Verwendung von PROC METADATA zur Steuerung eines zentralen Metadatenmodells

Dubravko Dolic
dsquare.de
Kaiserstraße 28
83022 Rosenheim
office@dsquare.de

Zusammenfassung

Um ein durchgehendes Metadatenamangement in einem Entwicklerteam realisieren zu können und ein zentrales Datenmodell zentral verwalten zu können, lässt sich mittels SAS Base eine Excelbasierte Steuerung im DI Studio realisieren. Der Ansatz, welche XML Steuerbefehle dazu notwendig sind und wir der Metadatenserver angesprochen werden kann, wird im Folgenden dargestellt.

Schlüsselwörter: XML, Metadaten, PROC METADATA, Projektmanagement

1 Ausgangssituation

Mit dem DI Studio wurde von SAS eine Software bereitgestellt, mit deren Hilfe eine ETL Entwicklung in einem Projekt über mehrere Entwickler organisiert und verteilt werden kann. Im Gegensatz zur Entwicklung von SAS Base Code kann mittels der im DI Studio zur Verfügung gestellten Transformationen und Metadaten eine standardisierte und auf ein gemeinsames Zielmodell hin harmonisierte Entwicklung in einem Team von Entwicklern stattfinden.
Diese Vorteile sollten in einem Projekt im Bankenumfeld zur Entwicklung eines Risiko Datamarts genutzt werden. Ein Team von sechs Entwicklern sollte mittels des DI Studios 3.4 ETL Strecken entwickeln, um Daten aus sieben Liefersystemen (entstanden aus verschiedenen Mergern) in ein zentrales Zieldatenmodell zu verbringen.

2 Problemstellung für die konkrete Anwendung

Nachdem das Zielmodell definiert und in den Metadaten abgelegt war, stellte sich sehr schnell heraus, dass die konzeptionellen Arbeiten an den Überleitungen und damit auch an den Zieldaten auf fachlicher Seite nicht abgeschlossen waren. Dadurch ergab sich immer wieder ein Änderungsbedarf, der sich auch auf das Zielmodell auswirkte. Konkret konnte das eine Änderung an der Länge eines Feldes sein aber auch das Hinzufügen von Feldern, Tabellen oder das Entfernen ganzer Felder. Diese Änderungen, motiviert aus der fachlichen Analyse, mussten durch das gesamte Zielmodell propagiert werden.
Aufgrund einer Projektentscheidung sah die Projektorganisation vor, dass die Entwicklung das Zieldatenmodell je Liefersystem einzeln zu beliefern hatte. Erst in einem wei-

teren Schritt sollte eine liefersystemübergreifende Konsolidierung stattfinden. Dies erforderte, dass das Zielmodell in den Metadatenmehrfach (einmal je Liefersystem) angelegt wurde. Eine Unterscheidung erfolgte nach DI Ordnern. Eine Änderung am Datenmodell war demnach sehr aufwändig zu pflegen. Um diese Pflege des Datenmodells in den Metadaten zu leisten, wurde eine Funktionalität in SAS Base entwickelt.

3 Gesamtüberblick der Lösung

Um die Metadaten des Zielmodells zentral pflegen zu können, wurde ein Prozess eingeführt, der von einer SAS Base basierten Lösung begleitet werden konnte. Dieser Prozess sah vor, dass alle Tabellen und Felder inklusive einer entsprechenden Beschreibung des Zieldatenmodells in einer Excelliste vorgehalten und gepflegt wurden. Die Pflege des Excelsheet oblag einer zentralen Person. Diese hielt die Kommunikation zum Fachbereich und gab in regelmäßigen Abständen ein neues Release des Zieldatenmodells frei. Erst nach dieser Freigabe wurde die neue Version mittels der SAS Skripte in die Metadaten propagiert. Somit konnte sichergestellt werden, dass alle Entwickler zu jeder Zeit immer auf die gleiche Version des Zieldatenmodells Zugriff hatten.
Die Lösung besteht aus zwei Modulen: zum einen muss das Zieldatenmodell einmalig komplett angelegt werden. Dies wurde auch über SAS Skripte gelöst, da bei einem Deployment das Zieldatenmodell in verschiedenen Umgebungen neu angelegt werden muss. In einem weiteren Modul kann das angelegte Modell gepflegt werden.
Die Excelliste zur Pflege des Modells enthält folgende Informationen über das Zieldatenmodell.

Diese Excelliste wird zu Beginn des Prozesses jeweils in SAS eingelesen. Die Liste zum jeweils vorherigen Release wurde historisiert, so dass das Delta aus dem Vergleich zweier Listen erstellt werden kann.

Target table	Target variable	Target data type	null option	is PK	is FK	Description
CUSTOMER_PARTNER	CUSTOMER_ID	CHAR(25)	NOT NULL	Yes	No	TextTextTextText
CUSTOMER_PARTNER	SOURCE_SYSTEM_CD	CHAR(03)	NOT NULL	Yes	Yes	TextTextTextText
CUSTOMER_PARTNER	SNAPSHOT_DT	DATE	NOT NULL	Yes	No	TextTextTextText
CUSTOMER_PARTNER	CUSTOMER_TYPE_CD	CHAR(03)	NOT NULL	No	No	TextTextTextText
CUSTOMER_PARTNER	FEVE_RISK_POLICY_TYPE_CD	CHAR(05)	NULL	No	No	TextTextTextText
CUSTOMER_PARTNER	EMPLOYEE_STATUS_CD	CHAR(03)	NULL	No	No	TextTextTextText
CUSTOMER_X_FINANCIAL_ACCOUNT	CUSTOMER_ID	CHAR(25)	NOT NULL	Yes	Yes	TextTextTextText
CUSTOMER_X_FINANCIAL_ACCOUNT	SOURCE_SYSTEM_CD	CHAR(03)	NOT NULL	Yes	Yes	TextTextTextText
CUSTOMER_X_FINANCIAL_ACCOUNT	SNAPSHOT_DT	DATE	NOT NULL	Yes	Yes	TextTextTextText
CUSTOMER_X_FINANCIAL_ACCOUNT	ACCOUNT_ID	CHAR(25)	NOT NULL	Yes	Yes	TextTextTextText
CUSTOMER_X_FINANCIAL_ACCOUNT	RELATIONSHIP_TO_ACCOUNT_CD	CHAR(03)	NOT NULL	Yes	Yes	TextTextTextText
FINANCIAL_ACCOUNT	ACCOUNT_ID	CHAR(25)	NOT NULL	Yes	No	TextTextTextText
FINANCIAL_ACCOUNT	SOURCE_SYSTEM_CD	CHAR(03)	NOT NULL	Yes	Yes	TextTextTextText
FINANCIAL_ACCOUNT	SNAPSHOT_DT	DATE	NOT NULL	Yes	No	TextTextTextText
FINANCIAL_ACCOUNT	FINANCIAL_ACCOUNT_TYPE_CD	CHAR(03)	NOT NULL	No	Yes	TextTextTextText

4 Problemstellungen bei der Initialerstellung

Bei einem initialen Deployment des Zieldatenmodells war die zentrale Herausforderung, die Ordner anzulegen. Im DI sollte die gleiche Ordnerstruktur für jedes Liefersystem neu angelegt werden. Den Ordner wurden SAS Libraries zugewiesen, die Liefersystemspezifisch waren. Bei der initialen Anlage der Ordner bestand der Hauptaufwand darin, die jeweiligen IDs der Objekte zu kennen und anzulegen. Jedes Objekt in den Metadaten (Ordner, Tabelle, Attribut, etc.) bekommt eine eindeutige ID. Wenn nun Unterordner angelegt werden sollen, dann muss die ID des jeweiligen übergeordneten Ordners bekannt sein. Somit musst ein Algorithmus gefunden werden, mit dem neue Ordner erzeugt werden konnten, die IDs in Makrovariablen geschrieben wurden und darauf basierend entsprechende Unterordner angelegt werden konnten.
Die grundlegende Idee war folgende: Mittels der Prozedur PROC METADATA wurden XML Dateien gegen den Metadatenserver geschickt. In der Rückgabe der Prozedur sind die gewünschten Daten auch wieder eingebettet in eine XML strukturierte Datei enthalten. Die Informationen sind daher gleichartig geformt und können somit über klassische SAS String-Funktionen extrahiert werden.
Auf diese Weise wurden die gewünschten IDs evaluiert und in Makrovariablen gespeichert.
Dazu ein Beispiel: im folgenden Code wird eine XML Datei erstellt, mit deren Hilfe ein Unterordner im DI Studio erzeugt werden kann.

```
    objref = "&id_of_parent.";
    name = "&name_of_dir.";
    descr = "&description.";
    file subdir;
    PUT @01 '<AddMetaData>';
    PUT @03 '<MetaData>';
    PUT @06 '<Tree Name = "' name +(-1) '" TreeType="KPI"
Desc="' descr +(-1) '">';
    PUT @06 '<ParentTree>';
    PUT @09 '<Tree ObjRef="' objref +(-1) '"/>';
    PUT @06 '</ParentTree>';
    PUT @06 '</Tree>';
    PUT @03 '</MetaData>';
    PUT @03 "<Reposid>&Reposid.</Reposid>";
    PUT @03 '<NS>SAS</NS>';
    PUT @03 '<!-- OMI_TRUSTED_CLIENT flag   -->';
    PUT @03 '<Flags>268435456</Flags>';
    PUT @03 '<Options/>';
    PUT @01 '</AddMetaData>';
```

Die zu Beginn referenzierte Makrovariable ID_OF_PARENT wird als Eingabewert vom Makro erwartet. Um einen Unterordner anzulegen, muss demnach zunächst der entsprechende Hauptordner ermittelt werden. Für den obersten Ordner in der Hierarchie wurde dies mittels einer weiteren XML Datei erreicht. Diese Datei hat als Output eine

SAS Datei erzeugt, in der alle DI Ordner mit ihren entsprechenden Objekt IDs enthalten waren. Hier der entscheidende Ausschnitt aus dem Code dieser XML Datei:

```
<TABLE name="Treeinfo">
<TABLE-PATH
syntax="XPath">/GetMetadataObjects/Objects/Tree</TABLE
-PATH>

<COLUMN name="Obj_ref">
<PATH
syntax="XPath">/GetMetadataObjects/Objects/Tree/@Id</P
ATH>
        <TYPE>character</TYPE>
        <DATATYPE>string</DATATYPE>
        <LENGTH>17</LENGTH>
</COLUMN>
```

Alle XML Codes wurden per PROC METADATA an den Metadataserver übergeben.
Das Makro zur Erzeugung der DI Ordner hat nun mit Hilfe der Eltern ID einen neuen Ordner anlegen können. PROC METADATA gibt für dieses Vorgehen im OUT Statement eine XML Datei zurück, aus der die ID des frisch angelegten Verzeichnisses ausgelesen werden kann:

```
call symput
("&mv_id_of_lib.",
substr(tags, index(tags, "Id=")+4,
%eval(%length(&reposid))));
```

Dieses grundlegende Prinzip wurde in allen weiteren Schritten beibehalten:

1. Erstellen einer XML Datei
2. Erzeugen eines Objektes
3. Auslesen der dazugehörigen ID

5 Wartung der Objekte

Nachdem die Metadaten initial angelegt wurde, musste eine Wartung stattfinden. Wie so oft in Projekten, stellte sich das Vorgehensmodell als mehr oder weniger intendiert agil heraus. In der Praxis bedeutete das ständige Korrekturen und Erweiterungen am bereits angelegten Zieldatenmodell.
Jede Änderung im Zieldatenmodell wurde zentral in der eingangs erwähnten Exceldatei durchgeführt. Erst wenn diese von einer zentralen Instanz freigegeben wurde, erfolgte das Deployment mittels des hier beschriebenen Frameworks. Um die verschiedenen Änderungstypen manifestieren zu können, waren folgende Vorgänge notwendig:

- Hinzufügen neuer Spalten
- Löschen vorhandener Spalten
- Ändern von bestehenden Spalteninformationen
- Löschen von Tabellen

Diese finden ihre Entsprechung in folgenden XML Klassen:

- Hinzufügen neuer Spalten:

```
put @01 '<UpdateMetadata>';
put @03 '<Metadata>';
put @06 '<PhysicalTable id="' id +(-1) '">';
put @08 '<Columns Function="Append">';
```

- Löschen vorhandener Spalten:

```
put @01 '<DeleteMetadata>';
put @03 '<Metadata>';
put @06 '<Column Id="' cid +(-1) '" Name="'
colname +(-1) '" />';
```

- Ändern von bestehenden Spalteninformationen

```
put @01 '<UpdateMetadata>';
put @03 '<Metadata>';
put @06 '<Column id="' id +(-1) '" SASFormat="'
fmt +(-1) '" ';
```

- Löschen von Tabellen

```
on = "&objname";
id = "%upcase(&id)";
put @01 '<DeleteMetadata>';
put @03 '<Metadata>';
put @06 '<' on +(-1) ' id="' id +(-1) '"/>';
```

Im letzten Code wird die Makrovariable OBJNAME verwendet. Diese kann die Werte PhysicalTable oder SASLibrary annehmen, da mit diesem Code beide Objektarten gelöscht werden können.

6 Dokumentation

Die größte Herausforderung während der Programmierung dieses Frameworks war die Evaluierung der notwendigen XML Statements zur Steuerung der Metadaten. Die Dokumentation von SAS ist zwar vorhanden, aber sehr verteilt. Als zentrales Dokument zur Bestimmung der jeweiligen Vorgehensweise muss die Dokumentation zur SAS Open Metadata Architecure genannt werden. Innerhalb dieser finden sich UML Diagramme, anhand derer die Hierarchien der Objekte erläutert werden. Damit verschafft sich der interessierte Leser einen guten Überblick, wann welche Objekte referenziert werden müssen.
Als Referenz unerlässlich ist der Abschnitt in der gleichen Dokumentation: Alphabetical Listing of SAS Namespace Metadata Types. Hier werden alle Attribute und Beziehungen, die ein Objekt aufweisen kann, beschrieben. Diffizile Unterschiede wie der zwischen den Attributen Name, SASTableName und TableName des Objektes PhysicalTable werden in dieser Referenz erläutert.

Wichtige Beispiele für den zu verwendenden XML Code findet der suchende Programmierer im SAS Open Metadata Interface: User's Guide. Dort werden die Methoden Adding, Updating und Deleting für Metadatenobjekte mit Beispielen beschrieben.
Sicher ist es sinnvoll, bei der Arbeit mit dem Metadatenserver, dass DI zu verwenden und durchgeführte Arbeiten dort zu überprüfen. Oftmals kann es aber auch hilfreich sein, die Metadaten im Display Manager zu betrachten. Dort hält der Metadaten Browser viele Informationen im schnellen Zugriff bereit. Der Metadatenbrowser wird über den Menüpunkt „Lösungen" aufgerufen: