



Avantis • SimSci • Wonderware

THE INDUSTRIAL SOFTWARE
REVOLUTION
BEGINS NOW

invensysTM

WW TSS-02\03 MS SQL Server Extended Performance & Tuning

Pierluigi Iodice

Regional Solution Support Engineer,
Wonderware – Invensys Software


Email: pierluigi.iodice@invensys.com

Javier Aldan

Technical Account Manager
Wonderware – Invensys Software


Email: javier.aldan@invensys.com

 social.invensys.com

 [@InvensysOpsMgmt](https://twitter.com/InvensysOpsMgmt) / [#SoftwareRevolution](https://twitter.com/SoftwareRevolution)

 [/InvensysVideos](https://www.youtube.com/InvensysVideos)

 [/Wonderware](https://www.facebook.com/Wonderware)

 [/company/Wonderware](https://www.linkedin.com/company/Wonderware)



Microsoft SQL Server



- Wonderware Products has developed with Microsoft Technologies
- All Microsoft development languages used are “Hand and Glove” with SQL Server

What we need to know about MS SQL Server?

- A. SQL Server within Wonderware Database
- B. Database Maintenance, Troubleshooting Tools & Diagnostic Query
- C. Hands on Custom Project with SQL



Maintenance: The plan

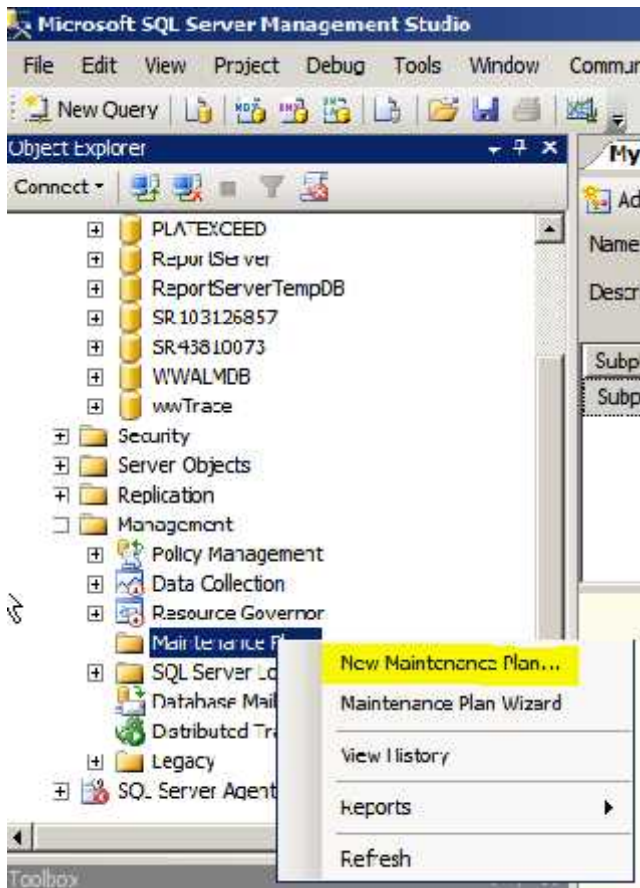
One of the **first tasks** for a new DB is set up a maintenance plan:

- Understand how to have a backup
- Define a recovery strategy
- Truncate Log and Shrink DB
- Check DB fragmentation
- Keep the database clean

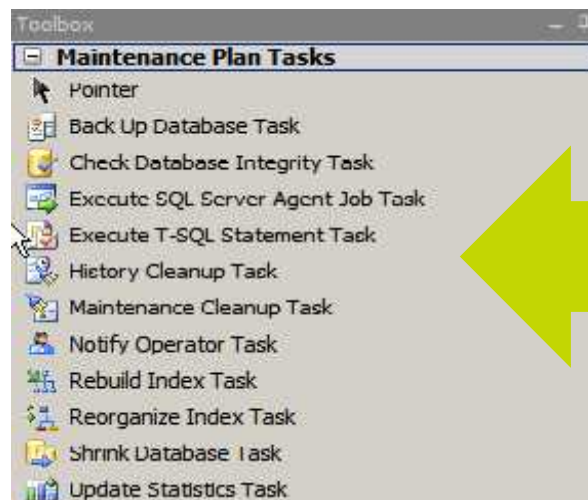


Maintenance: The backup strategy

You could use the embedded [SQL Server Maintenance](#):



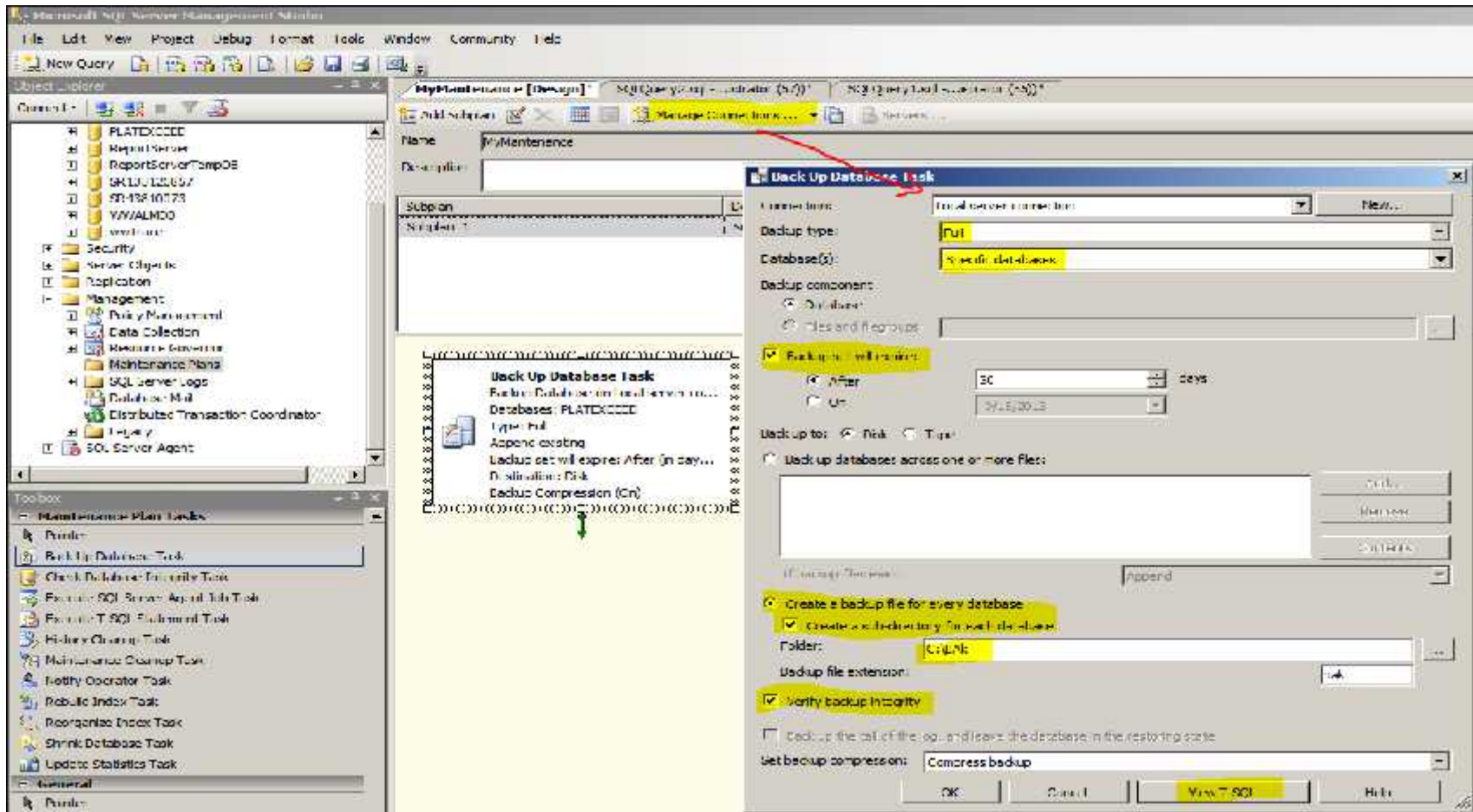
Maintenance plans create a workflow of the tasks required to make sure that your database is **optimized, regularly backed up, and free of inconsistencies**. The Maintenance Plan Wizard also creates core maintenance plans, but creating plans manually gives you much more flexibility.



Here all the task you can execute!

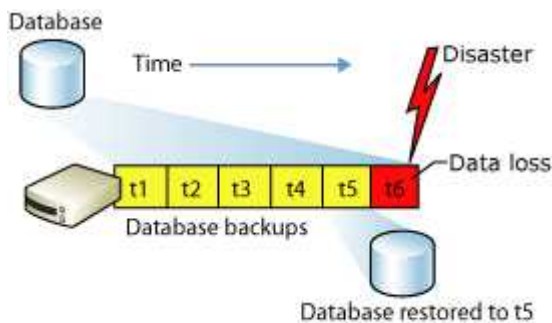
Maintenance: The backup strategy

Here below an example of backup with [SQL Server Maintenance](#):



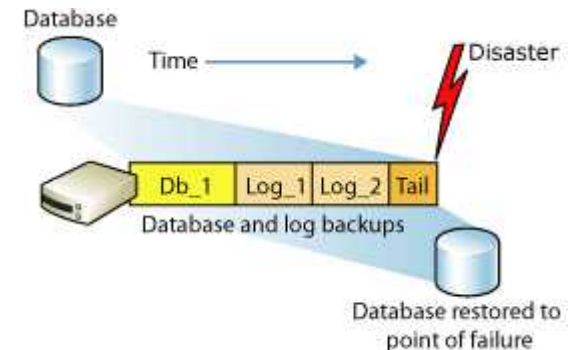
Maintenance: Backup and Restore mode

The DB administrator **needs to decide** on a backup and restore strategy, and choose a recovery mode accordingly:



- Backup Under the **Simple** Recovery Model
 - This recovery model supports both database backups and file backups, but does not support log backups.

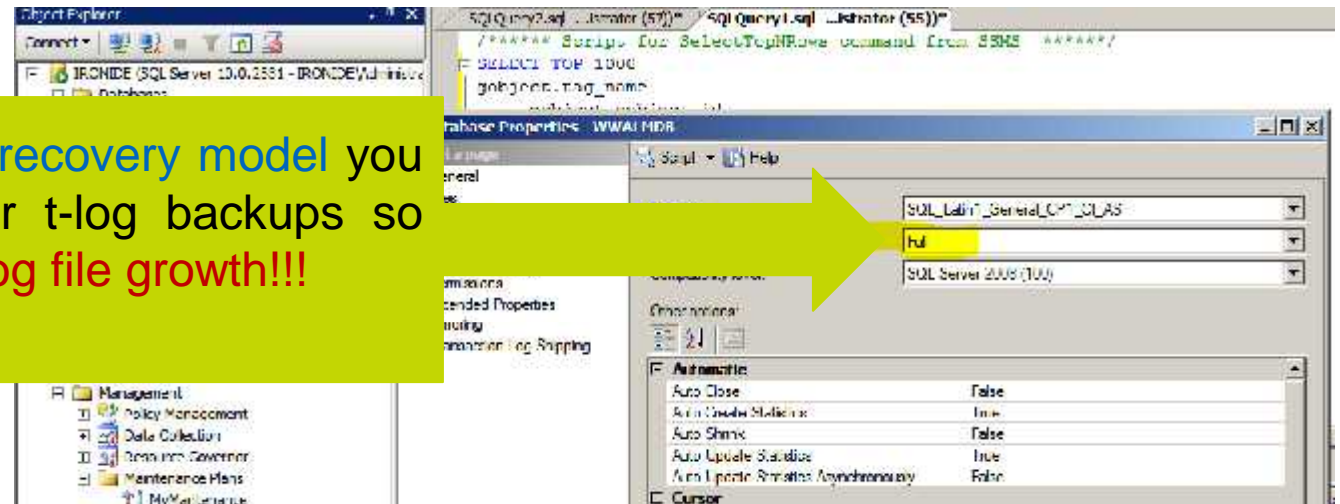
- Backup Under the **Full** Recovery Model
 - The full recovery model uses log backups to prevent data loss in the broadest range of failure scenarios, and backing and restoring the transaction log (log backups) is required.



Maintenance: Impact of recovery model

All the databases can have his own recovery model:

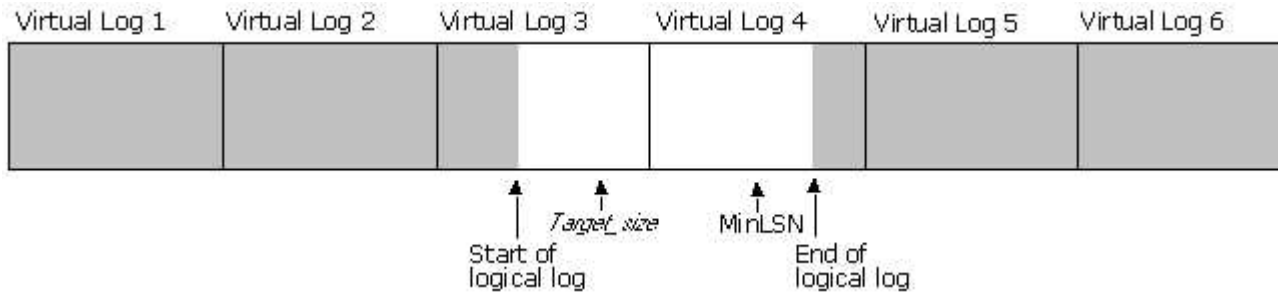
If you have **Full recovery model** you can take regular t-log backups so that means **the log file growth!!!**



Each Sql Statement can create a log lines, The following are the major causes for **transaction log growth**:

- Uncommitted transactions
- Rebuild/Create Index
- Run extremely large transaction like Bulk Insert
- Run Select INTO
- More information about the causes on <http://support.microsoft.com/kb/317375/>

Maintenance: Transaction Log

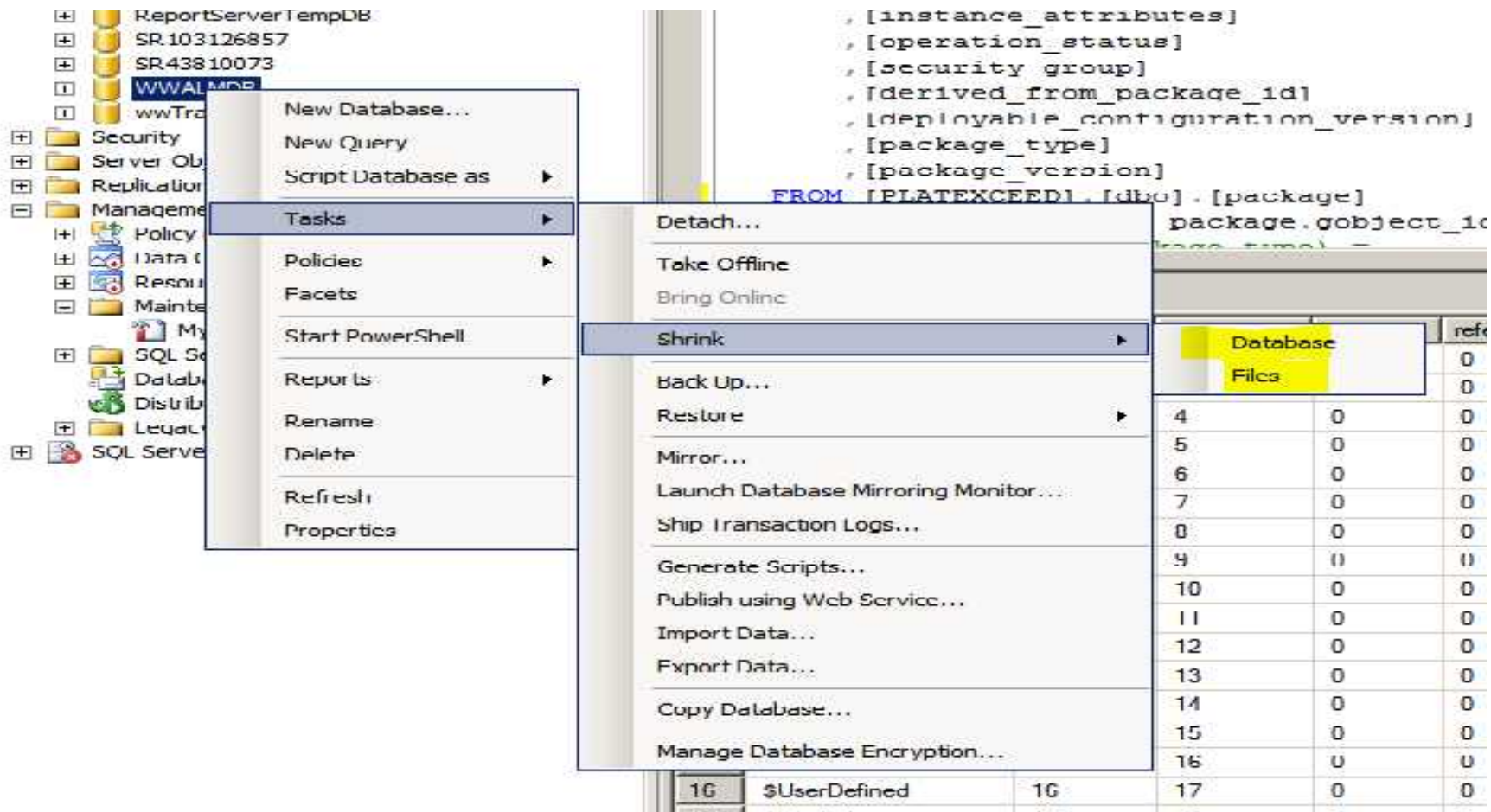


You could define some **rules** for Transaction growth:

The screenshot shows the SQL Server Enterprise Manager interface. A yellow starburst overlay contains the text: "You can have a Log Shrink, but... Perform a regular t-log backup would be better then run periodically the shrink command!". In the background, a dialog box for "Log Shrink" is visible, showing options for "Shrink to" (110 MB) and "Restrict File Growth" (2,000,102 MB).

Maintenance: Transaction Log

One way to shrink immediately would be using the [user interface](#):



Maintenance: Transaction Log

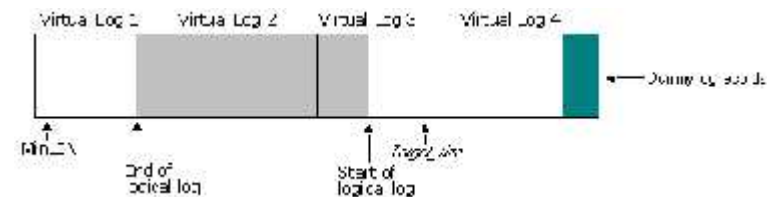
```
declare @DBNAME nvarchar(max)
set @DBNAME= 'GALAXY DB NAME' /* Set the correct galaxy name here*/
-----
-- Author: Iodice Pierluigi
-- Create date: 4/7/2013
-- Description: Clean Galaxy Database

declare @Rtp nvarchar(max)
select @Rtp = recovery_model_desc FROM sys.databases WHERE name = @DBNAME ;
if (@Rtp is not null)
begin
declare @LOGNAME varchar(100)
declare @sql nvarchar(max)
set @sql = 'ALTER DATABASE '+ @DBNAME + ' SET RECOVERY SIMPLE'
if (upper(@Rtp) <> 'SIMPLE')
exec (@sql)
if (coalesce(object_id(N'tempdb..##tt'),0)>0)
drop table ##tt;
set @sql = 'select name into ##tt from sys.database_files where type =0'
exec sp_executesql @sql
declare users_cursor CURSOR FOR select name from ##tt
OPEN users_cursor
FETCH NEXT FROM users_cursor
INTO @LOGNAME
WHILE @@FETCH_STATUS = 0
BEGIN
--Print @LOGNAME
DBCC SHRINKFILE (@LOGNAME , 1)
FETCH NEXT FROM users_cursor --have to fetch again within loop
INTO @LOGNAME
END
CLOSE users_cursor
DEALLOCATE users_cursor

set @sql = 'DBCC SHRINKDATABASE (''' + @DBNAME + ''', TRUNCATEONLY)'
exec (@sql)
set @sql = 'ALTER DATABASE '+ @DBNAME + ' SET RECOVERY FULL;'
if (upper(@Rtp) <> 'SIMPLE')
exec (@sql)

end
```

This an example of truncate log and shrink database using a little SQL Script



More information on [TN 599](#) or [837](#) on [WDN Site](#)

Maintenance: DB Fragmentation

Fragmentation occurs when data is modified in a table. When you insert or update data in a table (via INSERT or UPDATE), the table's corresponding [indexes are affected](#).



- The amount of fragmentation can be analyzed by using the `sys.dm_db_index_physical_stats` function.
- Fragmentation can be reduced by rebuilding and/or reorganizing indexes.
- The DB fill factor can help reduce fragmentation.
- Physical disk fragmentation can also help.

Maintenance: DB Fragmentation

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The query window contains the following SQL query:

```
SELECT * FROM sys.dm_db_index_physical_stats
(DB_ID(N'WVAALMDB'), OBJECT_ID(N'AlarmMaster'), NULL, NULL, 'DETAILED');
```

The results grid displays the following data:

id	obj_id	ind_id	ind_n	index_type	file	file_id	ind_id	avg_fragmentation_in_percent	avg_fragmentation_in_bytes	avg_page_space_in_use	avg_page_space_used_in_bytes	scan_count	avg_fragmentation_in_percent	avg_page_space_in_use	avg_page_space_used_in_bytes	scan_count	
1	10	101575400	1	CLUSTP	IN	3	0	95.7296845974349	1088000000000	78.3588459100888	1088000000000000	1	1	1	78.3588459100888	1088000000000000	1
2	10	101575400	1	CLUSTP	IN	3	1	5.4487794571785	5117	56.437270153373	5117000000000000	3	1	317	56.437270153373	5117000000000000	3
3	10	101575400	1	CLUSTP	IN	3	2	0	0	50.086400015177	5008640000000000	1	1	1	50.086400015177	5008640000000000	1
4	10	101575400	2	NONC II	IN	4	0	95.4845083361431	1088000000000	72.1033650036204	1088000000000000	1	1	1	72.1033650036204	1088000000000000	1
5	10	101575400	2	NONC II	IN	4	1	99.4475126121547	1088000000000	55.89465701627	1088000000000000	1	1	151	55.89465701627	1088000000000000	1
6	10	101575400	2	NONC II	IN	4	2	100	1088000000000	50.2903555023629	1088000000000000	2	1	2	50.2903555023629	1088000000000000	2
7	10	101575400	2	NONC II	IN	4	3	0	0	1.08729510501806	1088000000000000	1	1	1	1.08729510501806	1088000000000000	1
8	10	101575400	5	NONC II	IN	3	0	99.9610136450242	1088000000000	98.47355557188	1088000000000000	5	1	5	98.47355557188	1088000000000000	5
9	10	101575400	3	NONC II	IN	3	1	99.1538461538462	1088000000000	51.1670600125003	1088000000000000	2	104	25	51.1670600125003	1088000000000000	2
10	10	101575400	3	NONC II	IN	3	2	0	0	6.7210000181747	6721000000000000	1	1	1	6.7210000181747	6721000000000000	1
11	10	101575400	4	NONC II	IN	3	0	99.9512055709112	1088000000000	98.483659152459	1088000000000000	5	1	5	98.483659152459	1088000000000000	5
12	10	101575400	4	NONC II	IN	3	1	99.5714285714286	1088000000000	52.080266173473	1088000000000000	3	1	35	52.080266173473	1088000000000000	3
13	10	101575400	4	NONC II	IN	3	2	0	0	10.316207769138	1031620776913800	1	1	1	10.316207769138	1031620776913800	1

TIP

- DBCC INDEXDEFRAG
- ALTER INDEX REORGANIZE
- DBCC DBREINDEX
- ALTER INDEX REBUILD

Maintenance: clean Alarm DB



Helpful
Tips

```
-- Author: Iodice Pierluigi
-- Create date: 4/2/2013
-- Description: Defragmentation of Whole Database Tables

--SET NOCOUNT ON;
DECLARE @tablename varchar(255), @objectid int, @indexid int, @indexname varchar(400), @frag decimal
DECLARE @execstr nvarchar(max)

DECLARE indexes CURSOR FOR
select OBJECT_NAME (sys.dm_db_index_physical_stats.object_id) ObjectName, sys.dm_db_index_physical_stats.object_id Objectid,
sys.indexes.index_id IndexId, sys.indexes.name IndexName, avg_fragmentation_in_percent LogicalFrag
from sys.dm_db_index_physical_stats (DB_ID('MWAQMDS'), null, null, null, 'DETAILED')
inner join sys.indexes on sys.indexes.object_id = sys.dm_db_index_physical_stats.object_id
and sys.indexes.index_id = sys.dm_db_index_physical_stats.index_id
where INDEXPROPERTY (sys.dm_db_index_physical_stats.object_id, sys.indexes.name, 'IndexDepth') > 0
and avg_fragmentation_in_percent > 30 --means 30%

-- Open the cursor.
OPEN indexes;
-- Loop through the indexes.
FETCH NEXT FROM indexes INTO @tablename, @objectid, @indexid, @indexname, @frag;
WHILE @@FETCH_STATUS = 0
BEGIN
PRINT 'Executing DBCC INDEXDEFRAG (' + RTRIM(@tablename) + ', '
+ RTRIM(@indexid) + ') - fragmentation currently ' + RTRIM(CONVERT(varchar(15), @frag)) + '%';
SELECT @execstr = 'DBCC INDEXDEFRAG (' + RTRIM(@objectid) + ', ' + RTRIM(@indexid) + ')';
EXEC (@execstr);
FETCH NEXT FROM indexes INTO @tablename, @objectid, @indexid, @indexname, @frag;
END;
-- Close and deallocate the cursor.
CLOSE indexes;
DEALLOCATE indexes;
```

Maintenance: Galaxy Repository DB

```
#!/bin/bash
# Tests for validity of the database as a whole

returns two rows
- A single statement -- Removes all packages that are no longer referred to
- a table with -- directly or indirectly by any gobject
exec internal ...

--This would be used also during the validation of the object after importing
exec internal_delete_old_packages_after_checkin ...

all_finished OUTPUT
```

	allfinished	totalprocessedpackages
1	1	0

id



Maintenance: Runtime DB

- The Runtime DB size is not affected by the amount of history data.

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the 'Runtime' database under the 'Databases' folder. The 'Views' folder is expanded, showing several views including 'dbo.AnalogHistory'. A red arrow points from 'dbo.AnalogHistory' in the Object Explorer to the corresponding view definition in the SQL query window. The query window shows the following SQL code:

```
USE [Runtime]
GO

/***** Object: View [dbo].[AnalogHistory]    Script Date: 10.
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

create view [dbo].[AnalogHistory]
as
select * from [INSQL].[Runtime].dbo.AnalogHistory;
GO
```

The 'select * from [INSQL].[Runtime].dbo.AnalogHistory;' line is highlighted in yellow. The 'Connection' pane at the bottom shows the server name 'RACHFT' and the connection name 'RACHFT\Administrator'.

Maintenance: End! And now?

All the questions and doubts can be later treated in Hands On,



Troubleshooting and Diagnostic Tools

There are [several tools](#) that will allow you to:

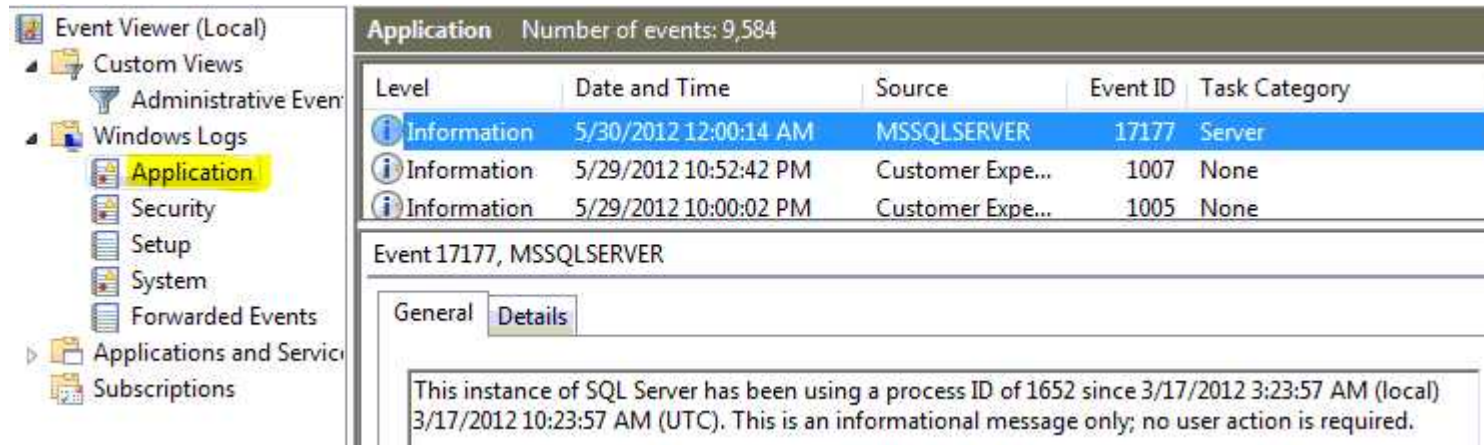
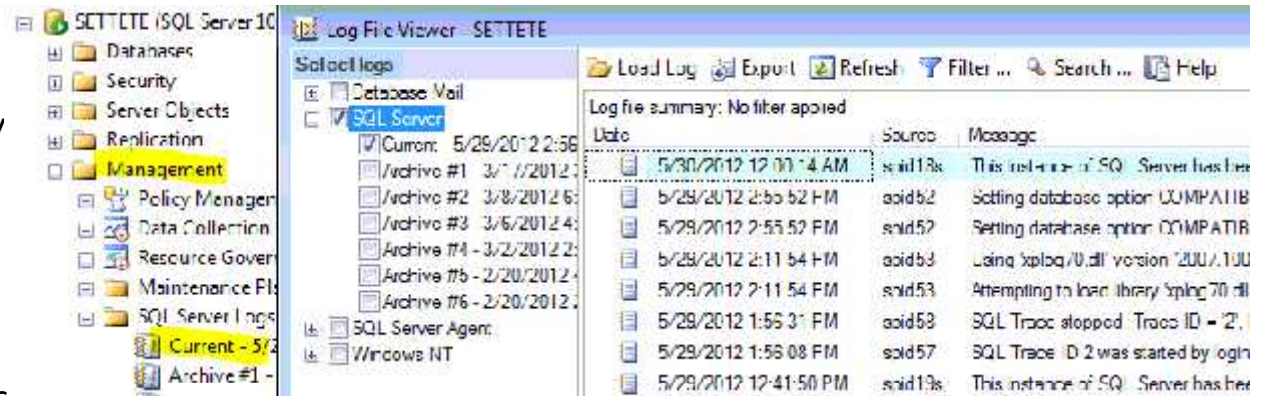
1. Detect errors and exceptions
2. Monitor the SQL Server
3. Monitor the performance with counters
4. Log the SQL statements executed
5. Know system function / stored procedure
6. Understand SQL in ArcestraA Script



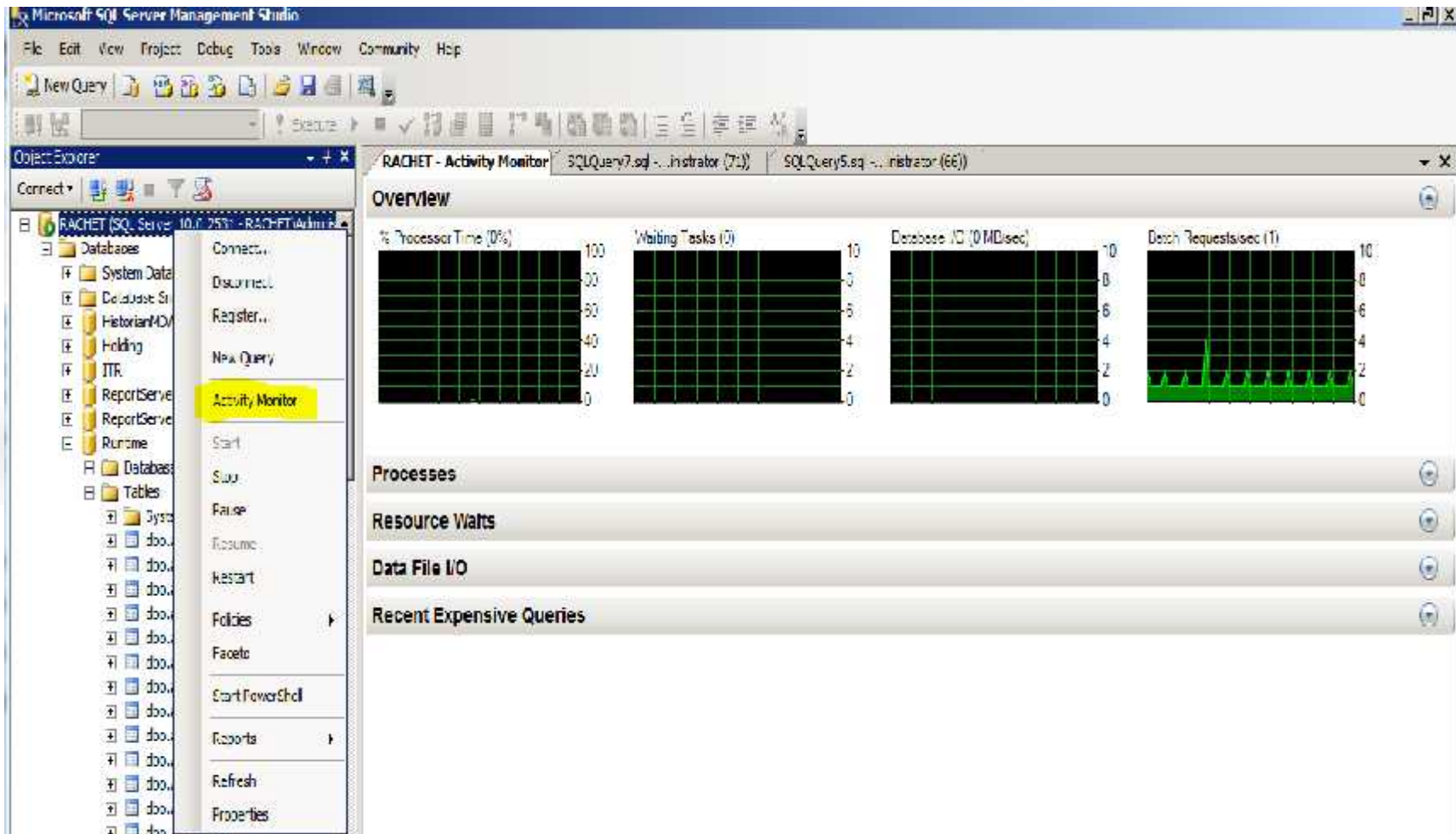
Detect errors: Logging Mechanisms

Useful logs to keep in mind:

- SQL Server Log
 - Export .log, .txt, .csv
 - Filter condition
 - Search
- Windows Event Viewer



Monitor SQL: Activity Monitor



Monitor SQL: Activity Monitor

Processor Time %: What could be helpful to know

The screenshot displays the SQL Server Activity Monitor interface. The main window is titled 'Processes' and shows a table with columns: Sess ID, User Proc, Login, Database, Task State, Command, Applicatio, Wait Time (ms), Wait type, Wait Resource, Eloc Ey, Heap Eloc, Memory Use (KB), Host Name, and Workload Group. The first row shows a process with Sess ID 72, User Proc RACHETVA..., Login ITR, Database Microsoft S..., Task State 0, Command Microsoft S..., Applicatio Microsoft S..., Wait Time (ms) 0, Wait type, Wait Resource, Eloc Ey 1, Heap Eloc, Memory Use (KB) 16, Host Name RACHET, and Workload Group interna.

A 'Session Details' window is open over the first row, titled 'Session Details - RACHET\Administrator - 72 - RACHET'. It shows the 'Last Transact-SQL command batch' with the following SQL commands:

```
BEGIN TRANSACTION  
insert into LIVFRK  
  
SELECT *  
FROM LIVFRK
```

A red question mark is drawn over the SQL commands. A yellow arrow points from the 'Kill Process' button at the bottom of the 'Session Details' window to the text box below.

There was a transaction in progress that **has not committed yet**, and is still causing a **table lock**

Monitor SQL: Activity Monitor

Resource Wait: Nice to know all the operations that were consuming resources

Wait Category	Wait Time (ms/sec)	Recent Wait Time (ms/sec)	Average Wait Count	Cumulative Wait Time (sec)
Lock	0	17	17	430
Memory	0	3	3	227
Compilation	0	2	2	64
Logging	2	1	1	23
Network I/O	0	0	0	8051
Other	0	0	0	10
Latch	0	0	0	0
Buffer I/O	0	0	0	329
Buffer Latch	0	0	0	0

Data file I/O: Nice to know all the Database files that were involved in R/W ops.

Database	File Name	MB/sec Read	MB/sec Written	Response Time (ms)
HistorianMDASBuffer	C:\Program Files (x86)\Microsoft SQL Server\W...	0.0	0.0	0
HistorianMDASBuffer	C:\Program Files (x86)\Microsoft SQL Server\W...	0.0	0.0	0
Holding	C:\Program Files (x86)\Microsoft SQL Server\W...	0.0	0.0	0
Holding	C:\Program Files (x86)\Microsoft SQL Server\W...	0.0	0.0	0
ITR	C:\Program Files (x86)\Microsoft SQL Server\W...	0.0	0.0	0
ITR	C:\Program Files (x86)\Microsoft SQL Server\W...	0.0	0.0	0
master	C:\Program Files (x86)\Microsoft SQL Server\W...	0.0	0.0	0
master	C:\Program Files (x86)\Microsoft SQL Server\W...	0.0	0.0	0
model	C:\Program Files (x86)\Microsoft SQL Server\W...	0.0	0.0	0
model	C:\Program Files (x86)\Microsoft SQL Server\W...	0.0	0.0	0

Monitor SQL: Activity Monitor

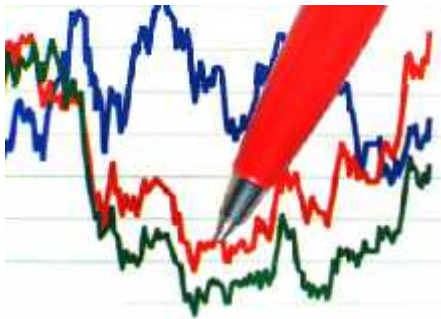
Recent Expensive Queries: Getting a look around

The screenshot displays the 'Recent Expensive Queries' window in SQL Server Enterprise Manager. The window title is 'Recent Expensive Queries'. The main table lists queries with columns for Query, Executions/mir, CPU (ms/sec), Physical Reads/sec, Logical Writes/sec, Logical Reads/sec, Average Duration (ms), Plan Count, and Database Name. The first query is highlighted, and a context menu is open over it, showing 'Edit Query Text' and 'Show Execution Plan' options. A yellow callout box with an arrow points to the 'Show Execution Plan' option, containing the text: 'This would be useful to understand the customization, in order to improve eventually'.

Query	Executions/mir	CPU (ms/sec)	Physical Reads/sec	Logical Writes/sec	Logical Reads/sec	Average Duration (ms)	Plan Count	Database Name
insert into dbc.AnalogSnapshot Snapshot..	0	0	0	0	0	27400	1	Runtime
select count(1) as S...	18					1	0	Runtime
select count(1) from ConfigSta	6					0	0	Runtime
UPDATE [Notifications] WITH	6					0	0	ReportServer
select count(1) from ConfigStatusSnapshot	6					0	0	ReportServer
UPDATE [Event] WITH (TABLOCKX)	6					0	0	ReportServer
select top 2						0	0	Runtime
select top 2						0	0	Runtime
select top 1 Vers						0	0	Runtime
select e.TanName e.ScanRate e.UseThread	3					0	0	Runtime

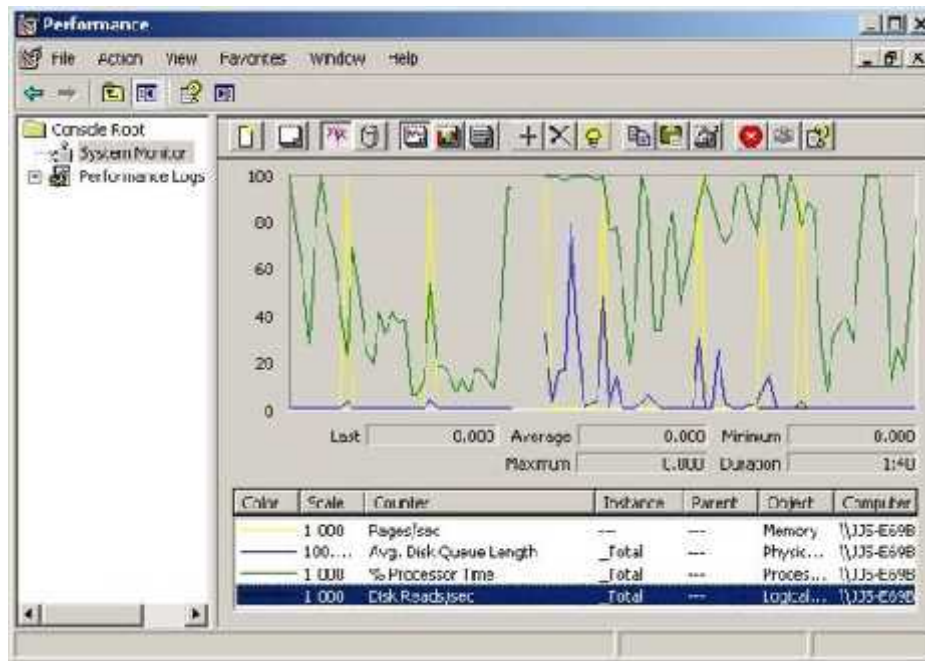
Performance Monitor

You cannot control what you don't measure



- MS SQL Server exposes a set of [performance counters](#) for virtually every subsystem
- These counters allow you to create a [performance baseline](#)

Performance Monitor



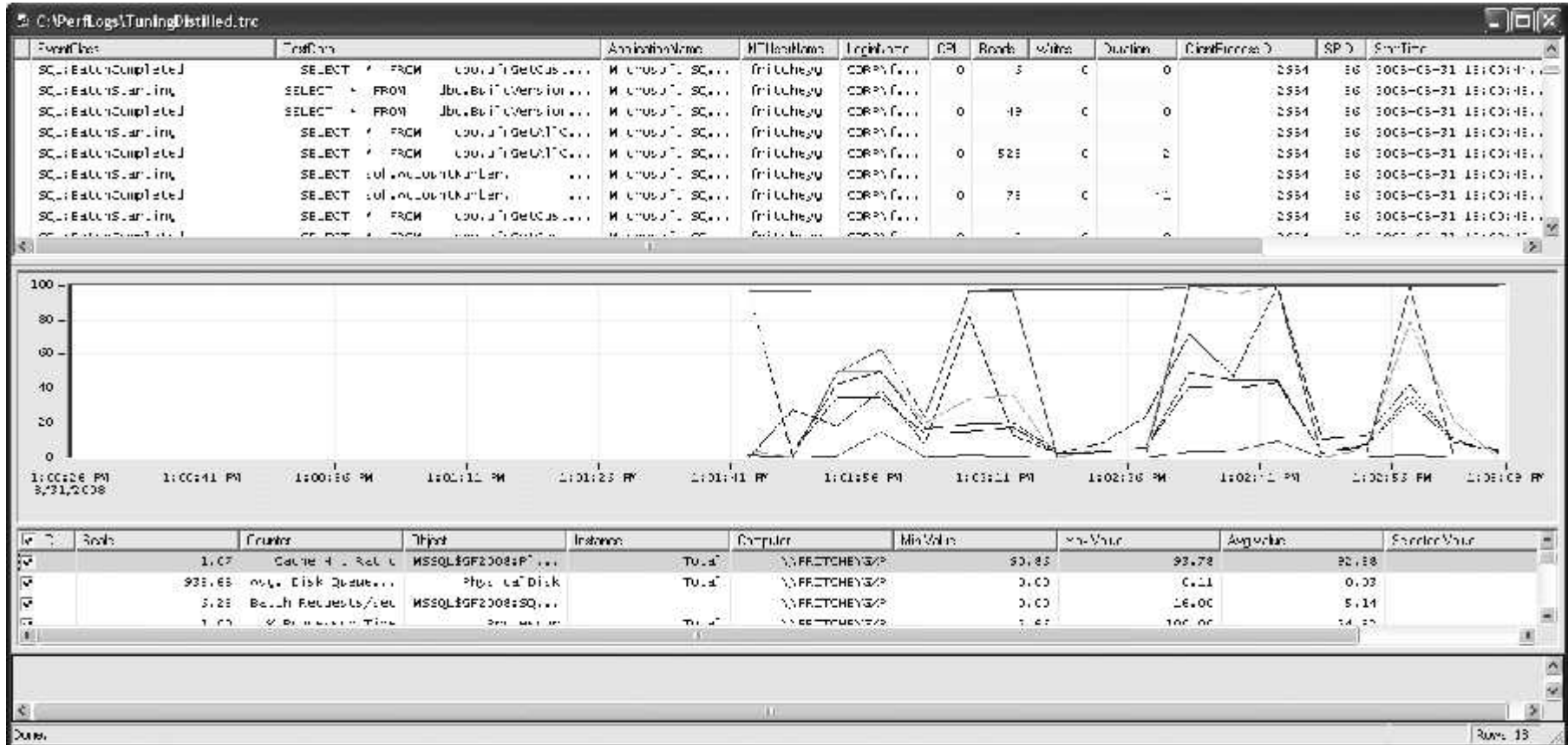
Collects detailed information about the utilization of operating system resources. SQL Server provides extensions to the Performance Monitor tool to track a variety performance counters.

- It allows you to track memory, disk, processor, and the network performance.
- Allows you to track both system-wide and SQL Server counters.
- Tracing can occur in real-time or captured as a log.

Performance Monitor: SQL Pen

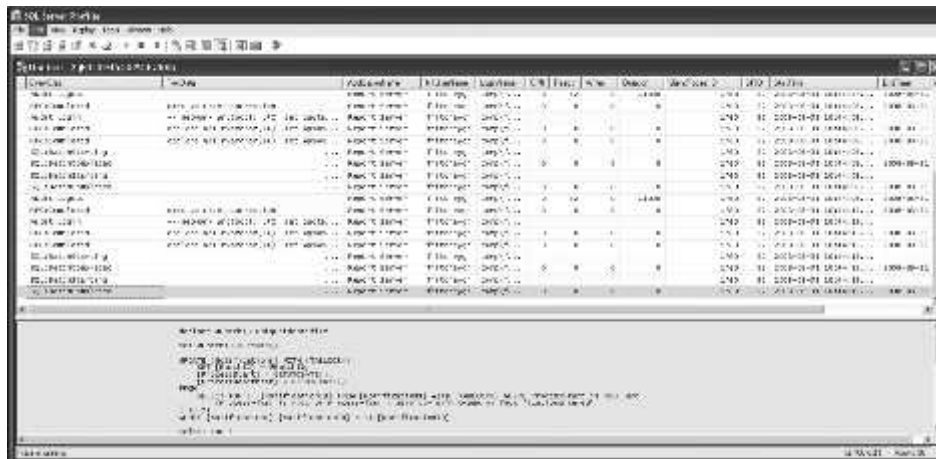
Object(Instance)	Counter	Description	Target Value
SQLServer:Access Methods	FreeSpace Scans/sec	Rate of inserts into tables with no indexes	No target. Should be monitored over time.
SQLServer:Access Methods	Full Scans/sec	Rate of unrestricted full scans on tables indexes	No target. Should be monitored over time.
SQLServer:Latches	Total Latch Wait Time	Wait time before latch requests are acquired	No target. Should be monitored over time.
SQLServer:Locks	Lock Timeouts/sec	Amount of locks that timeout and exit	Avg = 0
SQLServer:Locks	Lock Wait Time	Wait time before a lock can be acquired	Avg < 10 ms
SQLServer:Locks	Number of Deadlocks/sec	Amount of deadlocks	Avg = 0
SQLServer:General Statistics	Processes Blocked	Amount of processes that are denied connection to the DB	Avg = 0
SQLServer:General Statistics	User Connections	Total number of user connections	No target. Should be monitored over time.

Performance Monitor and Profiler Trace



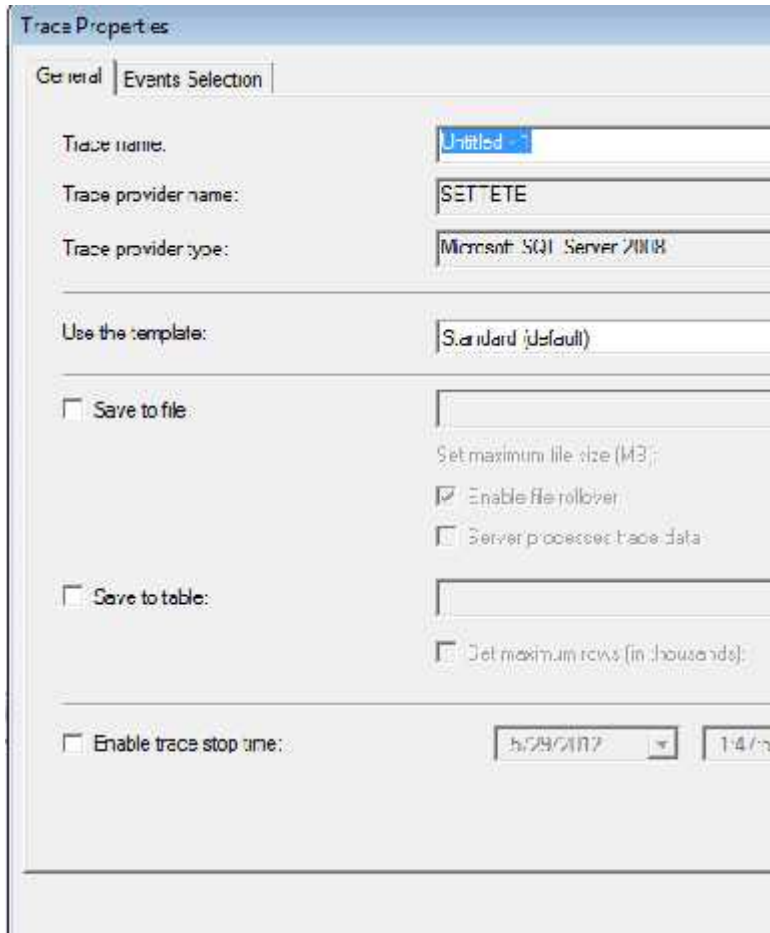
SQL Server Profiler

Is a rich interface to **create** and **manage** traces and **analyze** and **replay** trace results.



- Trace each query into SQL Server DB
- Analyze performance and diagnose problems.
- Debug a T-SQL statements and Stored Procedures.
- Replay SQL Server activity in a simulation.
- Combine with other debug instruments

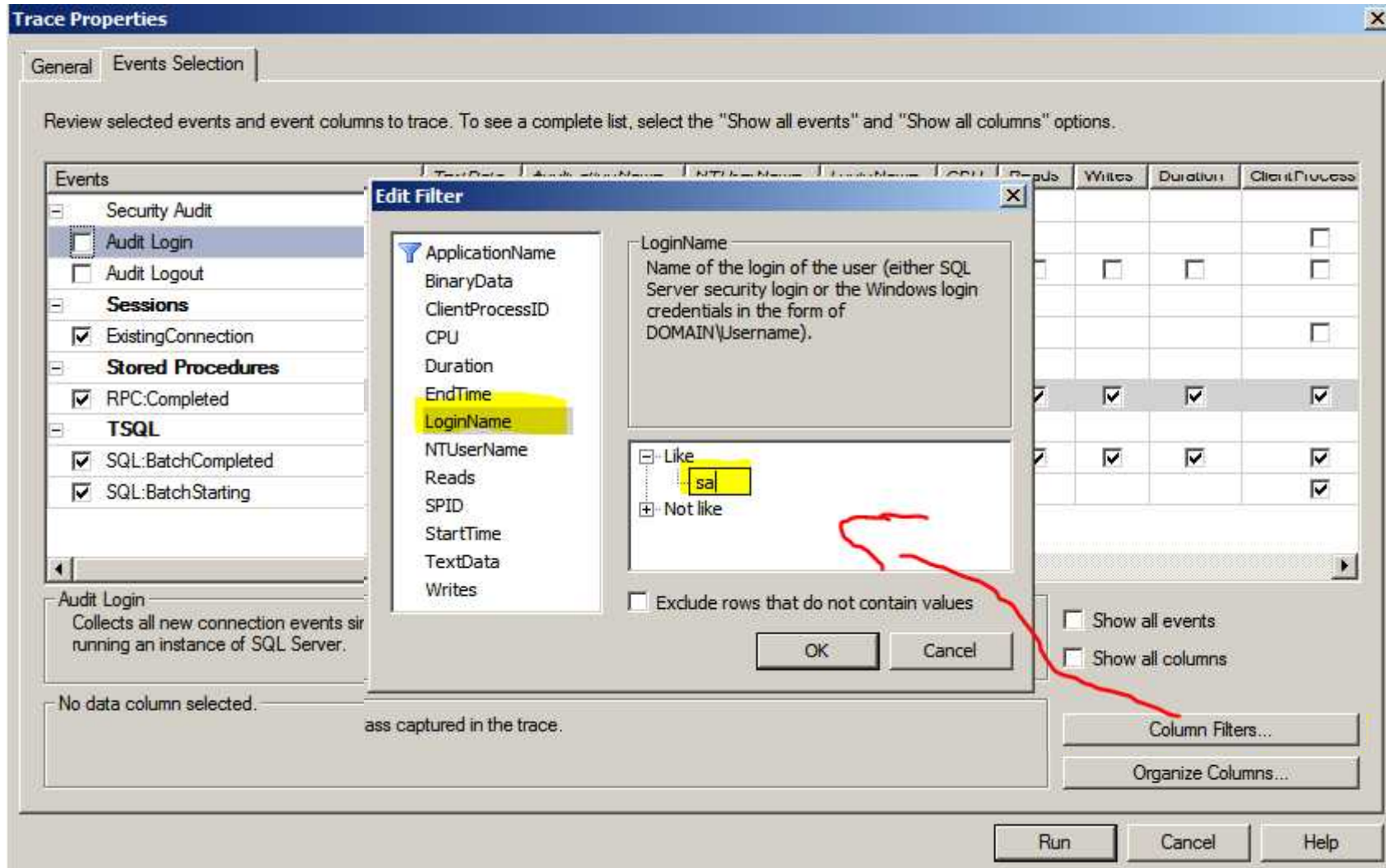
SQL Server Profiler: How to Run



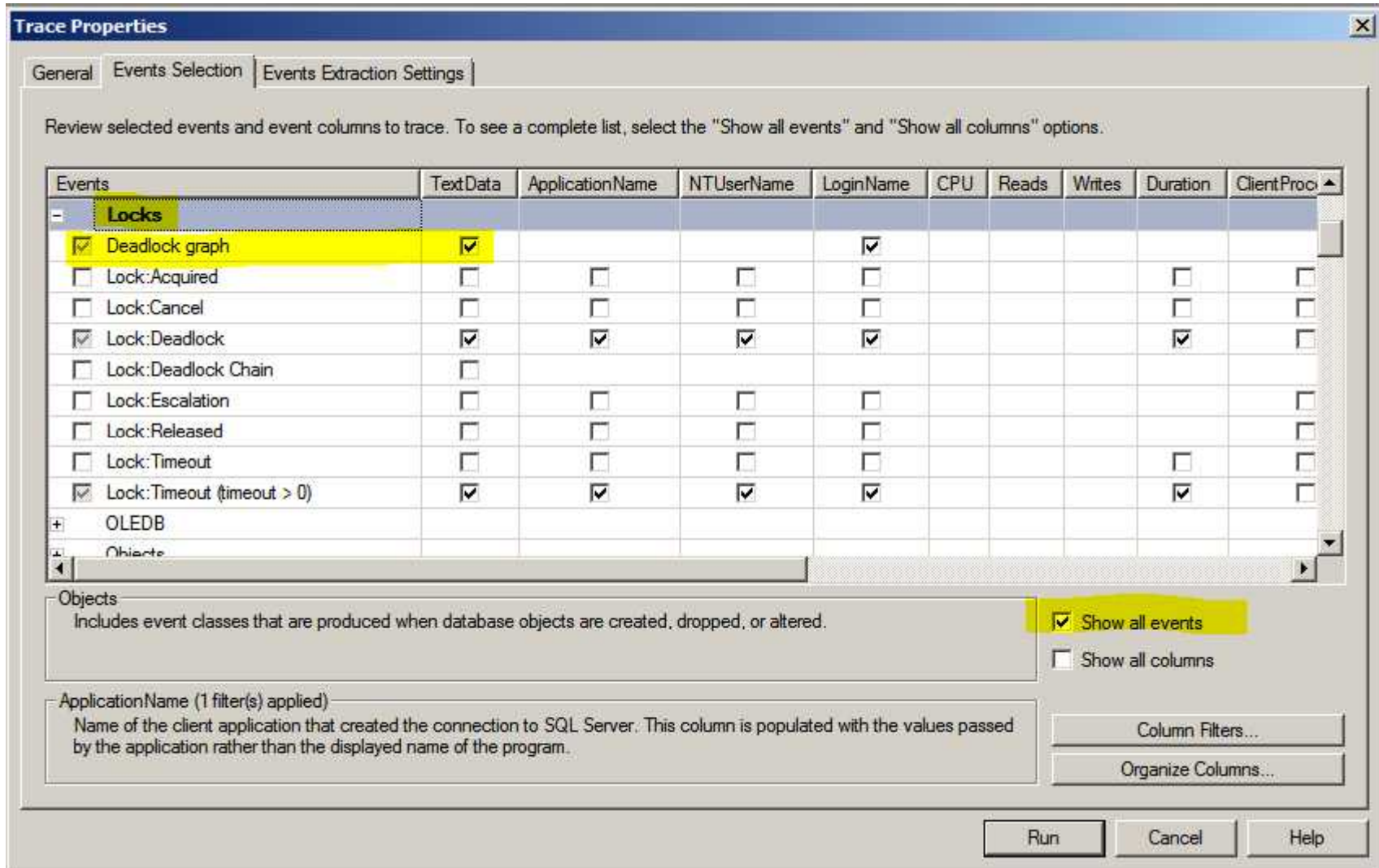
General Setting

- Trace Name
- Trace Provider name
- Trace Provider Type
- Use template:
 - ✓ Standard
 - ✓ TSQL
 - ✓ TSQL Duration
 - ✓ TSQL Lock
 - ✓ And so on...
- **Save to file** {.trc, xml, ...}
- Save to table
- Enable trace stop time

SQL Server Profiler: How to Run



SQL Server Profiler: How to Run



SQL Server Profiler: How to Run

General Events Selection

Review selected events and event columns to trace. To see a complete list, select the "Show all events" and "Show all columns" options.

Events	ClientProcess
<input type="checkbox"/> Security Audit	<input type="checkbox"/>
<input type="checkbox"/> Audit Login	<input type="checkbox"/>
<input type="checkbox"/> Audit Logout	<input type="checkbox"/>
<input type="checkbox"/> Sessions	<input type="checkbox"/>
<input type="checkbox"/> ExistingConnection	<input type="checkbox"/>
Stored Procedures	
<input checked="" type="checkbox"/> RPC:Completed	<input checked="" type="checkbox"/>
TSQL	
<input checked="" type="checkbox"/> SQL:BatchCompleted	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> SQL:BatchStarting	<input checked="" type="checkbox"/>

SQL:BatchStarting
Occurs when a Transact-SQL batch is starting.

ApplicationName (1 filter(s) applied)
Name of the client application that created the connection to SQL Server. This column is populated with the values passed by the application rather than the displayed name of the program.

Show all events
 Show all columns

Column Filters...
Organize Columns...

Run Cancel Help

This would be a standard way to catch only the SQL query and Stored procedure will be executed

SQL Server Profiler: the Results

SQL Server Profiler - [C:\Users\swadmin\Desktop\CPU-100%SQL TRACE CPU 100%(Trace 06-03-2012.trc)]

File Edit View Replay Tools Window Help

EventClass	TaskData	ApplicationName	N UserNa...	LoginName	StartTime	EndTime	UserP...	SPIU	CP...
Trace Start					2012-03-06 09:28:18.780				
RPC:Completed	exec sp_reset_conne...	.Net SqlClient Data Pro...	SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:19.007	2012-03-06 09:28:19.007	1888	92	
RPC:Completed	declare @p1 bigintNet SqlClient Data Pro...	SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:19.007	2012-03-06 09:28:19.007	1888	92	
RPC:Completed	exec sp_reset_conne...	.Net SqlClient Data Pro...	SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:20.020	2012-03-06 09:28:20.020	1888	92	
RPC:Completed	declare @p1 bigintNet SqlClient Data Pro...	SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:20.020	2012-03-06 09:28:20.020	1888	92	
SQL:BatchStarting	select ...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:20.490		8656	73	
SQL:BatchCompleted	select ...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:20.490	2012-03-06 09:28:20.490	8656	73	
RPC:Completed	exec sp_reset_conne...	.Net SqlClient Data Pro...	SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.033	2012-03-06 09:28:21.033	1888	92	
RPC:Completed	declare @p1 bigintNet SqlClient Data Pro...	SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.033	2012-03-06 09:28:21.033	1888	92	
RPC:Completed	exec sp_U_Audit_Tra...	.Net SqlClient Data Pro...	sa\SES	sa\SES	2012-03-06 09:28:21.257	2012-03-06 09:28:21.257	5172	60	
RPC:Completed	exec SP_SA_UTI_L_EXE...	.Net SqlClient Data Pro...	sa\SES	sa\SES	2012-03-06 09:28:21.257	2012-03-06 09:28:21.257	5172	60	
RPC:Completed	exec sp_reset_conne...	.Net SqlClient Data Pro...	sa\SES	sa\SES	2012-03-06 09:28:21.257	2012-03-06 09:28:21.257	5172	60	
SQL:BatchStarting	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.330		2208	63	
SQL:BatchCompleted	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.330	2012-03-06 09:28:21.330	2208	63	
SQL:BatchStarting	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.330		2208	63	
SQL:BatchCompleted	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.330	2012-03-06 09:28:21.330	2208	63	
SQL:BatchStarting	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.333		2208	63	
SQL:BatchCompleted	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.333	2012-03-06 09:28:21.333	2208	63	
SQL:BatchStarting	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.333		2208	63	
SQL:BatchCompleted	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.333	2012-03-06 09:28:21.333	2208	63	
SQL:BatchStarting	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.337		2208	63	
SQL:BatchCompleted	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.337	2012-03-06 09:28:21.337	2208	63	
SQL:BatchStarting	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.337		2208	63	
SQL:BatchCompleted	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.337	2012-03-06 09:28:21.337	2208	63	
SQL:BatchStarting	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.340		2208	63	
SQL:BatchCompleted	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.340	2012-03-06 09:28:21.340	2208	63	
SQL:BatchStarting	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.340		2208	63	
SQL:BatchCompleted	exec aaInternalTsqD...		SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:21.340	2012-03-06 09:28:21.340	2208	63	

exec sp_U_Audit_Tra..._SetContextInfo @session_id=-1, @creation_time=N'Client: Standard; Time...

SQL Server Profiler: Debug and Execute

The image displays two screenshots from Microsoft SQL Server. The top screenshot is a SQL Server Profiler trace showing various events such as SQLBatchStarting, SQLBatchCompleted, and RPC:Completed. The bottom screenshot is a screenshot of Microsoft SQL Server Enterprise Manager (SSMS) showing a query window with the following SQL code:

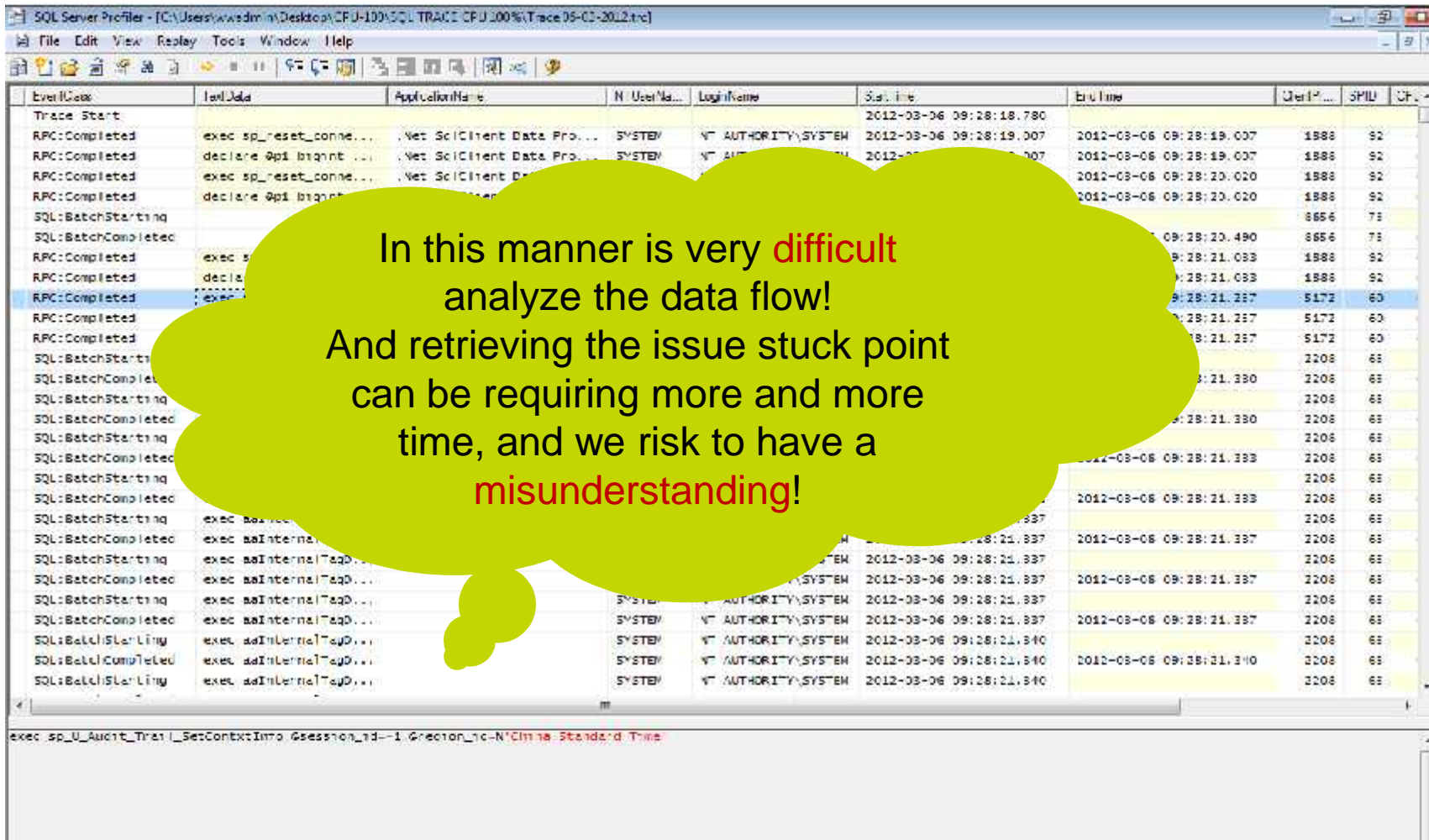
```
select count(1) as SyncRequestsCount
from dbo.ReplicationSyncRequest rsr
inner join dbo.ReplicationTagEntity rte
on rsr.ReplicationTagEntityKey =
where rte.ReplicationServerKey = 1
```

SQL Server Profiler: Lock Result

EventClass	TextData	BatchID	DatabaseID	TransactionID	SPID	HostName	ClientProcessID	ApplicationName	LoginName	SPID	Duration
SQL:BatchStarting	select * from sysobjects		9	274297996		ESTME...	2172	IntTouch	sa	74	
SQL:BatchCompleted	select * from sysobjects		9	274297996		ESTME...	2172	IntTouch	sa	74	
SQL:BatchStarting	insert into Comment (OperatorNode,D...		9	274297996		ESTME...	2172	IntTouch	sa	74	
SQL:BatchCompleted	insert into Comment (OperatorNode,D...		9	274297996		ESTME...	2172	IntTouch	sa	74	
SQL:BatchStarting	select * from Comment where Operat...		9	274297996		ESTME...	2172	IntTouch	sa	74	
SQL:BatchCompleted	select * from Comment where Operat...		9	274297996		ESTME...	2172	IntTouch	sa	74	
SQL:BatchStarting	update AlarmConsolidated set Return...		9	274297996		ESTME...	2172	IntTouch	sa	74	
Lock:Dead lock	{6900cb61a6a}		9	274297922		ESTME...	26676	IntTouch	sa	63	
Deadlock graph	<deadlock-list> <deadlock victim>...								sa	25	
SQL:BatchCompleted	select * from Comment where Comment...		9	274297922		ESTME...	26676	IntTouch	sa	63	
SQL:BatchCompleted	update AlarmConsolidated set Return...		9	274297996		ESTME...	2172	IntTouch	sa	74	
SQL:BatchStarting	SET TRANSACTION ISOLATION LEVEL REA...		9			ESTME...	26676	IntTouch	sa	63	
SQL:BatchCompleted	SET TRANSACTION ISOLATION LEVEL REA...		9			ESTME...	26676	IntTouch	sa	63	
SQL:BatchStarting	IF @@TRANCOUNT > 0 COMMIT TRAN		9	274297996		ESTME...	2172	IntTouch	sa	74	
SQL:BatchCompleted	IF @@TRANCOUNT > 0 COMMIT TRAN		9	274297996		ESTME...	2172	IntTouch	sa	74	
SQL:BatchStarting	SET TRANSACTION ISOLATION LEVEL REA...		9			ESTME...	2172	IntTouch	sa	74	
SQL:BatchCompleted	SET TRANSACTION ISOLATION LEVEL REA...		9			ESTME...	2172	IntTouch	sa	74	
SQL:BatchStarting	SET TRANSACTION ISOLATION LEVEL REP...		9			ESTME...	2172	IntTouch	sa	74	

The diagram illustrates a deadlock scenario. On the left, a server process (ID 63) is shown with a crossed-out oval, indicating it is a victim. It has requested a key lock. In the center, a 'Key Lock' box shows it is held by a server process (ID 74) on the right. The lock is associated with object ID 72057594048800912 and index name 'P_Comment'. The requester (63) is in 'Request Mode: RangeS' and the owner (74) is in 'Owner Mode: X'.

SQL Server Profiler: wall of text



In this manner is very **difficult** analyze the data flow!
And retrieving the issue stuck point can be requiring more and more time, and we risk to have a **misunderstanding!**

EventClass	TaskData	ApplicationName	N UserNa...	LoginName	StartTime	EndTime	ClientP...	SPIU	CP...
Trace Start					2012-03-06 09:28:18.780				
RPC:Completed	exec sp_reset_conne...	Net SqlClient Data Pro...	SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:19.007	2012-03-06 09:28:19.007	1888	92	
RPC:Completed	declare @pi bigint ...	Net SqlClient Data Pro...	SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:19.007	2012-03-06 09:28:19.007	1888	92	
RPC:Completed	exec sp_reset_conne...	Net SqlClient Data Pro...	SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:20.020	2012-03-06 09:28:20.020	1888	92	
RPC:Completed	declare @pi bigint ...	Net SqlClient Data Pro...	SYSTEM	NT AUTHORITY\SYSTEM	2012-03-06 09:28:20.020	2012-03-06 09:28:20.020	1888	92	
SQL:BatchStarting							8656	73	
SQL:BatchCompleted							8656	73	
RPC:Completed	exec s...				2012-03-06 09:28:21.033	2012-03-06 09:28:21.033	1888	92	
RPC:Completed	decla...				2012-03-06 09:28:21.033	2012-03-06 09:28:21.033	1888	92	
RPC:Completed	exec...				2012-03-06 09:28:21.217	2012-03-06 09:28:21.217	5172	60	
RPC:Completed					2012-03-06 09:28:21.217	2012-03-06 09:28:21.217	5172	60	
RPC:Completed					2012-03-06 09:28:21.217	2012-03-06 09:28:21.217	5172	60	
SQL:BatchStarting							2208	63	
SQL:BatchCompleted							2208	63	
SQL:BatchStarting							2208	63	
SQL:BatchCompleted							2208	63	
SQL:BatchStarting							2208	63	
SQL:BatchCompleted							2208	63	
SQL:BatchStarting							2208	63	
SQL:BatchCompleted							2208	63	
SQL:BatchStarting							2208	63	
SQL:BatchCompleted							2208	63	
SQL:BatchStarting	exec aa...				2012-03-06 09:28:21.337	2012-03-06 09:28:21.337	2208	63	
SQL:BatchCompleted	exec aaInternal...				2012-03-06 09:28:21.337	2012-03-06 09:28:21.337	2208	63	
SQL:BatchStarting	exec aaInternalTsqD...				2012-03-06 09:28:21.337	2012-03-06 09:28:21.337	2208	63	
SQL:BatchCompleted	exec aaInternalTsqD...				2012-03-06 09:28:21.337	2012-03-06 09:28:21.337	2208	63	
SQL:BatchStarting	exec aaInternalTsqD...				2012-03-06 09:28:21.337	2012-03-06 09:28:21.337	2208	63	
SQL:BatchCompleted	exec aaInternalTsqD...				2012-03-06 09:28:21.337	2012-03-06 09:28:21.337	2208	63	
SQL:BatchStarting	exec aaInternalTsqD...				2012-03-06 09:28:21.340	2012-03-06 09:28:21.340	2208	63	
SQL:BatchCompleted	exec aaInternalTsqD...				2012-03-06 09:28:21.340	2012-03-06 09:28:21.340	2208	63	
SQL:BatchStarting	exec aaInternalTsqD...				2012-03-06 09:28:21.340	2012-03-06 09:28:21.340	2208	63	

SQL Server Profiler: External free tool

The screenshot displays the SQL Server Profiler interface with the following sections:

- marzo Workload Summary (Show Details):** A summary view of the current workload.
- Top Consuming Applications:** A table listing applications and their resource usage.

Resource	Top Consumer	Event Count	Total	Average	More Information
CPU	Net SqlClient Data Provider	8139	19.14 sec	2 ms	Show Applications by CPU
Duration	Net SqlClient Data Provider	8139	20.12 sec	2.47 ms	Show Applications by Duration
Reads	Net SqlClient Data Provider	8139	5.13 M	755.07	Show Applications by Reads
Row Count	Net SqlClient Data Provider	8139	0	0	Show Applications by Row Count
				0.1	Show Applications by Writes
- Workload View:** A detailed view of the workload, showing a table with columns: User, Duration, CPU, Reads, Writes, Row Count. The 'CPU' column is highlighted in orange. Below this table, a bar chart shows resource consumption for various users and batch templates.
- Resource Consumption:** A table showing the overall resource usage for the workload.

Resource	Value	Percentage of Workload
Duration	4.84 sec	22.2% of Workload
CPU	4.84 sec	24.5% of Workload
Reads	2.81M	45.5% of Workload
Writes	248	27.1% of Workload
Row Count	0	
Event Count	9	0.1% of Workload

SQL Server: PID and SPID

- ✓ Client Process ID report exactly the **PID** of **Task Manager** or Process Explorer.
- ✓ It has used to **identify the Application** that still running something into SQL Server Database

ClientProces...	SPID
7388	51
5224	52
7528	53
7388	54
5224	55
1888	56
7388	57
1888	58
9664	59
5172	60
5172	61
1888	62
1000	63
5224	64
2208	65
2208	66
7718	67
2208	68
2208	69
7528	70
6636	71
6636	72
8524	74
8656	75
6980	76
6980	77
6980	78
2556	79

- ✓ A SPID in **SQL Server** is a Server **Process ID**. These process ID's are essentially sessions in SQL Server.
- ✓ Every time **an application connects** to SQL Server, **a new SPID is created**.
- ✓ This connection has a defined scope and memory space and cannot interact with other SPIDs.
- ✓ The term SPID is synonymous with Connection, or Session.

Diagnostic Query: From SPID to SQL

The screenshot shows a SQL Server Enterprise Manager window with several tabs. The active tab is titled "SQLQuery9.sql...istrator (59)*". The query editor contains the following text:

```
From SPID to Query  
--- To know which sessions are running currently, run the following command:  
SELECT @@SPID CurSession
```

Below the query editor, the "Results" pane shows a table with one row:

CurSession
59

The status bar at the bottom of the window indicates: "Query executed successfully. P12008R2 (10.0 SP1) FI2008R2\Administrator... W\WALMDB 00:00:00 1 rows".

TIP
Get Last
Running
Query Based
on SPID

Diagnostic Query: Concatenate SPID

```
SQLQuery1.sql -...istrator (52)* Object Explorer Details
SELECT p.spid
, convert(char(12), d.name) db_name
, program_name
, convert(char(12), l.name) login_name
, convert(char(12), hostname) hostname
, cmd
, p.status
, p.blocked
, login_time
, last_batch
, p.spid
FROM master..sysprocesses p
JOIN master..sysdatabases d ON p.dbid = d.dbid
JOIN master..syslogins l ON p.sid = l.sid
WHERE p.blocked = 0
AND EXISTS ( SELECT 1
FROM master..sysprocesses p2
WHERE p2.blocked = p.spid )
```

Results Messages

spid	db_name	program_name	login_name	hostname	cmd	status	blocked	login_time	last_batch	spid
------	---------	--------------	------------	----------	-----	--------	---------	------------	------------	------

TIP

See Who Is
[Blocking](#) Your
SQL Server

Diagnostic Query: Useful Query



- Execution related dynamic objects provide information about [current sessions](#), [connections](#), client [requests](#), [opened cursors](#) and [execution plans](#).
- These objects can be particularly helpful in [identifying](#) resource [bottlenecks](#) such as CPU, memory or disk. You can also peruse execution related objects to [troubleshoot blocking issues](#).
- Each object in this category is prefixed with "dm_exec".

TIP [Sys.dm_exec_query_stats](#)

provides a wealth of performance statistics for cached query plans.

Diagnostic Query: ...on CPU

The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query editor contains a T-SQL query to identify the top 10 most CPU-intensive SQL statements. The results pane shows a table with 10 rows, each representing a statement and its CPU time.

```
SELECT TOP 10
SUBSTRING(b.text, ((a.statement_start_offset/2) + 1),
((CASE statement_end_offset
WHEN -1 THEN DATALENGTH(b.text)
ELSE a.statement_end_offset END
- a.statement_start_offset)/2) + 1) AS statement_text,
c.query_plan,
total_worker_time AS CPU time
FROM sys.dm_exec_query_state a CROSS APPLY sys.dm_exec_sql_text (a.sql_handle) AS b
CROSS APPLY sys.dm_exec_query_plan (a.plan_handle) AS c
ORDER BY total_worker_time DESC
```

id	statement_text	query_plan	CPU_time
1	SELECT * FROM sys.dm_db_index_physical_state ...	<Show Plan XML xmlns="http://schemas.microsoft.com..."	26649525
2	select (@current_proxy_timestamp - max_proxy_f...	<Show Plan XML xmlns="http://schemas.microsoft.com..."	9090515
3	insert into @objobj WITH WARNING select distinct...	<Show Plan XML xmlns="http://schemas.microsoft.com..."	1411080
4	delete primitive_instance from primitive_instanc...	<Show Plan XML xmlns="http://schemas.microsoft.com..."	641006
5	delete pa from package pa inner join #check...	<Show Plan XML xmlns="http://schemas.microsoft.com..."	532050
6	delete visual_element_version from visual_element...	<Show Plan XML xmlns="http://schemas.microsoft.com..."	444025
7	update subject set subject_configuration_id	NULL	414023
8	update user_preferences set preferences = @prefa...	<Show Plan XML xmlns="http://schemas.microsoft.com..."	377027
9	delete from primitive_instance where subject_id ...	NULL	287016
10	insert #visual_element_ids_for_referenced_element...	NULL	226012

Query executed successfully. | P12008R2 (10.0 SP1) | P12008R2\Administrator... | WWWALPUB | 00:00:01 | 10 rows

TIP
Top 10 most
CPU intensive
SQL
Statements

Diagnostic Query: ...by frequency

TIP

Top 10 Stored Procedures ordered by the frequency of their execution

The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query editor contains the following SQL code:

```
SELECT TOP 10
b.text AS 'SP Name',
a.execution count AS 'Execution Count',
a.execution count/DATEDIFF(SECOND, a.creation time, GETDATE()) AS 'Calls/Second',
(a.total worker time/a.execution count AS 'AvgCPUTime',
a.total worker time AS 'TotalCPUTime',
a.total elapsed time/a.execution count AS 'AvgElapsedTime',
a.max logical reads,
a.max logical writes,
a.total physical reads,
DATEDIFF(MINUTE, a.creation time, GETDATE()) AS 'Age in Cache' FROM sys.dm_exec_query_stats a
CROSS APPLY sys.dm_exec_sql_text(a.sql_handle) b
WHERE b.dbid = DB_ID() -- only for current database
ORDER BY a.execution count DESC
```

The results pane shows the following data:

SP Name	Execution Count	Calls/Second	AvgCPU Time	TotalCPU Time	AvgElapsedTime	max	max	total_pr	Age in Cache
/	2	0	0	0	0	3	0	0	133/3
/	2	0	500	1000	1000	3	0	2	13373

The status bar at the bottom indicates: Query executed successfully. P12008R.2 (JC.D SP L) P12008R.2\Administrator... SR.10312307E 00:00:00 2 rows

Diagnostic Query: ...by recompile

The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query editor contains the following SQL code:

```
SELECT TOP 10
  b.text AS query_text,
  plan_generation_num,
  execution_count,
  DB_NAME(mbid) AS database_name,
  OBJECT_NAME(objectid) AS [object name]
FROM sys.dm_exec_query_stats a
CROSS APPLY sys.dm_exec_sql_text(sql_handle) AS b WHERE plan_generation_num > 1
ORDER BY plan_generation_num DESC
```

The results pane displays a table with 10 rows of data. The columns are query_text, plan_generation_num, execution_count, database_name, and object_name. The data shows that the most frequently re-compiled statements are 'create procedure internal_multi_object_check...' statements.

	query_text	plan_generation_num	execution_count	database_name	object_name
1	create procedure internal_multi_object_check...	31	1	sr103124108	NULL
2	create procedure internal_multi_object_check...	30	1	sr103124108	NULL
3	create procedure internal_multi_object_check...	29	1	sr103124108	NULL
4	create procedure internal_multi_object_check...	28	1	sr103124108	NULL
5	create procedure internal_multi_object_check...	27	1	sr103124108	NULL
6	create procedure internal_multi_object_check...	26	2	sr103124108	NULL
7	create procedure internal_multi_object_check...	25	2	sr103124108	NULL
8	create procedure internal_multi_object_check...	24	2	sr103124108	NULL
9	create procedure internal_multi_object_check...	23	2	sr103124108	NULL
10	create procedure internal_multi_object_check...	22	2	sr103124108	NULL

Query executed successfully. P[2008R2 (10.0 SP1) P:2008R2Administrator... WWA_MDD 00:00:00 10 rows

TIP
most frequently
re-compiled
statements

Diagnostic Query: ...by I/O

TIP

The most I/O intensive queries:

The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results grid. The query is designed to identify the top 10 most I/O intensive queries based on the sum of logical reads and logical writes.

```
SELECT TOP 10
    total_logical_reads,
    total_logical_writes,
    execution_count,
    total_logical_reads+total_logical_writes AS [IO total],
    b.text AS query_text,
    db_name(b.dbid) AS database_name,
    o.objectid AS object_id
FROM sys.dm_exec_query_stats a
CROSS APPLY sys.dm_exec_sql_text(sql_handle) b
WHERE total_logical_reads+total_logical_writes > 0
ORDER BY [IO total] DESC
```

	total_logical_reads	total_logical_writes	execution_count	IO_total	query_text	database_name	object_id
1	2079597	0	603193	2079597	create procedure internal_get_current_db_time...	sr103124100	930818375
2	140620	0	1	140620	SPI FACT FROM sys.dm_db_index_physical_s	NULL	NULL
3	90086	0	1	90086	(@_msparam_0 nvarchar(1000),@_msparam_1...	NULL	NULL
4	87477	0	1	87477	(@_msparam_0 nvarchar(4000),@_msparam_1...	NULL	NULL
5	83169	0	1	83169	(@_msparam_0 nvarchar(4000),@_msparam_1...	NULL	NULL
6	83029	0	1	83029	(@_msparam_0 nvarchar(4000),@_msparam_1...	NULL	NULL
7	42120	0	1	42120	(@_msparam_0 nvarchar(4000),@_msparam_1...	NULL	NULL
8	18270	0	290	18270	-- This stored procedure will return the GalaxySt...	sr103124108	930818373
9	17400	0	290	17400	-- This stored procedure will return the GalaxySt...	sr103124108	930818373
10	17400	0	290	17400	-- This stored procedure will return the GalaxySt...	sr103124108	930818373

Query executed successfully. | 6/20/2008 7:10:05 PM | P12008R2\Administrator... | W/WAIT/F | 00:00:00 | 0 rows

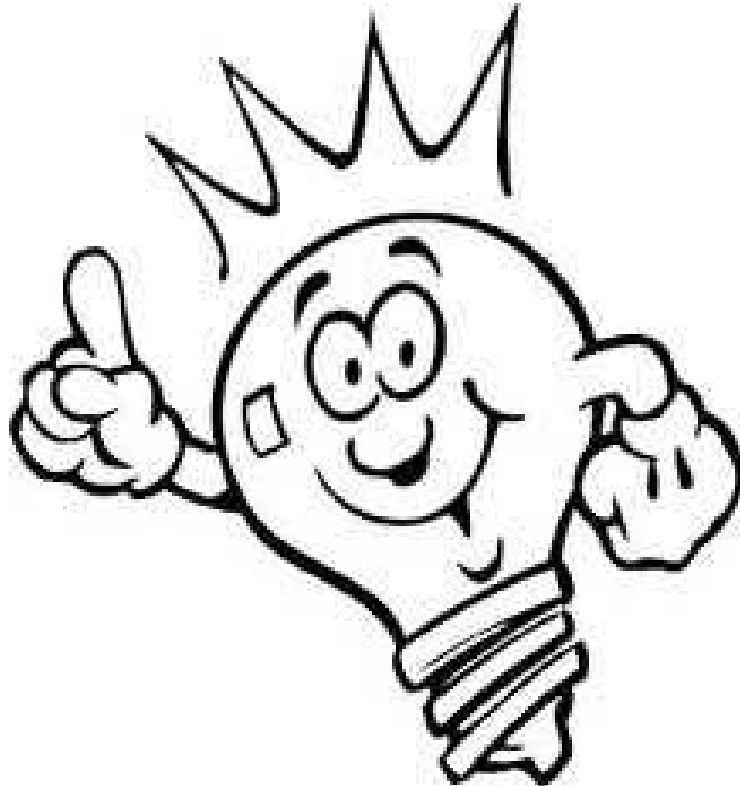
Diagnostic Query: CTRL+1 and CTRL+2

Key short cut on Management Studio:

The screenshot shows the SQL Server Enterprise Manager interface. A yellow starburst graphic points to the text 'SP_LOCK' in the top pane. Below it, a keyboard overlay shows the '2' key highlighted in yellow. The bottom pane displays a table with 17 rows of data.

	spid	stuid	ObjId	InstId	Type	Resource	Mode	Status
1	51	7	0	0	DB		S	GRANT
2	52	7	0	0	DB		S	GRANT
3	53	7	0	0	DB		S	GRANT
4	54	7	0	0	DD		S	GRANT
5	55	7	0	0	DD		S	GRANT
6	56	7	0	0	DD		S	GRANT
7	57	7	0	0	DD		S	GRANT
8	58	/	0	0	DU		S	GRANT
9	59	/	0	0	DB		S	GRANT
10	61	/	0	0	DB		S	GRANT
11	62	/	0	0	DB		S	GRANT
12	63	/	0	0	DB		S	GRANT
13	64	7	0	0	DB		S	GRANT
14	65	7	0	0	DB		S	GRANT
15	66	11	0	0	DB		S	GRANT
16	67	7	0	0	DB		S	GRANT
17	68	7	0	0	DB		S	GRANT

Tips: about SQL Server Customization



Nice To Know!!!

Last part will be treating some topics that could help out us with software customization.

Tips: SQL Server CLR Integration

The Common Language Runtime (CLR) is the heart of the Microsoft [.NET Framework](#) and provides the execution environment for all [.NET Framework](#) code.

With the [CLR hosted in Microsoft SQL Server](#) (called CLR integration), [you can create](#) stored procedures, triggers, user-defined functions, user-defined types, and user-defined aggregates [in managed code](#).

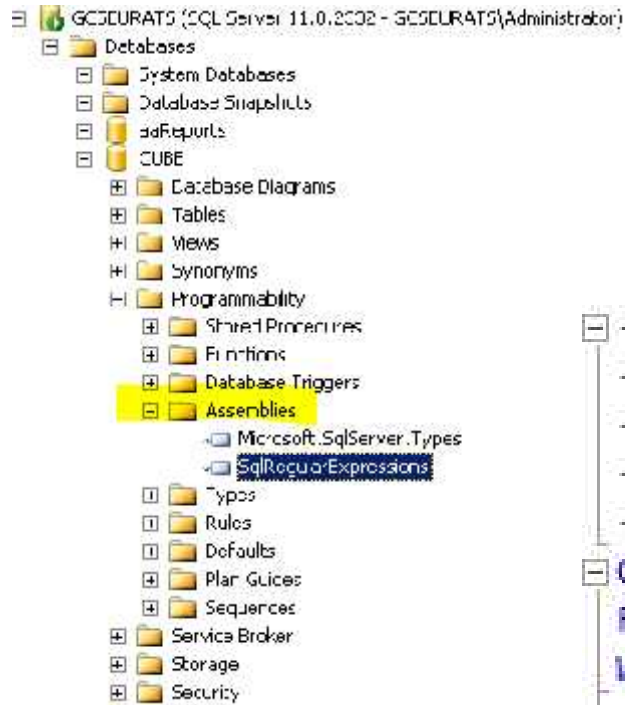
Because managed code compiles to native code prior to execution, you can achieve [significant performance increases](#) in some scenarios.

How to Enable CLR on SQL server:



```
sp_configure 'show advanced options', 1
RECONFIGURE
go
sp_configure 'clr enabled', 1
RECONFIGURE
go
```

Tips: SQL Server CLR Integration



On Sql Server side we can see all the assemblies hosted.

This is how to add an assembly on SQL side:

```
-- =====  
-- Author:      Pierluigi Iodice  
-- Create date: 19/09/2012  
-- Description: regular expression in SQL  
-- =====  
CREATE ASSEMBLY [SqlRegularExpressions]  
FROM 'c:\SqlRegularExpressions.dll'  
WITH PERMISSION_SET = SAFE
```



REMIND: If you need to have a **UNSAFE** permission, you have to change the database **TRUSTWORTHY**

Tips: SQL Server CLR Integration

Only few rules creating your .NET Assembly for SQLSrv Hosting:

1. Remove the **Namespace**
2. Using **Microsoft.SqlServer.Server**
3. Use **[SqlFunction]** Attribute

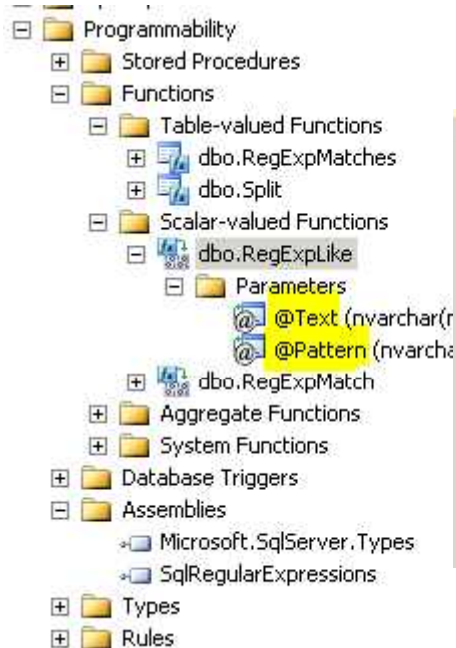


```
Microsoft Visual Studio (Administrator)
File Edit View Refactor Project Build Debug Team Data Tools Test Window Help
Debug add assembly
SqlRegularExpressions.cs* .xc
Server Explorer Toolbox
SqlRegularExpressions
using System;
using System.Data.SqlTypes; //SqlChars
using System.Collections; //IEnumerable
using System.Text.RegularExpressions; //Match, Regex
using Microsoft.SqlServer.Server; //SqlFunctionAttribute

/// <summary>
/// Class that allows to support regular expressions in MS SQL Server 2005/2000
/// </summary>
public partial class SqlRegularExpressions
{
    /// <summary>
    /// Checks string on match to regular expression
    /// </summary>
    /// <param name="text">string to check</param>
    /// <param name="pattern">regular expressions</param>
    /// <returns>true text consists match one at least, false no matches</returns>
    [SqlFunction]
    public static bool Like(string text, string pattern)
    {
        Match match = Regex.Match(text, pattern);
        return (match.Value != String.Empty);
    }
}
```

Tips: SQL Server CLR Integration

Mind to create the Function on SQL side:



```
-- =====  
-- Author: Pierluigi Iodice  
-- Create date: 19/09/2012  
-- Description: regular expression Like match  
-- =====  
  
CREATE FUNCTION [dbo].[RegExpLike](@Text [nvarchar](max), @Pattern [nvarchar](255))  
RETURNS [bit] WITH EXECUTE AS CALLER  
AS  
EXTERNAL NAME [SqlRegularExpressions].[SqlRegularExpressions].[Like]  
GO
```

The name of SQL Function parameters need to match **Exactly** with name of Net Function parameters!!!



Tips: SQL Server CLR Integration

H
O
W

i
t

W
O
R
K
S

```
select distinct id HotFixID, productname, synopsis from [dbo].[VN_CQUEST_CRHotFix]
where dbo.RegExpLike(calesce(synopsis, ''), 'Archestra.+graph[a-z]+')= 1
```

	HotFixID	productname	synopsis
1	L00072430	Industrial Application Serve	million ID bench - If you have an application with Archestra Graphics running on multiple Terminal Server sessions running on
2	L00075419	IrTouch	Customer Wrigley SR #31610315 Archestra graphics change are no: deployed to the client: (HF-87)
3	L00093964	Industrial Application Serve	SR:38370056 - Named Scripts inside an Archestra graphics are no: fired if birded to a bad quality IrTouch Tag. Also custo
4	L00103368	Industrial Application Serve	33430525: Windows in intouch with an Archestra graphics containing CF client controls seems to automatically be brought th
5	L00122558	IrTouch	SR 45770056 - Archestra graphic objects aren't validated with "Custom Property xxx has a circular reference" error message
6	L00123875	AAWebGraphics	[SR 43610658] Archestra graphic - scroll bar issue.



This would be a simple regular expression, but can work also with something of strongest!!!
i.e. email RegExp: `^[a-z._-]+\@[a-z._-]+\.[a-z]{2,4}$`

Tips: Optimizing your customization

Everything can be improved!

Next few slide will show you how to write a performing query, what to use, what would best to avoid, the means of some SQL clause, ...



Tips: Optimize SQL Queries

When Identified Long-Running Query:

- Put it on SQLS Management Studio and start to analyze the cause of slowness:
 1. Using SET statements: `SHOWPLAN_ALL`, `STATISTIC IO/TIME/PROFILE ON`
 2. Using `SQL Query Analyzer` options

SQLQuery5.sql -...istratur (57)*

```
select * from dbo.MyTempTable  
left join dbo.MyTempTableDet on MyTempTable.NodeId = MyTempTableDet.NodeId
```

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

```
select * from dbo.MyTempTable left join dbo.MyTempTableDet on MyTempTable.NodeId = MyTempTableDet.NodeId
```

Execution Plan:

- SELECT (Cost: 0%)
- Nested Loops (Left Outer Join) (Cost: 0%)
- Table Scan [MyTempTable] (Cost: 19%)
- Table Scan [MyTempTableDet] (Cost: 51%)

Tips: Optimize SQL Queries

Table Scan	
Physical Operation	Table Scan
Logical Operation	Table Scan
Estimated I/O Cost	0.0039442
Estimated CPU Cost	0.0000818
Estimated Number of Executions	3
Estimated Operator Cost	0.0041096 (51%)
Estimated Subtree Lost	0.0041896
Estimated Number of Rows	3
Estimated Row Size	27 B
Ordered	False
Node ID	2
Object	
[MyTempDB].[dbo].[MyTempTableDet]	
Output List	
[MyTempDB].[dbo].[MyTempTableDet].RowId,	
[MyTempDB].[dbo].[MyTempTableDet].ParentId,	
[MyTempDB].[dbo].[MyTempTableDet].NodeId,	
[MyTempDB].[dbo].[MyTempTableDet].NodeParentId,	
[MyTempDB].[dbo].[MyTempTableDet].NodeOwner	

- Analyzing the Results:
 - Physical Operation: **Avoid Table Scan**
 - Estimated cost: **I/O** vs **CPU** intensive
 - Estimated Number of Execution **Avoid no needed loops**
 - Estimate Row Size: **Avoid Large size**

Fastest query improve your application



Tips: Writing SQL Queries



- How to write an efficient query:
 - Write correctly formed queries, using correct **ON** clause and avoid **DISINCT**
 - Return only the rows and columns needed, **avoid *** and use **TOP x**
 - Avoid expensive operators such as **NOT LIKE**.
 - Avoid explicit or implicit functions in **WHERE** clauses.
 - Use **stored procedures** or parameterized queries.
 - Minimize **cursor** use.
 - Avoid long actions in triggers, or best, **Avoid Trigger!**
 - Use **temporary tables** and table variables appropriately.
 - Limit **query** and **index hints** use.
 - Use **SET NO COUNT ON**
 - **NOT/IN** vs **NOT/EXIST** especially with null value matching
 - Use **with (nolock/readpast)** to have a dirty read

Tips: Writing SQL Queries

Next video will show the difference writing a performing query.



SQL Server from ArcestraA scripts



- The recommended method is to use the included SQL Server Data Components:
 - \$SQLData object, scripting library, and SQLGrid control.

- These components offer a lot of flexibility:
 - Connection pooling
 - Transaction support
 - Synchronous and Asynchronous execution

SQL Server from ArcestraA scripts



If you **absolutely need** to use .NET scripting to access the DB:

- Ensure that scripts are asynchronous.
- Use Multiple Active Result Set option (MARS)
- Manage the DB connection lifecycle.
- Use Stored Procedures
- Use System.Data.SqlClient

Don't care about all these recommendations to make an



Question/ feedbacks/ request



...just to clarify any doubt before the Hands On!

Hands on Time! Custom SQL Project

Project: Replicate the Historian tag on Performance Monitor

Knowledge: .NET, SQL, and Performance Monitor Tools



Hands on Time! Summary

Project: Replicate the Historian tag on Performance Monitor

Step 1: Creating .NET Function

Step 2: Creating SQL Function

Step 3: Creating SQL Job scheduled each minute

Step 4: Observing the results on Process Monitor



Hands on Time! Create .NET Function



```
using System;
using System.Collections.Generic;
using System.Text;
using System.Diagnostics;
using Microsoft.SqlServer.Server;

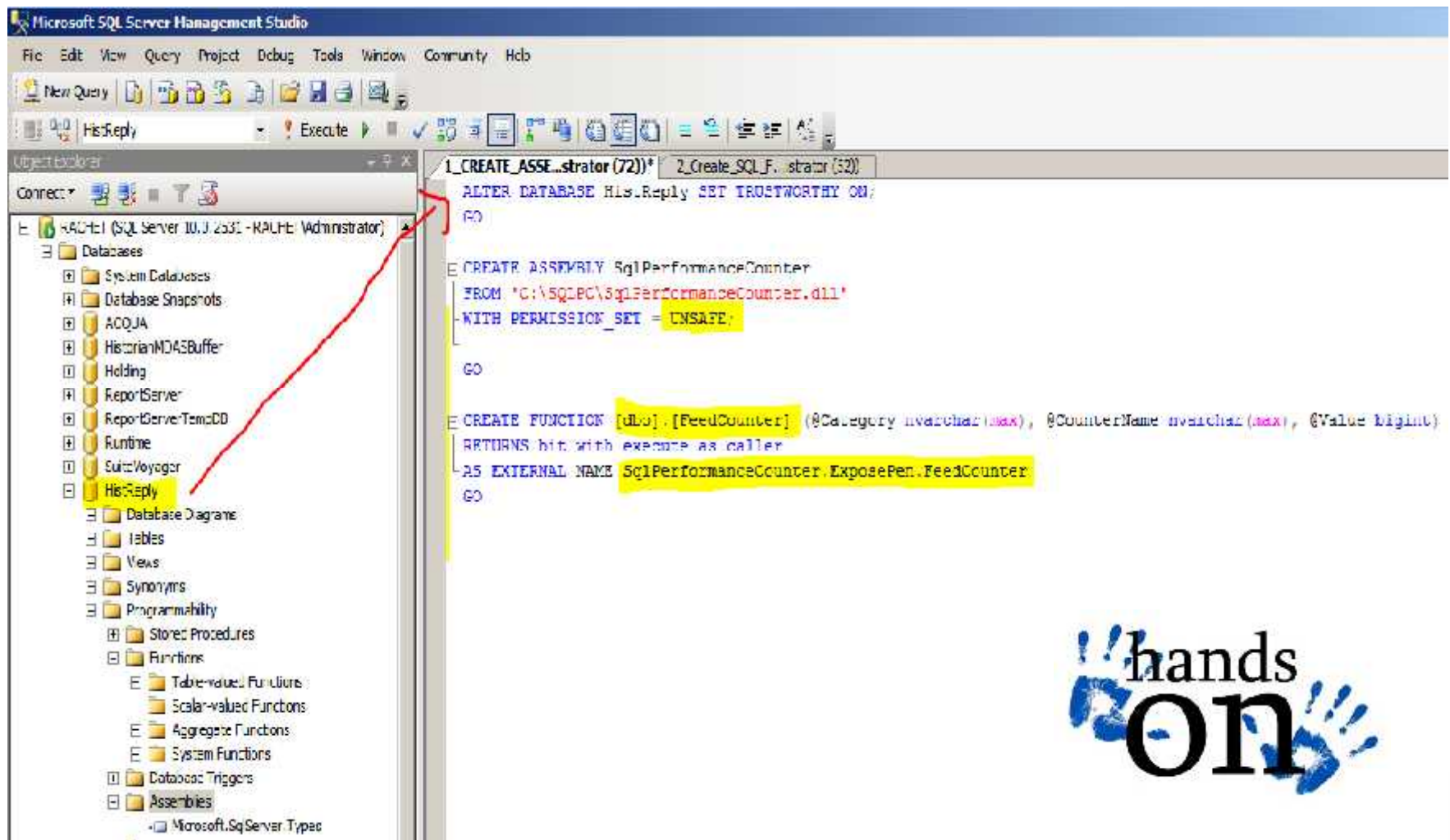
public class ExposePer
{
    [SqlFunction]
    public static bool FoodCounter(string Category, string CounterName, Int64 Value)
    {
        // Create a collection of type CounterCreationDataCollection.
        if (!System.Diagnostics.PerformanceCounterCategory.Exists(Category))
        {
            System.Diagnostics.CounterCreationDataCollection CounterData = new System.Diagnostics.CounterCreationDataCollection();
            // Create the counters and set their properties.
            System.Diagnostics.CounterCreationData cdCounter1 = new System.Diagnostics.CounterCreationData();
            cdCounter1.CounterName = CounterName;
            cdCounter1.CounterHelp = CounterName;
            cdCounter1.CounterType = System.Diagnostics.PerformanceCounterType.NumberOfItems64;
            // Add both counters to the collection.
            CounterData.Add(cdCounter1);
            // Create the category and pass the collection to it.
            System.Diagnostics.PerformanceCounterCategory.Create(Category, Category,
                PerformanceCounterCategoryType.SingleInstance, CounterData);
        }
        Delete_Existing_category

        System.Diagnostics.PerformanceCounter x = new PerformanceCounter(Category, CounterName, false);
        x.RawValue = Value;
        x.Close();

        return true;
    }
}
```



Hands on Time! Create SQL Function



The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the server hierarchy for 'RAUHE1 (SQL Server 10.0.2531 - RAUHE: Administrator)'. The 'Hi:Reply' database is selected, and the 'Assemblies' folder is expanded. The right pane shows a query window with the following SQL script:

```
ALTER DATABASE Hi:Reply SET TRUSTWORTHY ON;
GO

CREATE ASSEMBLY SqlPerformanceCounter
FROM 'C:\SQLPC\SqlPerformanceCounter.dll'
WITH PERMISSION_SET = UNSAFE;
GO

CREATE FUNCTION [dbo].[FeedCounter] (@Category nvarchar(max), @CounterName nvarchar(max), @Value bigint)
RETURNS bit with execute as caller
AS EXTERNAL NAME SqlPerformanceCounter.ExposePer.FeedCounter;
GO
```



Hands on Time! Create SQL Job

hands ON

4_Update_coun...strator (72)*

New Job

Select a step:

- General
- Steps
- Schedules
- Alerts
- Notifications
- Alerts

New Job Step

Select a step:

- General
- Advanced

Step name: **Update Log**

Type: Transact-SQL script (T-SQL)

Database: HistReply

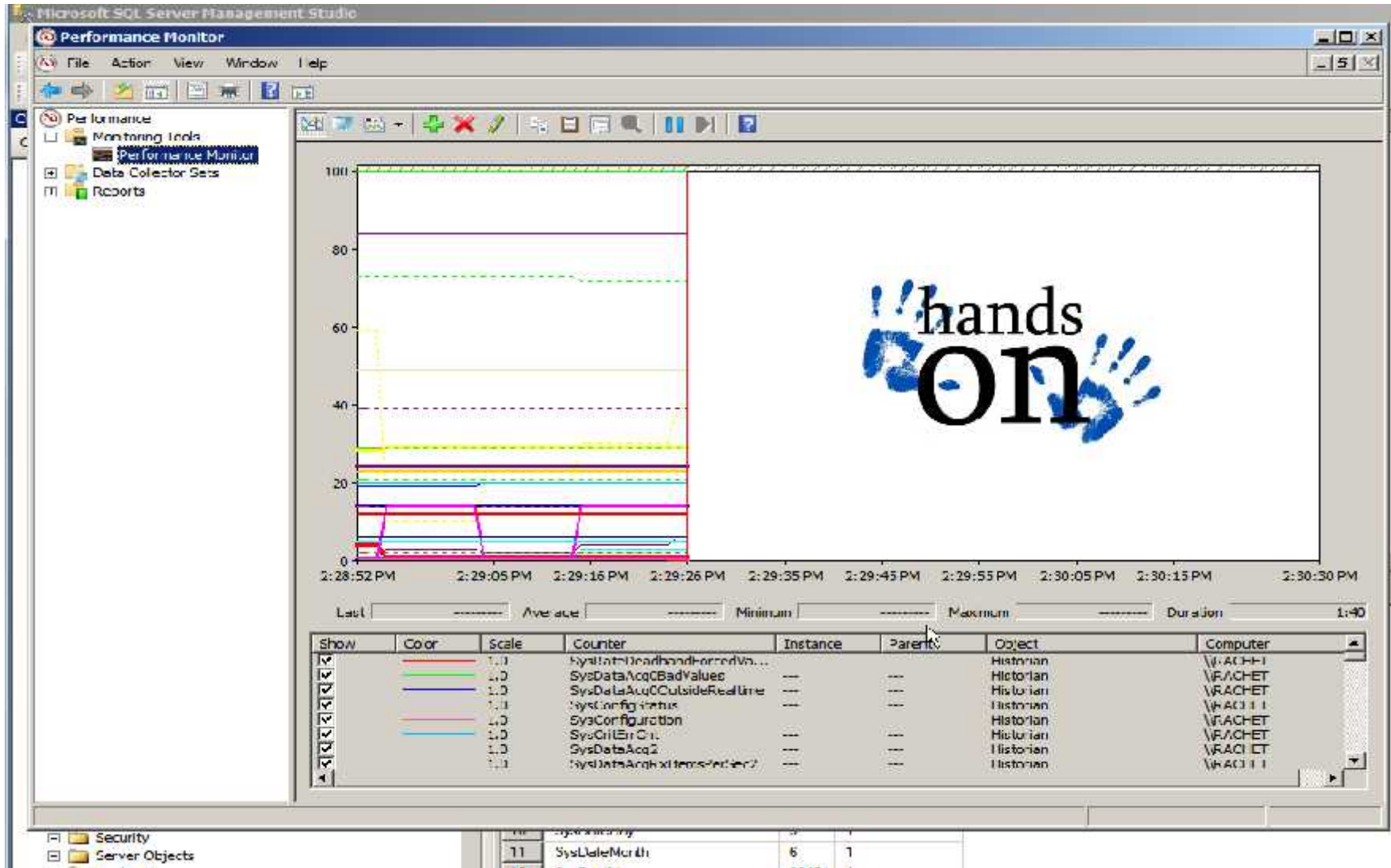
Command:
`select TagName, count(*) Value from
dbo.FeedCounter(History) where
TagName = coalesce(Value,0) CounterUpdated
from Rules.Live
where TagName in ('Sys'
or TagName in ('?Tags', 'Receptor_001.ReactTemp', 'Receptor_00' ReactLevel))`

Server: RACHET
Connection: RACHET\Administrator
View connection properties

Progress: Ready

Next Previous OK Cancel

Hands on Time! See Perf. Monitor!



Question/ feedbacks/ request

invenSYS™

Customer FIRST



Thanks !!!



Avantis • SimSci • Wonderware

THE INDUSTRIAL SOFTWARE
REVOLUTION
BEGINS NOW

inven.s.y.s.TM