BISSANTZ

Der kleine Process zwischendurch

Autor	Bernd Werther
Datum der Veröffentlichung	18.01.2019

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

Dieser Beitrag zeigt Möglichkeiten nur Teile einer Analysis Services Datenbank zu verarbeiten. Das kann in sehr großen Systemen, in denen sich im zugrundeliegenden Snowflake Schema nur bestimmte Measuregroups ändern, eine große Zeitersparnis darstellen. Dabei gibt es jedoch einige Tücken, die zu beachten sind.

Der kleine Process zwischendurch

In sehr großen Systemen mit vielen Measuregroups und großen Datenmengen kann es vorkommen, dass die Verarbeitung der kompletten OLAP-Datenbank viel Zeit benötigt. Gerade wenn relational nur kleine Teile des Systems aktualisiert werden (z. B. eine einzelne Measuregroup über einen separaten Transformationsjob, der kein vollständiges Transform-All von DeltaMaster ETL ausführt) ist es ärgerlich, wenn die Gesamtzeit dieser Verarbeitung durch die vollständige Verarbeitung der OLAP-Datenbank erheblich erhöht wird. Ein inkrementelles Verarbeiten der OLAP-Datenbank wäre hier sehr hilfreich.

Tatsächlich bietet Analysis Services im Verarbeitungsdialog für Dimensionen und Measuregroups zahlreiche Einstellungsmöglichkeiten, die auf den ersten Blick sehr vielversprechend klingen. In der Praxis zeigt sich jedoch, dass die meisten dieser Einstellungen entweder wirkungslos sind, oder im schlimmsten Fall sogar die Verfügbarkeit der OLAP-Datenbank verhindern!

Im folgenden Beitrag sollen die einzelnen Optionen erläutert werden und es wird ein Weg beschrieben, der eine inkrementelle Verarbeitung ermöglicht. Dabei wird ein mit DeltaMaster ETL erstelltes Datenmodell vorausgesetzt.

1 Aktualisierung von Dimensionen

Die Hauptproblematik beim inkrementellen Verarbeiten einer OLAP-Datenbank liegt in der Änderung von Dimensionsdaten.

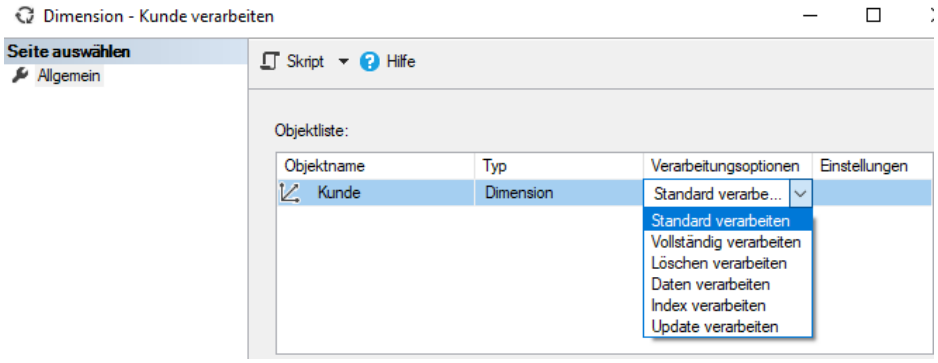

Abbildung 1 Verarbeitungsoptionen bei Dimensionen (im Dialog des Management Studios)

Zunächst müssen wir uns überlegen, welche Fälle relational auftreten können. Diese Fälle müssen entsprechend berücksichtigt werden.

- Es kommen neue Elemente hinzu
- Attribute oder die Bezeichnungen von Elementen ändern sich
- Der Parent von Elementen ändert sich
- Es fallen Elemente weg (in einem ETL-Modell eher unwahrscheinlich, weil die Ausführung einer *P_DIM*-Prozedur lediglich ergänzt, jedoch keine Elemente entfernt. Daher wird dieser Fall hier nur der Vollständigkeit halber erwähnt).

Im Folgenden werden diese vier Fälle für die oben gezeigten Verarbeitungsoptionen durchgespielt und die Auswirkungen betrachtet.

1.1 Hinzufügen eines neuen Elements

Option „*Standard verarbeiten*"

In der Kundendimension wird ein Element ergänzt. Die Ausführung der Option läuft erfolgreich, jedoch taucht das Element anschließend nicht in der Dimension auf. Auch der Zeitstempel der letzten Verarbeitung der Dimension ändert sich nicht. Das System bleibt jedoch erreichbar.

Auch das Hinzufügen eines Elements auf einer höheren Ebene ändert nichts.

Das deckt sich auch mit der Dokumentation von Microsoft. Diese besagt, dass mit der Option „Standard verarbeiten" Objekte, die nicht oder nicht vollständig verarbeitet sind, in den Status vollständig verarbeitet versetzt werden. Ist das Objekt bereits im Status „vollständig verarbeitet", wird nichts passieren.

Option „*Vollständig verarbeiten*"

Wie zuvor wird erneut ein Element ergänzt und anschließend die Option ausgeführt. Die Auswirkungen sind in diesem Fall fatal. Zunächst wird beim Ausführen angezeigt, dass auch die an die Dimension angebundenen Measuregroups vollständig verarbeitet werden. Das kostet unter Umständen viel Zeit und führt somit nicht zum gewünschten Effekt. Noch problematischer: beim Versuch nach der Verarbeitung auf die OLAP-Datenbank mit DeltaMaster zuzugreifen ist der Cube verschwunden und kann nicht mehr genutzt werden. Interessant ist, dass bei der Ausführung dieser Option kein Fehler gemeldet wird. Von dieser Option sollte also in jedem Fall Abstand genommen werden!

Option „*Löschen verarbeiten*"

Diese Option dürfte fachlich kaum Sinn machen und führt zudem zu denselben Problemen wie „Vollständig verarbeiten".

Option „*Daten verarbeiten*"

Auch hier treten wieder die vorherigen Probleme auf.

Option „*Index verarbeiten*"

Die Option verursacht zwar keine Fehler, jedoch taucht das neue Element auch nicht in der Dimension auf. Laut Dokumentation werden bei Ausführung dieser Option Indizes und Aggregationen neu erstellt. Somit ist diese Option für den gegebenen Anwendungsfall auch nicht hilfreich.

Option „*Update verarbeiten*"

Die Option ergänzt das neue Element in der Dimension. Allerdings werden auch alle angebundenen Measuregroups mitverarbeitet, was für unser Szenario eigentlich nicht gewünscht ist. Der Zeitstempel der letzten Verarbeitung bleibt jedoch für die betroffenen Measuregroups unverändert.

Option „*Hinzufügung verarbeiten*"

In der Dokumentation von Microsoft wird auf eine weitere Option hingewiesen, die jedoch nur per XMLA-Skript und nicht über die GUI ausgeführt werden kann: „Hinzufügung verarbeiten". Dafür muss im XMLA-Tag <*Type*> die Option **ProcessAdd** angegeben werden.

Der XMLA-Code für unser Beispiel sieht dann wie folgt aus:

```
<Batch xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
  <Parallel>
    <Process xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-in-
stance" xmlns:ddl2="http://schemas.microsoft.com/analysisservices/2003/engine/2"
xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/engine/2/2"
xmlns:ddl100_100="http://schemas.microsoft.com/analysisservices/2008/engine/100/100"
```

```
xmlns:ddl200="http://schemas.microsoft.com/analysisservices/2010/engine/200"
xmlns:ddl200_200="http://schemas.microsoft.com/analysisservices/2010/engine/200/200"
xmlns:ddl300="http://schemas.microsoft.com/analysisservices/2011/engine/300"
xmlns:ddl300_300="http://schemas.microsoft.com/analysisservices/2011/engine/300/300"
xmlns:ddl400="http://schemas.microsoft.com/analysisservices/2012/engine/400"
xmlns:ddl400_400="http://schemas.microsoft.com/analysisservices/2012/engine/400/400"
xmlns:ddl500="http://schemas.microsoft.com/analysisservices/2013/engine/500"
xmlns:ddl500_500="http://schemas.microsoft.com/analysisservices/2013/engine/500/500">
      <Object>
        <DatabaseID>FoodMart_plus_PROCESS</DatabaseID>
        <DimensionID>Kunde</DimensionID>
      </Object>
      <Type>ProcessAdd</Type>
      <WriteBackTableCreation>UseExisting</WriteBackTableCreation>
    </Process>
  </Parallel>
</Batch>
```

Im Ergebnis wird das neue Element in der Dimension korrekt ergänzt.

1.2 Änderungen bei Attributen und Bezeichnung

Insbesondere wenn die „Update by Key"-Eigenschaft in DeltaMaster ETL für die zu verarbeitende Dimension aktiviert ist, kann es zu Änderungen von Attributen oder Bezeichnungen bei bestehenden Dimensionselementen kommen.
Gemäß den Testergebnissen aus 1.1 können wir bereits einige Verarbeitungsoptionen ausschließen, da sie dieses Testszenario nicht korrekt umsetzen konnten bzw. sogar zu erheblichen Fehlern geführt haben. Erfolgreich waren lediglich die Optionen „Update verarbeiten" und „Hinzufügung verarbeiten".

Option „*Update verarbeiten*"

Nach Änderung der Element-Bezeichnung (oder eines Attributs) kommt es bei der Durchführung dieser Option zu folgendem Fehler:

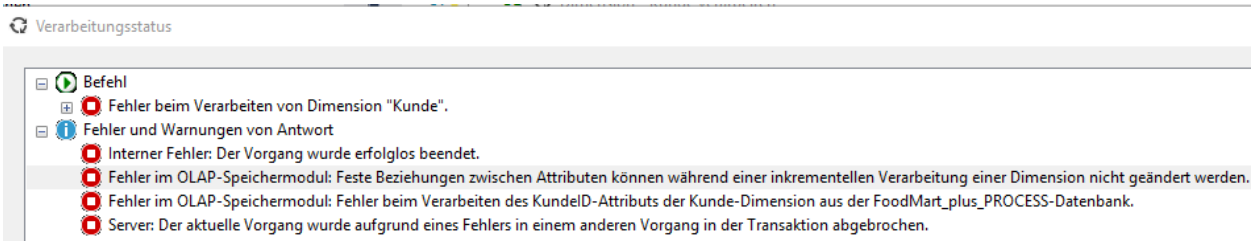

Abbildung 2 Fehler bei „Update verarbeiten"

Immerhin bleibt die OLAP-Datenbank erreichbar, sodass der Endanwender nichts von diesem Fehler mitbekommt. Die Option selbst scheidet jedoch aufgrund des Fehlers aus.

Option „*Hinzufügung verarbeiten*"

Wie der Name dieser Option schon sagt, werden hier nur neue Elemente ergänzt, bestehende aber nicht geändert. Folglich wird das geänderte Element in der OLAP-Datenbank nicht aktualisiert. Die Option läuft jedoch ohne Fehler durch und der Würfel bleibt erreichbar.

1.3 Der Parent von Elementen ändert sich

Die bisherigen Tests lassen uns lediglich eine Verarbeitungsoption offen, um die Verarbeitung zumindest erfolgreich durchführen zu können: „*Hinzufügung verarbeiten*"

Allerdings bekommt diese Option Änderungen des Parent-Elements ebenso wenig mit, wie die zuvor betrachteten Änderungen bei Attributen und Bezeichnungen.

Der folgende Fall kann also nicht inkrementell aktualisiert werden:

Dimension Kunde:

Ebene 1: *Land*
Ebene 2: *Stadt*
Ebene 3: *Kunde*

Der Kunde „Max Mustermann" zieht von „Hamburg" nach „Nürnberg". Entsprechend wird der Kunde relational von Hamburg nach Nürnberg umgemappt. In der OLAP-Datenbank kann diese Änderung erst nach einer vollständigen Verarbeitung der OLAP-Datenbank beobachtet werden.

2 Aktualisierung von Measuregroups

Im folgenden Abschnitt betrachten wir den Fall, dass es keine Änderungen an den Dimensionsdaten gegeben hat und lediglich eine Faktentabelle neu befüllt wurde („Truncate Table" und Ausführung der entsprechenden *P_FACT*-Prozedur).

Option „*Standard verarbeiten*"

Wie schon bei den Dimensionen findet hier keine Aktualisierung statt, weil die Measuregroup bereits im Zustand „vollständig verarbeitet" ist.

Option „*Vollständig verarbeiten*"

Dies ist in der Benutzeroberfläche standardmäßig ausgewählt. Die Option führt zum Erfolg, d. h. die Werte kommen im DeltaMaster an.

Option „*Daten verarbeiten*"

Auch bei dieser Option kommen die Werte im DeltaMaster an, allerdings werden keine Aggregationen und Indizes erstellt. Deshalb ist von dieser Option eher abzuraten.

Option „*Löschen verarbeiten*"

Nach Ausführung dieser Option ist der Cube nicht mehr erreichbar. Zudem klingt diese Variante auch fachlich wenig sinnvoll.

Option „*Hinzufügung verarbeiten*"

Diese Variante ist am kompliziertesten zu konfigurieren. Es muss in den Einstellungen nochmals die (oder eine weitere) Quelltabelle angegeben werden. Neue Datensätze werden dann erfolgreich hinzugefügt, jedoch findet kein Löschen bzw. keine Aktualisierung statt. Somit ist diese Variante in der Praxis nicht brauchbar.

Option „*Index verarbeiten*"

Es findet – wie der Name der Option schon vermuten lässt – keine Aktualisierung der Werte statt.

Somit ist für die Aktualisierung einzelner Measuregroups in jedem Fall zur Option „Vollständig verarbeiten" zu raten.

3 Aktualisierung Measuregroups bei geänderten Dimensionsdaten

In der Praxis wird wohl das folgende Szenario das am häufigsten benötigte sein:

1. Relationale Aktualisierung der benötigten Dimensionen (abhängig von den in Schritt 2 neu zu ladenden Measuregroups oder Partitionen).

2. Relationale Aktualisierung der benötigten Measuregroups oder Partitionen (Truncate Table und Ausführung der entsprechenden *P_FACT*-Prozedur).

3. Inkrementelle OLAP-Verarbeitung

In den Dimensionstabellen können also die folgenden Fälle auftreten:

- Es kommen neue Elemente hinzu
- Attribute oder Bezeichnungen bestehender Elemente ändern sich (wenn „Update by Key"-Eigenschaft in DeltaMaster ETL aktiv ist)
- Der Parent bestehender Elemente ändert sich (wenn „Update by Key"-Eigenschaft in Delta-Master ETL aktiv ist)

Die betroffenen Faktentabellen wurden vollständig geleert und neu befüllt.

Für die OLAP-Verarbeitung der Fakten bietet sich somit eigentlich nur die Option „Vollständig verarbeiten" an (vgl. 2).

Führt man diese jedoch aus, ohne sich zuvor um die neu hinzugekommenen Dimensionselemente gekümmert zu haben, so kommt es zum Fehler während der Verarbeitung (ähnlich einem Foreign-Key-Fehler im Snowflake Schema), sofern diese neuen Elemente auch in der entsprechenden Measuregroup verwendet werden.

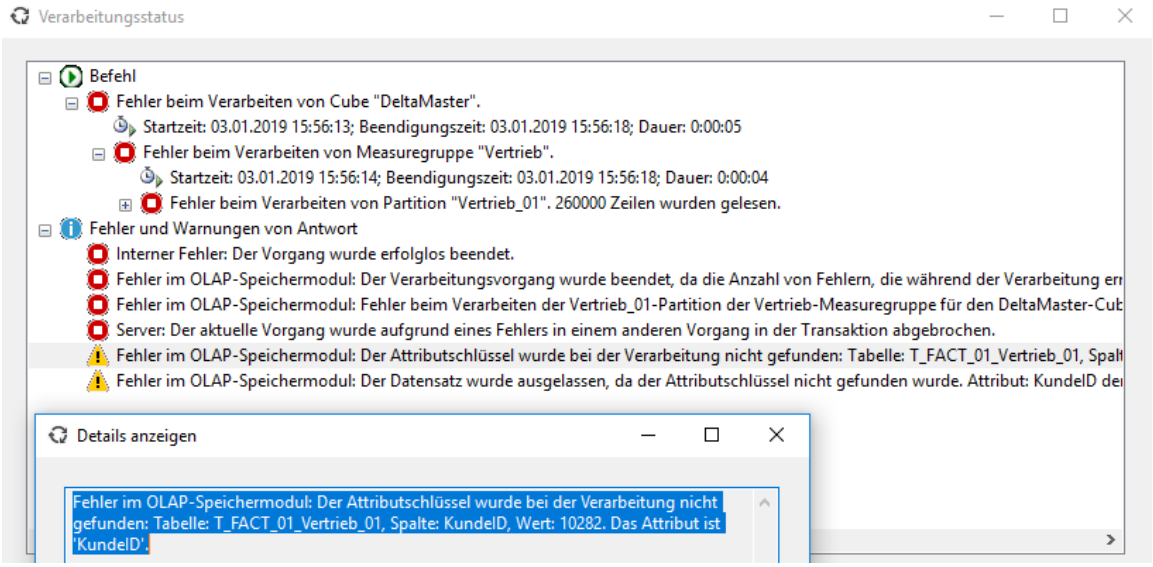

Abbildung 3 Verarbeitungsfehler, verursacht durch nicht aktualisierte Dimension

Auch die Option „*Betroffene Objekte verarbeiten*" in den Process-Einstellungen hilft in diesem Fall nicht, obwohl diese besagt, dass Abhängige Objekte beim Verarbeiten mitverarbeitet werden.

Somit müssen also die stabil funktionierenden Varianten aus Kapitel 1 und 2 kombiniert werden.

Zur Aktualisierung der Dimension muss auf die Option „*Hinzufügung verarbeiten*" zurückgegriffen werden und für die Measuregroups auf die Option „*Vollständig verarbeiten*".

Zunächst muss ermittelt werden, welche, an die betroffenen Measuregroups angebundenen Dimensionen, Änderungen unterliegen könnten. Im einfachsten Fall werden einfach alle angebundenen Dimensionen berücksichtigt. Für diese muss jeweils ein ProcessAdd-Skript, wie in Kapitel 1 beschrieben, erstellt werden.
Für die Measuregroups muss jeweils ein ProcessFull-Skript (= „Vollständig verarbeiten") erstellt

werden. Diese Skripte können dann gemeinsam in einen Batch-Tag gepackt werden und als XMLA-Skript ausgeführt werden, wobei zunächst neu hinzugekommene Elemente in den Dimensionen ergänzt werden und anschließend die Measuregroups verarbeitet werden.

Ein solches Skript sieht dann etwa wie folgt aus:

```
<Batch xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
        <Parallel>
                <Process xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ddl2="http://schemas.microsoft.com/analy-
sisservices/2003/engine/2" xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/engine/2/2"
xmlns:ddl100_100="http://schemas.microsoft.com/analysisservices/2008/engine/100/100"
xmlns:ddl200="http://schemas.microsoft.com/analysisservices/2010/engine/200"
xmlns:ddl200_200="http://schemas.microsoft.com/analysisservices/2010/engine/200/200"
xmlns:ddl300="http://schemas.microsoft.com/analysisservices/2011/engine/300"
xmlns:ddl300_300="http://schemas.microsoft.com/analysisservices/2011/engine/300/300"
xmlns:ddl400="http://schemas.microsoft.com/analysisservices/2012/engine/400"
xmlns:ddl400_400="http://schemas.microsoft.com/analysisservices/2012/engine/400/400"
xmlns:ddl500="http://schemas.microsoft.com/analysisservices/2013/engine/500"
xmlns:ddl500_500="http://schemas.microsoft.com/analysisservices/2013/engine/500/500">
                        <Object>
                                <DatabaseID>FoodMart_plus_PROCESS</DatabaseID>
                                <DimensionID>Periode</DimensionID>
                        </Object>
                        <Type>ProcessAdd</Type>
                        <WriteBackTableCreation>UseExisting</WriteBackTableCreation>
                </Process>
                <Process xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ddl2="http://schemas.microsoft.com/analy-
sisservices/2003/engine/2" xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/engine/2/2"
xmlns:ddl100_100="http://schemas.microsoft.com/analysisservices/2008/engine/100/100"
xmlns:ddl200="http://schemas.microsoft.com/analysisservices/2010/engine/200"
xmlns:ddl200_200="http://schemas.microsoft.com/analysisservices/2010/engine/200/200"
xmlns:ddl300="http://schemas.microsoft.com/analysisservices/2011/engine/300"
xmlns:ddl300_300="http://schemas.microsoft.com/analysisservices/2011/engine/300/300"
xmlns:ddl400="http://schemas.microsoft.com/analysisservices/2012/engine/400"
xmlns:ddl400_400="http://schemas.microsoft.com/analysisservices/2012/engine/400/400"
xmlns:ddl500="http://schemas.microsoft.com/analysisservices/2013/engine/500"
xmlns:ddl500_500="http://schemas.microsoft.com/analysisservices/2013/engine/500/500">
                        <Object>
                                <DatabaseID>FoodMart_plus_PROCESS</DatabaseID>
                                <DimensionID>Wertart</DimensionID>
                        </Object>
                        <Type>ProcessAdd</Type>
                        <WriteBackTableCreation>UseExisting</WriteBackTableCreation>
                </Process>

[… weitere Dimensionen …]

                <Process xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ddl2="http://schemas.microsoft.com/analy-
sisservices/2003/engine/2" xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/engine/2/2"
xmlns:ddl100_100="http://schemas.microsoft.com/analysisservices/2008/engine/100/100"
xmlns:ddl200="http://schemas.microsoft.com/analysisservices/2010/engine/200"
xmlns:ddl200_200="http://schemas.microsoft.com/analysisservices/2010/engine/200/200"
xmlns:ddl300="http://schemas.microsoft.com/analysisservices/2011/engine/300"
xmlns:ddl300_300="http://schemas.microsoft.com/analysisservices/2011/engine/300/300"
xmlns:ddl400="http://schemas.microsoft.com/analysisservices/2012/engine/400"
xmlns:ddl400_400="http://schemas.microsoft.com/analysisservices/2012/engine/400/400"
xmlns:ddl500="http://schemas.microsoft.com/analysisservices/2013/engine/500"
xmlns:ddl500_500="http://schemas.microsoft.com/analysisservices/2013/engine/500/500">
                        <Object>
                                <DatabaseID>FoodMart_plus_PROCESS</DatabaseID>
                                <CubeID>DeltaMaster</CubeID>
                                <MeasureGroupID>Vertrieb</MeasureGroupID>
                        </Object>
                        <Type>ProcessFull</Type>
                        <WriteBackTableCreation>UseExisting</WriteBackTableCreation>
                </Process>

[… weitere Measuregroups …]
```

```
        </Parallel>
</Batch>
```

Zu beachten: in der Praxis hat sich gezeigt, dass dieses Verfahren bei extrem großen Measuregroups mit vielen Partitionen unter Umständen langsamer ist, als die vollständige Verarbeitung der gesamten OLAP-Datenbank. Die Ursache liegt vermutlich in einer unterschiedlichen Parallelisierung bei den einzelnen Verfahren. Die genauen Grenzen müssen jedoch im jeweiligen Projekt herausgefunden werden.

4 Automatische Skript-Erstellung

Die Erstellung eines solchen XMLA-Skripts ist mühsam und auch fehleranfällig. Daher wird im Folgenden eine Prozedur vorgestellt, die für DeltaMaster ETL-Modelle passende XMLA-Skripte generieren kann. Das Skript zur Erstellung dieser Prozedur ist im Anhang zu finden.

Zunächst muss eine Parametertabelle für die Skriptgenerierung angelegt werden (*T_S_Incremental-Process_Parameter*). Diese enthält diverse Parameter aus denen das Skript anschließend generiert werden kann.

Zu beachten: insbesondere Parameter 1 und 3 können sich je nach Analysis Services Instanz unterscheiden. Diese Parameter sollten in jedem Fall vor der ersten Skriptgenerierung auf einer neuen Umgebung angepasst werden!
Dafür am besten einfach im Management Studio ein Verarbeitungsskript für ein beliebiges Objekt der betroffenen OLAP-Datenbank (z. B. eine Measuregroup) erstellen.

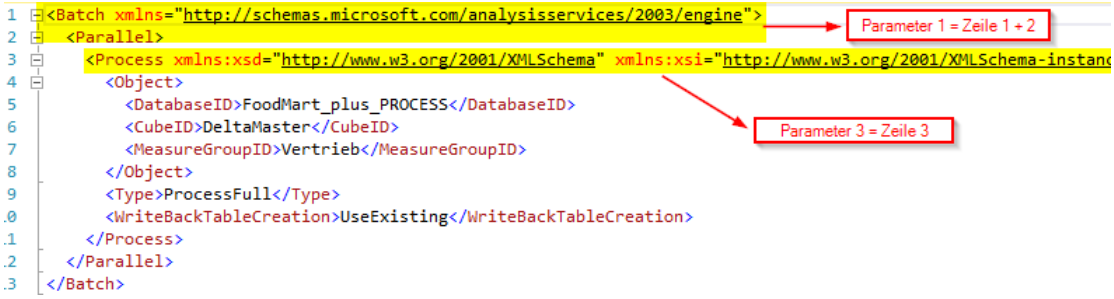

Abbildung 4 Ermittlung der anzupassenden Parameter 1 und 3

Der Eintrag für Parameter 1 ist in den Zeilen 1 und 2 des Skripts enthalten. Der Eintrag für Parameter 3 in der Zeile 3. Diese Einträge sind einfach per Copy & Paste in die Steuertabelle einzufügen.

	ParameterID	ParameterValue	ParameterDESC
1	1	<Batch xmlns="http://schemas.microsoft.com/analysiss...	Header
2	2	</Parallel> </Batch>	End
3	3	<Process xmlns:xsd="http://www.w3.org/2001/XMLSc...	Process Object Header
4	4	<WriteBackTableCreation>UseExisting</WriteBackTab...	Process Object Tail
5	5	<Object>	Object start tag
6	6	</Object>	Object end tag
7	7	<DatabaseID>	Database start tag
8	8	</DatabaseID>	Database end tag
9	9	<DimensionID>	Dimension start tag
10	10	</DimensionID>	Dimension end tag
11	11	<Type>ProcessAdd</Type>	ProcessAdd for dimension object
12	12	<CubeID>	Cube start tag
13	13	</CubeID>	Cube end tag
14	14	<MeasureGroupID>	Msrgrp start tag
15	15	</MeasureGroupID>	Msrgrp end tag
16	16	<Type>ProcessFull</Type>	ProcessFull for Measuregroup

Abbildung 5 Parameter-Tabelle T_S_IncrementalProcess_Parameter

Die Prozedur zur Erzeugung des XMLA-Skripts wird z. B wie folgt aufgerufen:

```
exec [dbo].[P_Generate_IncrementalProcess_Script] @FactIDs = '1,3'
```

Die Parameter setzen sich wie folgt zusammen:

@FactIDs (varchar(100)): Kommagetrennte IDs der zu verarbeitenden Measuregroups. Die IDs können beispielsweise im DeltaMaster ETL-Bericht „*Measure groups*“ in der Spalte „*Meas group ID*“ ermittelt werden.

Measure groups

Eingabe

Cube	Meas group ID	Measure group name
DeltaMaster	1	Vertrieb
DeltaMaster	2	Logistik
DeltaMaster	3	Personal

Abbildung 6 Ermittlung des Parameters @FactIDs

@OLAPDB (varchar(100)): **Optionaler Parameter**. Name der zu verarbeitenden Microsoft Analysis Services OLAP-Datenbank. Wird kein Wert mitgeliefert (wie im obigen Beispiel) wird die OLAP-Datenbank aus der *T_Model_Parameter*-Tabelle des DeltaMaster ETL-Metamodells ermittelt.

Alle weiteren benötigten Informationen werden automatisch aus den *T_Model*-Tabellen des Metamodells ermittelt.

Nach dem erfolgreichen Prozeduraufruf erscheint im Ergebnis-Fenster eine Zelle „ProcessScript“:

Ergebnisse | Meldungen

	ProcessScript
1	<Batch xmlns="http://schemas.microsoft.com/analy...

Abbildung 7 Das unscheinbare Abfrageergebnis

Der Inhalt dieser Zelle muss nun kopiert werden. Es handelt sich dabei um das gewünschte XMLA-Skript das nun entweder direkt im Management Studio oder an anderer Stelle (z. B. ServerAgent-Job, SSIS-Paket etc.) ausgeführt werden kann.

5 Anhang

Steuertabelle T_S_IncrementalProcess_Parameter

Achtung: Parameter 1 und 3 sind nach dem Einspielen auf das jeweilige System anzupassen!

```
CREATE TABLE [dbo].[T_S_IncrementalProcess_Parameter](
        [ParameterID] [int] NOT NULL,
        [ParameterValue] [varchar](max) NULL,
        [ParameterDESC] [varchar](max) NULL,
 CONSTRAINT [PK_T_S_IncrementalProcess_Parameter] PRIMARY KEY CLUSTERED
(
        [ParameterID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(1, N'<Batch xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
  <Parallel>', N'Header')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(2, N'</Parallel>
</Batch>', N'End')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(3, N'<Process xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:ddl2="http://schemas.microsoft.com/analysisservices/2003/engine/2"
xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/engine/2/2"
xmlns:ddl100_100="http://schemas.microsoft.com/analysisservices/2008/engine/100/100"
xmlns:ddl200="http://schemas.microsoft.com/analysisservices/2010/engine/200"
xmlns:ddl200_200="http://schemas.microsoft.com/analysisservices/2010/engine/200/200"
xmlns:ddl300="http://schemas.microsoft.com/analysisservices/2011/engine/300"
xmlns:ddl300_300="http://schemas.microsoft.com/analysisservices/2011/engine/300/300"
xmlns:ddl400="http://schemas.microsoft.com/analysisservices/2012/engine/400"
xmlns:ddl400_400="http://schemas.microsoft.com/analysisservices/2012/engine/400/400"
xmlns:ddl500="http://schemas.microsoft.com/analysisservices/2013/engine/500"
xmlns:ddl500_500="http://schemas.microsoft.com/analysisservices/2013/engine/500/500">', N'Process Object
Header')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(4, N' <WriteBackTableCreation>UseExisting</WriteBackTableCreation>
    </Process>', N'Process Object Tail')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(5, N'<Object>', N'Object start tag')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(6, N'</Object>', N'Object end tag')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(7, N'<DatabaseID>', N'Database start tag')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(8, N'</DatabaseID>', N'Database end tag')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(9, N'<DimensionID>', N'Dimension start tag')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(10, N'</DimensionID>', N'Dimension end tag')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(11, N'<Type>ProcessAdd</Type>', N'ProcessAdd for dimension object')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(12, N'<CubeID>', N'Cube start tag')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(13, N'</CubeID>', N'Cube end tag')
GO
```

```
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(14, N'<MeasureGroupID>', N'Msrgrp start tag')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(15, N'</MeasureGroupID>', N'Msrgrp end tag')
GO
INSERT [dbo].[T_S_IncrementalProcess_Parameter] ([ParameterID], [ParameterValue], [ParameterDESC]) VALUES
(16, N'<Type>ProcessFull</Type>', N'ProcessFull for Measuregroup')

GO
```

Prozedur P_Generate_IncrementalProcess_Script

```
CREATE PROCEDURE [dbo].[P_Generate_IncrementalProcess_Script] (@FactIDs AS Varchar(100), @OLAPDB AS
Varchar(100) = NULL) AS
        /*
        Parameter:
        @FactsIDs: comma separated measuregroup IDs from: SELECT FactID, FactName FROM T_Model_Facts
        @OLAPDB: name of OLAP-Database, DEFAULT = NULL --> Value comes from SELECT * FROM
T_Model_Parameter WHERE ParameterID = 210

        Example:
        exec [P_Generate_IncrementalProcess_Script] @FactIDs = '1,3';

        Process-Parameter-Table:
        SELECT * FROM   T_S_IncrementalProcess_Parameter -- has to be modified for system configuration,
especially ParameterIDs 1 & 3

        */
BEGIN

        DECLARE @ProcessScript AS VARCHAR(MAX);
        DECLARE @ObjectName AS VARCHAR(MAX);
        DECLARE @CubeName AS VARCHAR(50);

        IF @OLAPDB IS NULL
                BEGIN
                        SELECT @OLAPDB = ParameterValue FROM T_Model_Parameter WHERE ParameterID = 210
                END

        IF OBJECT_ID('tempdb..#T_RelevantMeasuregroups') IS NOT NULL
                BEGIN
                        DROP TABLE #T_RelevantMeasuregroups
                END

        IF OBJECT_ID('tempdb..#T_RelevantDimensions') IS NOT NULL
                BEGIN
                        DROP TABLE #T_RelevantDimensions
                END

        CREATE TABLE #T_RelevantMeasuregroups   (
                FactID INT,
                FactName VARCHAR(50),
                CubeName VARCHAR(50)
        )

        CREATE TABLE #T_RelevantDimensions      (
                DimensionID INT,
                DimensionName VARCHAR(100)
        )

        INSERT INTO #T_RelevantMeasuregroups (FactID, FactName, CubeName)
        SELECT
                split.SepString AS FactID
                ,facts.FactName
                ,cubes.CubeName
        FROM
                dbo.F_BC_Split(@FactIDs,',',1,1) split
        LEFT JOIN T_Model_Facts facts ON
                split.SepString = facts.FactID
        INNER JOIN T_Model_Cubes cubes ON
```

```
                facts.CubeID = cubes.CubeID

        INSERT INTO #T_RelevantDimensions (DimensionID, DimensionName)
        SELECT DISTINCT
                dims.DimensionID_Orig
                ,dimNames.DimensionName
        FROM
                V_Model_Fact_Dimensions dims
        LEFT JOIN T_Model_Dimensions dimNames ON
                dims.DimensionID_Orig = dimNames.DimensionID
        WHERE
                dims.FactID IN (SELECT FactID FROM #T_RelevantMeasuregroups)


        -- Build Script:
        -- 1. Header
        SET @ProcessScript = (SELECT ParameterValue FROM [dbo].[T_S_IncrementalProcess_Parameter] WHERE
ParameterID = 1)


        -- 2. Process Dims
        DECLARE stpr_cursor CURSOR FOR
        SELECT
                DimensionName
        FROM #T_RelevantDimensions

        OPEN stpr_cursor

        FETCH NEXT FROM stpr_cursor INTO @ObjectName

        WHILE @@FETCH_STATUS = 0
                BEGIN
                        SET @ProcessScript =     @ProcessScript
                                                                + (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 3)
                                                                + (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 5)
                                                                + (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 7)
                                                                + @OLAPDB
                                                                + (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 8)
                                                                + (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 9)
                                                                + @ObjectName
                                                                + (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 10)
                                                                + (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 6)
                                                                + (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 11)
                                                                + (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 4)


                FETCH NEXT FROM stpr_cursor INTO @ObjectName

                END

        CLOSE stpr_cursor
        DEALLOCATE stpr_cursor


        -- 2. Process Facts
        DECLARE stpr_cursor CURSOR FOR
        SELECT
                FactName
                ,CubeName
        FROM #T_RelevantMeasuregroups

        OPEN stpr_cursor
```

```
	FETCH NEXT FROM stpr_cursor INTO @ObjectName, @CubeName

	WHILE @@FETCH_STATUS = 0
		BEGIN
			SET @ProcessScript =	@ProcessScript
+ (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 3)
+ (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 5)
+ (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 7)
+ @OLAPDB
+ (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 8)
+ (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 12)
+ @CubeName -- Cubename
+ (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 13)
+ (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 14)
+ @ObjectName -- MsrGrp name
+ (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 15)
+ (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 6)
+ (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 16)
+ (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 4)


			FETCH NEXT FROM stpr_cursor INTO @ObjectName, @CubeName

			END

	CLOSE stpr_cursor
	DEALLOCATE stpr_cursor

	-- 3. Tail
	SET @ProcessScript =	@ProcessScript
+ (SELECT ParameterValue FROM
[dbo].[T_S_IncrementalProcess_Parameter] WHERE ParameterID = 2)


	select @ProcessScript AS ProcessScript

	IF OBJECT_ID('tempdb..#T_RelevantMeasuregroups') IS NOT NULL
		BEGIN
			DROP TABLE #T_RelevantMeasuregroups
		END

	IF OBJECT_ID('tempdb..#T_RelevantDimensions') IS NOT NULL
		BEGIN
			DROP TABLE #T_RelevantDimensions
		END


END
```