ODI Projekt Lifecycle – Segen oder Fluch

Michael Klose
Logica Deutschland GmbH & Co. KG, now Part of CGI
Sulzbach (Taunus)

Schlüsselworte
Oracle Data Integrator, Entwicklung, Best Practice, Datenintegration, Datenqualität, ETL, Datawarehouse

Einleitung
Der Einsatz des Oracle Data Integrator bietet vor allem in heterogenen Systemlandschaften Vorteile gegenüber dem Oracle Warehouse Builder. Anhand eines Kundenprojekts werden die besonderen Herausforderungen und Fallstricke im Projekt Lifecycle mit unterschiedlichen Datenbanken (Oracle, DB2, MySQL) aufgezeigt. Daraus ergeben sich Best-Practice Ansätze welche an praktischen Beispielen veranschaulicht werden und Themen umfassen wie die Anbindung verschiedener Quell- und Ziel-Datenbanken, Topologie und Ordnerstrukturen, User Defined Functions und Data Quality Regeln, Best Fit Knowledge Module, Versionierung, LoadPlans sowie das Deployment.

Anbindung verschiedener Quell- und Ziel Datenbanken
Der Oracle Data Integrator hat seine Stärken vor Allem in sehr heterogenen Systemlandschaften mit Nicht-Oracle Datenbanken. Im Projekt bestand die Herausforderung als Datenquellen MySQL Datenbanken und Dateien (CSV) einzubinden. Das ODI Repository und Datawarehouse ist auf einer DB2 UDB 9.7 realisiert. Weiterhin werden Auszüge der im Datawarehouse aufbereiteten Daten in eine Oracle Datenbank oder als Datei exportiert.

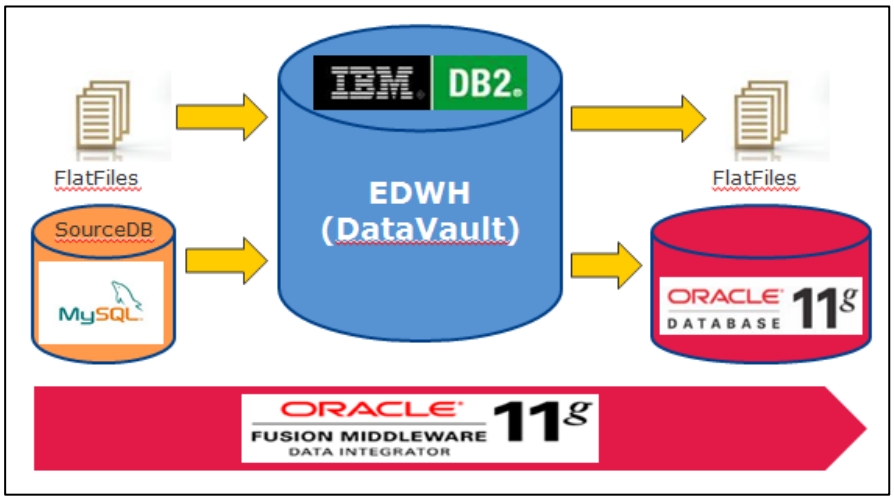

Abb. 1: Systemlandschaft

Aus Kostengründen wurde der direkte Konnektor von DB2 zu MySQL (vergleichbar mit Database Gateways in Oracle) nicht lizensiert. Dadurch bestand die Herausforderung einen performanten Weg zu finden die Daten aus den operativen Systemen in den Staging Layer des Data Warehouse zu

transportieren. Mögliche Alternativen waren der Export in eine Datei und das Laden über den ODI Agent.
In kleinen Proof of Concepts wurden die Technologien gegenübergestellt um die Vor- und Nachteile abzuwägen. Der Transport über Dateien (CSV) ist zwar aus Performancesicht der bessere Weg, allerdings musste viel Textkontext bewegt werden welcher Steuerzeichen und Zeilenumbrüche beinhaltete und somit nicht ohne weiteres mit den Standardverfahren (LOAD in DB2) abgebildet werden konnte.
Somit stellte sich der Transport der Daten über den ODI Agent als praktikabelster Weg heraus. Allerdings ist hier zu beachten, dass die „Array Fetch Size" und „Batch Update Size" in der physikalischen Architektur im ODI Repository entsprechend angepasst werden müssen. Diese Parameterwerte sollten nicht zu klein gewählt werden (vergleichbar mit Bulk Size in PL/SQL) um hier eine möglichst gute Performance zu erzielen. Dadurch steigt natürlich das Datenvolumen und somit auch der Arbeitsspeicherbedarf für das Result Set (durch die Array Fetch Size bestimmt) im ODI Agent. Dies bedingt dass in der Datei odiparams.sh die Speicherparameter angepasst werden, da ansonsten der Fehler „Out of Memory" während der Datenextraktion auftritt.

```
ODI_MAX_HEAP = 4096M  (Default 256M)
ODI_INIT_HEAP  =  256M  (Default 32M)
```

Diese Einstellungen ermöglichen auch größere Fetch Sizes. Voraussetzung ist, dass der Server mit entsprechend Speicherplatz ausgestattet ist. Falls der JDBC Treiber Streaming unterstützt sollte diese Option unbedingt zum Einsatz kommen.
Empfehlenswert ist auch der Einsatz mehrerer ODI Agenten. Dies bietet Vorteile, z.B. dass der speicherintensive Ladeprozess über JDBC nicht mit anderen Ladeprozessen innerhalb des Datawarehouse kollidiert. Im Projekt werden getrennte Agenten für den Datei-, Datenbank- und Extraktionsprozess eingesetzt welche auf die jeweilige Anforderung optimierte Speicherparameter besitzen.

Topologie und Ordnerstrukturen
Häufig werden in Projekten mit toolgestützter Entwicklung keine klaren Vorgaben bezüglich der Benennung von Objekten vorgegeben. Dies führt in größeren Projektteams während der Entwicklung und spätestens bei der Übergabe an den Betrieb oder bei der Weiterentwicklung zu erhöhten Aufwänden. Diese begründen sich z.B. auf Dubletten (z.B. Verbindungen werden doppelt angelegt), „Copy-Of"-Interfaces (Versionierung nicht durchgeführt) oder auch User Functions welche mehrfach für den gleichen Zweck angelegt werden.

Im Bereich der Topologie ist es sinnvoll einheitliche Benennungsregeln für die Daten Server der physikalischen Architektur und die logischen Schemas in der logischen Architektur festzulegen. Erfolgt dies nach klaren Regeln ist in der Folge die Zuordnung der Kontexte und das Arbeiten im Model wesentlich vereinfacht da Entwickler, Architekten und Administratoren genau wissen mit welchem System und welcher Technologie sie gerade arbeiten.

Beispiel für Namenskonventionen:
Physikalische Architektur Data Server:
Syntax: `<TECHNOLOGIE>_<HOST>_<DBINSTANZ>` → `Oracle_ProdSrv_odwhprd`
Logische Architektur Logical Schema:
Syntax: `<TECHNOLOGIE>_<FACHLICHER SYSTEMNAME>_<LAYER>` → `Oracle_EDWH_Stage`

Vergleichbares muss im Entwicklerbereich innerhalb der ODI Projektstruktur vorgegeben werden um die spätere Wartbarkeit zu gewährleisten. Dafür ist es unbedingt notwendig klare Ordnerstrukturen

innerhalb eines Projekts einzuführen, die durchaus nicht auf Layer-Ebene (z.B. Stage, Foundation, Access) enden. In der folgenden Abbildung ist ein Beispiel aufgeführt.
Der Stage Ordner wurde hier aus technischer Sicht gesehen in Unterordner für verschiedene Quellsysteme (Datei- und Datenbankbasiert) unterteilt. Die Interfaces welche die Daten aus den verschiedenen Systemen laden werden in die entsprechenden Ordner gespeichert. Somit wird vermieden dass im Ordner Stage z.B. hunderte von Interfaces abgelegt sind und die Übersichtlichkeit verloren geht.
Im Foundation Ordner wird hingegen eine eher fachliche Unterteilung durchgeführt. Als erstes werden Ordner erstellt welche die Daten nach Stamm- und Bewegungsdaten unterscheiden. Innerhalb dieser beiden Ordner erfolgt die weitere Strukturierung nach fachlichen Domänen wie Kunde, Auftrag und Rechnung.
Im Access Ordner welcher die Data Marts repräsentiert kann eine Ordnerstrukturierung nach fachlichen Bereichen wie Controlling und Marketing durchgeführt werden. Alle Interfaces welche Dimensionen- und Faktentabellen für den Controlling-Bereich befüllen sind im Ordner Controlling abzulegen.
Durch diese klaren Strukturen findet selbst ein neuer Mitarbeiter im Projektteam schnell die entsprechenden Interfaces welche z.B. die Kundentabellen im Foundation Layer befüllen.

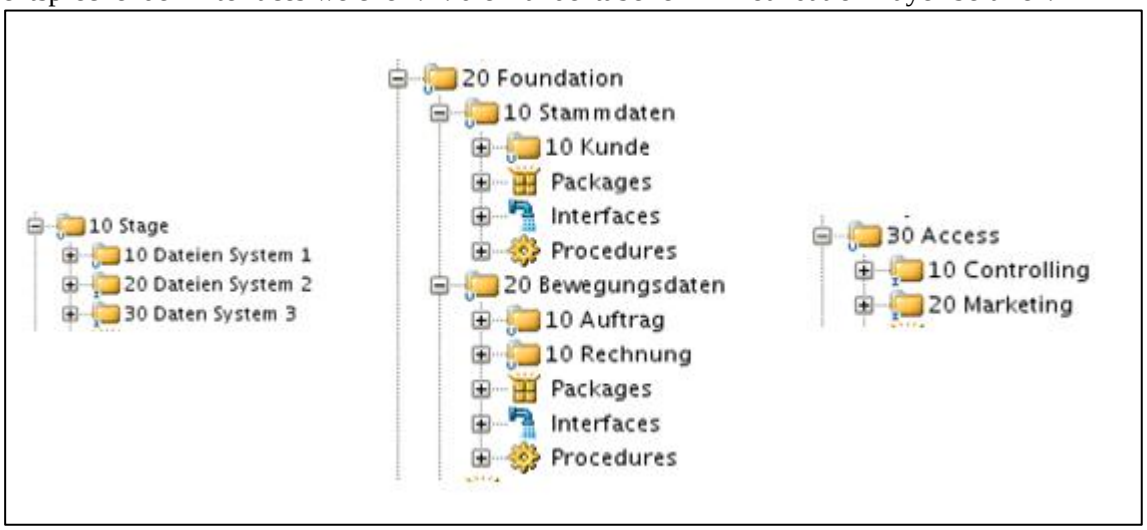

Abb. 2: Ordnerstrukturen im ODI

User Defined Functions und Data Quality Regeln
Im ODI besteht die Möglichkeit sogenannte User Defined Functions (vergleichbar mit Datenbankfunktionen) zu erstellen. Diese Funktionen können Parametrisiert und für die verschiedenen Technologien (Oracle, MySQL, DB2,…) und somit Implementierungen gesondert definiert werden. Im Projektbeispiel werden die User Functions z.B. für Typtransformationen (String → Timestamp, String → Number) oder auch Datenqualitätsprüfungen (z.B. Geburtsdatum) eingesetzt.
Die so abgelegten Typkonvertierungen werden beim Laden der Daten in Staging oder Foundation häufig verwendet. So muss ohne Verwendung von User Functions in den Mappingn Spalten bei jeder Konvertierung der komplette Code (z.B. TO_NUMBER(umsatz, '9999.99') eingetragen werden. Die hinterlegte Funktion vereinfacht die Erstellung, bietet aber auch den großen Vorteil, dass bei Änderungen (z.B. das gelieferte Format ändert sich und die Konvertierung muss geändert werden) lediglich die User Function geändert werden muss. Anschließend werden die Interfaces welche die User Function verwendet neu getestet und ausgeliefert. Beschreitet man den konventionellen Weg müsste jedes Mapping in welchem die Transformation durchgeführt wird angepasst werden.

Gleiches gilt für das Beispiel der Datenqualitätsprüfung. Wird hier die Data Quality Rule erweitert oder angepasst ist dies ohne weitere Eingriffe in die Interfaces möglich.

Best Fit Knowledge Module

Der ODI bringt eine Vielzahl von Knowledge Modulen für diverse Technologien und Lademechanismen bereits mit. In diesen sind sämtliche Funktionalitäten wie z.B. Journalizing, Flow Control und Check enthalten. Sicherlich macht der Einsatz dieser Knowledge Module ohne größere Anpassungen Sinn wenn diese Funktionalitäten im Projekt auch wirklich benötigt und eingesetzt werden. Anderenfalls ist es empfehlenswert die mitgelieferten Knowledge Module auf seine eigenen Bedürfnisse im Projekt anzupassen und den Umfang und Komplexität zu reduzieren. Dafür ist es nicht notwendig „das Rad neu zu erfinden" und somit das Knowledge Modul komplett neu zu schreiben. Vielmehr kann auf bestehende Module zurückgegriffen und der benötigte Inhalt kopiert werden.

Sehr gut lässt sich dies Anhand eines Beispiels darstellen. Die Daten sollen hierfür aus einer anderen Datenbank 1:1 in den DWH Staging Layer geladen werden ohne Journalizing, Check oder Flow Control. Dafür können z.B. die zwei folgenden Knowledge Module verwendet werden:
Das Load Knowledge Modul „LKM SQL to Oracle" für den Extrakt aus dem Quellsystem und das Integrations Knowledge Modul „IKM SQL Control Append" um die Daten in die Staging Tabelle zu laden.
LKM: Das Knowledge Modul erstellt eine C$-Arbeitstabelle in welches es die Daten lädt
IKM: Das Knowledge Modul lädt die Daten aus der C$-Arbeitstabelle in die Stage Tabelle

Der ODI erfordert hier zwingend zwei Knowledge Module. Nachteil dieser Methode ist, dass die Daten zweimal geschrieben (C$ und Stage Tabelle) und zweimal gelesen (Quellsystem über JDBC und C$ Tabelle) werden müssen. Dadurch wird das Storage System mit zusätzlichen Lese- und Schreiboperationen belastet welche keinen Mehrwert generieren.

Der Lösungs- und Optimierungsansatz im Projektbeispiel war, dass das bestehende LKM dahingehend angepasst wird dass die Daten direkt in die Staging Tabelle eingefügt werden und nicht in die C$-Arbeitstabelle. Dafür ist lediglich eine kleine Codeanpassung im Knowledge Modul und das Entfernen dann überflüssiger Schritte notwendig.
Nun muss ein neues IKM erstellt werden, da das bestehende die Daten aus der C$-Arbeitstabelle zu laden versucht, diese aber bereits in der Stage Tabelle gespeichert sind. Das neue IKM kann aus dem bestehenden erstellt werden. Hierfür alle Schritte aus dem Knowledge Modul löschen und diesem einen neuen Namen (z.B: IKM_Dummy) geben um zu verdeutlichen dass keine Arbeitsschritte durchgeführt werden.

Versionierung, Load Plans und Deployment

Seit der ODI Version 11.1.1.5 besteht die Möglichkeit Load Plans zur Ausführung der Ladeprozesse zu erstellen. Damit ist der Einsatz eines externen Schedulers nicht mehr notwendig, da auch die Anforderungen an Parallelisierung, Exception Handling, Scheduling und Wiederaufsetzbarkeit damit durchgeführt werden können. Die Load Plans können dann mit den restlichen Entwicklungsobjekten (Szenarios) zu einer Solution zusammengefasst werden und an das Test- und Produktivsystem übergeben.
Um eine Solution sinnvoll zu erstellen und eine Weiterentwicklung durchzuführen sollte das ODI Versionsmanagement verwendet werden. Nahezu jedes ODI Objekt (Model, Interface, Package, Procedure, Szenario,…) kann versioniert werden.
Die Versionierung im ODI entspricht nicht der eines Code Versionierungs Werkzeuges inkl. aller Automatismen sondern muss durch Vorgaben für die Entwickler gesteuert werden. Empfehlenswert ist

die Einführung eines Versionierungslogik. Diese könnte z.B. vierstellig sein, wobei die erste Stelle für Major Releases, die zweite für Minor Releases und die dritte für Hotfixes verwendet wird. Die vierte Stelle steht den Entwicklern zur Verfügung um auch während der Entwicklungsphase „Versionen“ erzeugen zu können.
Historisch gewachsen gibt es beim ODI noch eine zweite Art der Version im Bereich Szenario. Wird z.B. aus einem Interface ein Szenario generiert wird eine Versionsnummer (beginnend bei 001) für den Szenario Namen generiert. Diese hat NICHTS mit der Versionierung wie vorher erläutert zu tun und muss daher mit Vorsicht eingesetzt werden.

Fazit

Der Oracle Data Integrator bringt sehr viele Funktionen mit um den kompletten Datenbewirtschaftungsprozess und Data Warehouse Lifecycle Metadatengestützt für ein Enterprise Datawarehouse abzubilden. Ebenso wie bei Nicht-Toolgestützter Entwicklung ist es unbedingt erforderlich Richtlinien in Form von Namenskonventionen und Prozessen VOR der Projektdurchführung festzulegen.

Kontaktadresse:

Michael Klose
Logica Deutschland GmbH & Co. KG | Now part of CGI
Am Limespark 2
D-65843 Sulzbach (Taunus)

Telefon: +49 (0) 171-977 90 99
E-Mail Michael.Klose@logica.com
Internet: www.logica.com | www.cgi.com