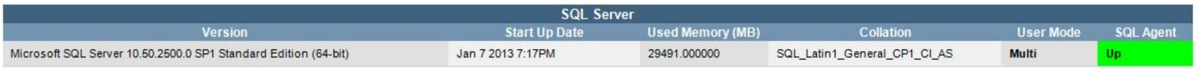
System: This section contains the instance name, processor specifications, server operating system, total server memory, instance uptime, and is clustered



System

Name	Processor	Operating System	Total Memory (GB)	Uptime	Clustered
[redacted]	Intel(R) Xeon(R) CPU E5-2667 0 @ 2.90GHz	Windows NT x64 Version 6.1 (7601)	32.00	10 days, 11 hours & 48 minutes	**Yes**

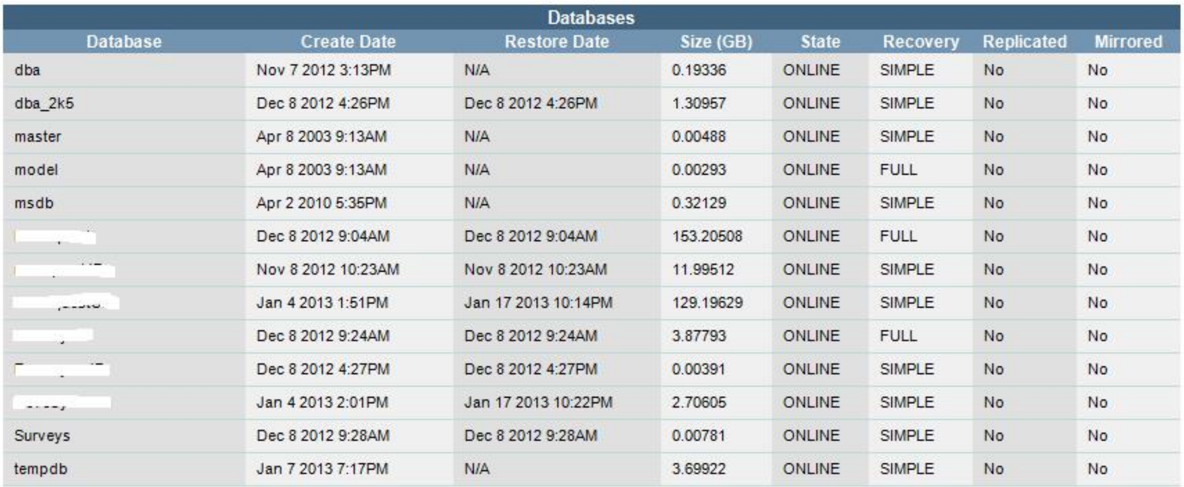
SQL Server: This section contains the SQL Server version, instance startup date, memory foot print, default collation, user mode, and SQL Agent status.



SQL Server

Version	Start Up Date	Used Memory (MB)	Collation	User Mode	SQL Agent
Microsoft SQL Server 10.50.2500.0 SP1 Standard Edition (64-bit)	Jan 7 2013 7:17PM	29491.000000	SQL_Latin1_General_CP1_CI_AS	**Multi**	**Up**

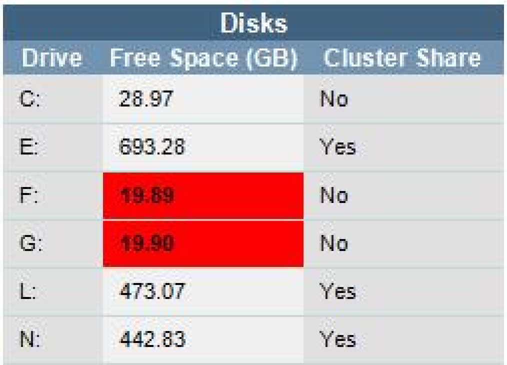
Databases: This section contains a list of databases with creation date, database size, state (ie. online, offline), recovery model, is replicated and is mirrored. SQL Agent will be green if up and red if down.



Databases

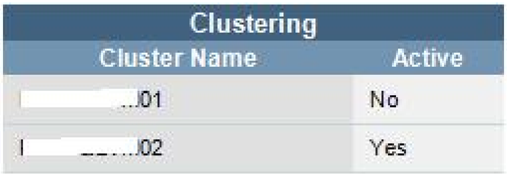
Database	Create Date	Restore Date	Size (GB)	State	Recovery	Replicated	Mirrored
dba	Nov 7 2012 3:13PM	N/A	0.19336	ONLINE	SIMPLE	No	No
dba_2k5	Dec 8 2012 4:26PM	Dec 8 2012 4:26PM	1.30957	ONLINE	SIMPLE	No	No
master	Apr 8 2003 9:13AM	N/A	0.00488	ONLINE	SIMPLE	No	No
model	Apr 8 2003 9:13AM	N/A	0.00293	ONLINE	FULL	No	No
msdb	Apr 2 2010 5:35PM	N/A	0.32129	ONLINE	SIMPLE	No	No
[redacted]	Dec 8 2012 9:04AM	Dec 8 2012 9:04AM	153.20508	ONLINE	FULL	No	No
[redacted]	Nov 8 2012 10:23AM	Nov 8 2012 10:23AM	11.99512	ONLINE	SIMPLE	No	No
[redacted]	Jan 4 2013 1:51PM	Jan 17 2013 10:14PM	129.19629	ONLINE	SIMPLE	No	No
[redacted]	Dec 8 2012 9:24AM	Dec 8 2012 9:24AM	3.87793	ONLINE	FULL	No	No
[redacted]	Dec 8 2012 4:27PM	Dec 8 2012 4:27PM	0.00391	ONLINE	SIMPLE	No	No
[redacted]	Jan 4 2013 2:01PM	Jan 17 2013 10:22PM	2.70605	ONLINE	SIMPLE	No	No
Surveys	Dec 8 2012 9:28AM	Dec 8 2012 9:28AM	0.00781	ONLINE	SIMPLE	No	No
tempdb	Jan 7 2013 7:17PM	N/A	3.69922	ONLINE	SIMPLE	No	No

Disks: This section includes the disk drives available, free space on each drive, and whether or not each drive is a cluster share. The free space will show up as red if the drive space is less than 20 GB. We had trouble finding a reliable way to get the total drive space via SQL, otherwise this section would be included and the red would show up if a certain percentage of the disk drive was free.



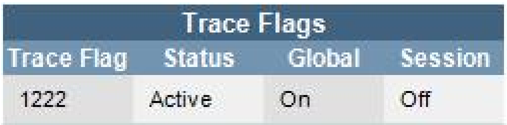
Disks		
Drive	Free Space (GB)	Cluster Share
C:	28.97	No
E:	693.28	Yes
F:	19.89	No
G:	19.90	No
L:	473.07	Yes
N:	442.83	Yes

Clustering: This section includes cluster node name and which node is active.



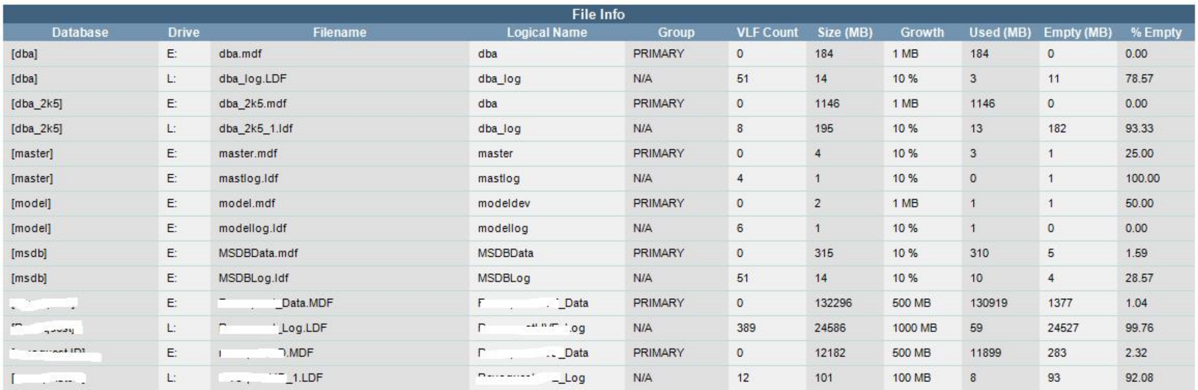
Clustering	
Cluster Name	Active
[redacted]01	No
[redacted]02	Yes

Trace flags: This section will include active global trace flags.



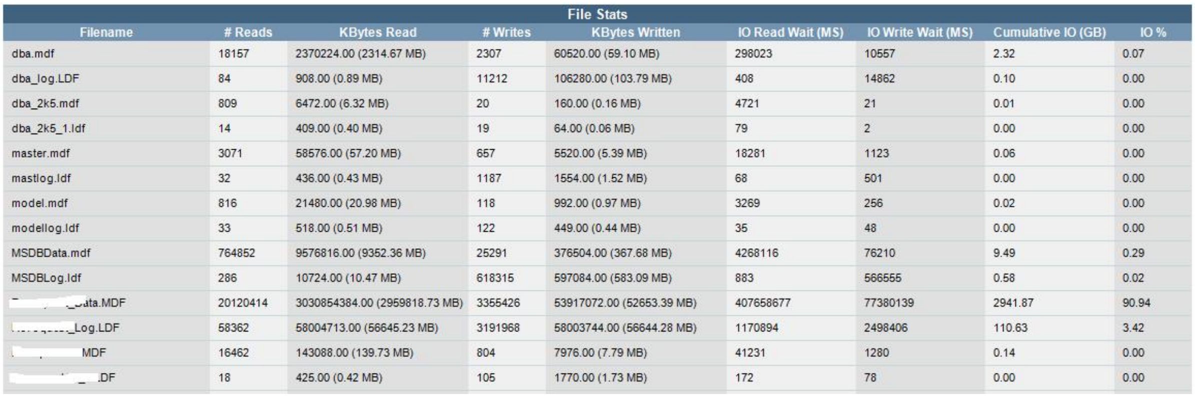
Trace Flags			
Trace Flag	Status	Global	Session
1222	Active	On	Off

File Info: This section contains the database file names, what drive they are located on, the files logical name, what group they are located on, virtual log file counts, growth settings, space used, space empty and percentage empty. If an LDF is larger than an MDF, the LDF size will show up in yellow to highlight a potential problem or configuration issue (i.e. Database mode is set incorrectly)



File Info										
Database	Drive	Filename	Logical Name	Group	VLF Count	Size (MB)	Growth	Used (MB)	Empty (MB)	% Empty
[dba]	E:	dba.mdf	dba	PRIMARY	0	184	1 MB	184	0	0.00
[dba]	L:	dba_log.LDF	dba_log	N/A	51	14	10 %	3	11	78.57
[dba_2k5]	E:	dba_2k5.mdf	dba	PRIMARY	0	1146	1 MB	1146	0	0.00
[dba_2k5]	L:	dba_2k5_1.ldf	dba_log	N/A	8	195	10 %	13	182	93.33
[master]	E:	master.mdf	master	PRIMARY	0	4	10 %	3	1	25.00
[master]	E:	mastlog.ldf	mastlog	N/A	4	1	10 %	0	1	100.00
[model]	E:	model.mdf	modeldev	PRIMARY	0	2	1 MB	1	1	50.00
[model]	E:	modellog.ldf	modellog	N/A	6	1	10 %	1	0	0.00
[msdb]	E:	MSDBData.mdf	MSDBData	PRIMARY	0	315	10 %	310	5	1.59
[msdb]	E:	MSDBLog.ldf	MSDBLog	N/A	51	14	10 %	10	4	28.57
[redacted]	E:	[redacted]_Data.MDF	[redacted]_Data	PRIMARY	0	132296	500 MB	130919	1377	1.04
[redacted]	L:	[redacted]_Log.LDF	[redacted]og	N/A	389	24586	1000 MB	59	24527	99.76
[redacted]	E:	[redacted].MDF	[redacted]_Data	PRIMARY	0	12182	500 MB	11899	283	2.32
[redacted]	L:	[redacted]_1.LDF	[redacted]_Log	N/A	12	101	100 MB	8	93	92.08

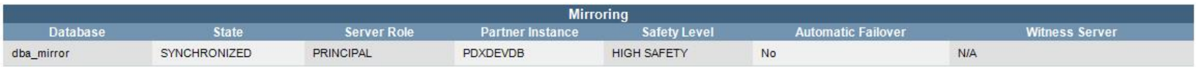
File Stats: This section includes each database file, with recent read and write statistics.



File Stats

Filename	# Reads	KBytes Read	# Writes	KBytes Written	IO Read Wait (MS)	IO Write Wait (MS)	Cumulative IO (GB)	IO %
dba.mdf	18157	2370224.00 (2314.67 MB)	2307	60520.00 (59.10 MB)	298023	10557	2.32	0.07
dba_log.LDF	84	908.00 (0.89 MB)	11212	106280.00 (103.79 MB)	408	14862	0.10	0.00
dba_2k5.mdf	809	6472.00 (6.32 MB)	20	160.00 (0.16 MB)	4721	21	0.01	0.00
dba_2k5_1.ldf	14	409.00 (0.40 MB)	19	64.00 (0.06 MB)	79	2	0.00	0.00
master.mdf	3071	58576.00 (57.20 MB)	657	5520.00 (5.39 MB)	18281	1123	0.06	0.00
mastlog.ldf	32	436.00 (0.43 MB)	1187	1554.00 (1.52 MB)	68	501	0.00	0.00
model.mdf	816	21480.00 (20.98 MB)	118	992.00 (0.97 MB)	3269	256	0.02	0.00
modellog.ldf	33	518.00 (0.51 MB)	122	449.00 (0.44 MB)	35	48	0.00	0.00
MSDBData.mdf	764852	9576816.00 (9352.36 MB)	25291	376504.00 (367.68 MB)	4268116	76210	9.49	0.29
MSDBLog.ldf	286	10724.00 (10.47 MB)	618315	597084.00 (583.09 MB)	883	566555	0.58	0.02
[redacted]ata.MDF	20120414	3030854384.00 (2959818.73 MB)	3355426	53917072.00 (52653.39 MB)	407658677	77380139	2941.87	90.94
[redacted]_Log.LDF	58362	58004713.00 (56645.23 MB)	3191968	58003744.00 (56644.28 MB)	1170894	2498406	110.63	3.42
[redacted]MDF	16462	143088.00 (139.73 MB)	804	7976.00 (7.79 MB)	41231	1280	0.14	0.00
[redacted].DF	18	425.00 (0.42 MB)	105	1770.00 (1.73 MB)	172	78	0.00	0.00

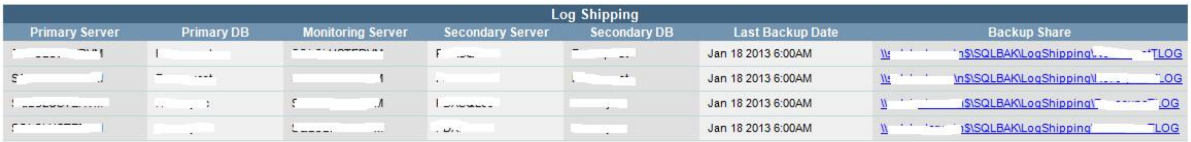
Mirroring: This section will show up if you have mirroring set up. It will tell you the database(s) being mirrored, the state, server role, partner instance, safety level, if automatic failover is on, and if it is a witness server.



Mirroring

Database	State	Server Role	Partner Instance	Safety Level	Automatic Failover	Witness Server
dba_mirror	SYNCHRONIZED	PRINCIPAL	PDXDEVDB	HIGH SAFETY	No	N/A

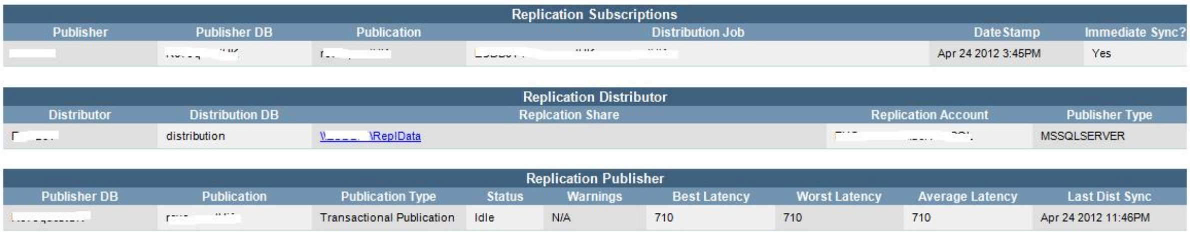
Log Shipping: This section will show up if log shipping is set up and configured as a Primary, Secondary or Monitoring server.



Log Shipping

Primary Server	Primary DB	Monitoring Server	Secondary Server	Secondary DB	Last Backup Date	Backup Share
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Jan 18 2013 6:00AM	\\[redacted]n$\SQLBAK\LogShipping\[redacted]TLOG
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Jan 18 2013 6:00AM	\\[redacted]n$\SQLBAK\LogShipping\[redacted]LOG
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Jan 18 2013 6:00AM	\\[redacted]$\SQLBAK\LogShipping\[redacted]LOG
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Jan 18 2013 6:00AM	\\[redacted]$\SQLBAK\LogShipping[redacted]TLOG

Replication: This section will show up if Replication is set up and configured as a Publisher, Subscriber or Distributor on the server.



Replication Subscriptions

Publisher	Publisher DB	Publication	Distribution Job	Date Stamp	Immediate Sync?
[redacted]	[redacted]	[redacted]	[redacted]	Apr 24 2012 3:45PM	Yes

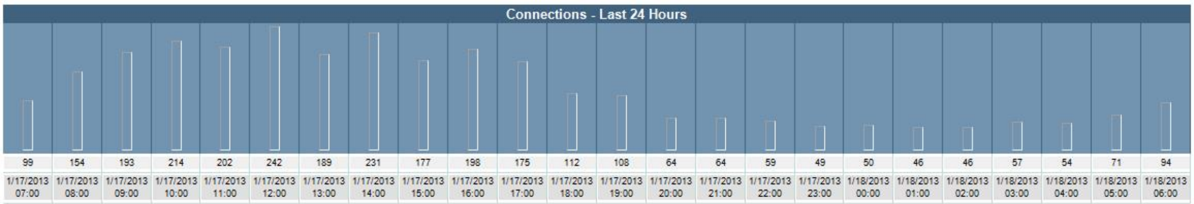
Replication Distributor

Distributor	Distribution DB	Replcation Share	Replication Account	Publisher Type
[redacted]	distribution	\\[redacted]\ReplData	[redacted]	MSSQLSERVER

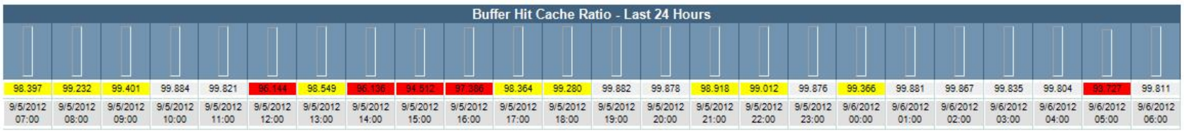
Replication Publisher

Publisher DB	Publication	Publication Type	Status	Warnings	Best Latency	Worst Latency	Average Latency	Last Dist Sync
[redacted]	[redacted]	Transactional Publication	Idle	N/A	710	710	710	Apr 24 2012 11:46PM

Connections: If the SQL job to collect Performance statistics is on and the Health Report parameter (@IncludePerfStats) is set to 1, then this section will show up displaying the connection count in hourly intervals.



Buffer: If the SQL job to collect Performance statistics is on and the Health Report parameter (@IncludePerfStats) is set to 1, then this section will show up displaying the buffer cache percentage in hourly intervals.

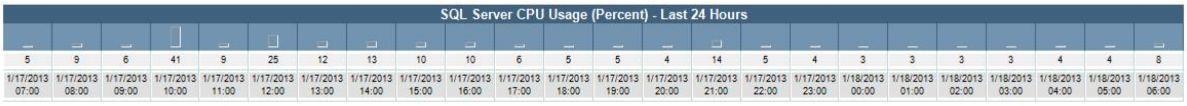


Buffer Hit Cache Ratio - Last 24 Hours

9/5/2012 07:00	9/5/2012 08:00	9/5/2012 09:00	9/5/2012 10:00	9/5/2012 11:00	9/5/2012 12:00	9/5/2012 13:00	9/5/2012 14:00	9/5/2012 15:00	9/5/2012 16:00	9/5/2012 17:00	9/5/2012 18:00
98.397	99.232	99.401	99.884	99.821	96.144	98.549	96.136	94.512	97.386	98.364	99.280

9/5/2012 19:00	9/5/2012 20:00	9/5/2012 21:00	9/5/2012 22:00	9/5/2012 23:00	9/6/2012 00:00	9/6/2012 01:00	9/6/2012 02:00	9/6/2012 03:00	9/6/2012 04:00	9/6/2012 05:00	9/6/2012 06:00
99.882	99.878	98.918	99.012	99.876	99.366	99.881	99.867	99.835	99.804	93.727	99.811

CPU: If the SQL job to collect Performance statistics is on and the Health Report parameter (@IncludePerfStats) is set to 1, then this section will show up displaying the SQL Server CPU usage in hourly intervals.

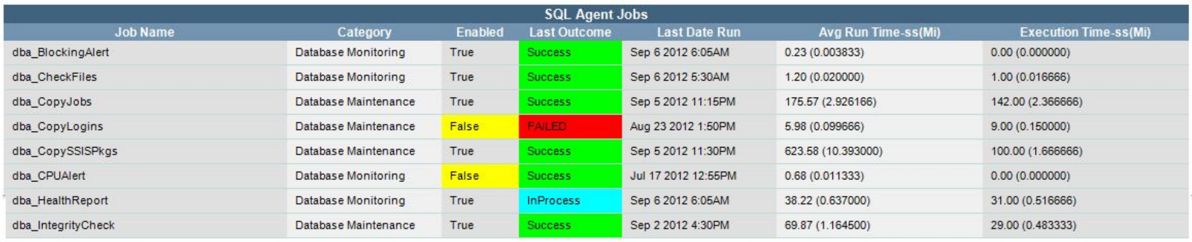


SQL Server CPU Usage (Percent) - Last 24 Hours

1/17/2013 07:00	1/17/2013 08:00	1/17/2013 09:00	1/17/2013 10:00	1/17/2013 11:00	1/17/2013 12:00	1/17/2013 13:00	1/17/2013 14:00	1/17/2013 15:00	1/17/2013 16:00	1/17/2013 17:00	1/17/2013 18:00
5	9	6	41	9	25	12	13	10	10	6	5

1/17/2013 19:00	1/17/2013 20:00	1/17/2013 21:00	1/17/2013 22:00	1/17/2013 23:00	1/18/2013 00:00	1/18/2013 01:00	1/18/2013 02:00	1/18/2013 03:00	1/18/2013 04:00	1/18/2013 05:00	1/18/2013 06:00
5	4	14	5	4	3	3	3	3	4	4	8

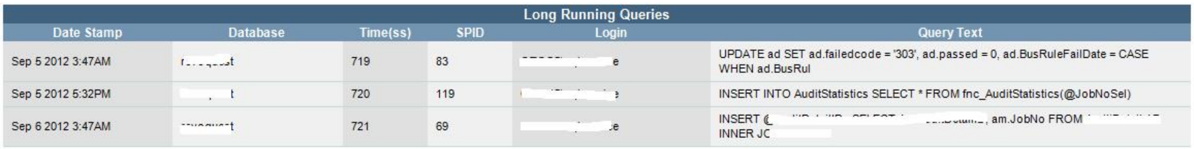
SQL Agent Jobs: This section will display all SQL Jobs and their current state, last run outcome, average run times and last run execution time.



SQL Agent Jobs

Job Name	Category	Enabled	Last Outcome	Last Date Run	Avg Run Time-ss(Mi)	Execution Time-ss(Mi)
dba_BlockingAlert	Database Monitoring	True	Success	Sep 6 2012 6:05AM	0.23 (0.003833)	0.00 (0.000000)
dba_CheckFiles	Database Monitoring	True	Success	Sep 6 2012 5:30AM	1.20 (0.020000)	1.00 (0.016666)
dba_CopyJobs	Database Maintenance	True	Success	Sep 5 2012 11:15PM	175.57 (2.926166)	142.00 (2.366666)
dba_CopyLogins	Database Maintenance	False	FAILED	Aug 23 2012 1:50PM	5.98 (0.099666)	9.00 (0.150000)
dba_CopySSISPkgs	Database Maintenance	True	Success	Sep 5 2012 11:30PM	623.58 (10.393000)	100.00 (1.666666)
dba_CPUAlert	Database Monitoring	False	Success	Jul 17 2012 12:55PM	0.68 (0.011333)	0.00 (0.000000)
dba_HealthReport	Database Monitoring	True	InProcess	Sep 6 2012 6:05AM	38.22 (0.637000)	31.00 (0.516666)
dba_IntegrityCheck	Database Maintenance	True	Success	Sep 2 2012 4:30PM	69.87 (1.164500)	29.00 (0.483333)

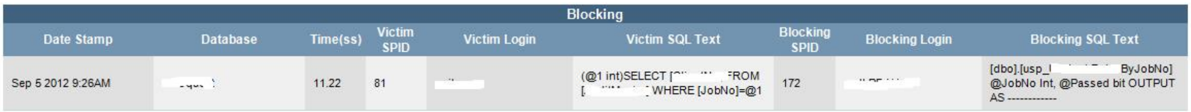
Long Running Queries: This section will show details if there have been any long running queries on the server. It will display session information and a limited amount of the query text that was run.



Long Running Queries

Date Stamp	Database	Time(ss)	SPID	Login	Query Text
Sep 5 2012 3:47AM	[redacted]	719	83	[redacted]	UPDATE ad SET ad.failedcode = '303', ad.passed = 0, ad.BusRuleFailDate = CASE WHEN ad.BusRul
Sep 5 2012 5:32PM	[redacted]	720	119	[redacted]	INSERT INTO AuditStatistics SELECT * FROM fnc_AuditStatistics(@JobNoSel)
Sep 6 2012 3:47AM	[redacted]	721	69	[redacted]	INSERT [redacted], am.JobNo FROM [redacted] INNER [redacted]

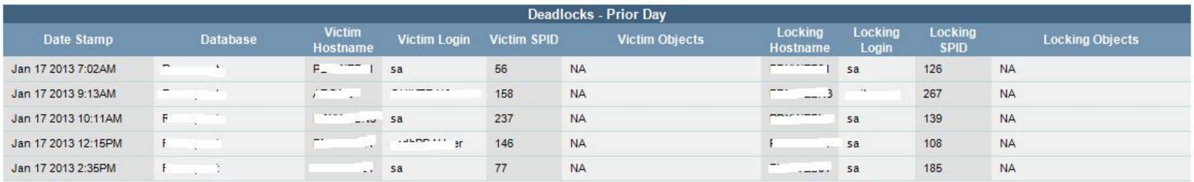
Blocking: This section will show details of any blocking that occurred on the server. It will display session information and a limited amount of the query text for blocking and blocked sessions.



Blocking

Date Stamp	Database	Time(ss)	Victim SPID	Victim Login	Victim SQL Text	Blocking SPID	Blocking Login	Blocking SQL Text
Sep 5 2012 9:26AM	[redacted]	11.22	81	[redacted]	(@1 int)SELECT [redacted] FROM [redacted] WHERE [JobNo]=@1	172	[redacted]	[dbo].[usp_[redacted] ByJobNo] @JobNo Int, @Passed bit OUTPUT AS ------------

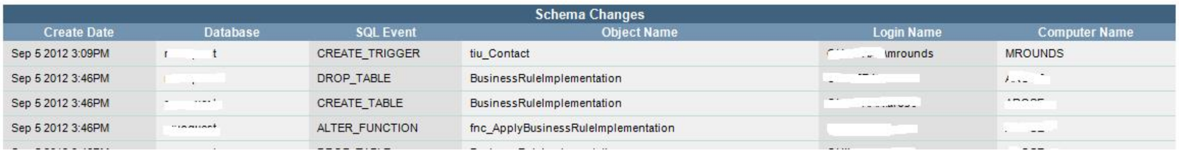
Deadlocks: If Traceflag 1222 is on globally and you have recorded deadlocks in the error log, they will be displayed here. This section is a work in progress and will show the locking and victim's login and session information. The idea is to also display the database objects that were involved.



Deadlocks - Prior Day

Date Stamp	Database	Victim Hostname	Victim Login	Victim SPID	Victim Objects	Locking Hostname	Locking Login	Locking SPID	Locking Objects
Jan 17 2013 7:02AM	[redacted]	[redacted]	sa	56	NA	[redacted]	sa	126	NA
Jan 17 2013 9:13AM	[redacted]	[redacted]	[redacted]	158	NA	[redacted]	[redacted]	267	NA
Jan 17 2013 10:11AM	[redacted]	[redacted]	sa	237	NA	[redacted]	sa	139	NA
Jan 17 2013 12:15PM	[redacted]	[redacted]	[redacted]	146	NA	[redacted]	sa	108	NA
Jan 17 2013 2:35PM	[redacted]	[redacted]	sa	77	NA	[redacted]	sa	185	NA

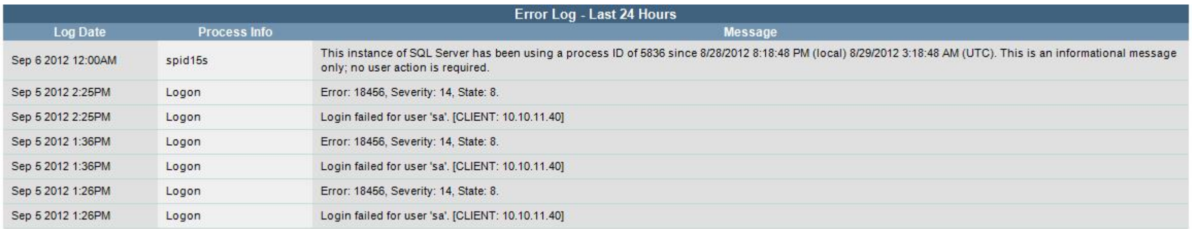
Schema Changes: This section will show any schema changes that have occurred on the databases that have been set up to have schema changes monitored via the trigger in dbo.DatabaseSettings.SchemaTracking. It includes create date for the date the change was made, the database affected, the type of event (ie. CREATE_TRIGGER, DROP_TABLE), the object affected, the login of the session that made the change, and the computer name of the session that made the change.



Schema Changes

Create Date	Database	SQL Event	Object Name	Login Name	Computer Name
Sep 5 2012 3:09PM	[redacted]	CREATE_TRIGGER	tiu_Contact	[redacted]mrounds	MROUNDS
Sep 5 2012 3:46PM	[redacted]	DROP_TABLE	BusinessRuleImplementation	[redacted]	[redacted]
Sep 5 2012 3:46PM	[redacted]	CREATE_TABLE	BusinessRuleImplementation	[redacted]	[redacted]
Sep 5 2012 3:46PM	[redacted]	ALTER_FUNCTION	fnc_ApplyBusinessRuleImplementation	[redacted]	[redacted]

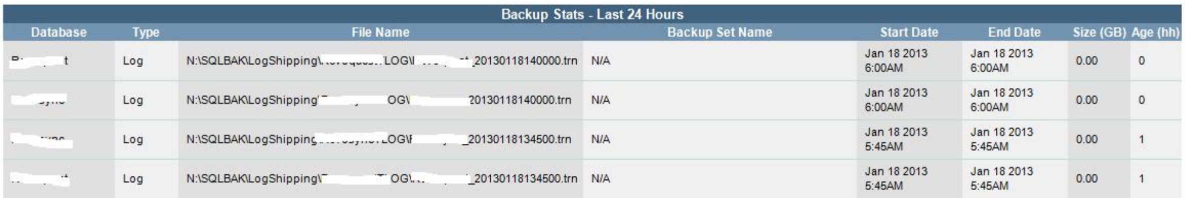
Error Log: This section shows the information that has been written to the error log in the last 24 hours. It excludes deadlock and backup details that have their own sections in this report.



Error Log - Last 24 Hours

Log Date	Process Info	Message
Sep 6 2012 12:00AM	spid15s	This instance of SQL Server has been using a process ID of 5836 since 8/28/2012 8:18:48 PM (local) 8/29/2012 3:18:48 AM (UTC). This is an informational message only; no user action is required.
Sep 5 2012 2:25PM	Logon	Error: 18456, Severity: 14, State: 8.
Sep 5 2012 2:25PM	Logon	Login failed for user 'sa'. [CLIENT: 10.10.11.40]
Sep 5 2012 1:36PM	Logon	Error: 18456, Severity: 14, State: 8.
Sep 5 2012 1:36PM	Logon	Login failed for user 'sa'. [CLIENT: 10.10.11.40]
Sep 5 2012 1:26PM	Logon	Error: 18456, Severity: 14, State: 8.
Sep 5 2012 1:26PM	Logon	Login failed for user 'sa'. [CLIENT: 10.10.11.40]

Backup: This section will show all of the backups that have been taken in the last 24 hours. It lists the database, the type of backup (ie. Full, Differential, Log), the File name and location, the backup set name if it is part of a set, the start date and the end date, the size in GB, and the age in hours.



Backup Stats - Last 24 Hours

Database	Type	File Name	Backup Set Name	Start Date	End Date	Size (GB)	Age (hh)
[redacted]	Log	N:\SQLBAK\LogShipping\[redacted]LOG\[redacted]_20130118140000.trn	N/A	Jan 18 2013 6:00AM	Jan 18 2013 6:00AM	0.00	0
[redacted]	Log	N:\SQLBAK\LogShipping\[redacted]OG\[redacted]20130118140000.trn	N/A	Jan 18 2013 6:00AM	Jan 18 2013 6:00AM	0.00	0
[redacted]	Log	N:\SQLBAK\LogShipping[redacted]LOG\[redacted]_20130118134500.trn	N/A	Jan 18 2013 5:45AM	Jan 18 2013 5:45AM	0.00	1
[redacted]	Log	N:\SQLBAK\LogShipping\[redacted]OG\[redacted]_20130118134500.trn	N/A	Jan 18 2013 5:45AM	Jan 18 2013 5:45AM	0.00	1

Summary

Not everyone can afford database monitoring software. Tight budgets often dictate how effective a DBA can be at keeping their servers running smoothly. Our goal with DBWarden is to help you monitor efficiently, without killing your budget.

DBWarden is a work in progress, and we’re always adding new features to it and fixing any bugs found. We are very excited to provide what we’ve done to the community to use and to contribute to its evolvement.

If you'd like to give DBWarden a try, the code can be found on our Sourceforge page at: http://sourceforge.net/projects/dbwarden/.