DWH-Automatisierung mit ODI12c
- Entwicklung und Deployment -

Stefan Raabe
C.ST Raabe
Hamburg

Schlüsselworte
ODI, 12c, Oracle Data Integrator, DWH, DWA, Datenintegration, Automatisierung, Data Vault, Standardisierung, Agile, Versionierung, API, SDK, Lifecycle Management

Einleitung
Rein manuelle Entwicklung von Datawarehouse Systemen war gestern! An der (Teil-) Automatisierung der Erstellung von Datenintegrationsprozessen führt kein Weg mehr vorbei, denn durch die fortschreitende Digitalisierung von Geschäftsprozessen steigen die Datenmengen in Unternehmen kontinuierlich an und auch der Bedarf diese neuen Daten schnell auswerten zu können wächst. Mit Release-Zyklen von Monaten und Quartalen für die Anbindung neuer Quellen im Data Warehouse gibt sich heute niemand mehr zufrieden. In agil gesteuerten Projekten müssen Änderungen schon eher im Wochenrythmus geliefert werden.

Durch den Einsatz moderner Modellierungstechniken - wie Data Vault - und der damit einhergehenden Standardisierung, wird mittlerweile ein Großteil von Datenintegrationsprozessen anhand der immer gleichen Muster erstellt. Ideale Ausgangsbasis für die automatisierte Generierung dieser Prozesse. Aber funktioniert das auch, wenn ein grafisches ETL-Werkzeug wie der Oracle Data Integrator eingesetzt wird?

Dazu wird im Folgenden ein Überblick über die Möglichkeiten der DWH-Automatisierung im ODI gegeben.

Gründe für DWH-Automatisierung
Doch was sind im Detail die Gründe sich mit DWA (= DWH-Automatisierung) zu beschäftigen?
Hinter DWA verbirgt sich nicht nur die Möglichkeit eine höhere Umsetzungs- bzw. Ausliefergeschwindigkeit in Datawarehouse-Projekten zu erreichen, sondern sie unterstützt an vielen Stellen im Prozess: von der Entwicklung, über die Qualitätssicherung, bis hin zu Deployment und Changemanagement.
In der Entwicklung liegt der Vorteil der DWH-Automatisierung auf der Hand. Bisher manuell erstellte Datenintegrationsprozesse werden durch metadatengesteuerte Generierung auf Knopfdruck erzeugt, dadurch entstehen konsistente ETL-Prozesse mit dem immer gleichen, getesteten und optimierten Code.
Eine Standardisierung von Vorgehensweisen ist dabei eine zwingende Voraussetzung für die Automatisierung. Diese führt aber gleichzeitig dazu, dass diese Standards auch eingehalten werden und dadurch das Fehlerpotenzial verringert wird und die Transparenz über die Prozesse steigt. Zusätzlich ermöglicht dies eine leichtere Einbindung von Testautomatisierung, was in Summe mit den anderen Punkten auf Dauer zu einer höheren Qualität im DWH führt.

Wenn zusätzlich noch der Deployment-Prozess automatisiert wird, lässt sich auch das gesamte Changemanagement verbessern. Denn gerade in größeren Projekten, in denen teils mehrere Entwickler an unterschiedlichen Release-Ständen arbeiten ist es schwer die Übersicht zu behalten. Die automatische Generierung von Deployment-Paketen bietet hier eine Lösung.

Neben den prozessualen Verbesserungen, optimiert DWA auch den Ressourceneinsatz. In Zeiten der Entwickler-Knappheit muss der Einsatz der zur Verfügung stehenden Kapazitäten optimiert werden. So sollten sich Entwickler nicht mit repetitiven Tasks aufhalten, die nicht nur langweilig, sondern auch fehleranfällig sind. Durch die Automatisierung gerade dieser Aufgaben, kann die Konzentration der Entwickler auf komplexere, höherwertige Themen gerichtet werden.

Schlussendlich sollte es durch die aufgezeigten Vorteile der DWA gelingen die Entwicklungs- und Betriebskosten zu verringern, oder mehr mit dem zur Verfügung stehenden Budget zu erreichen.

Automatisierung im ODI

Also sollte es sich durchaus lohnen über Automatisierung nachzudenken, selbst wenn bereits ein „klassisches" ETL-Werkzeug für die Umsetzung von Datenintegrationsprozessen genutzt wird.
Die nächsten Abschnitte beschreiben dafür sieben Methoden, die der ODI für die Umsetzung von DWH-Automatisierung bietet und bei Welchen es sich lohnt diese näher zu betrachten.

Knowledge Module

Der Oracle Data Integrator ist durch seine grundsätzliche Funktionsweise von jeher ideal für das Thema Automatisierung geeignet. Die Knowledge Module, als integraler Bestandteil, sind bereits der erste Schritt der DWA, da sich hierüber viel Prozesslogik wiederholbar abbilden lässt ohne diese jedes Mal explizit in einem Mapping implementieren zu müssen. Dazu gehört zum Beispiel das Anlegen temporärer Objekte, das Ab-/Anschalten von Indizes, das Analysieren von Statistiken, die Verwaltung von Slowly Changing Dimensions und vieles mehr.

monolithic substitution API

Die Knowledge Module selbst sind dabei als Templates aufgebaut und nutzen vielfach die „monolithic substitution API" des ODI, bzw. das odiRef-Objekt, um dynamisch auf Bestandteile des logischen Mapping-Designs, oder auf andere Metadaten des ODI-Repository zuzugreifen und diese zu manipulieren. So zeigt beispielsweise Listing 1 den Code, der innerhalb von Integration Knowledge Modulen verwendet wird um aus der Zieltabelle innerhalb eines Mappings eine strukturell identische Integrationstabelle zu erstellen.
Für Automatisierungszwecke kann die odiRef-API verwendet werden um Knowledge Module zu erstellen/anzupassen, oder innerhalb von Mappings und Packages Code dynamisch einzufügen.

```
create table <%=odiRef.getTable("L", "INT_NAME", "W")%>
(
   <%=odiRef.getColList("",  "[COL_NAME]\t\t[DEST_WRI_DT]  NULL",  ",\n\t",
"", "")%>,
   IND_UPDATE CHAR(1)
)
<%=odiRef.getOption("FLOW_TABLE_OPTIONS")%>
```

Listing 1: Code der odiRef-API zum dynamischen Erzeugen der Integrationstabelle

Das Repository
Um auf die Design- oder Laufzeitmetadaten des ODI zuzugreifen, können die Repository-Tabellen auch direkt abgefragt werden. Das bietet sich insbesondere an, um die Ausführungsinformationen der Datenintegrationsprozesse auszuwerten und diese beispielsweise in ein automatisiertes Monitoring einzubinden.

Diese Methode sollte jedoch mit Bedacht eingesetzt werden, da die Struktur des Repositories mit jedem ODI-Release Änderungen unterliegen kann und die Abfragen später eventuell ins Leere laufen.

ODI Tools
Eine weitere Möglichkeit für die Automatisierung von Vorgängen im ODI sind die ODI Tools. Das sind vordefinierte Kommandos, die parametrisiert innerhalb von Packages und Prozeduren innerhalb des ODI aufgerufen werden können, oder direkt über einen Agent auf Kommandozeilenebene.

Im Rahmen des Lifecycle Management bieten sich die Tools zur Unterstützung des Deploymentvorgangs an. So gibt es vordefinierte Tools für die Regenerierung, den Export und Import von Szenarien und in den neusten Versionen des ODI werden auch die Arbeit mit Deployment Archiven (mehr dazu im letzten Abschnitt) durch ODI Tools unterstützt. Abb. 1 zeigt ein ODI-Package in dem mit Hilfe von ODI Tools alle Szenarien eines Projektes, deren Mappings und Packages mit dem Marker „Releasestand“ versehen sind, erst erstellt und dann exportiert werden.

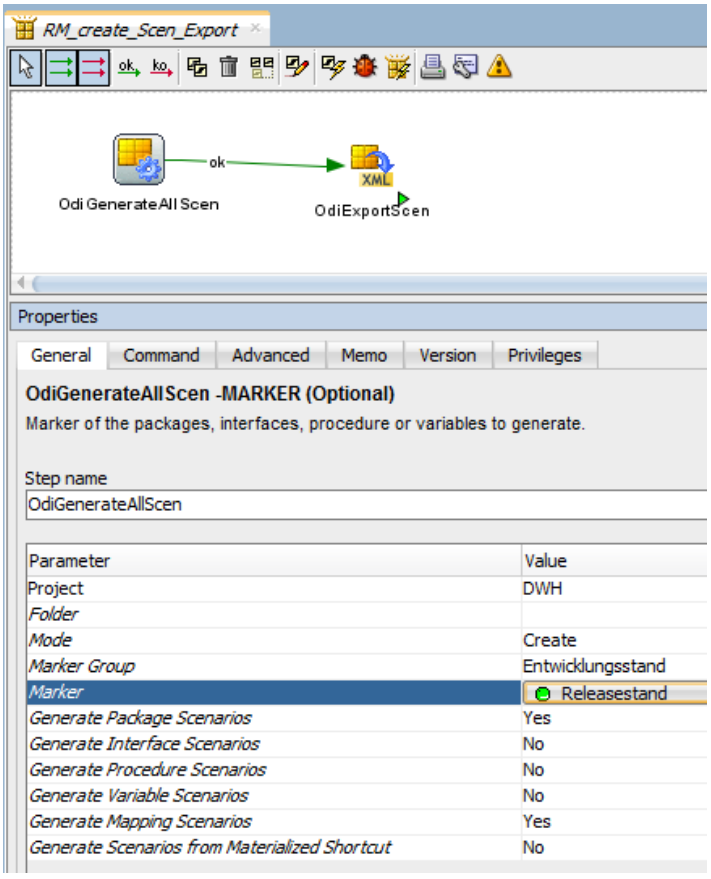

Abb. 1: Package für Release Management mit ODI Tools

Ein großer Vorteil der ODI Tools ist, dass sie auf Kommandozeilenebene ausgeführt werden können und damit beispielsweise geeignet sind per Skript Änderungen einzuspielen, ohne manuelle Interaktion über das Design Studio.

Ein Nachteil ist jedoch, dass die Tools trotz der möglichen Parametrisierung, in ihrem Umfang limitiert sind. So kann beim Exportieren von Szenarien die betreffende Menge durch Filter zwar eingeschränkt werden, aber über die reine Gruppierung anhand von Markern und Ordnern reichen die Möglichkeiten hier nicht hinaus. Eine feinere Unterteilung ist nicht möglich.

ODI SDK Public API

Viel interessanter für die Betrachtung der Automatisierungsmöglichkeiten des ODI ist daher die bestehende Java API, bzw. das ODI SDK, das seit Version 11g des ODI ein Bestandteil Desselbigen ist. Über diese Java Schnittstelle lässt sich so gut wie alles, was auf der grafischen Oberfläche des ODI Studios umgesetzt werden kann auch als Skript ausführen. Das reicht von der einfachen Anpassung an mehreren Objekten gleichzeitig, über die komplette Erstellung von Mappings, inkl. Generierung der zugehörigen Szenarien, bis hin zum Erzeugen von Deployment Archiven.
Eine ideale Möglichkeit also um wiederholbare Aufgaben zu automatisieren, oder gleich komplette DWH-Bestandteile zu generieren.

Für einfaches Skripting kann auf die SDK dabei mit Groovy zugegriffen werden. Das ODI Studio enthält hierfür eine Konsole, bei welcher der Code direkt gegen das verbundene Repository ausgeführt wird (siehe Abb. 2).

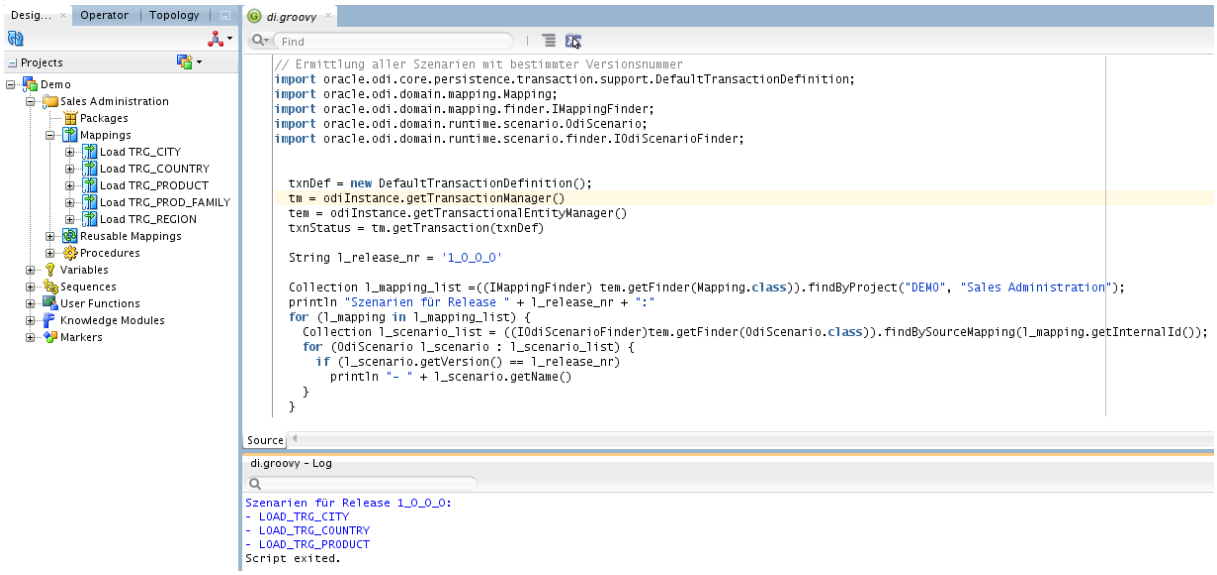

Abb. 2: Groovy-Scripting im Design Studio

Für die Erstellung umfangreicherer Groovy-Skripte, oder ganzer Java-Programme, auf Basis der SDK bietet es sich aber an eine eigenständige IDE wie Eclipse, JDeveloper oder IntelliJ IDEA zu nutzen. Denn Funktionen wie Code-Completion, Refactoring und der automatischen Import von benötigten Java-Klassen vereinfachen die Arbeit mit der ODI SDK ungemein.

Versionierung
Insbesondere in den letzten Releases hat Oracle viel Wert auf die Verbesserung des Lifecycle Management im ODI gelegt. So ist beispielsweise seit Version 12.2.1.0.0 die Anbindung von Subversion für die Versionskontrolle direkt aus dem Design Studio möglich und mit Version 12.2.1.2.6 kam auch die Unterstützung von Git dazu. Damit können nun durch, in der reinen Softwareentwicklung bereits etablierte, Vorgehensweisen wie Branching / Tagging mehrere Entwicklungspfade gleichzeitig bearbeitet und getestet werden. Der Aufbau eines neuen Entwicklungsrepositories aus einem bestimmten Branch, oder die Wiederherstellung eines bestimmten Release-Standes sind damit kein Problem mehr.

Deployment Archive
Die mit Version 12.2.1.0.0 eingeführten Deployment-Archive zielen außerdem konkret darauf ab das Releasemanagement zu verbessern, indem bei Ihrer Erzeugung automatisch je ein Archiv mit allen zugehörigen Designobjekten und ein Archiv mit ausschließlich den ausführbaren Objekten (Szenarien und Ladeplänen) erstellt wird. Dadurch ist es möglich einen Change inkl. aller Objekte auf einem weiteren Entwicklungsrepository, zum Beispiel für eine Hotfix-Umgebung, einzuspielen und den gleichen Change auch auf ein reines Ausführungsrepository, wie es meist in der produktiven Umgebung eingesetzt wird, zu bringen.

Ein weiterer Vorteil der Deployment-Archive ist, dass bei ihrem Einspielen automatisch ein passendes Rollback-Archiv erzeugt werden kann, mit dem das betreffende Repository wieder auf den Zustand versetzt werden kann, der vor dem Einspielen bestand. Wenn also nach dem Einspielen eines neuen Releases auf einer produktiven Umgebung unvorhergesehene Fehler auftreten, dann kann problemlos der vorhergehende Stand wiederhergestellt werden.
In Kombination mit den bereits beschriebenen ODI Tools kann dieser Vorgang auch komplett von der Kommandozeile ausgeführt werden um Fehlern beim manuellen Einspielen vorzubeugen. Abb. 1 zeigt den zugehörigen Befehl.

```
File  Edit  View  Search  Terminal  Help
[oracle@                bin]$ cd /u01/Middleware/ODI12c/user_projects/domains/agent_domain/bin
[oracle@                bin]$ ./startcmd.sh -INSTANCE=PRD_OracleDIAgent1 OdiApplyDeploymentArchive
"-ARCHIVE_FILE_NAME=/media/sf_Share/05-odi/Deployment_Archive/R_1_0_0_0_Topology.zip" "-APPLY_WITHOUT
_CIPHER_DATA=NO" "-EXPORT_KEY=1ImOlRUPWxjxOofCpMKnVw==" "-CREATE_ROLLBACK_ARCHIVE=YES" "-ROLLBACK_FIL
E_NAME=/media/sf_Share/05-odi/Deployment_Archive/ROLLBACK_R_1_0_0_0_Topology.zip" "-INCLUDE_PHYSICAL_
TOPOLOGY=YES"
```

Abb. 3: Kommando zum Einspielen eines Deployment Archivs

Ausblick
Die beschriebenen Möglichkeiten zeigen, dass der ODI durchaus geeignet ist viele Vorgänge bei der Erstellung und Verwaltung von DWH-Umgebungen zu automatisieren. Allerdings ist es aufgrund der Vielzahl der zur Verfügung stehenden Schnittstellen und Tools nicht immer ganz einfach zu entscheiden was an welcher Stelle eingesetzt werden sollte.

Der Vortrag wird daher an konkreten Beispielen demonstrieren, wie komplette Ladestrecken mit Hilfe der ODI SDK automatisch generiert werden können und wie durch Nutzung der Deployment Archive in Verbindung mit der Versionierung in Git das Deployment einzelner Releases automatisiert werden kann.

Kontaktadresse:
Stefan Raabe
C.ST Raabe
Consulting – Solutions – Training
Gärtnerstr. 23
D-25469 Halstenbek

Telefon: +49 151 175 172 23
E-Mail consulting@st-raabe.de