BISSANTZ

Eine Eingabe, viele Änderungen – Rollbackfähigkeit in der Hybridplanung

Autor	Lukas Müller
Datum der Veröffentlichung	07.12.2018

Abstract

Schnell passiert es, dass man mit einer Eingabe nicht nur die beschriebene Kennzahl für die beschriebenen Datensätze anpassen möchte, sondern auch weitere Kennzahlen für zum Beispiel alle folgenden Monate. Wie dies im Zusammenhang mit der (Delta-)Hybridplanung möglich ist, ohne die Möglichkeit eines Rollbacks zu verlieren, soll dieser Beitrag zeigen.

Eine Eingabe viele Änderungen – Rollbackfähigkeit in der Hybridplanung

1 Einleitung

In Planungsanwendungen kommt es vor, dass mit einer Eingabe auf eine Kennzahl auch andere, abhängige Kennzahlen geändert werden sollen. Diese Anpassungen können mit Hilfe der im Standard vorgesehenen Pre- und PostProcess-Prozeduren der (Delta-)Hybridplanung vorgenommen werden. In diese müssen einige kurze Update- und / oder Insert-Statements eingebaut werden und man kann bei einer Steigerung des Absatzes gleich den Lagerbestand senken. Sofern der Lagerbestand nur in Datensätzen angepasst wurde, in denen auch der Absatz geändert wurde, ist alles in bester Ordnung. Man kann nach Belieben die Eingabe bestätigen und alles wird korrekt angezeigt oder man kann die Eingabe abbrechen und Dank des Rollbacks wird der alte Stand angezeigt. Wird aber im Januar der Absatz erhöht und kann nicht gleichzeitig nur für den Januar den Bestand verringert werden. Die zusätzlich abgesetzten Einheiten sind auch in den Folgemonaten nicht verfügbar, weshalb auch diese angepasst werden müssen. Bei einem Rollback werden hier aber nur die Werte für den Januar zurückgesetzt. Für die restlichen Monate ist im Standard kein Rollback möglich. Jedoch gibt es zwei Möglichkeiten diesen Umstand zu ändern. Diese werden in diesem Beitrag vorgestellt.

2 Rollback-Prozess

Der Writeback-Prozess zwischen dem Start einer Eingabe und dem Bestätigen bzw. Abbrechen der (Delta-)Hybridplanung ist eine Transaktion. Innerhalb dieser Transaktion geänderte Stände werde in der Rollback-Tabelle gesichert. Was gesichert wird kann innerhalb einer Transaktion beeinflusst werden. Der Abbildung 1 kann der Writeback-Prozess in gekürzter Form entnommen werden. Hier sind alle für das Rollback-Thema relevanten Schritte aufgeführt.

PreProcess:
Alle Manipulation auf die Writeback-Tabelle, die an dieser Stelle vorgenommen werden, können durch ein Ablehnen der Eingabe nicht mehr rückgängig gemacht werden, da die Rollback-Tabelle erst nach diesem Schritt gefüllt wird.

Rollback:
Alle Datensätze der Writeback-Tabelle, für die sich der Wert der Kennzahl ändert, auf der im Delta-Master eingegeben wurde, werden in die Rollback-Tabelle geschrieben. Beim oben beschriebenen Beispiel würde also nur der Januar gesichert werden. Bei Änderung des Absatzes auf Jahresebene würden hingegen alle Monate des Jahres in der Rollback-Tabelle gesichert werden.

Wird die Eingabe bestätigt, werden die Einträge in aus der Rollback-Tabelle und die Log-Informationen (*Locked*-Spalte, TransactionID, EntryID etc.) aus der Writeback-Tabelle gelöscht. Wenn die Abfrage abgebrochen wird, werden die Einträge in der Writeback-Tabelle, die auf Locked gesetzt sind und die TransaktionsID der abgebrochenen Transaktion enthalten, gelöscht. Anschließend werden die Datensätze der Rollback-Tabelle wieder in die Writeback-Tabelle geschrieben. Somit wurde der Stand, der vor der Eingabe vorlag, wiederhergestellt.

PostProcess:
Alle Manipulationen der Writeback-Tabelle, die an dieser Stelle vorgenommen werden, können durch ein Ablehnen der Eingabe rückgängig gemacht werden, da die Rollback-Tabelle zu diesem Zeitpunkt bereits gefüllt ist und die Log-Informationen in der Writeback-Tabelle stehen.

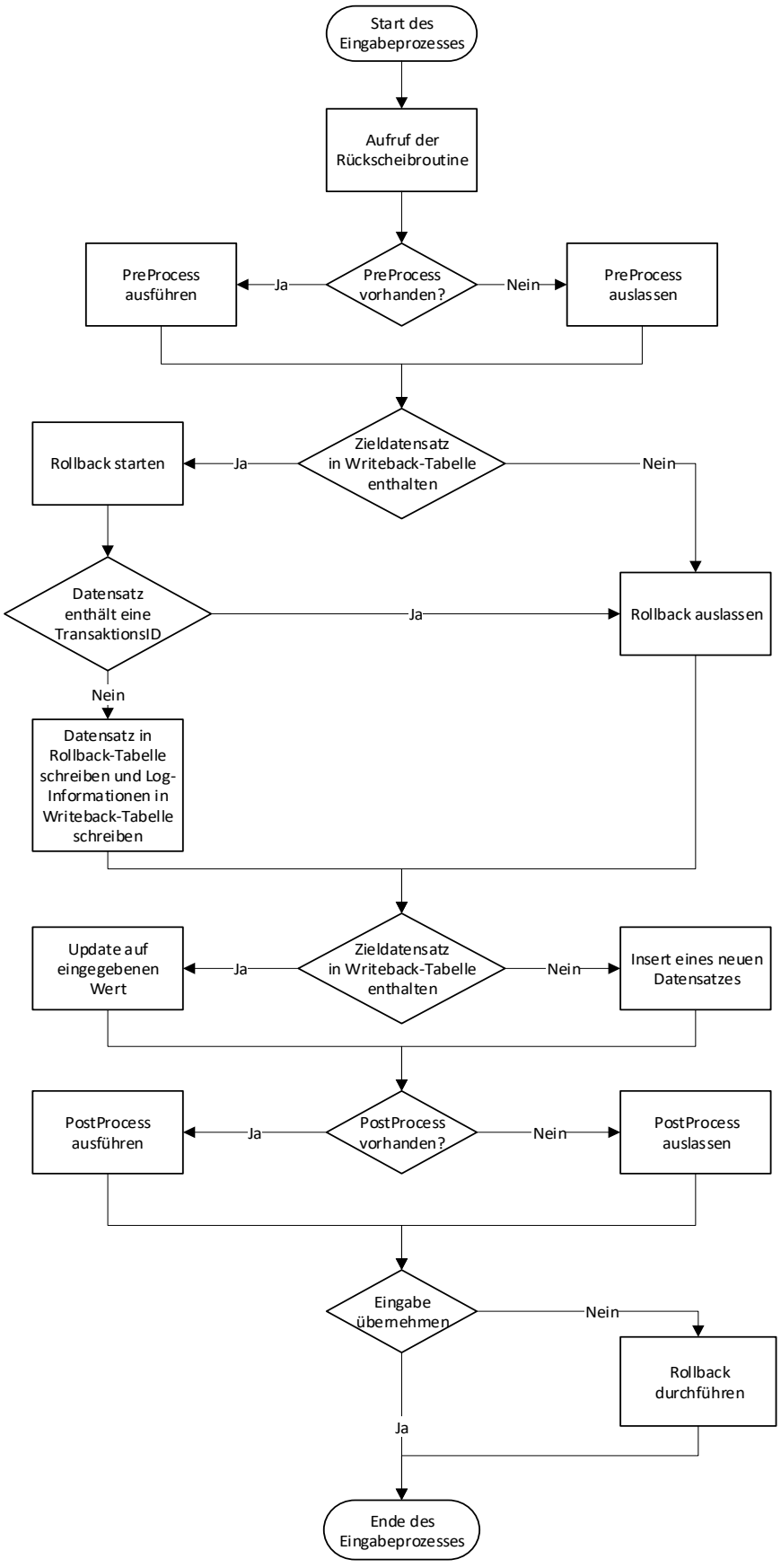

Abbildung 1 - Writeback-Prozess

3 Zusätzlich Datensätze rollbackfähig bearbeiten

3.1 Aufruf der gesamten Writeback-Routine

Wird die Writeback-Routine innerhalb der eigentlichen Routine aufgerufen, sollte dies nach Möglichkeit in der PostProcess-Prozedur erfolgen. So ist sichergestellt, dass die eigentliche Eingabe ohne Probleme durchgeführt werden konnte, bevor die Parameter für den erneuten Aufruf der Routine angepasst werden. Hier müssen unweigerlich die Parameter der Dimension überschrieben werden, für die weitere Elemente im Zuge der Eingabe angepasst werde sollen.

Der Vorteil hierbei ist, dass man sich keine Gedanken mehr machen muss, ob prozessual alles korrekt ist, da auf diese Weise der Standard verwendet wird. Wichtig hierbei ist sicherzustellen, dass keine Dauerschleife geschaffen wird, in der immer wieder die Writeback-Routine aufgerufen wird. Hier können die *AvoidProProcess*- und *AvoidPostProcess*-Parameter helfen. Sie können beim erneuten Aufruf der Routine auf den Wert „1“ gesetzt werden, wodurch ein entsprechender Aufruf vermieden wird. Der Vorteil, dass der Standard verwendet wird, ist aber auch der Nachteil. Anstelle eines einfachen *Inserts* in die Rollback-Tabelle und eines *Inserts* bzw. *Updates* innerhalb der PostProcess-Prozedur müsste für jedes Dimensionselement, das zusätzlich manipuliert wird, gesamte Routine aufgerufen werden. Dies kann zu Laufzeitproblemen führen.

3.2 Erstellen eigener Statements

Die hier vorgestellte Lösung des zusätzlich erstellten Statements besteht aus zwei Parts. Der erste Part schreibt in der PreProcess-Prozedur alle zusätzlich anzupassenden Datensätze in die Rollback-Tabelle. Der zweite Part führt in der PostProcess-Prozedur die Manipulationen der Datensätze in der Writeback-Tabelle durch, die nicht durch die eigentliche Eingabe erreicht werden.

Die im Folgenden gezeigten Skript-Ausschnitte stammen von einem fiktiven Planungsbeispiel, bei dem auf Monats-, Filial- und Produktebene die Absätze einzelner Filialen für ein Kalenderjahr geplant werden. Wird der Absatz einer Filiale verändert, verändert sich auch der Lagerbestand in der jeweiligen Filiale. Wird zum Beispiel der Absatz im Oktober für ein Produkt um 3 erhöht, muss der Lagerbestand für Oktober, November und Dezember um 3 verringert werden, da diese 3 Einheiten für den Rest des Kalenderjahres nicht mehr zur Verfügung stehen.

Da die Eingabe auf der untersten Ebene stattfindet, wird immer nur der Monat der Eingabe an die Writeback-Routine übergeben. Somit wird auch nur der Datensatz für diese Kombination aus Monat, Filiale und Produkt in der Rollback-Tabelle gesichert. Wenn aber in der PostProcess-Prozedur die Lagerbestände für November und Dezember angepasst werden und nach Abschluss der Eingabe festgestellt wird, dass man die Transaktion doch abbrechen möchte., bleiben die manipulierten Bestände für November und Dezember bestehen. Dieser Umstand kann durch folgendes Vorgehen verhindert werden:

```
SELECT
       [MonatID]
      ,[PeriodenansichtID]
      ,[KumulationID]
      ,[WertartID]
      ,[FilialeID]
      ,[ProduktID]
      ,[Absatz]
      ,[Lagerbestand]
      ,[SourceID]
      ,[Locked]
      ,[TransactionID]
```

```
	,[EntryID]
	,[Deleted]
	,[ZeroEntered]
	,[CreationTime]
	,[LastChangeTime]
	,[LastChangeUser]
	,@TransactionID AS LastChangeTransactionID
	,[ChangeCount]
INTO #T_Rollback_Data
FROM
	T_WriteBackSQL_FACT_01_Absatzplanung
WHERE
	MonatID > @Periode_Periode_MemberKey
	AND FilialeID = @Filiale_Filiale_MemberKey
	AND ProduktID = @Produkt_Produkt_MemberKey
	AND TransactionID IS NULL
```

Zunächst werden alle Datensätze der Writeback-Tabelle der nachfolgenden Monate für die relevante Filiale-Produkt-Kombination in eine temporäre Tabelle geschrieben. Wichtig hierbei ist, dass nur solche Datensätze weggeschrieben werden, bei denen die TransactionID den Wert NULL hat. Steht eine TransactionID im Datensatz, wurden dieser innerhalb der aktuellen Transaktion bereits bearbeitet und steht somit auch in der Rollback-Tabelle. Der Monat, auf dem eingegeben wurde, wird nicht benötigt, da hier der Standardprozess für ein korrektes Rollback Sorge trägt.

```
MERGE T_WriteBackSQL_FACT_01_Absatzplanung_Rollback AS tgt
	USING #T_Rollback_Data AS src
	ON (
		tgt.MonatID = src.MonatID
		AND tgt.FilialeID = src.FilialeID
		AND tgt.ProduktID = src.ProduktID
	)
WHEN NOT MATCHED BY TARGET THEN

INSERT (
	[MonatID]
	,[PeriodenansichtID]
	,[KumulationID]
	,[WertartID]
	,[FilialeID]
	,[ProduktID]
	,[Absatz]
	,[Lagerbestand]
	,[SourceID]
	,[Locked]
	,[TransactionID]
	,[EntryID]
	,[Deleted]
	,[ZeroEntered]
	,[CreationTime]
	,[LastChangeTime]
	,[LastChangeUser]
	,[LastChangeTransactionID]
	,[ChangeCount]
)
VALUES(
	[MonatID]
	,[PeriodenansichtID]
	,[KumulationID]
	,[WertartID]
```

```
	,[FilialeID]
	,[ProduktID]
	,[Absatz]
	,[Lagerbestand]
	,[SourceID]
	,[Locked]
	,[TransactionID]
	,[EntryID]
	,[Deleted]
	,[ZeroEntered]
	,[CreationTime]
	,[LastChangeTime]
	,[LastChangeUser]
	,[LastChangeTransactionID]
	,[ChangeCount]
);
```

Anschließend werden die Datensätze aus der temporären Tabelle in die Rollback-Tabelle geschrieben. Hier wird durch die Verwendung von *Merge* sichergestellt, dass keine Kombination aus Monat, Filiale und Produkt doppelt in der Rollback-Tabelle steht. Wäre dies der Falle, würde es bei einem Wiederherstellen des Standes aus dieser Tabelle zu einem Fehler kommen, da diese einfach in die Writeback-Tabelle geschrieben werden würden. In dieser darf aber jede Kombination der einzelnen Dimensionselemente nur einmal aufgeführt werden.

Somit ist der erste Part abgeschlossen. Es bleibt noch zu sagen, dass alle zusätzlichen Plausibilitätsprüfungen, die im PreProcess stattfinden, vor diesem Schritt erfolgen sollten.

Anschließend kann mit dem zweiten Part die Manipulation der Writeback-Tabelle in der PostProcess-Prozedur begonnen werden.

```
SELECT
	m.MonthID
FROM
	T_DIM_01_03_Monat m
INTO #T_MergeMonths
WHERE

	m.MonatID BETWEEN CONVERT(INT, @Periode_Periode_MemberKey) AND (CONVERT(INT,
@Periode_Periode_MemberKey) + 11)
```

Im ersten Schritt der Manipulation der zusätzlichen Datensätze wird in diesem Beispiel eine temporäre Tabelle mit allen Monaten vom Monat der Eingabe bis zum Jahresende erstellt.

```
SELECT
	[MonatID]
	,[PeriodenansichtID]
	,[KumulationID]
	,[WertartID]
	,[FilialeID]
	,[ProduktID]
	,[Absatz]
	,ISNULL([Lagerbestand], 0) + (@MeasureValueOld - @MeasureValue)
	,[SourceID]
	,[Locked]
	,[TransactionID]
	,[EntryID]
	,[Deleted]
	,[ZeroEntered]
	,[CreationTime]
	,[LastChangeTime]
	,[LastChangeUser]
```

```
	,[LastChangeTransactionID]
	,[ChangeCount]
INTO #T_MergeWriteback
FROM
	T_WriteBackSQL_FACT_01_Absatzplanung
WHERE
	MonatID >= @Periode_Periode_MemberKey -- Monat der Dateneingabe und folgende
	AND FilialeID = @Filiale_Filiale_MemberKey
	AND ProduktID = @Produkt_Produkt_MemberKey

UNION ALL
-- Datensätze, die noch in Writeback-Tabelle stehen
SELECT
	m.[MonthID]
	,1 AS PeriodenansichtID
	,1 AS KumulationID
	,2 AS WertartID
	,@Filiale_Filiale_MemberKey
	,@Produkt_Produkt_MemberKey
	,NULL AS Absatz
	,@MeasureValueOld - @MeasureValue AS Lagerbestand
	,4 AS SourceID
	,1 AS Locked
	,@TransactionID AS [TransactionID]
	,@EntryID AS [EntryID]
	,0 AS [Deleted]
	,0 AS [ZeroEntered]
	,GETDATE() AS [CreationTime]
	,GETDATE() AS [LastChangeTime]
	,@UserName AS [LastChangeUser]
	,@TransactionID AS [LastChangeTransactionID]
	,1 AS [ChangeCount]
FROM
	#T_MergeMonths m
WHERE
	NOT EXISTS(
		SELECT *
		FROM
			T_WriteBackSQL_FACT_01_Absatzplanung wb
		WHERE
			wb.MonatID = m.MonatID
			AND wb.FilialeID = m.FilialeID
			AND wb.ProduktID = m.ProduktID

	)
```

Anschließend wird eine weitere temporäre Tabelle erstellt. In diese werden die Datensätze geschrieben, die sich bereits in der Writeback-Tabelle befinden, bei denen der Lagerbestand jedoch angepasst werden muss. Zusätzlich werden in diese Tabelle auch neue Datensätze geschrieben, sofern für eine Filiale-Produkt-Kombination Monate vorhanden sind, die zwischen dem Monat der Eingabe und dem Jahresende liegen und noch nicht in der Writeback-Tabelle stehen. Hierfür wird die im vorherigen Schritt erstellte temporäre Tabelle mit den relevanten Monaten verwendet. Außerdem werden für die neu erstellten Datensätze alle Logging-Informationen mit in die Tabelle geschrieben.

Für die Berechnung der Lagerbestandsveränderung können hier einfach die Parameter @MeasureValueOld und @Measurevalue voneinander subtrahiert werden.

```
MERGE T_WriteBackSQL_FACT_01_Absatzplanung t
USING #T_MergeWritebackFD s
	ON t.MonatID = s.MonatID
	AND t.FilialeID = s.FilialeID
```

```
	AND t.ProduktID = s.ProduktID
WHEN MATCHED
	THEN UPDATE
	SET
		t.Lagerbestand = s.Lagerbestand
		,t.[LastChangeTime] = GETDATE()
		,t.[LastChangeUser] = @UserName
		,t.[LastChangeTransactionID] = @TransactionID
		,t.EntryID = @EntryID
		,t.Locked = 1
		,t.TransactionID = @TransactionID
WHEN NOT MATCHED
	THEN INSERT(
		[MonatID]
		,[PeriodenansichtID]
		,[KumulationID]
		,[WertartID]
		,[FilialeID]
		,[ProduktID]
		,[Absatz]
		,[Lagerbestand]
		,[SourceID]
		,[Locked]
		,[TransactionID]
		,[EntryID]
		,[Deleted]
		,[ZeroEntered]
		,[CreationTime]
		,[LastChangeTime]
		,[LastChangeUser]
		,[LastChangeTransactionID]
		,[ChangeCount]
	)
	VALUES(
		[MonatID]
		,[PeriodenansichtID]
		,[KumulationID]
		,[WertartID]
		,[FilialeID]
		,[ProduktID]
		,[Absatz]
		,[Lagerbestand]
		,[SourceID]
		,[Locked]
		,[TransactionID]
		,[EntryID]
		,[Deleted]
		,[ZeroEntered]
		,[CreationTime]
		,[LastChangeTime]
		,[LastChangeUser]
		,[LastChangeTransactionID]
		,[ChangeCount]

	);
```

Abschließend werden die bereits in der Writeback-Tabelle vorhandenen Datensätze aktualisiert und die Logging-Informationen für diese anpasst bzw. die neuen Datensätze in die Writeback-Tabelle geschrieben. Hierbei wird wieder eine *Merge* verwendet, da so sichergestellt wird, dass definitiv keine zwei Datensätze mit den gleichen Dimensionselementen in der Writeback-Tabelle stehen.

Auf diese Weise kann mit relativ wenig Aufwand die bereits vorhandene Rollback-Logik verwendet werden, sodass das Zurückrollen des vorherigen Standes ohne weitere Anpassungen funktioniert und bei einer Bestätigung der Eingabe der komplette Inhalt der Rollback-Tabelle verworfen wird.

Der Nachteil dieser Lösung ist der Aufwand, der mit der Einrichtung verbunden ist. Auch die Wartbarkeit und zusätzlichen Stellen, die bei der Abbildung weiterer Logiken berücksichtigt werden, müssen berücksichtigt werden. Jedoch ist das Prozedere selber nicht sonderlich kompliziert und wenn diese Lösung einmal korrekt implementiert ist, ist sie sehr stabil. Auch ist die Performance dieser Lösung besser, da bei dem aktuellen Beispiel für jeden zusätzlichen Monat die gesamte Routine erneut aufgerufen werden würde. Außerdem sind hier mehr zusätzliche Freiheiten für projektspezifische Anpassungen gegeben.

Abschließend ist noch zu erwähnen, dass die vorgestellte Lösung sowohl für die Hybridplanung als auch für die Delta-Hybridplanung verwendet werden kann. Im Falle der Delta-Hybridplanung muss lediglich die Delta-Tabelle verwendet werden, sodass immer die Tabelle, die als ROLAP-Partition angebunden ist, die ist, in der die Manipulationen stattfinden.

4 Fazit

Zusammenfassend ist zu sagen, dass es immer von den Anforderungen des Projekts abhängt welcher Weg gewählt wird. Das erneute Aufrufen der Routine ist definitiv die einfachere und sichere Lösung, jedoch auch die unflexiblere und im Zweifel auch die, mit der schlechteren Performance. Die Lösung mit dem zusätzlich erstellten Prozess ist gerade, wenn mehrere Datensätze angepasst werden müssen, deutlich flexibler und wenn man das vorgestellte Muster verwendet auch übersichtlich. Die Komplexität kommt hier durch die zusätzlichen Geschäftslogiken, die abgebildet werden sollen. Diese müssen aber für jede Lösung implementiert werden und dürfen deshalb nicht für die Entscheidung für oder wider einer der vorgestellten Lösungen beeinflussen. Lediglich die Wartbarkeit muss im Auge behalten werden und man sollte vorab immer überlegen, welche Fälle überhaupt möglich sind bzw. erlaubt sein dürfen.