Poor man’s SQL Server job monitoring with R

Thomas Hütter

Poor man’s SQL Server job monitoring with R

Thomas Hütter, Diplom-Betriebswirt

- Application developer, consultant, accidental DBA, author
- Worked at consultancies, ISVs, end user companies
- SQL Server > 6.5, former „Navision“ > 3.0, R > 3.1.2
- Speaker at SQL events around Europe

@DerFredo https://twitter.com/DerFredo

de.linkedin.com/in/derfredo

www.xing.com/profile/Thomas_Huetter

Agenda

- My background and motivation for this session
- The system databases: where to find the relevant data for job monitoring
- Introducing R and the RStudio IDE
- Gathering and preparing our data in R
- Visualizing history and trendlines
- Bringing it all to SQL Server
- Round-up; resources & credits; Q&A

Background and motivation

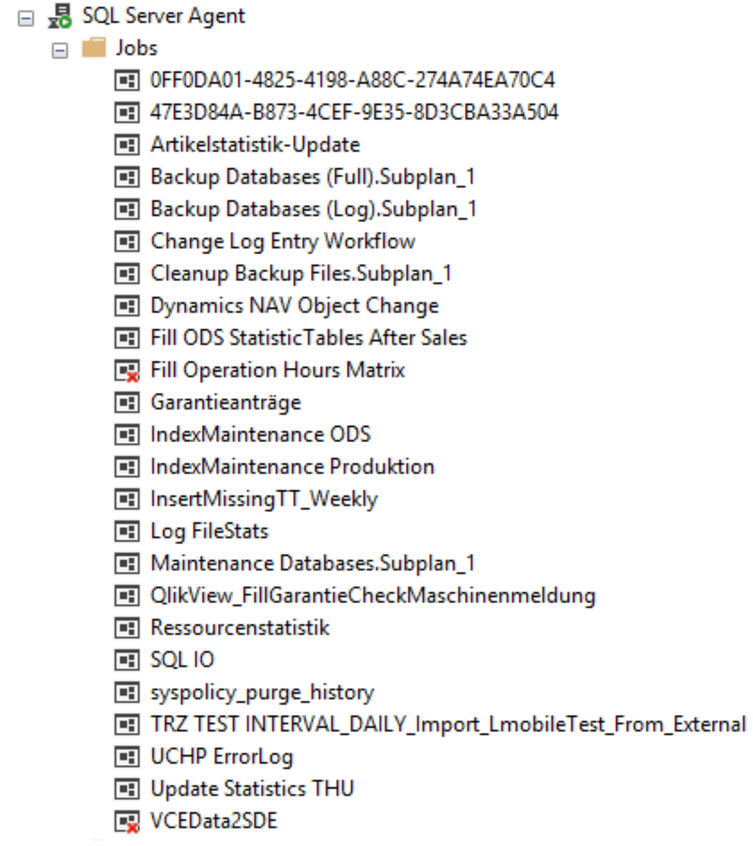
- Tasks as a developer:
  developing software solutions, i.e. modifying D365 Business Central, documenting changes, supporting the users, designing interfaces, developing views, stored procedures, SQL reports, …

- Tasks as an „accidental“ DBA:
  analyze lockings and blockings in the SQL Server databases, optimize query performance, change indexes, update statistics, create and monitor SQL Server agent jobs, …

Background and motivation

SQL Server agent jobs

- can contain one or more action steps, administrative tasks as database maintenance, data manipulation, import, export
- actions can run T-SQL scripts, execute SSIS packages or issue other commands
- can be run according to a schedule, in response to an alert or manually



Background and motivation

Then your agent log may look something like this (for one job),
can you spot at one glance where the job took longer than usual?

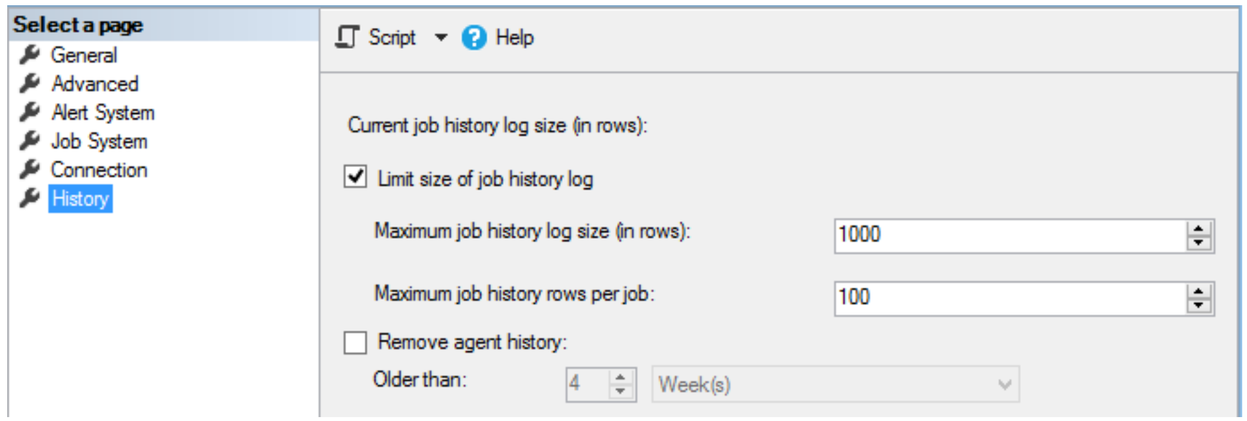
Log file summary: No filter applied

Date	Job Name	Step Name	Notifications	Message	Duration
03.11.2019 15:15:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:00:02
03.11.2019 15:00:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:00:05
03.11.2019 14:45:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:00:05
03.11.2019 14:30:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:00:04
03.11.2019 14:15:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:00:04
03.11.2019 14:00:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:00:04
03.11.2019 13:45:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:00:05
03.11.2019 13:30:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:00:04
03.11.2019 13:15:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:00:06
03.11.2019 13:00:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:00:07
03.11.2019 12:45:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:03:49
03.11.2019 12:30:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:03:30
03.11.2019 12:15:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:04:23
03.11.2019 12:00:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:07:02
03.11.2019 11:45:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:04:01
03.11.2019 11:30:00	Backup Databases (Log).Subplan_1			The job succeeded. The Job was invoked by Schedule 35 (Backup Database Log). The last step to run was step 1 (Subplan_1).	00:03:47

Background and motivation

Job history log defaults

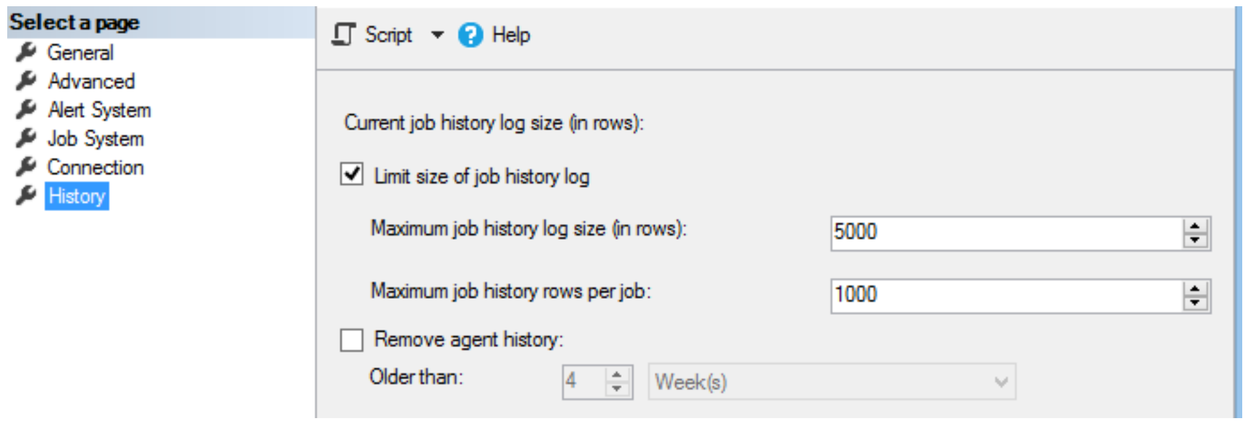
- 1000 rows for all jobs,
  100 for any individual job



- when running a log backup every 15 minutes -> 96 rows per day,
  meaning you can't even compare two whole days

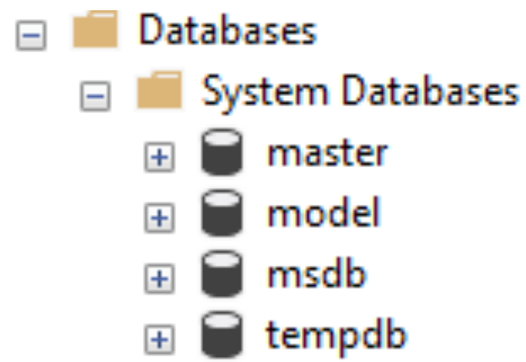
According to your mix of jobs, play around with those agent properties

- 5000 rows for all jobs,
  1000 for any individual job



The system databases: where to find what

- master
  holds system-level information of the instance
- model
  is used as a template for newly created databases
- msdb
  contains all the SQL Server agent information, also used by backup and restore processes, Service Broker, Database Mail
- tempdb
  workspace for temporary objects, e.g. while sorting occurs
  „the public toilet of SQL Server“
- Resource [mssqlsystemresource]
  „hidden“, read-only database contains the sys schema objects



The system databases: where to find what

What's in the msdb databases for us (job related system views):

- `sysjobhistory`
- `sysjobs`
- `sysjobservers`
- `sysjobactivity`
- `sysjobsteps`
- `sysjobstepslogs`
- `sysjobschedules`

The system databases: where to find what

What's in the msdb databases for us (job related system views):

- `sysjobs`
  retrieves the general information about all jobs,
  for us mainly the job name (instead of the job_id GUID)
  is of interest to make it humanly recognizable,
  but also has description, category, owner, several notify properties
- `sysjobhistory`
  has information about each run of a job step (step 0 = job outcome),
  we are interested in the date, time, duration and the run_status
  (0 = failed, 1 = succeeded, 2 = Retry, 3 = Canceled, 4 = In progress)

The system databases: where to find what

So, basically we need something like this:

- `SELECT sj.name, sh.run_status, sh.run_date, sh.run_time, sh.run_duration`

```
FROM sysjobs sj
JOIN sysjobhistory sh
ON sh.job_id = sj.job_id

WHERE sh.step_id = 0

ORDER BY sj.name, sh.run_date, sh.run_time
```

Introducing R and the RStudio IDE

- R is a programming language for statistical computing, analysis and visualization, widely used by statisticians, data miners, data scientists
- Created in 1993, GNU project, for MacOS, Linux, Windows, extensible through user-created packages (> 16000 available on CRAN mirror)
- open source, commercial support e.g. since 2007 by Revolution Analytics, acquired by Microsoft in 2015, R in SQL Server, Power BI, Azure ML

- RStudio = the de-facto standard for R IDEs
- R Tools for Visual Studio deprecated from VS 2019 on

Gathering and preparing our data in R

- Call all the packages we need

- Create a connection to the SQL Server instance

- Run a query to retrieve the data we need and store it in a tibble

- Modify date and time columns, add colour coding

Visualizing history and trendlines

- Call a plot command to display our data

- Generate a faceted plot with free scales

- Add the colouring for the run_status outcome

- Add trend lines, following a linear model or others

Bringing it all to SQL Server

- Configure external scripts to be enabled
- Construct the call to `sp_call_external_script`
- Use the SQL query for your input data set
- Make sure to mask single quotes within the quoted R script
- Call `png()`, encapsulate ggplot call with `print()`, call `dev.off()`

Round-up

- (Accidental) DBA's tasks include monitoring agent jobs
- Better to have a graphical overview than to follow endless columns of nos
- Combining SQL and R leads to a useful first result relatively fast
- RStudio is better suited for developing and debugging than SSMS
- Our solution should run on Windows & Mac & Linux,
  in RStudio & SQL Server Management Studio & Azure Data Studio
  and requires no extra $$$/~~€€€~~ to be invested

Resources on- and offline, credits

- System databases: https://docs.microsoft.com/de-de/sql/relational-databases/databases/system-databases
- SQL Server agent tables in msdb: https://docs.microsoft.com/de-de/sql/relational-databases/system-tables/sql-server-agent-tables-transact-sql
- R download and packages from https://cran.r-project.org/index.html
- RStudio IDE and more (see „products“ or „download"): https://rstudio.com/
- „What is SQL Server Machine Learning Services“, includes Quickstarts and Tutorials on R and Python for SQL developers: https://docs.microsoft.com/en-us/sql/advanced-analytics/what-is-sql-server-machine-learning

Poor man’s SQL Server job monitoring with R

Thank you for your time and interest & keep in touch:

- @DerFredo https://twitter.com/DerFredo
- de.linkedin.com/in/derfredo
- www.xing.com/profile/Thomas_Huetter

This file and all demo scripts can be found at:
https://github.com/SQLThomas/Conferences/tree/master/Erding2020

Poor man’s SQL Server job monitoring with R

Time for some Q & A?