Speed up Developement – ODI Mapping Generierung für Data Vault.

Martin Wichert
CGI
Sulzbach

Schlüsselworte
ODI, Groovy, Data Vault, Generierung,Mapping

Einleitung
In diesem Vortrag stelle ich vor, wie wir im Projekt die Mappings zur Befüllung eines Data Vault Modells in einer Oracle Datenbank mit Hilfe eines groovy Skriptes automatisiert generieren. Das Groovy Script verwendet dafür Klassen und vorgefertigte Pakete von Oracle.Es verbindet sich mit dem ODI-Repository, liest dort vorgegebene Schemata aus, findet zusammengeörige Quell und Zieltabellen, und legt dann Mappings mit den notwendigen Komponenten an.

Namenskonventionen, und Vorgehensweisen

Ein Data Vault hat immer auch Namenskonventionen. Sind diese günstig gewählt kann man diese auch bei der Generierung nutzen.

In unserem Projekt werden die Tabellen der Quellsysteme zunächst 1:1 in die Stage kopiert. Dabei bleiben die Namen erhalten, und werden durch ein Prefix erweitert, so dass am Tebellenname erkannt werden kann, aus welchem System die Daten kommen.

Im Data Vault heißen die Hubs im allgemeinen so wie die Tabellen aus der Stage, jedoch ohne das Prefix, dafür mit dem Suffix _H. In Fällen in denen dies nicht möglich ist muss für den Generator ein special Table Mapping zur Verfügung gestellt werden. Damit die Quelltabelle zur Zieltabelle zugeordnet werden kann wird dann über die Namenskonvention geprüft ob es eine passende Verbindung gibt, und Alternativ ob es eine Verknüpfung im special Table Mapping gibt.

Die Business Keys in den Hubs werden alle mit ID benannt, die Surrogate Keys werden immer aus dem Namen des Hubs bestimmt indem das Suffix _ID angehängt wird.

Um zu verhindern, dass durch Prefixe und Suffixe Tabellen- oder Spaltennamen entstehen die länger als zulässig sind werden alle Namen auf 25 Stellen ohne Prä- und Suffixe reduziert. Bei Automatischen Namensabgleichen werden auch nur 25 Zeichen berücksichtigt.

Data Vault verwendet im Allgemeinen Surrogate Key Felder um Hubs Links und Satelliten miteinander zu verbinden. Wir beziehen diese nicht aus einer Sequenz, sondern berechnen sie mittels eines Hash-Verfahrens aus den Business Key. Hash-Berechnungen bieten zwar theoretisch die Gefahr einer Hash-Kollision, wenn zwei Business Keys zum selben Hash-Wert führen ergibt sich daraus ein Problem für die Datenbank. Die Wahrscheinlichkeit dass dies in Realen Daten passiert ist jedoch so gering, dass es auch bei großen Datenmengen nicht zu erwarten ist. Die Vorteile dieses Vorgehens überwiegen dieses Risiko bei weitem: Dadurch dass der Surrogate Key deterministisch vom Business Key abhängt müssen bei der Befüllung von Links und Stelliten, wo in den Quelldaten die Business Keys enthalten sind die Surrogate Keys nicht per Lookup aus den Hubs abgerufen werden, sondern können direkt aus den Quelldaten bestimmt werden.

Vorteile automatisierter Generierung von Mappings

Zunächst stellt sich natürlich die Frage warum man den Aufwand betreiben sollte einen Mapping-Generator zu entwickeln und zu testen. Wenn nur wenige Mappings benötigt werden ist dies in der Tat nicht sinnvoll. In Datawarehouse Projekten mit einer Datenhaltung in Data Vault Modellierung müssen viele gleichartig strukturierte Mappings gebaut werden. Durch den Mappinggenerator bleibt der Aufwand weitgehend unabhängig von der Anzahl der zu entwickelnden Mappings. Allein dadurch ist es schon nur noch eine Frage der Anzahl der Mappings die entwickelt werden müssen ab wann sich die Erstellung eines Mappinggenerators lohnt. Zusätzlich bieten sich noch die Vorteile, dass man wenn der Generator erstmal entwicklet ist bei späteren Erweiterungen des Datawarehouses für neue Tabellen im Data Vault Modell keine mappings mehr entwicklen muss, sondern diese durch den Generator einfach erzeugen kann, und man muss generiete Mappings deutlich weniger intensiv testen, wenn man einmal geprüft hat, dass generierte Mappings funktionieren.

Last but not least hat man eine konstantere Qualität der Mappings. Im Gegensatz zu manuell entwickleten Mappings wird nicht jedes Mapping von Hand erzeugt, und somit gibt es keine Flüchtigkeitsfehler. Selbst mit vorgefertigten Grundmappings in die man nur noch die unterschiedlichen Quellen und Ziele einfügen und verbinden muss kann es manuell immer noch zu kleineren Fehlern kommen, die im automatisiertem Prozess ausgeschlossen sind.

Warum sich gerade Data Vault für die Modellierung eignet

Data Vault baut auf den Elementen Hub, Link und Satellit auf. Alle Hubs sind gleich strukturiert, sie beinhalten den Business Key, mit dem das Geschäftsobjekt in den Quellsystemen eindeutig identifiziert ist, einen Surrogate Key, mit dem das Geschäftsobjekt im Data Vault verlinkt und referenziert wird sowie die Metadatenfelder MD_DWH_ID, MD_LOAD_ID, MD_LOAD_DATE, MD_LOAD_USER, MD_UPDATE_USER, MD_UPDATE_DATE, MD_SRC_SYSTEM.

Der Business Key heißt grundsätzlich ID, und der Name des Surrogate Key ergibt sich aus dem Tabellennamen indem an den Tabellennamen das Suffix _ID angehängt wird. Damit unterscheiden sich die Mappings inhaltlich nur durch den Namen von Quell und Zieltabelle, dem Namen des Surrogate Key und den Namen der Spalte aus dem der Business Key gelesen wird. Durch Namenskonventionen folgt der Name des Surrogate Key aus dem Namen des Hub, so dass man dem Generator nur noch die Zuordnung der Quellspalte für den Business Key zur Verfügung stellen muss, sowie für Fälle, in denen sich die Namen von Quell und Zieltabelle nicht auseinander ergeben , die Zuordnung der Tabellen.

Für die Satelliten sieht es ähnlich aus, nur werden hier statt der Business Keys eine variable Anzahl an Spalten gemappt. Da jedoch die Namenskonventionen vorsehen dass alle Spalten die Nutzdaten enthalten dieselben Namen haben wie in den Quelltabellen kann hier aus der Zieltabelle ausgelesen werden, welche Spalten aus der Quelle ausgelesen werden müssen.

Da Data Vault grundsätzlich die Daten so aufnimmt wie sie geliefert werden sind keine Transformationen nötig.

Bei den Links gibt es etwas mehr zu beachten, da hier die Keys der durch den Link verbundenen Hubs auftreten und Informationen aus den Spalten der Quelltabelle benötigt werden, die die Business Keys der Hubs liefern. Da hier die Namen der Quellspalten nicht von den Tabellennamen abhängen müssen dem Generator auch hier noch einige Informationen mitgegeben werden.

Vorraussetzungen

Damit Mappings automatisiert generiert werden können muss das Datenmodell für Quelle und Core Datawarehouse erstellt sein, und im ODI im Designer als Modell verfügbar sein. Dazu braucht der Generator eine so genannte special field mapping liste in der aufgelistet wird aus welchen Quellspalten welche Business Keys kommen, eine Special Table Mapping Liste in der zuordnungen von Quell und Zieltabelle enthalten sind, die nicht durch Namenskonventionen erkenntlich sind, sowie Informationen über das Repository in dem die Mappings generiert werden sollen und in welchem Projekt und Folder im ODI gearbeitet werden soll.

Vorgehensweise des Generators

Schritt 1 Informationen sammeln

Zunächst verbindet sich der Generator mit dem Repository. Dann liest der das Modell mit den Tabellen des Data Vaults aus. Zu jeder Tabelle dort sucht er einen passenden Datastore im Modell der Stage, ist dieser gefunden wird der Name des Mappings erzeugt und bereits der Konstruktor für das Mapping aufgerufen, und somit ein leeres Mapping angelegt. Das Objekt was der Konstruktor zurückliefert wird in einer Variablen gespeichert um es anschließend fertigzustellen.

. Je nach Art des Mappings werden dann zu den Schlüsselspalten in der Special Field Mapping Liste die zugehörigen Quellspalten gesucht.

Schritt 2 Das Mapping erstellen

Sind alle Informationen beisammen werden 3 Datastores in das Mapping eingefügt: Die Quelltabelle, die Zieltabelle, sowie ein weiteres mal die Zieltabelle um sie auszulesen, damit bereits vorhandene Datensätze nicht erneut geschrieben werden Diese nenne ich im Folgenden Abgleichquelle. Für die Datastores steht ebenfalls eine Klasse zur Verfügung, hier wird dem Konstruktor das Mapping mitgegeben, in das der Datastore eingefügt werden soll. Die Datastores werden ebenfalls in Variablen gespeichert, damit man sie referenzieren kann.

Ich gehe nun zunächst auf die Erstellung eines Satellitenmappings ein.

Hier wird zunächst der Datenstrom von der Abgleichquelle durch eine Expression geleitet.

Wie alle Komponenten wird die Expression durch den Konstruktor der entsprechenden Klasse erzeugt. Dann werden alle für den Abgleich nötigen Spalten als Attribute in die Expression eingefügt, wofür die Klasse eine Prozedur mitbringt. Die Namen der Attribute sind in Abgleichquelle, ziel und Expression gleich, die Ausdrücke sind einfach Übernahmen aus der Abgleichquelle, also

```
ABGLEICHQUELLE.ATTRIBUTNAME
```

Die Expression soll immer nur den Aktuellsten Datensatz zu jedem Business Key weitergeben, dies wird mit analytischen Funktionen realisiert:

```
MAX(MD_LOAD_ID) KEEP (DENSE_RANK LAST ORDER BY MD_LOAD_ID)
OVER (PARTITION BY <HUB_ID>)
```

Beim Quelldatenstrom wird ein Aggregator eingefügt - wiederum mit dem Konstruktor aus der entsprechenden Klasse - um Dubletten in der Quelle herauszufiltern. Hier werden alle Felder der Quelle die in die Zieltabelle geschrieben werden sollen in den Aggregator aufgenommen. Es wird nach allen Feldern außer MD_DWH_ID gruppiert und von der MD_DWH_ID wird das Maximum genommen.

Danach werden die beiden Datenströme zusammengejoint. Hiebei wird als Schlüssel die HUB_ID verwendet und ein Outer Join gemacht, so dass alle Datensätze der Quelle auf jeden Fall weitergeleitet werden und mit dem Datensatz der Abgleichquelle zur gleichen HUB_ID sofern Vorhanden verbunden werden.

Im anschließendem Filter werden die Datensätze durchgelassen zu deren HUB_ID es noch keinen Eintrag in der Abgleichquelle gab, oder sich in einem der nicht Metadatenattribute unterscheiden.

Zum Schluss wird noch der Targetdatastore mit dem Filter verbunden und die Spalten mit den Werten aus der Quelle verbunden.

Andere Mapping Typen

Für Hub Mappings ist das Vorgehen ähnlich, nur gibt es hier keine weiteren Spalten die zu Überprüfen sind, und die Einträge werden nicht aktualisiert, so dass nur ein Abgleich der vorhandenen Schlüssel mit den neuen Schlüsseln durchgeführt werden muss.

Für Link Mappings müssen ebenfalls nur die vorhandenen Verknüpfungen mit den neu gelieferten Verknüpfungen verglichen werden um keine Verknüpfung doppelt einzufügen, jedoch sind hier mehr Schlüsselspalten zu betrachten.

Fazit

Anstatt immer wieder gleichartige Mappings manuell zu erzeugen bietet es sich an diesen Prozess zu automatisieren. Dadurch kann man eine Vielzahl an Mappings in gleichbleibender Qualität in hoher Geschwindigkeit entwickeln.

Kontaktadresse:
Martin Wichert
CGI Deutschland
Am Limespark 2
D-65843 Sulzbach

Telefon: +49 (0) 178 9277847
Fax: +49 (0) 6196 7742555
E-Mail Martin.Wichert@cgi.com