Auftragshistorie für Datenaufbereitungen mit DeltaMaster visualisieren

Autor	Sigrid Kauter-Miersch
Datum der Veröffentlichung	01.12.2017

Bissantz & Company GmbH
Nordring 98 90409 Nürnberg
T +49 911 935536 - 0
F +49 911 935536 - 10
service@bissantz.de

Geschäftsführer
Dr. Nicolas Bissantz
Michael Westphal

USt-IdNr. DE181064258
Steuer-Nr. 241/118/25048
Amtsgericht Nürnberg
HRB 15590

Sparkasse Nürnberg
IBAN DE88 7605 0101 0570 1792 91
BIC SSKNDE77XXX

www.bissantz.de

Abstract

Die Auftragshistorie von Datenaufbereitungen mittels SQL-Server-Agent-Aufträgen ist im Allgemeinen nur im SQL-Server Management-Studio sichtbar. Damit bleiben zeitliche Veränderungen bei der Auftragsausführung häufig unbemerkt und unerkannt. Dieser Blog zeigt, wie man diese Daten mit DeltaMaster visualisieren und so Veränderungen bei der Auftragsausführung schnell erkennen kann um ggf. frühzeitig einer weiteren Verschlechterung der Performance entgegenzuwirken.

Auftragshistorie für Datenaufbereitungen mit DeltaMaster visualisieren

Sind die Datenimporte erstellt, das Modell aufgebaut, werden Datenimport und -aufbereitung meist automatisiert. Dies geschieht häufig als Auftrag (Job) in dem von SQL-Server zur Verfügung gestellten SQL-Server-Agent. Die einzelnen Aufträge, die im SQL Server-Agent eingestellt sind, werden zu definierten Zeiten automatisch gestartet und alle Schritte eines Auftrags nacheinander abgearbeitet. Das ist bekannt und funktioniert seit vielen Jahren zuverlässig. Für den Fall, dass bei der Abarbeitung des Auftrags ein Fehler auftritt, können ein oder mehrere E-Mail-Empfänger hinterlegt werden, sodass im Fehlerfall reagiert werden kann.

Ob ein Auftrag im Laufe der Monate und Jahre immer länger läuft, weil das Datenvolumen mit jedem Datenimport größer geworden oder der Server durch zusätzliche Aufgaben an den Rand seiner Leistungsfähigkeit geraten ist, behält kaum jemand im Auge. Erst wenn die Daten nicht mehr zum gewohnten Zeitpunkt zur Verfügung stehen oder sich Aufträge gegenseitig behindern und Fehler produzieren, fällt es auf. Dann ist schneller Rat gefragt.

Dabei ließe sich dies mit den Daten der Aufträge, die in den Tabellen der Systemdatenbank msdb gespeichert werden, sehr gut beobachten.

In meinem Blog: „SQL Server-Agent Auftragsdetails mit SQL ermitteln“ habe ich beschrieben, welche Tabellen welche Informationen enthalten und wie man diese extrahieren kann.

Mit diesem Blog soll gezeigt werden, wie diese Daten historisiert, aufbereitet und in einer DeltaMaster-Analysesitzung in gewohnter Übersichtlichkeit dargestellt werden können.

1 Auftragsdaten historisieren

In den Tabellen der Systemdatenbank msdb werden die Daten der Auftragsausführung gespeichert, aber nicht über einen längeren Zeitraum historisiert. Für das Speichern der historischen Daten muss eine Tabelle in einer vorhandenen oder auch einer neuen Datenbank angelegt werden sowie eine Prozedur erstellt werden, die diese Daten aus den Tabellen der msdb-Datenbank in die neue Tabelle übernimmt.

Eine solche Tabelle kann z.B. mit folgendem SQL-Statement erstellt werden:

```
CREATE TABLE [dbo].[T_S_Auftragshistorie](
        [job_id] [uniqueidentifier] NOT NULL,
        [job_name] [nvarchar](150) NOT NULL,
        [Job_Beschreibung] [nvarchar](512) NULL,
        [Datum_Ausführung] [date] NULL,
        [Zeit_Ausführung] [varchar](8) NULL,
        [Dauer_Ausführung] [varchar](8) NULL,
        [Status_Ausführung] [smallint] NULL,
        [Fehlermeldung] [nvarchar](4000) NULL,
        [Fehler_bei_Schritt] [nvarchar](128) NULL

) ON [PRIMARY]
```

Das Skript für eine Übernahmeprozedur könnte dann wie folgt aussehen:

```
CREATE PROCEDURE [dbo].[P_Insert_T_S_Auftragshistorie]
(
        @Auftrag varchar(250)
)
AS
BEGIN
INSERT INTO T_S_Auftragshistorie
SELECT
  Job.job_id
  , Job.name
  , Job.description
  , CONVERT(date,CAST(JobHist.run_date as varchar(8)))
  , STUFF(STUFF(RIGHT('000000' + CAST(JobHist.run_time AS VARCHAR(6)), 6), 3, 0, ':'), 6, 0, ':')
  , STUFF(STUFF(RIGHT('000000' + CAST(JobHist.run_duration AS VARCHAR(6)), 6), 3, 0, ':'), 6, 0, ':')
  , JobHist.run_status
  , CASE WHEN JobHist.run_status = 0 THEN JobHist2.message ELSE NULL END
  , CASE WHEN JobHist.run_status = 0 THEN JobHist2.step_name ELSE NULL END

FROM
  msdb.dbo.sysjobs AS Job
  LEFT JOIN (
          SELECT
            job_id
            , step_id
            , step_name
            , run_date
            , run_time
            , run_status
            , run_duration
            , message
            , ROW_NUMBER() OVER (PARTITION BY job_id ORDER BY run_date DESC, run_time DESC) AS RowNumber
          FROM msdb.dbo.sysjobhistory
          WHERE step_id = 0
-- nur im Step = 0 stehen das Ausführungsergebnis des gesamten Jobs; in den weiteren Step werden die Teil-
schritte aufgelistet
             ) AS JobHist
          ON Job.job_id = JobHist.job_id
          AND JobHist.RowNumber = 1
-- Fehlermeldung selektieren, wenn Job auf Fehler gelaufen ist
-- run_status = 0
  LEFT JOIN (
            SELECT
              job_id
              , step_id
              , step_name
              , run_date
              , run_time
              , message
              , ROW_NUMBER() OVER (PARTITION BY job_id ORDER BY run_date DESC, run_time DESC) AS RowNumber
            FROM msdb.dbo.sysjobhistory
            WHERE step_id > 0
            AND run_status = 0
            ) AS JobHist2
            ON JobHist.job_id = JobHist2.job_id AND JobHist.RowNumber = JobHist2.RowNumber
WHERE Job.name = @Auftrag
END
GO
```

Die Prozedur wird im Anschluss als letzter Schritt in den Auftrag eingefügt, dessen Ausführungsdaten gespeichert werden sollen:

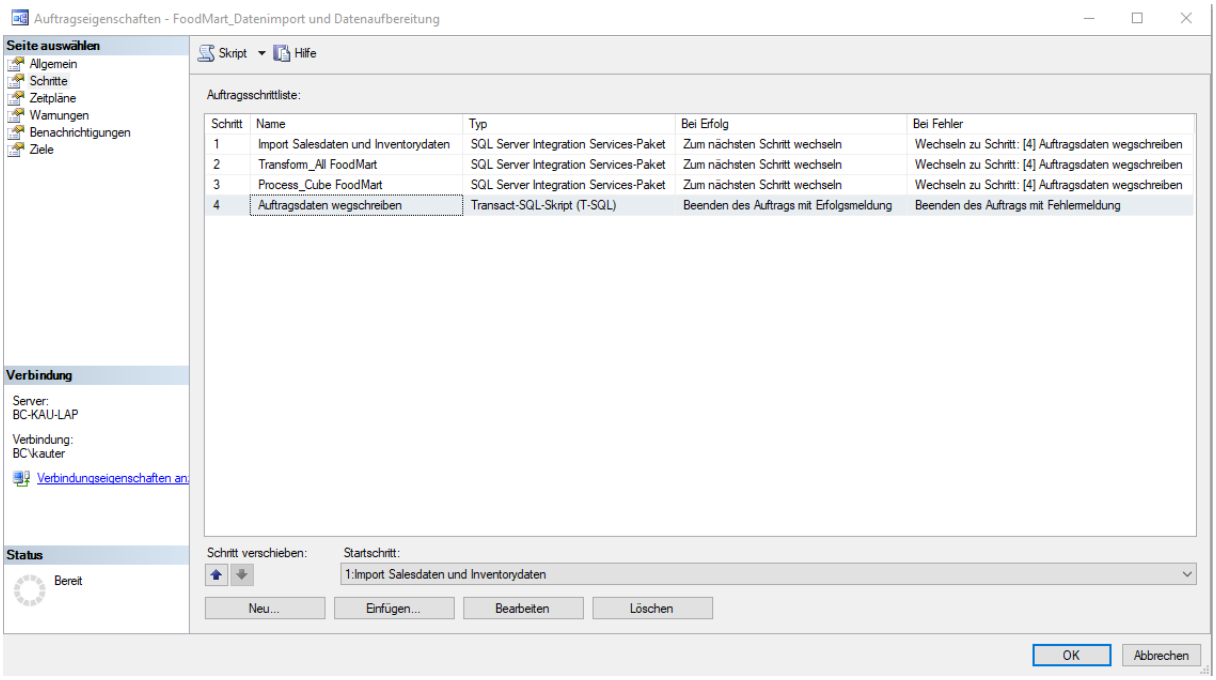

Abbildung 1 Auftragsschritte ergänzen

Sollen auch Fehler beim Datenimport oder der Datenaufbereitung dokumentiert werden, ist zu beachten, dass bei jedem Schritt (Schritt 1-3 in Abbildung 1) in der Spalte „Bei Fehler" der neu hinzugefügte Schritt (hier Schritt 4 „Auftragsdaten wegschreiben") eingetragen wird.

Der Schritt 4 „Auftragsdaten wegschreiben" enthält den Aufruf der oben aufgeführten Prozedur „P_Insert_T_S_Auftragshistorie":

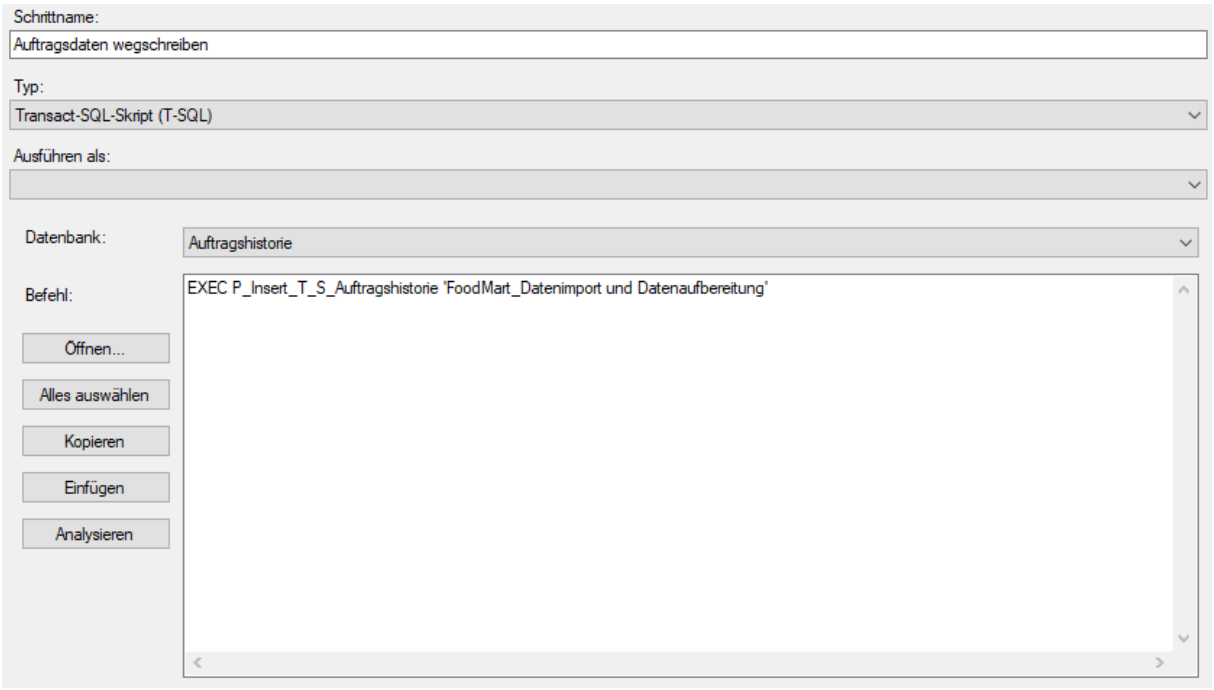

Abbildung 2 Schritt Auftragsdaten wegschreiben

Als Parameter wird der frei wählbare Name des Auftrags an die Prozedur übergeben. Durch die Angabe und Speicherung eines Auftragsnamens in der Tabelle T_S_Auftragshistorie ist es möglich in der gleichen Tabelle die Daten mehrere zu überwachender Jobs zu speichern.

Nun werden bei jedem Aufruf des Auftrags Daten gesammelt und die Tabelle T_S_Auftragshistorie füllt sich.

2 Darstellung im DeltaMaster

Ein kleines OLAP-Modell ist schnell erstellt und soll hier nicht Gegenstand des Blogs sein. Selbstverständlich kann man die Tabelle T_S_Auftragshistorie auch im Selfservice als Tabelle laden und modellieren.

Für alle, die das vorliegende Beispiel nachvollziehen möchten, befinden sich im Anhang die Datenbanksicherungen der relationalen Datenbank und der OLAP-Datenbank sowie die DeltaMaster-Analysesitzung.

Entscheidender ist, dass die Daten nun im DeltaMaster vorliegen und der zeitliche Verlauf der Auftragsausführung dargestellt und einem entsprechenden Nutzerkreis zugänglich gemacht werden kann.

Für die OLAP-Modellierung wurden für das hier gezeigte Beispiel folgende Dimensionstabellen angelegt:

Eine Tabelle T_S_Status mit den Status_Ids und Bezeichnungen:

Status_ID	Status_Name
0	Fehler
1	Erfolgreich
2	Wiederholen
3	Abgebrochen
4	Wird ausgeführt

Eine entsprechende View dazu:

```
CREATE VIEW [dbo].[V_Import_Dim_Status]
AS
SELECT DISTINCT
        Status_ID,
        Status_Name
FROM T_S_Status
```

Eine View mit den Jobnamen und der Jobbeschreibung:

```
CREATE VIEW [dbo].[V_Import_Dim_Job]
AS
SELECT DISTINCT
        job_name,
        Job_Beschreibung
FROM T_S_Auftragshistorie
```

Eine View für die Periodendimension:

```
CREATE VIEW V_Import_Dim_Periode
AS
SELECT
        Periode
FROM T_Import_Periode_manuell
```

Eine Tabelle mit den Perioden auf Tagesbasis befindet sich meist in einer bereits bestehenden Datenbank und kann hier anstelle der T_Import_Periode_Manuell als Basistabelle verwendet werden.

Sowie eine View als Faktentabelle :

```
CREATE VIEW V_Import_Fact_Auftragshistorie
AS
SELECT
        job_name,
        Datum_Ausführung,
        Dauer_Ausführung,
        CAST(LEFT(Dauer_Ausführung,2) as int) * 360 + CAST(SUBSTRING(Dauer_Ausführung,4,2) as int) * 60 +
CAST(RIGHT(Dauer_Ausführung,2) as int) AS Dauer_in_Sekunden,
        Status_Ausführung Status_Ausführung_ID,
        Fehler_bei_Schritt,
        Fehlermeldung
FROM T_S_Auftragshistorie
```

Hier ist anzumerken, dass in der Tabelle T_S_Auftragshistorie die Dauer der Ausführung gut lesbar in der Form „00:00:00“ gespeichert wird. Um die Ausführungsdauer als Kennzahl verwenden zu können, muss diese umgerechnet werden. In meinem Beispiel erfolgt die Umrechnung in Sekunden. Bei größeren Modellen muss hier möglicherweise eine größere Zeiteinheit gewählt werden. Dann ist die Umrechnung entsprechend anzupassen.

Alle erforderlichen SQL-Statements befinden sich im Anhang zu diesem Blog in der Datei: „Auftragshistorie.sql“.

In der Analysesitzung könnte die Auswertung zum Beispiel als Zeitreihe erfolgen:

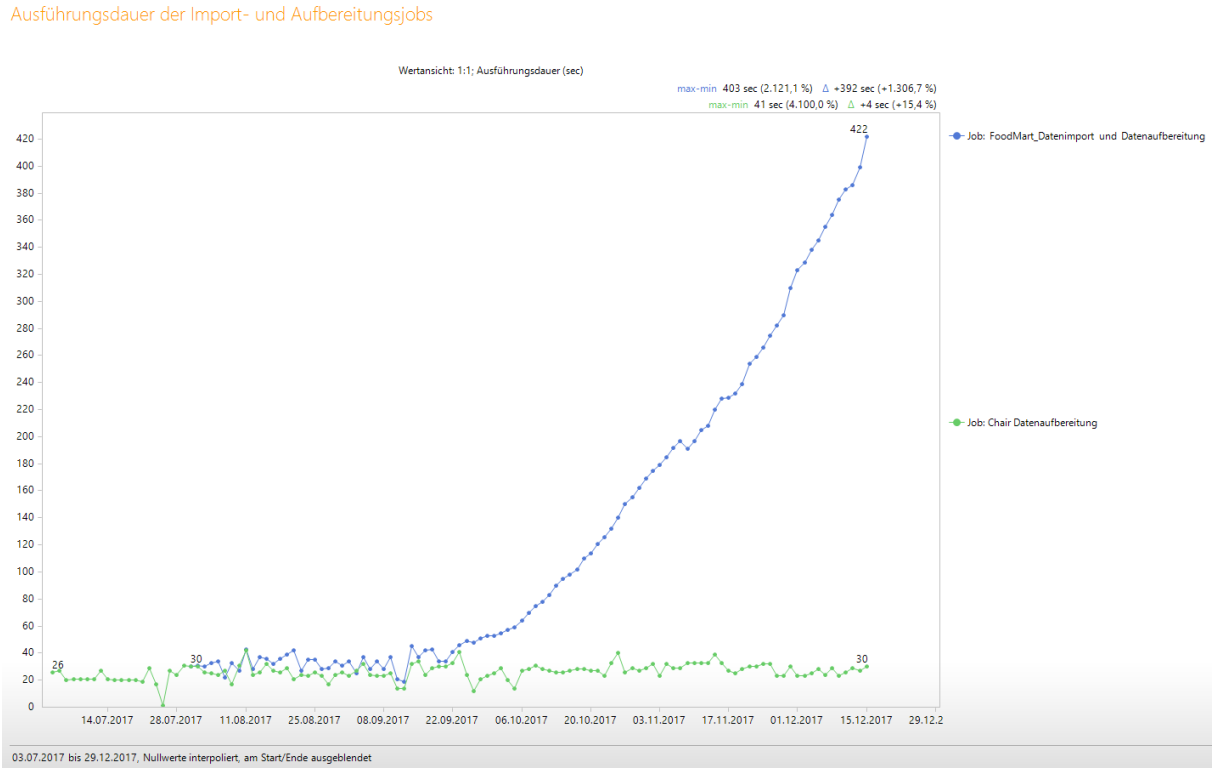

Abbildung 3 Ausführungsdauer als Zeitreihe visualisiert

In dieser Darstellung ist auf einen Blick ersichtlich, dass bei dem Auftrag „FoodMart_Datenimport und Datenaufbereitung“ (blaue Zeitreihe) die Zeiten deutlich aus dem Ruder laufen. Auch wenn es sich hier im Beispiel nur um Sekunden bzw. wenige Minuten handelt, ist ein solcher Verlauf ein deutliches Warnsignal und Handeln erforderlich. Dagegen ist der Verlauf der „Chair Datenaufbereitung“ (grüne Zeitreihe) beruhigend unauffällig. Warum die Zeiten der Datenaufbereitung so deutlich angestiegen sind, kann verschieden Ursachen haben, die aus den Daten der Systemdatenbank msdb nicht hervorgehen. Die gilt es herauszufinden. In dem im Beispiel gezeigten Fall wurde der Anstieg der Ausführungszeiten durch den Import immer größerer Datenmengen und zusätzlich komplexen Datenaufbereitungen gezielt herbeigeführt.

Gründe, die zu einer Verschlechterung der Performance führen können, sind vielfältig und können im Server selbst begründet sein, in zusätzlichen Datenimporten inklusive komplexen Datenaufbereitungen, in der Zunahme des Datenvolumens, Überschneidungen von verschiedenen Aufbereitungssaufträgen uvm. Das muss individuell ermittelt werden.

Nicht nur die Zeitreihe zeigt den Verlauf deutlich auch Sparklines sind dafür gut geeignet:

Abbildung 4 Ausführungsdauer mit Sparklines visualisiert

,

Ebenso visualisieren Bissantz'Numbers den Verlauf eindrucksvoll:

Ausführungsdauer (sec) Job	Periode_KW KW38 2017	KW39 2017	KW40 2017	KW41 2017	KW42 2017	KW43 2017	KW44 2017	KW45 2017	KW46 2017	KW47 2017	KW48 2017
FoodMart_Datenimport und Datenaufbereitung	194	247	288	396	519	669	840	962	1.090	1.250	1.480

Abbildung 5 Ausführungsdauer mit Bissantz'Numbers visualisiert

Oder eine Variante mit Deltaberechnung und Wetterzelle:

Δ Ausführungsdauer im Zeitraum 01.12.2017 ./. 24.11.2017: 57 Δ-Bericht

Ausführungsdauer (sec)	24.11.2017	27.11.2017	28.11.2017	29.11.2017	30.11.2017	01.12.2017	01.12.2017 ./. 24.11.2017
FoodMart_Datenimport und Datenaufbereitung	266	275	282	290	310	323	57

Abbildung 6 Differenz der Ausführungsdauer als Wetterzelle

3 Fazit

Die Historisierung der Verlaufsdaten des entsprechenden SQL-Server-Agent-Auftrags und die anschließende Darstellung im DeltaMaster eignet sich sehr gut dazu, den zeitlichen Verlauf und die Entwicklung der Ausführungszeiten von Datenimporten und -aufbereitungen im Auge zu behalten ohne selbst über einen Zugriff auf das SQL-Server-Managementstudio und damit auf den SQL-Server-Agent zu verfügen.

Die Umsetzung wie sie im Artikel beschrieben ist, ist denkbar einfach, der Nutzen ggf. groß, da Trends in der Auftragsausführung frühzeitig erkannt und Gegenmaßnahmen eingeleitet werden können, bevor Daten nicht mehr zum gewohnten Zeitpunkt zur Verfügung stehen oder die Aufbereitung sogar auf einen Fehler läuft.