# Utilizzo delle registrazioni

## Contents



# 1. Versioni

| Ver. | Note |
|---|---|
| 1.0.0 | Prima stesura |

# 2. Analisi dei comandi

Il servizio di SQL Easy Monitor non ha l'obiettivo di sostituire le funzionalità di SQL Server, bensì va a compensare una importante funzionalità mancante ovvero quella di sapere cosa sta girando sull'istanza SQL Server in un certo punto nel tempo. Si tratta di una metodologia che si basa sulla registrazione del payload ogni 1 minuto (configurabile). Il servizio lavora in modalità multithread in modo che se un'istanza non risponde, il payload delle altre viene comunque registrato. Si tratta inoltre di un servizio testato e che fa un carico minimo per il server in quanto sfrutta comandi ottimizzati e scrive le registrazioni sul database in modalità bulk.

Questo ha lo scopo di identificare in modo semplice i problemi quando accadono e poter intervenire anziché avere la preoccupazione che il problema possa riaccadere senza sapere cosa è accaduto. Questa metodologia si basa su un metodo utilizzato da anni su vari clienti e che ha permesso nel 99% dei casi di individuare i problemi occorsi in un certo punto nel tempo quando viene segnalata la lentezza del sistema.

La stessa cosa si potrebbe fare con SQL Profiler oppure Extended Events, ma con i seguenti svantaggi:

- La rilevazione dei comandi è allo START oppure END del comando e quindi l'analisi ad un punto nel tempo diventa più complicata
- La configurazione per la registrazione continua in un unico punto di consultazione non è di semplice implementazione nonché la consultazione dei dati registrati che potrebbero essere costituiti da moltissime righe e quindi di difficile consultazione

Vediamo di seguito alcuni screenshot che mostrano come individuare i problemi di lentezze dell'istanza SQL Server monitorata.

## a. Segnalazione di rallentamenti in un range temporale

Viene segnalato che il giorno 7 luglio 2023 ci sono stati rallentamenti nella fascia oraria che va dalle 17.42 alle 17.55 sull'istanza "sql04\p04".

```
use SEM
go
select top (5000)
           DtTimeStamp = [TicketIdGroup],total_elapsed_time/1000 as Seconds_From_Start ,[db],[host_name]
           ,[program_name],[blocking_session_id],[cmd]  = replace ( replace([cmd] , char(10), ''), char(13), ' ')
           ,[cmdStmt] = left(replace ( replace([cmdStmt] , char(10), ''), char(13), ' '),8000)
           ,[InstanceName], wait_type],[last_wait_type],[session_id],[start_time],[reads],[writes],[logical_reads],[row_count]
           ,[login_name],[cpu_time],[total_elapsed_time],[command],[wait_time],[wait_resource],[status]
           ,[last_request_start_time],[last_request_end_time]
           ,[percent_complete],[estimated_completion_time],[scheduler_id],[granted_query_memory], rs_governor_grp
from [Svc].[Tb_PrActiveData] with (nolock)
where 1=1
           and isnull(cmd,'') not like '%UTIL_EXEC_TRACE%' --> non prendo rek che testimoniano registrazione.
           AND TicketIdGroup between '2023-7-7 17:30:2.0' and '2023-7-7 18:00:2.0'
           AND InstanceName in ('sql04\p04') and db<>'master'
order by TicketIdGroup, cmd, cmdStmt
```


| | DtTimeStamp | Second... | db | host_name | program_name | blocking_session_id | cmd | cmdStmt | InstanceName | wait_type | last_wait_type |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 291 | 2023-07-07 17:45:04.830 | 0 | OrderProcess | SRV_ELAB12_PRD_1615 | Service_LoadAnagData_05 | 0 | NULL | CREATE PROCEDURE [dbo].[tb_spPSOPRDeCfeefare] ... | sql04\p04 | NULL | MEMORY_ALLOCATION_EXT |
| 292 | 2023-07-07 17:45:04.830 | 0 | OrderProcess | SRV_ELAB12_PRD_1613 | Service_LoadAnagData_05 | 0 | NULL | CREATE PROCEDURE [dbo].[tb_spPSOPRDeCfeefare] ... | sql04\p04 | MEMORY_AL... | MEMORY_ALLOCATION_EXT |
| 293 | 2023-07-07 17:45:04.830 | 0 | OrderProcess | SRV_ELAB12_PRD_1612 | Service_LoadAnagData_05 | 0 | NULL | sp_xml_preparedocument | sql04\p04 | NULL | MEMORY_ALLOCATION_EXT |
| 294 | 2023-07-07 17:45:04.830 | 0 | OrderProcess | SRV_ELAB12_PRD_1612 | Web_LoadAnagData_01 | 0 | pp_DecT | upDt st set st.IdRowCowe = swq.IdRow from #sourceTabl... | sql04\p04 | NULL | MEMORY_ALLOCATION_EXT |
| 295 | 2023-07-07 17:45:04.830 | 0 | OrderProcess | SRV_ELAB02_PRD_PROD03 | Web_Elab_Analysis_02 | 0 | pp_GE143 | | sql04\p04 | | CXPACKET |
| 296 | 2023-07-07 17:45:04.830 | 0 | OrderProcess | SRV_ELAB02_PRD_PROD03 | Web_Elab_Analysis_02 | 0 | pp_GE143 | | sql04\p04 | | CXPACKET |
| 297 | 2023-07-07 17:45:04.830 | 0 | OrderProcess | SRV_ELAB02_PRD_PROD03 | Web_Elab_Analysis_02 | 0 | pp_GE143 | | sql04\p04 | | CXPACKET |
| 298 | 2023-07-07 17:45:04.830 | 18 | OrderProcess | SRV_ELAB02_PRD_PROD04 | Web_Elab_Analysis_02 | 0 | pp_GE206 | SELECT convert(varchar(50),IdRow) AS IdRow, --IdRow, ... | sql04\p04 | CXCONSUMER | CXCONSUMER |
| 299 | 2023-07-07 17:45:04.830 | 289 | OrderProcess | SRV_ELAB02_PRD_PROD01 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 300 | 2023-07-07 17:45:04.830 | 207 | OrderProcess | SRV_ELAB02_PRD_PROD05 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 301 | 2023-07-07 17:45:04.830 | 266 | OrderProcess | SRV_VIRTBTCH_PRD_07 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 302 | 2023-07-07 17:45:04.830 | 239 | OrderProcess | SRV_ELAB02_PRD_PROD08 | Web_Elab_Analysis_02 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 303 | 2023-07-07 17:45:04.830 | 41 | OrderProcess | SRV_VIRTBTCH_PRD_08 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 304 | 2023-07-07 17:45:04.830 | 371 | OrderProcess | SRV_VIRTBTCH_PRD_02 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD |
| 305 | 2023-07-07 17:45:04.830 | 308 | OrderProcess | SRV_ELAB02_PRD_PROD04 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD |
| 306 | 2023-07-07 17:45:04.830 | 5 | OrderProcess | SRV_VIRTBTCH_PRD_01 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 307 | 2023-07-07 17:45:04.830 | 259 | OrderProcess | SRV_ELAB02_PRD_PROD03 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 308 | 2023-07-07 17:45:04.830 | 325 | OrderProcess | SRV_ELAB02_PRD_PROD02 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD |
| 309 | 2023-07-07 17:45:04.830 | 275 | OrderProcess | SRV_VIRTBTCH_PRD_05 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 310 | 2023-07-07 17:45:04.830 | 239 | OrderProcess | SRV_ELAB02_PRD_PROD08 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 311 | 2023-07-07 17:45:04.830 | 1 | OrderProcess | SRV_ELAB02_PRD_PROD1601 | Web_LoadAnagData_01 | 0 | pp_GE221 | Select isnull(CfeeRs.IdRow, RTF.IdRow) IdRow , isnull(Cfee... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 312 | 2023-07-07 17:45:04.830 | 10 | OrderProcess | SRV_ELAB02_PRD_PROD05 | Web_Elab_Analysis_02 | 0 | pp_GE225 | Select sqlScore.IdRow , sqlScore.Dt --, Dtadd(day, -1, (LEA... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 313 | 2023-07-07 17:45:04.830 | 1 | OrderProcess | SRV_ELAB12_PRD_1614 | Service_LoadAnagData_05 | 0 | pp_GE37 | select T.CfeOPR , X.CD , X.Value , T.fare into #temp_f... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 314 | 2023-07-07 17:45:04.830 | 1 | OrderProcess | SRV_ELAB12_PRD_1615 | Service_LoadAnagData_05 | 0 | pp_GE37 | select T.CfeOPR , X.CD , X.Value , T.fare into #temp_f... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 315 | 2023-07-07 17:45:04.830 | 1 | OrderProcess | SRV_ELAB12_PRD_1612 | Service_LoadAnagData_05 | 0 | pp_GE37 | select T.CfeOPR , X.CD , X.Value , T.fare into #temp_f... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 316 | 2023-07-07 17:45:04.830 | 1 | OrderProcess | SRV_ELAB12_PRD_1611 | Service_LoadAnagData_05 | 0 | pp_GE37 | select T.CfeOPR , X.CD , X.Value , T.fare into #temp_f... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 317 | 2023-07-07 17:45:04.830 | 0 | OrderProcess | SRV_ELAB02_PRD_PROD03 | Web_Elab_Analysis_02 | 0 | pp_GE46 | SET @BAopiAPITALLEG1=( SELECT TOP 1 BaopiapitalLeg1 ... | sql04\p04 | CXPACKET | CXPACKET |
| 318 | 2023-07-07 17:45:04.830 | 0 | OrderProcess | SRV_ELAB02_PRD_PROD03 | Web_Elab_Analysis_02 | 0 | pp_GE46 - (... | /* ---------- Aut... | sql04\p04 | | CXPACKET |
| 319 | 2023-07-07 17:45:04.830 | 0 | OrderProcess | SRV_ELAB12_PRD_1611 | Service_LoadAnagData_05 | 0 | pp_GE48 | SELECT TR.*, RTRIM(UPwid(O.relk)) AS relk FROM dbo.tb_... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 320 | 2023-07-07 17:45:04.830 | 255 | OrderProcess | SRV_ELAB02_PRD_PROD01 | Web_Elab_Analysis_02 | 0 | pp_GE71 | SELECT distinct maxt(t1.idTimeSttb_p) over (parOigion by t1.Id... | sql04\p04 | CXPACKET | CXPACKET |
| 321 | 2023-07-07 17:46:06.790 | 9 | OrderProcess | SRV_ELAB12_PRD_1612 | Web_LoadAnagData_01 | 0 | pp_DecT | Insert Into #temp_DeCfee ( ragfd , relk , Cfeicesawi... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD |
| 322 | 2023-07-07 17:46:06.790 | 9 | OrderProcess | SRV_ELAB12_PRD_1610 | Web_LoadAnagData_01 | 0 | pp_DecT | Insert Into #temp_DeCfee ( ragfd , relk , Cfeicesawi... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD |
| 323 | 2023-07-07 17:46:06.790 | 9 | OrderProcess | SRV_ELAB12_PRD_1612 | Web_LoadAnagData_01 | 0 | pp_DecT | Insert Into #temp_DeCfee ( ragfd , relk , Cfeicesawi... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD |
| 324 | 2023-07-07 17:46:06.790 | 8 | OrderProcess | SRV_ELAB12_PRD_1612 | Web_LoadAnagData_01 | 0 | pp_DecT | upDt st set st.IdRowCowe = swq.IdRow from #sourceTabl... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD |
| 325 | 2023-07-07 17:46:06.790 | 18 | OrderProcess | SRV_ELAB12_PRD_1610 | Web_LoadAnagData_01 | 0 | pp_GE19 | Select sql1.IdRoworeiolio , sql1.idCD , sql1.Dt , max(CQ.DtUp... | sql04\p04 | CXCONSUMER | CXCONSUMER |
| 326 | 2023-07-07 17:46:06.790 | 307 | OrderProcess | SRV_ELAB02_PRD_PROD08 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 327 | 2023-07-07 17:46:06.790 | 343 | OrderProcess | SRV_VIRTBTCH_PRD_05 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 328 | 2023-07-07 17:46:06.790 | 74 | OrderProcess | SRV_VIRTBTCH_PRD_01 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 329 | 2023-07-07 17:46:06.790 | 328 | OrderProcess | SRV_ELAB02_PRD_PROD03 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |

Osservando la figura sopra, si nota che il problema è dato dalla chiamata alla stored "pp_GE220" che esegue lo statement "`Select convert(varchar(50),allTable.IdRow) IdRow , convert(varch...`" che è troppo invasivo per l'istanza e viene chiamato troppe volte parallelamente. Queste chiamate vengono fatte principalmente dal "program_name=' Service_Elab_Analysis_01'". Guardando la colonna "host_name" si vede che questa chiamata è fatta da più macchine differenti.

| | DtTimeStamp | Second... | db | host_name | program_name | blocking_session_id | cmd | cmdStmt | InstanceName | wait_type | last_wait_type |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 570 | 2023-07-07 17:55:35.283 | 1 | OrderProcess | SRV_ELAB02_PRD_PROD02 | Service_Elab_Analysis_01 | 0 | pp_GE147 | select a.IdRow , Dt , widFO_DAILY , a.Cowe_VALUE , a.D... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 571 | 2023-07-07 17:55:35.283 | 18 | OrderProcess | SRV_ELAB02_PRD_PROD05 | Web_Elab_Analysis_02 | 0 | pp_GE208 | Select Cfeicesawi , Priority , IdRow , Cfeice , IdUdiType , Id... | sql04\p04 | ASYNC_NET... | ASYNC_NETWORK_IO |
| 572 | 2023-07-07 17:55:35.283 | 930 | OrderProcess | SRV_ELAB02_PRD_PROD01 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | ASYNC_NET... | ASYNC_NETWORK_IO |
| 573 | 2023-07-07 17:55:35.283 | 177 | OrderProcess | SRV_ELAB02_PRD_PROD05 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | NULL | RESERVED_MEMORY_ALLOCATION_EXT |
| 574 | 2023-07-07 17:55:35.283 | 262 | OrderProcess | SRV_ELAB02_PRD_PROD08 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD |
| 575 | 2023-07-07 17:55:35.283 | 16 | OrderProcess | SRV_VIRTBTCH_PRD_01 | Service_Elab_Analysis_01 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 576 | 2023-07-07 17:55:35.283 | 174 | OrderProcess | SRV_ELAB02_PRD_PROD01 | Web_Elab_Analysis_02 | 0 | pp_GE220 | Select convert(varchar(50),allTable.IdRow) IdRow , convert(v... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 577 | 2023-07-07 17:55:35.283 | 6 | OrderProcess | SRV_ELAB02_PRD_PROD04 | Web_Elab_Analysis_02 | 0 | pp_GE221 | Select isnull(CfeeRs.IdRow, RTF.IdRow) IdRow , isnull(Cfee... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 578 | 2023-07-07 17:55:35.283 | 38 | OrderProcess | SRV_ELAB02_PRD_PROD04 | Service_Elab_Analysis_01 | 0 | pp_GE225 | Select sqlScore.IdRow , sqlScore.Dt --, Dtadd(day, -1, (LEA... | sql04\p04 | CXPACKET | CXPACKET |
| 579 | 2023-07-07 17:55:35.283 | 2 | OrderProcess | SRV_ELAB01_PRD_1616 | Service_ProcessorOrder_01 | 0 | pp_GE232 | Select Pr1 , Data , CfeIdV , CfePae , DescCumcy , IdRo... | sql04\p04 | CXPACKET | CXPACKET |
| 580 | 2023-07-07 17:55:35.283 | 1 | OrderProcess | SRV_ELAB12_PRD_1613 | Service_LoadAnagData_05 | 0 | pp_GE37 | select T.CfeOPR , X.CD , X.Value , T.fare into #temp_f... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 581 | 2023-07-07 17:55:35.283 | 0 | OrderProcess | SRV_ELAB12_PRD_1613 | Service_LoadAnagData_05 | 0 | pp_GE37 | select T.CfeOPR , X.CD , X.Value , T.fare into #temp_f... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 582 | 2023-07-07 17:55:35.283 | 0 | OrderProcess | SRV_ELAB12_PRD_1612 | Service_LoadAnagData_05 | 0 | pp_GE37 | select T.CfeOPR , X.CD , X.Value , T.fare into #temp_f... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 583 | 2023-07-07 17:55:35.283 | 0 | OrderProcess | SRV_ELAB12_PRD_1610 | Service_LoadAnagData_05 | 0 | pp_GE37 | select T.CfeOPR , X.CD , X.Value , T.fare into #temp_f... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 584 | 2023-07-07 17:55:35.283 | 0 | OrderProcess | SRV_ELAB12_PRD_1614 | Service_LoadAnagData_05 | 0 | pp_GE37 | select T.CfeOPR , X.CD , X.Value , T.fare into #temp_f... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 585 | 2023-07-07 17:55:35.283 | 0 | OrderProcess | SRV_ELAB12_PRD_1613 | Service_LoadAnagData_05 | 0 | pp_GE37 | select T.CfeOPR , X.CD , X.Value , T.fare into #temp_f... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 586 | 2023-07-07 17:56:35.733 | 9 | OrderProcess | SRV_ELAB02_PRD_PROD03 | Web_Elab_Analysis_03 | 106 | pp_GE10 | INSERT INTO #TEMP_RESULT select s.IdRow ,s.CfeIdV, s.... | sql04\p04 | LCK_M_S | LCK_M_S |
| 587 | 2023-07-07 17:56:35.733 | 1 | OrderProcess | SRV_ELAB02_PRD_PROD04 | Web_Elab_Analysis_03 | 106 | pp_GE10 | select s.IdRow ,s.CfeIdV, s.Duration, s.CfePae,t1.r as Spread,s... | sql04\p04 | LCK_M_S | LCK_M_S |
| 588 | 2023-07-07 17:56:35.733 | 10 | OrderProcess | SRV_ELAB12_PRD_1614 | Service_LoadAnagData_05 | 0 | pp_GE136 | Insert Into tb_OPRExternalUdi ( IdExternalUdi , IdOwidat... | sql04\p04 | NULL | MEMORY_ALLOCATION_EXT |
| 589 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB12_PRD_1614 | Service_LoadAnagData_05 | 0 | pp_GE136 - (... | create PROCEDURE [dbo].[tb_spWSJuristUpDtRTUdi] @Owi... | sql04\p04 | | SOS_SCHEDULER_YIELD |
| 590 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE143 | | sql04\p04 | | CXCONSUMER |
| 591 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE143 | | sql04\p04 | | CXCONSUMER |
| 592 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE143 | | sql04\p04 | | CXCONSUMER |
| 593 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE143 | | sql04\p04 | | CXCONSUMER |
| 594 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE143 | | sql04\p04 | | CXPACKET |
| 595 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 359 | pp_GE143 | | sql04\p04 | | LCK_M_S |
| 596 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE143 | | sql04\p04 | | CXPACKET |
| 597 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE143 | | sql04\p04 | | CXPACKET |
| 598 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE143 | | sql04\p04 | | CXCONSUMER |
| 599 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE143 | | sql04\p04 | | CXCONSUMER |
| 600 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE143 | | sql04\p04 | | CXCONSUMER |
| 601 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE143 | | sql04\p04 | | CXCONSUMER |
| 602 | 2023-07-07 17:56:35.733 | 15 | OrderProcess | SRV_ELAB02_PRD_PROD04 | Web_Elab_Analysis_02 | 0 | pp_GE147 | select a.IdRow , Dt , widFO_DAILY , a.Cowe_VALUE , a.D... | sql04\p04 | NULL | CXCONSUMER |
| 603 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_VIRTBTCH_PRD_02 | Service_Elab_Analysis_01 | 0 | pp_GE15 | SELECT p.IdRow , Pr1 PLoc , p.CfeIdV , isnull(Pr1opic... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 604 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB02_PRD_PROD06 | Service_Elab_Analysis_01 | 0 | pp_GE15 | SELECT p.IdRow , Pr1 PLoc , p.CfeIdV , isnull(Pr1opic... | sql04\p04 | RESOURCE_... | RESOURCE_SEMAPHORE |
| 605 | 2023-07-07 17:56:35.733 | 12 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE208 | Select distinct tb_OPRExternalUdi.Cfeicesawi, tb_OPRCD.Prior... | sql04\p04 | CXPACKET | CXPACKET |
| 606 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB01_PRD_1601 | Service_ProcessorOrder_02 | 0 | pp_GE208 - (... | Create Proc [dbo].[tb_spGetDwhf2OPRGirs] As Set NoCount ... | sql04\p04 | | CXPACKET |
| 607 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB02_PRD_PROD07 | Service_Elab_Analysis_01 | 0 | pp_GE214 | UPDt @tmpReturn SET UtilInnerJoin__vtb_sr1izioneveeaz... | sql04\p04 | NULL | PAGELATCH_EX |
| 608 | 2023-07-07 17:56:35.733 | 0 | OrderProcess | SRV_ELAB02_PRD_PROD05 | Web_Elab_Analysis_02 | 0 | pp_GE22 | Select ns * From [dbo].[tb_OPRCDGirs] ns Where IdCD = isnull... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD |

Scorrendo i comandi registrati, si vede che il problema si presenta ancora in forma meno pesante alle 17:55 e non è più presente alle 17:56. Questo permette d'individuare la fascia oraria di lentezza dell'istanza SQL Server e provvedere ad intervenire in modo puntuale cercando di fare tuning sulla stored e\o cercando di capire come mail il servizio ha effettuato tutte queste chiamate consultando i log applicativi. Questo è di fondamentale importanza in quanto previene le stesse situazioni di blocco dell'istanza e di conseguenza permette di vivere sereni senza la paura che il blocco riaccada.

## b. Analisi risposta server nel range temporale

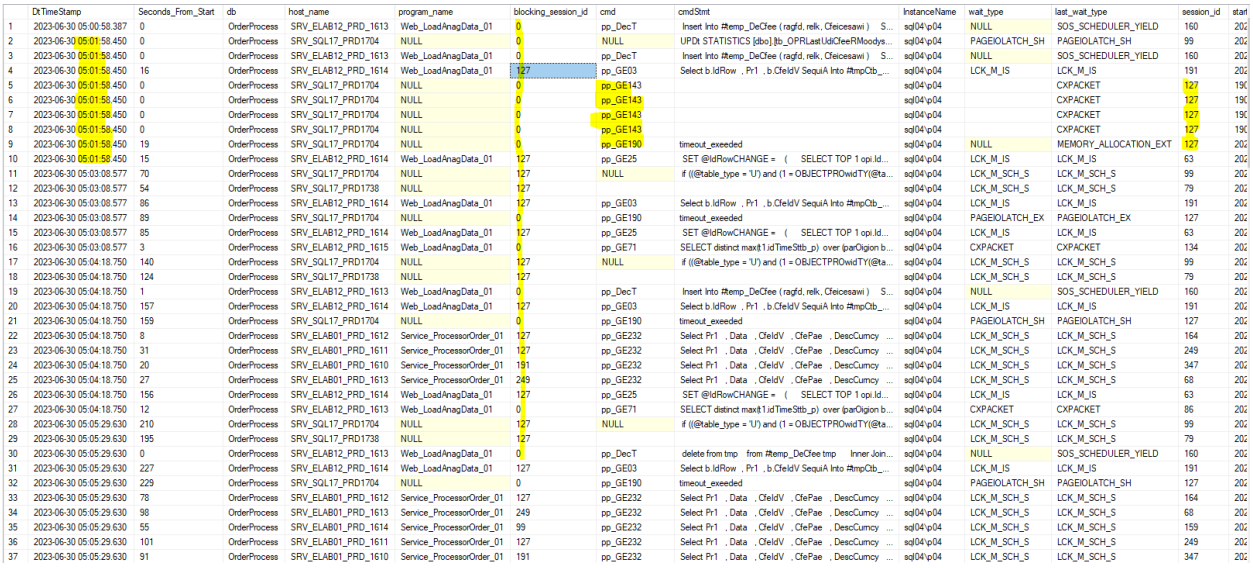
```
use SEM
go
select top (5000)
           DtTimeStamp = [TicketIdGroup],total_elapsed_time/1000 as Seconds_From_Start ,[db],[host_name]
           ,[program_name],[blocking_session_id],[cmd]  = replace ( replace([cmd] , char(10), ''), char(13), ' ') --> rem x copy su excel
           ,[cmdStmt] = left(replace ( replace([cmdStmt] , char(10), ''), char(13), ' '),8000) --> rem x copy su excel
           ,[InstanceName],
[wait_type],[last_wait_type],[session_id],[start_time],[reads],[writes],[logical_reads],[row_count],[login_name]
           ,[cpu_time],[total_elapsed_time],[command],[wait_time],[wait_resource],[status],[last_request_start_time],[last_request_end_time
]
           ,[percent_complete],[estimated_completion_time],[scheduler_id],[granted_query_memory], rs_governor_grp
-- select distinct host_name, host_name_New=''
-- into _temp_host_name
from [Svc].[Tb_PrActiveData] with(nolock)
where 1=1
           and isnull(cmd,'') like '%UTIL_EXEC_TRACE%' --> non prendo rek che testimoniano registrazione.
           AND TicketIdGroup between '2023-7-7 17:30:2.0' and '2023-7-7 18:00:2.0'
    AND InstanceName in ('sql04\p04')
order by TicketIdGroup, cmd, cmdStmt
```

| | DtTimeStamp | Seconds_From_Start | db | host_name | program_name | blocking_session_id | cmd | cmdStmt | InstanceName | wait_type | last_wait_type | session_id | start_time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2023-07-07 17:30:49.890 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 2 | 2023-07-07 17:31:49.230 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 3 | 2023-07-07 17:32:49.403 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 4 | 2023-07-07 17:33:49.557 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 5 | 2023-07-07 17:34:49.680 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 6 | 2023-07-07 17:35:49.880 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 7 | 2023-07-07 17:36:50.583 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 8 | 2023-07-07 17:37:50.880 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 9 | 2023-07-07 17:38:51.970 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 10 | 2023-07-07 17:40:51.830 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 11 | 2023-07-07 17:41:55.610 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 12 | 2023-07-07 17:42:56.313 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 13 | 2023-07-07 17:43:57.893 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 14 | 2023-07-07 17:45:04.830 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 15 | 2023-07-07 17:46:06.790 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 16 | 2023-07-07 17:47:14.813 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 17 | 2023-07-07 17:48:16.673 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 18 | 2023-07-07 17:49:24.627 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 19 | 2023-07-07 17:50:25.193 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 20 | 2023-07-07 17:51:26.813 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 21 | 2023-07-07 17:52:27.347 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 22 | 2023-07-07 17:53:28.580 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 23 | 2023-07-07 17:54:34.620 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 24 | 2023-07-07 17:55:35.283 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 25 | 2023-07-07 17:56:35.733 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 26 | 2023-07-07 17:57:36.300 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 27 | 2023-07-07 17:58:36.853 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |
| 28 | 2023-07-07 17:59:37.220 | 0 | | | NULL | 0 | UTIL_EXEC_TRACE | | sql04\p04 | 0 | | 0 | 1900-01-01 00:00:00.000 |

Il software ogni 1 minuto registra i comandi in running. Se non ci sono comandi in corso, scrive ugualmente 1 riga con "cmd=UTIL_EXEC_TRACE". Analizzando la figura sopra, si vede che c'è un buco di registrazione alle 17:44. Questo significa che probabilmente l'istanza in quella fascia oraria non ha risposto entro 1 minuto. Allargando la ricerca sui

comandi in corso si può comprendere se questo era dovuto ad un eccessivo carico dell'istanza.

## c. Rilevazione dei blocchi

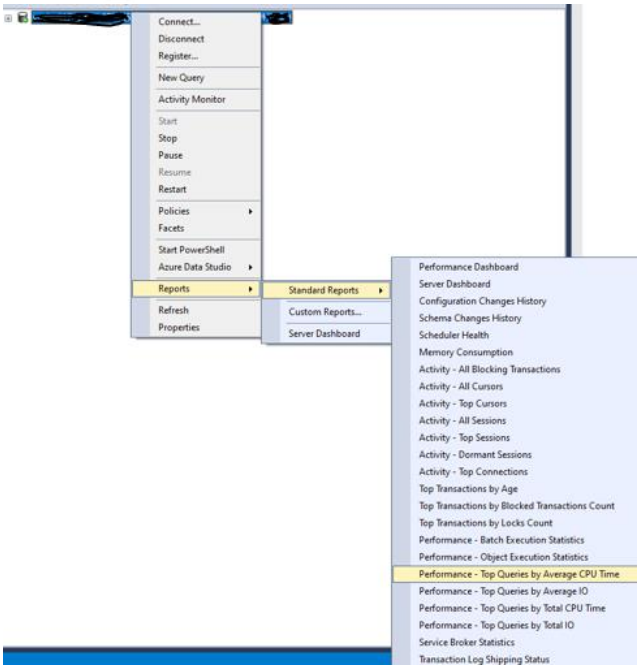


| | DtTimeStamp | Seconds_From_Start | db | host_name | program_name | blocking_session_id | cmd | cmdStmt | InstanceName | wait_type | last_wait_type | session_id | start |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2023-06-30 05:00:58.387 | 0 | OrderProcess | SRV_ELAB12_PRD_1613 | Web_LoadAnagData_01 | 0 | pp_DecT | Insert Into #temp_DeCfee ( ragfd, relk, Cfeicesawi ) S... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD | 160 | 202 |
| 2 | 2023-06-30 05:01:58.450 | 0 | OrderProcess | SRV_SQL17_PRD1704 | NULL | 0 | NULL | UPDt STATISTICS [dbo].[tb_OPRLastUdiCfeeRMoodys... | sql04\p04 | PAGEIOLATCH_SH | PAGEIOLATCH_SH | 99 | 202 |
| 3 | 2023-06-30 05:01:58.450 | 0 | OrderProcess | SRV_ELAB12_PRD_1613 | Web_LoadAnagData_01 | 0 | pp_DecT | Insert Into #temp_DeCfee ( ragfd, relk, Cfeicesawi ) S... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD | 160 | 202 |
| 4 | 2023-06-30 05:01:58.450 | 16 | OrderProcess | SRV_ELAB12_PRD_1614 | Web_LoadAnagData_01 | 127 | pp_GE03 | Select b.IdRow , Pr1 , b.CfeIdV SequiA Into #tmpCtb_... | sql04\p04 | LCK_M_IS | LCK_M_IS | 191 | 202 |
| 5 | 2023-06-30 05:01:58.450 | 0 | OrderProcess | SRV_SQL17_PRD1704 | NULL | 0 | pp_GE143 | | sql04\p04 | | CXPACKET | 127 | 19( |
| 6 | 2023-06-30 05:01:58.450 | 0 | OrderProcess | SRV_SQL17_PRD1704 | NULL | 0 | pp_GE143 | | sql04\p04 | | CXPACKET | 127 | 19( |
| 7 | 2023-06-30 05:01:58.450 | 0 | OrderProcess | SRV_SQL17_PRD1704 | NULL | 0 | pp_GE143 | | sql04\p04 | | CXPACKET | 127 | 19( |
| 8 | 2023-06-30 05:01:58.450 | 0 | OrderProcess | SRV_SQL17_PRD1704 | NULL | 0 | pp_GE143 | | sql04\p04 | | CXPACKET | 127 | 19( |
| 9 | 2023-06-30 05:01:58.450 | 19 | OrderProcess | SRV_SQL17_PRD1704 | NULL | 0 | pp_GE190 | timeout_exeeded | sql04\p04 | NULL | MEMORY_ALLOCATION_EXT | 127 | 202 |
| 10 | 2023-06-30 05:01:58.450 | 15 | OrderProcess | SRV_ELAB12_PRD_1614 | Web_LoadAnagData_01 | 127 | pp_GE25 | SET @IdRowCHANGE = ( SELECT TOP 1 opi.Id... | sql04\p04 | LCK_M_IS | LCK_M_IS | 63 | 202 |
| 11 | 2023-06-30 05:03:08.577 | 70 | OrderProcess | SRV_SQL17_PRD1704 | NULL | 127 | NULL | if (@table_type = 'U') and (1 = OBJECTPROwidTY(@ta... | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 99 | 202 |
| 12 | 2023-06-30 05:03:08.577 | 54 | OrderProcess | SRV_SQL17_PRD1738 | NULL | 127 | | | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 79 | 202 |
| 13 | 2023-06-30 05:03:08.577 | 86 | OrderProcess | SRV_ELAB12_PRD_1614 | Web_LoadAnagData_01 | 127 | pp_GE03 | Select b.IdRow , Pr1 , b.CfeIdV SequiA Into #tmpCtb_... | sql04\p04 | LCK_M_IS | LCK_M_IS | 191 | 202 |
| 14 | 2023-06-30 05:03:08.577 | 89 | OrderProcess | SRV_SQL17_PRD1704 | NULL | 0 | pp_GE190 | timeout_exeeded | sql04\p04 | PAGEIOLATCH_EX | PAGEIOLATCH_EX | 127 | 202 |
| 15 | 2023-06-30 05:03:08.577 | 85 | OrderProcess | SRV_ELAB12_PRD_1614 | Web_LoadAnagData_01 | 127 | pp_GE25 | SET @IdRowCHANGE = ( SELECT TOP 1 opi.Id... | sql04\p04 | LCK_M_IS | LCK_M_IS | 63 | 202 |
| 16 | 2023-06-30 05:03:08.577 | 3 | OrderProcess | SRV_ELAB12_PRD_1615 | Web_LoadAnagData_01 | 0 | pp_GE71 | SELECT distinct max(t1.idTimeSttb_p) over (parOigion b... | sql04\p04 | CXPACKET | CXPACKET | 134 | 202 |
| 17 | 2023-06-30 05:04:18.750 | 140 | OrderProcess | SRV_SQL17_PRD1704 | NULL | 127 | NULL | if (@table_type = 'U') and (1 = OBJECTPROwidTY(@ta... | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 99 | 202 |
| 18 | 2023-06-30 05:04:18.750 | 124 | OrderProcess | SRV_SQL17_PRD1738 | NULL | 127 | | | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 79 | 202 |
| 19 | 2023-06-30 05:04:18.750 | 1 | OrderProcess | SRV_ELAB12_PRD_1613 | Web_LoadAnagData_01 | 0 | pp_DecT | Insert Into #temp_DeCfee ( ragfd, relk, Cfeicesawi ) S... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD | 160 | 202 |
| 20 | 2023-06-30 05:04:18.750 | 157 | OrderProcess | SRV_ELAB12_PRD_1614 | Web_LoadAnagData_01 | 127 | pp_GE03 | Select b.IdRow , Pr1 , b.CfeIdV SequiA Into #tmpCtb_... | sql04\p04 | LCK_M_IS | LCK_M_IS | 191 | 202 |
| 21 | 2023-06-30 05:04:18.750 | 159 | OrderProcess | SRV_SQL17_PRD1704 | NULL | 0 | pp_GE190 | timeout_exeeded | sql04\p04 | PAGEIOLATCH_SH | PAGEIOLATCH_SH | 127 | 202 |
| 22 | 2023-06-30 05:04:18.750 | 8 | OrderProcess | SRV_ELAB01_PRD_1612 | Service_ProcessorOrder_01 | 127 | pp_GE232 | Select Pr1 , Data , CfeIdV , CfePae , DescCurncy ... | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 164 | 202 |
| 23 | 2023-06-30 05:04:18.750 | 31 | OrderProcess | SRV_ELAB01_PRD_1611 | Service_ProcessorOrder_01 | 127 | pp_GE232 | Select Pr1 , Data , CfeIdV , CfePae , DescCurncy ... | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 249 | 202 |
| 24 | 2023-06-30 05:04:18.750 | 20 | OrderProcess | SRV_ELAB01_PRD_1610 | Service_ProcessorOrder_01 | 191 | pp_GE232 | Select Pr1 , Data , CfeIdV , CfePae , DescCurncy ... | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 347 | 202 |
| 25 | 2023-06-30 05:04:18.750 | 27 | OrderProcess | SRV_ELAB01_PRD_1613 | Service_ProcessorOrder_01 | 249 | pp_GE232 | Select Pr1 , Data , CfeIdV , CfePae , DescCurncy ... | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 68 | 202 |
| 26 | 2023-06-30 05:04:18.750 | 156 | OrderProcess | SRV_ELAB12_PRD_1614 | Web_LoadAnagData_01 | 127 | pp_GE25 | SET @IdRowCHANGE = ( SELECT TOP 1 opi.Id... | sql04\p04 | LCK_M_IS | LCK_M_IS | 63 | 202 |
| 27 | 2023-06-30 05:04:18.750 | 12 | OrderProcess | SRV_ELAB12_PRD_1613 | Web_LoadAnagData_01 | 0 | pp_GE71 | SELECT distinct max(t1.idTimeSttb_p) over (parOigion b... | sql04\p04 | CXPACKET | CXPACKET | 86 | 202 |
| 28 | 2023-06-30 05:05:29.630 | 210 | OrderProcess | SRV_SQL17_PRD1704 | NULL | 127 | NULL | if (@table_type = 'U') and (1 = OBJECTPROwidTY(@ta... | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 99 | 202 |
| 29 | 2023-06-30 05:05:29.630 | 195 | OrderProcess | SRV_SQL17_PRD1738 | NULL | 127 | | | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 79 | 202 |
| 30 | 2023-06-30 05:05:29.630 | 0 | OrderProcess | SRV_ELAB12_PRD_1613 | Web_LoadAnagData_01 | 0 | pp_DecT | delete from tmp from #temp_DeCfee tmp Inner Join... | sql04\p04 | NULL | SOS_SCHEDULER_YIELD | 160 | 202 |
| 31 | 2023-06-30 05:05:29.630 | 227 | OrderProcess | SRV_ELAB12_PRD_1614 | Web_LoadAnagData_01 | 127 | pp_GE03 | Select b.IdRow , Pr1 , b.CfeIdV SequiA Into #tmpCtb_... | sql04\p04 | LCK_M_IS | LCK_M_IS | 191 | 202 |
| 32 | 2023-06-30 05:05:29.630 | 229 | OrderProcess | SRV_SQL17_PRD1704 | NULL | 0 | pp_GE190 | timeout_exeeded | sql04\p04 | PAGEIOLATCH_SH | PAGEIOLATCH_SH | 127 | 202 |
| 33 | 2023-06-30 05:05:29.630 | 78 | OrderProcess | SRV_ELAB01_PRD_1612 | Service_ProcessorOrder_01 | 127 | pp_GE232 | Select Pr1 , Data , CfeIdV , CfePae , DescCurncy ... | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 164 | 202 |
| 34 | 2023-06-30 05:05:29.630 | 98 | OrderProcess | SRV_ELAB01_PRD_1613 | Service_ProcessorOrder_01 | 249 | pp_GE232 | Select Pr1 , Data , CfeIdV , CfePae , DescCurncy ... | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 68 | 202 |
| 35 | 2023-06-30 05:05:29.630 | 55 | OrderProcess | SRV_ELAB01_PRD_1614 | Service_ProcessorOrder_01 | 99 | pp_GE232 | Select Pr1 , Data , CfeIdV , CfePae , DescCurncy ... | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 159 | 202 |
| 36 | 2023-06-30 05:05:29.630 | 101 | OrderProcess | SRV_ELAB01_PRD_1611 | Service_ProcessorOrder_01 | 127 | pp_GE232 | Select Pr1 , Data , CfeIdV , CfePae , DescCurncy ... | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 249 | 202 |
| 37 | 2023-06-30 05:05:29.630 | 91 | OrderProcess | SRV_ELAB01_PRD_1610 | Service_ProcessorOrder_01 | 191 | pp_GE232 | Select Pr1 , Data , CfeIdV , CfePae , DescCurncy ... | sql04\p04 | LCK_M_SCH_S | LCK_M_SCH_S | 347 | 202 |

Dalla figura sopra, si nota che la colonna "blocking_sessioni_id" è diversa da 0. Alle 5.01 del 30/06/2023 la session_id=127 sta bloccando altre sessioni_id. Questo permette di comprendere se i comandi vanno lunghi a causa di blocchi e non a causa di lentezza dell'istanza SQL Server e\o di problemi hardware.

## d. Comandi più invasivi nelle istanze

I comandi più invasivi per un'istanza possono essere presi con le dynamic views di SQL Server (`sys.dm_exec_query_stats`) e sono anche interfacciate dai report di SQL Server Management Studio.




Queste estrazioni però vengono fatte dall'avvio dell'istanza oppure da quando si resetta la cache di SQL Server.

Utilizzando Sql Easy Monitor si possono invece interrogare dei range temporali ben precisi.

```
select top (5000)
          DtTimeStamp = tb.[TicketIdGroup],total_elapsed_time/1000 as Seconds_From_Start ,[db],[host_name]
          ,[program_name],[blocking_session_id],[cmd]  = replace ( replace([cmd] , char(10), ''), char(13), ' ') --> rem x copy su excel
          ,[cmdStmt] = left(replace ( replace([cmdStmt] , char(10), ''), char(13), ' '),8000) --> rem x copy su excel
          ,tb.[InstanceName],
[wait_type],[last_wait_type],[session_id],[start_time],[reads],[writes],[logical_reads],[row_count],[login_name]
          ,[cpu_time],[total_elapsed_time],[command],[wait_time],[wait_resource],[status],[last_request_start_time],[last_request_end_time
]
          ,[percent_complete],[estimated_completion_time],[scheduler_id],[granted_query_memory], rs_governor_grp
-- select distinct host_name, host_name_New=''
-- into _temp_host_name
from [Svc].[Tb_PrActiveData] tb with(nolock)
          inner join
          (
                    select [TicketIdGroup], [InstanceName]--, tot=COUNT(*)
                    from [Svc].[Tb_PrActiveData] with(nolock)
                    where 1=1
                              and total_elapsed_time/1000>5
                    group by [TicketIdGroup],[InstanceName]
                    having COUNT(*)>10
          ) as sqCountBig
          on tb.TicketIdGroup = sqCountBig.TicketIdGroup
          and tb.[InstanceName] = sqCountBig.[InstanceName]
where 1=1
          and cmdStmt<>'sp_server_diagnostics'
          --and isnull(cmd,'') like '%UTIL_EXEC_TRACE%' --> non prendo rek che testimoniano registrazione.
          --AND TicketIdGroup between '2023-06-30 05:00:39.597' and '2023-06-30 05:15:39.597'
    AND tb.InstanceName in ('sql04\p04') and db<>'master'
          --and blocking_session_id<>0
order by tb.TicketIdGroup, cmd, cmdStmt
```

Il comando sopra estrae tutte le volte che ci sono state più di 10 righe in running nello stesso minuto di registrazione con durata di più di 5 secondi. A livello di parametri ogni

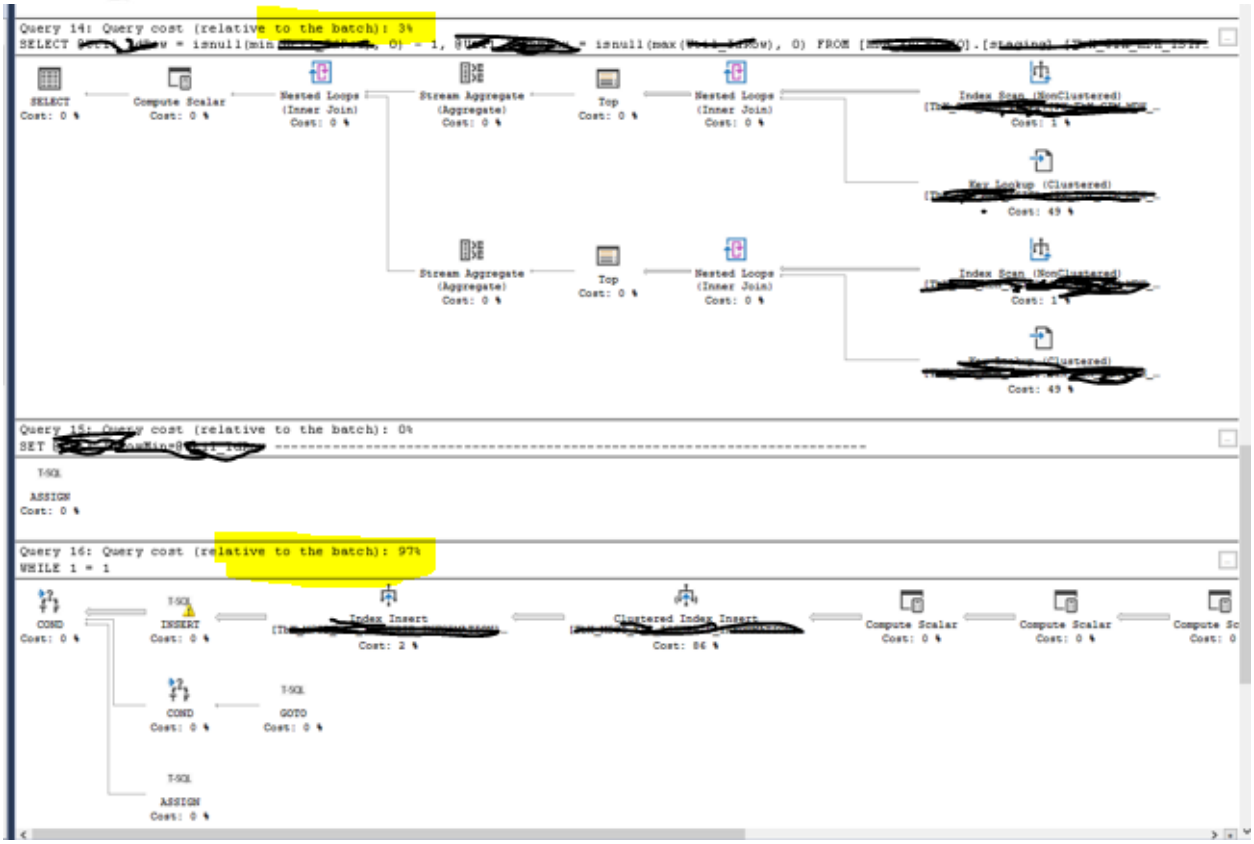
istanza\database potrà tarare delle soglie che sono soglie di Alert ovvero situazioni nelle quali si è verificato un problema per cui oltre una certa capacità l'istanza SQL Server inizia ad accodare e rispondere con tempi più lunghi a causa di blocchi oppure di termine delle risorse hardware.

| | DtTimeStamp | Seconds_From_Start | db | host_name | program_name | blocking_session_id | cmd | cmdStmt | InstanceNa |
|---|---|---|---|---|---|---|---|---|---|
| 808 | 2023-06-30 20:47:01.523 | 1 | OrderProcess | SRV_ELAB12_PRD_1612 | Service_LoadAnagData_05 | 0 | pp_GE136 | If exists( Select top 1 ts.Cfeicesawi from tb_OPRR... | sql04\p04 |
| 809 | 2023-06-30 20:47:01.523 | 36 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 810 | 2023-06-30 20:47:01.523 | 32 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 811 | 2023-06-30 20:47:01.523 | 28 | OrderProcess | SRV_VIRTBTCH_PRD_... | Service_Elab_Analysis_01 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 812 | 2023-06-30 20:47:01.523 | 36 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 813 | 2023-06-30 20:47:01.523 | 32 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 353 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 814 | 2023-06-30 20:47:01.523 | 12 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 178 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 815 | 2023-06-30 20:47:01.523 | 31 | OrderProcess | SRV_ELAB02_PRD_PR... | Service_Elab_Analysis_01 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 816 | 2023-06-30 20:47:01.523 | 3 | OrderProcess | SRV_ELAB02_PRD_PR... | Service_Elab_Analysis_01 | 109 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 817 | 2023-06-30 20:47:01.523 | 36 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 818 | 2023-06-30 20:47:01.523 | 37 | OrderProcess | SRV_ELAB02_PRD_PR... | Service_Elab_Analysis_01 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 819 | 2023-06-30 20:47:01.523 | 37 | OrderProcess | SRV_VIRTBTCH_PRD_... | Service_Elab_Analysis_01 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 820 | 2023-06-30 20:47:01.523 | 35 | OrderProcess | SRV_ELAB02_PRD_PR... | Service_Elab_Analysis_01 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 821 | 2023-06-30 20:47:01.523 | 11 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 323 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 822 | 2023-06-30 20:47:01.523 | 32 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 823 | 2023-06-30 20:47:01.523 | 32 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 824 | 2023-06-30 20:47:01.523 | 19 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 353 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 825 | 2023-06-30 20:47:01.523 | 32 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 325 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 826 | 2023-06-30 20:47:01.523 | 12 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 827 | 2023-06-30 20:47:01.523 | 12 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 828 | 2023-06-30 20:47:01.523 | 36 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 829 | 2023-06-30 20:47:01.523 | 36 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 830 | 2023-06-30 20:47:01.523 | 38 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 831 | 2023-06-30 20:47:01.523 | 37 | OrderProcess | SRV_VIRTBTCH_PRD_... | Service_Elab_Analysis_01 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 832 | 2023-06-30 20:47:01.523 | 39 | OrderProcess | SRV_VIRTBTCH_PRD_... | Service_Elab_Analysis_01 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 833 | 2023-06-30 20:47:01.523 | 36 | OrderProcess | SRV_ELAB02_PRD_PR... | Service_Elab_Analysis_01 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 834 | 2023-06-30 20:47:01.523 | 37 | OrderProcess | SRV_VIRTBTCH_PRD_... | Service_Elab_Analysis_01 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 835 | 2023-06-30 20:47:01.523 | 36 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 325 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 836 | 2023-06-30 20:47:01.523 | 36 | OrderProcess | SRV_ELAB02_PRD_PR... | Web_Elab_Analysis_02 | 297 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 837 | 2023-06-30 20:47:01.523 | 13 | OrderProcess | SRV_ELAB02_PRD_PR... | Service_Elab_Analysis_01 | 310 | pp_GE218 | IF ( Select count(*) From tb_OPRoreiolio Inner Join ... | sql04\p04 |
| 838 | 2023-06-30 20:47:01.523 | 38 | OrderProcess | SRV_ELAB12_PRD_1612 | Web_LoadAnagData_01 | 0 | pp_GE71 | SELECT distinct max(t1.idTimeSttb_p) over (parOigion ... | sql04\p04 |
| 839 | 2023-06-30 20:47:01.523 | 58 | OrderProcess | SRV_SQL17_PRD1731 | NULL | 0 | pp_GE87 | Delete from tb_OPRhistoricsr1izione with(updlock) --> ... | sql04\p04 |
| 840 | 2023-06-30 20:47:01.523 | 0 | OrderProcess | SRV_SQL17_PRD1731 | NULL | 0 | pp_GE8... | CREATE proc [dbo].[tb_spSvecchiaTabelleGenerico] ... | sql04\p04 |
| 841 | 2023-06-30 20:48:01.990 | 4511 | OrderProcess | | NULL | 0 | NULL | backup database "SPtb_OPR" to VIRRTL_DEVICE-' | sql04\p04 |

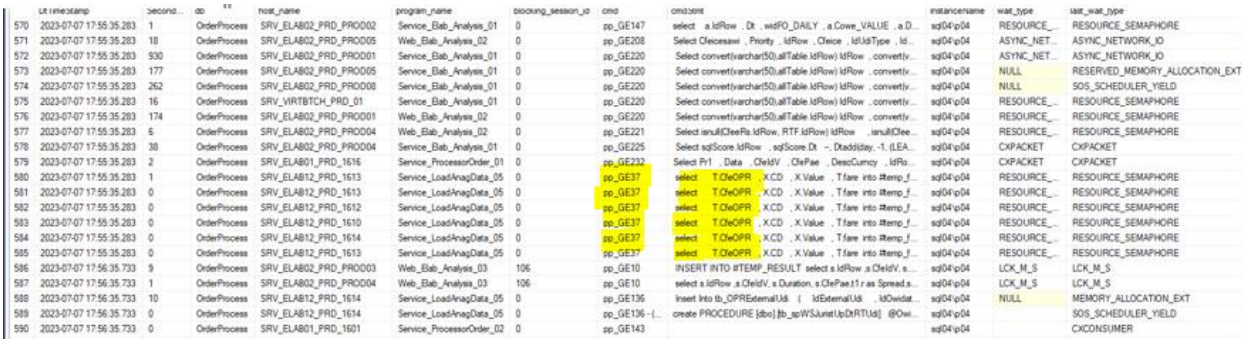
Il risultato nella figura sopra, mostra ad esempio che nella fascia oraria delle 20:47 del giorno 30/6 ci sono state situazioni di blocchi prolungati.

## e. Statement più invasivi nelle stored

A volte l'analisi degli execution plan assegna percentuali alte di utilizzo a degli statement che sono meno invasivi di quelli che realmente rimangono in esecuzione per più tempo all'interno di una sequenza di comandi chiamati da una stored procedure.

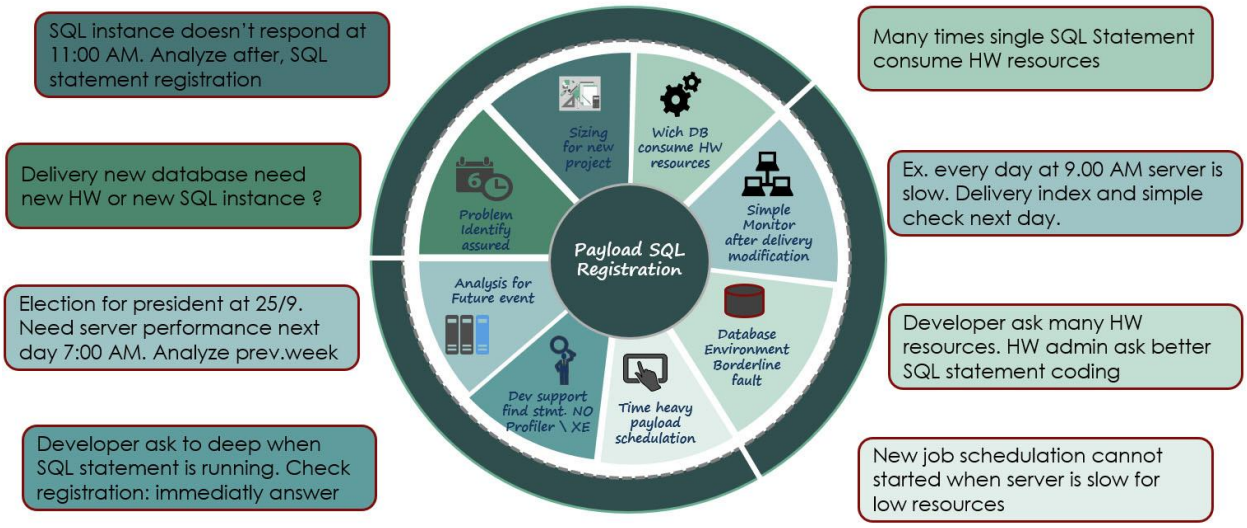



Ad esempio nel plan della figura sopra il primo statement occupa il 3%, mentre il secondo il 97%.



Può accadere che il comando al 3% è quello che si vede come "Select T.CfeOPR…" della figura sopra ovvero scoprire che nonostante l'exec plan lo indichi come il meno invasivo è quello su cui è necessario fare il tuning in quanto è quello che rimane in corso per più tempo.

# 3. Altre funzionalità utili

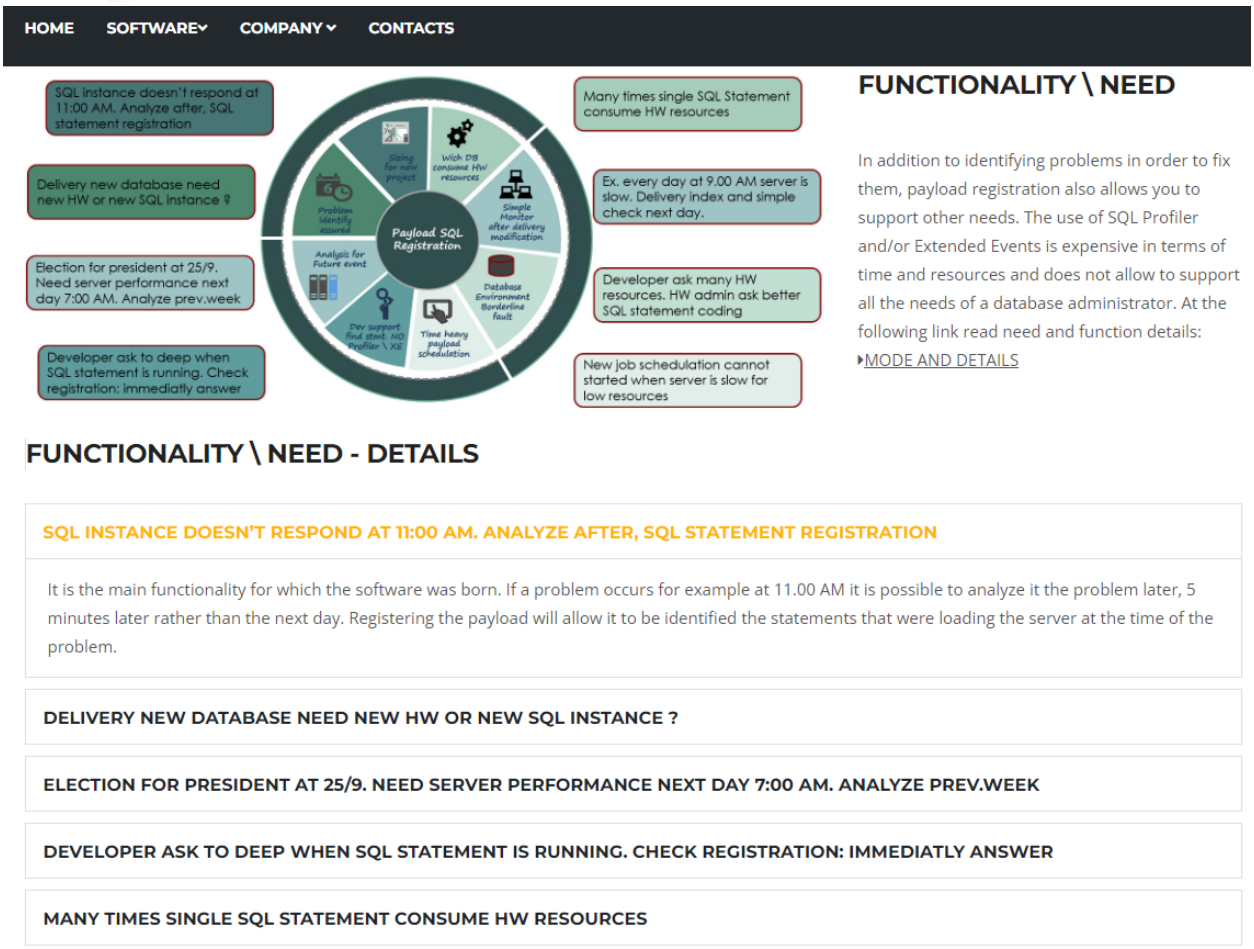



La funzionalità per cui Sql Easy Monitor è più utile, è sicuramente la prima "`SQL instance doesn't respond at 11:00 AM. Analyze after, SQL statement registration`". Essa + ampiamente discussa nei capitoli precedenti.

Ora vediamo le varie funzionalità della figura sopra.

## f. Funzionalità

Il commento di ciascuna funzionalità presente sulla pagina Home Page del sito: https://www.sqleasymonitor.com/Default.aspx




*SQL instance doesn't respond at 11:00 AM. Analyze after, SQL statement registration*

Funzionalità per cui è nato il servizio Sql Easy Monitor. Vedere i capitoli precedenti.

*Delivery new database need new HW or new SQL instance ?*

Se tramite la funzionalità riportata al paragrafo "Comandi più invasivi nelle istanze", si nota che spesso ci sono situazioni di rallentamenti non dovute a blocchi, è meglio fare approfondimenti e tuning dell'istanza prima di fare il delivery di un nuovo database sull'istanza.

*Election for president at 25/9. Need server performance next day 7:00 AM. Analyze prev.week*

Se è necessario che gli utenti abbiano il sistema disponibile in una fascia oraria di lavoro insolita e\o dedicata alla maintenance è bene fare una query nella stessa fascia oraria la settimana precedente e capire quali sono i comandi che girano, da quali macchine e da quali applicazioni vengono chiamati, sfruttando la baseline delle registrazioni precedenti. Questo permette di analizzare in modo semplice il carico dell'istanza SQL Server e capire se ci

possono essere situazioni di blocco in tale fascia oraria causati da maintenance piuttosto che attività batch che in tal caso andranno rischedulate.

```
use SEM
go
select top (5000)
           DtTimeStamp = [TicketIdGroup],total_elapsed_time/1000 as Seconds_From_Start ,[db],[host_name]
           ,[program_name],[blocking_session_id],[cmd]  = replace ( replace([cmd] , char(10), ''), char(13), ' ') --> rem x copy su excel
           ,[cmdStmt] = left(replace ( replace([cmdStmt] , char(10), ''), char(13), ' '),8000) --> rem x copy su excel
           ,[InstanceName],
[wait_type],[last_wait_type],[session_id],[start_time],[reads],[writes],[logical_reads],[row_count],[login_name]
           ,[cpu_time],[total_elapsed_time],[command],[wait_time],[wait_resource],[status],[last_request_start_time],[last_request_end_time
]
           ,[percent_complete],[estimated_completion_time],[scheduler_id],[granted_query_memory], rs_governor_grp
-- select distinct host_name, host_name_New=''
-- into _temp_host_name
from [Svc].[Tb_PrActiveData] with(nolock)
where 1=1
           AND TicketIdGroup between '2023-7-10 07:00' and '2023-7-107 08:00' --> fascia oraria 7/8 del lunedì mattina
order by TicketIdGroup, cmd, cmdStmt
```

*Developer ask to deep when SQL statement is running. Check registration: immediatly answer*

```
use SEM
go
select top (5000)
           DtTimeStamp = [TicketIdGroup],total_elapsed_time/1000 as Seconds_From_Start ,[db],[host_name]
           ,[program_name],[blocking_session_id],[cmd]  = replace ( replace([cmd] , char(10), ''), char(13), ' ') --> rem x copy su excel
           ,[cmdStmt] = left(replace ( replace([cmdStmt] , char(10), ''), char(13), ' '),8000) --> rem x copy su excel
           ,[InstanceName],
[wait_type],[last_wait_type],[session_id],[start_time],[reads],[writes],[logical_reads],[row_count],[login_name]
           ,[cpu_time],[total_elapsed_time],[command],[wait_time],[wait_resource],[status],[last_request_start_time],[last_request_end_time
]
           ,[percent_complete],[estimated_completion_time],[scheduler_id],[granted_query_memory], rs_governor_grp
-- select distinct host_name, host_name_New=''
-- into _temp_host_name
from [Svc].[Tb_PrActiveData] with(nolock)
where 1=1
           AND cmd like 'pp_Get%128%' --> ricerca delle stored Get%128
order by TicketIdGroup, cmd, cmdStmt
```

Con il comando sopra si estraggono tutte le rilevazioni dei comandi "'pp_Get%128%'" ed il risultato mostrerà da quali macchine e programmi viene fatta la chiamata nonché i tempi di esecuzione. Paragonando i tempi di esecuzione su una baseline dei giorni precedenti, si riesce anche a comprendere in modo semplice eventuali degradi di perfromance.

*Many times single SQL Statement consume HW resources*

Tramite la funzionalità riportata al paragrafo "Comandi più invasivi nelle istanze" si individuano le stored ed i comandi all'interno di esse che sono più invasivi. Questo permette di fare tuning aggiungendo eventualmente indici e controllando in modo semplice nei giorni successivi se l'azioni ha avuto successo.


*Ex. every day at 9.00 AM server is slow. Delivery index and simple check next day*

```
use SEM
go
select top (5000)
          DtTimeStamp = [TicketIdGroup],total_elapsed_time/1000 as Seconds_From_Start ,[db],[host_name]
          ,[program_name],[blocking_session_id],[cmd]  = replace ( replace([cmd] , char(10), ''), char(13), ' ') --> rem x copy su excel
          ,[cmdStmt] = left(replace ( replace([cmdStmt] , char(10), ''), char(13), ' '),8000) --> rem x copy su excel
          ,[InstanceName],
[wait_type],[last_wait_type],[session_id],[start_time],[reads],[writes],[logical_reads],[row_count],[login_name]
          ,[cpu_time],[total_elapsed_time],[command],[wait_time],[wait_resource],[status],[last_request_start_time],[last_request_end_time
]
          ,[percent_complete],[estimated_completion_time],[scheduler_id],[granted_query_memory], rs_governor_grp
-- select distinct host_name, host_name_New=''
-- into _temp_host_name
from [Svc].[Tb_PrActiveData] with(nolock)
where 1=1
          AND cmd like 'pp_Get%128%' --> ricerca delle stored Get%128
          AND TicketIdGroup > '2023-7-10 07:00' --> fascia > di quando fatto il tuning
order by TicketIdGroup, cmd, cmdStmt
```

Una volta fatto tuning su un comando è semplice ricercare il comando ad una TimeStamp maggiore di quello in cui è stato fatto il rilascio del tuning.

*Developer ask many HW resources. HW admin ask better SQL statement coding*

```
use SEM
go
select top (5000)
          DtTimeStamp = [TicketIdGroup],total_elapsed_time/1000 as Seconds_From_Start ,[db],[host_name]
          ,[program_name],[blocking_session_id],[cmd]  = replace ( replace([cmd] , char(10), ''), char(13), ' ') --> rem x copy su excel
          ,[cmdStmt] = left(replace ( replace([cmdStmt] , char(10), ''), char(13), ' '),8000) --> rem x copy su excel
          ,[InstanceName],
[wait_type],[last_wait_type],[session_id],[start_time],[reads],[writes],[logical_reads],[row_count],[login_name]
          ,[cpu_time],[total_elapsed_time],[command],[wait_time],[wait_resource],[status],[last_request_start_time],[last_request_end_time
]
          ,[percent_complete],[estimated_completion_time],[scheduler_id],[granted_query_memory], rs_governor_grp
-- select distinct host_name, host_name_New=''
-- into _temp_host_name
from [Svc].[Tb_PrActiveData] with(nolock)
order by TicketIdGroup, cmd, cmdStmt
```

Analizzando il carico ed in particolare le colonne "`[reads],[writes],[logical_reads]`" si dovrebbe comprendere i comandi più invasivi che eventualmente necessitano di tuning e\o revisione del design e\o dell'implementazione. Solo dopo questa fase è sensato capire se il problema è causato da un sottodimensionamento hardware.

*New job schedulation cannot started when server is slow for low resources*

```
use SEM
go
select top (5000)
           DtTimeStamp = [TicketIdGroup],total_elapsed_time/1000 as Seconds_From_Start ,[db],[host_name]
           ,[program_name],[blocking_session_id],[cmd]  = replace ( replace([cmd] , char(10), ''), char(13), ' ') --> rem x copy su excel
           ,[cmdStmt] = left(replace ( replace([cmdStmt] , char(10), ''), char(13), ' '),8000) --> rem x copy su excel
           ,[InstanceName],
[wait_type],[last_wait_type],[session_id],[start_time],[reads],[writes],[logical_reads],[row_count],[login_name]
           ,[cpu_time],[total_elapsed_time],[command],[wait_time],[wait_resource],[status],[last_request_start_time],[last_request_end_time
]
           ,[percent_complete],[estimated_completion_time],[scheduler_id],[granted_query_memory], rs_governor_grp
-- select distinct host_name, host_name_New=''
-- into _temp_host_name
from [Svc].[Tb_PrActiveData] with(nolock)
where 1=1
           AND InstanceName in ('sql04\p04') 00' --> istanza dove si vuole fare la schedulazione
           AND TicketIdGroup > '2023-7-10 07:00' --> fascia > di necessità
order by TicketIdGroup, cmd, cmdStmt
```

Osservando la baseline dei giorni precedenti si capisce quale è l'orario dove arrivano più chiamate all'istanza.