Das ungleiche Paar – Koexistenz von OWB und ODI

Till Sander
OPITZ CONSULTING GmbH
Hamburg

Schlüsselworte:

Oracle Warehouse Builder, Oracle Data Integrator, Code Templates, Knowledgemodule, Mappings, Interfaces

Einleitung

Der Oracle Data Integrator (ODI) und der Oracle Warehouse Builder (OWB) Enterprise Edition werden über die Data Integrator Enterprise Edition gemeinsam lizensiert, beide Produkte können also prinzipiell gemeinsam benutzt werden.
Es stellt sich die Frage, ob im Rahmen von Neu- oder Weiterentwicklungen tatsächlich beide Produkte benutzt oder ob beispielsweise die "gleitende" Migration von OWB zum ODI das in diesem Kontext das relevante Szenario ist.
Die verschiedenen Facetten eines solchen Zusammenspiels werden nachfolgend ausführlich betrachtet.

ETL-Werkzeuge im Oracle-Produktportfolio

Bis zum Zukauf der Firma Sunopsis im Jahre 2006 war die Produktvielfalt im Bereich ETL bei Oracle sehr übersichtlich. Der Oracle Warehouse Builder wurde als einziges Produkt angeboten. Die Basisfunktionalitäten des OWBs, die für die allermeisten ETL-Projekte ausreichten, konnten zudem mit der Datenbanklizenz genutzt werden. So fiel die Wahl eines ETL-Werkzeuges in einer Vielzahl von Projekten auf den OWB.
Begründet man die starke Verbreitung lediglich mit den geringen Kosten, so wird man diesem Werkzeug sicherlich nicht gerecht. Bewegt man sich in einem überwiegend von Oracle Datenbanken und Textdateien geprägtem Umfeld, so ist der OWB nach wie vor eines der besten Werkzeuge in diesem Bereich. In einigen Bereichen bietet der OWB sogar Funktionen und Möglichkeiten, die so von keinem anderen ETL-Werkzeug beherrscht werden. Als Beispiele sind hier etwa die dimensionale Modellierung und das implizite Generieren von Code für das Laden von Dimensionen vom Typ SCD2 und SCD3 genannt (siehe Abbildung 1). Weiterhin ist der generierte Code, abgelegt als PL/SQL-Package, konsequent auf schnellen Datendurchsatz getrimmt, so dass es schwerfällt, mit einer manuellen Programmierung ein gleicher Zeit eine ähnliche Performance zu erzielen.

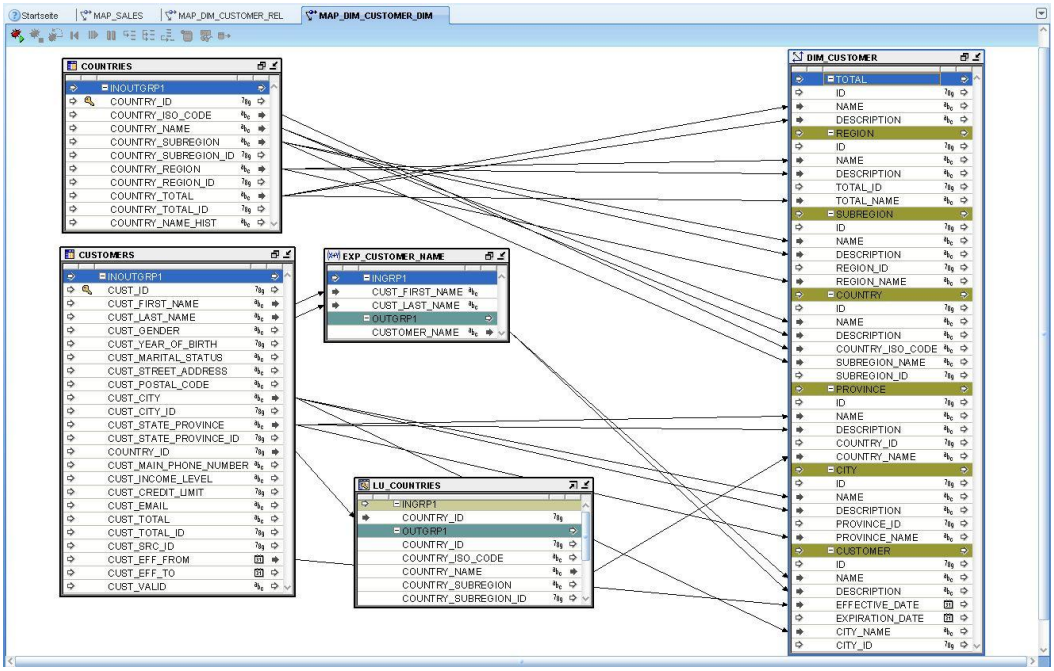

Abbildung 1: Das "Mapping" ist zentraler Bestandteil des OWBs. In diesem Fall wird eine Dimension befüllt. Durch Definition der Metadaten kann der OWB einen Großteil des Codes implizit im Hintergrund generieren, (Joins bzw. Lookups, SCD2/SCD3).

Nun ist die Welt der Datenintegration in vielen Fällen nicht so homogen auf Oracle abgestimmt, wie es sich vielleicht manch einer wünscht. DWH-Projekte haben immer mehr die Aufgabe, vielfältigste Datenquellen zu integrieren. Damit wachsen die Anforderungen an die beteiligten Werkzeuge. Diesen Trend hat Oracle frühzeitig erkannt und mit dem Zukauf von Sunopsis den Oracle Data Integrator (ODI) in sein Produktportfolio aufgenommen. Im Gegensatz zum OWB kann der ODI die unterschiedlichsten Datenquellen und –ziele bedienen. Mit dem Konzept der "Knowledgemodule" (KM) ist es zudem möglich, eigene ETL-Abläufe in der Sprache der Quellen/Ziele zu definieren und dynamisch und parametrisierbar in die ETL-Abläufe zu integrieren. Es werden KMs für alle Phasen der Datenintegration und –aufbereitung: Auslesen von Metadaten, nutzen von "Change Data Capture", Laden aus den Quellen, Prüfen der Daten, Integrieren der Daten ins Ziel und Weiterreichen von Informationen per Web Service (siehe Abbildung 2).
Damit kann der ODI als ETL-Framework bezeichnet werden, das dem Entwickler vielfältige Möglichkeiten der Erweiterung und Standardisierung bietet.

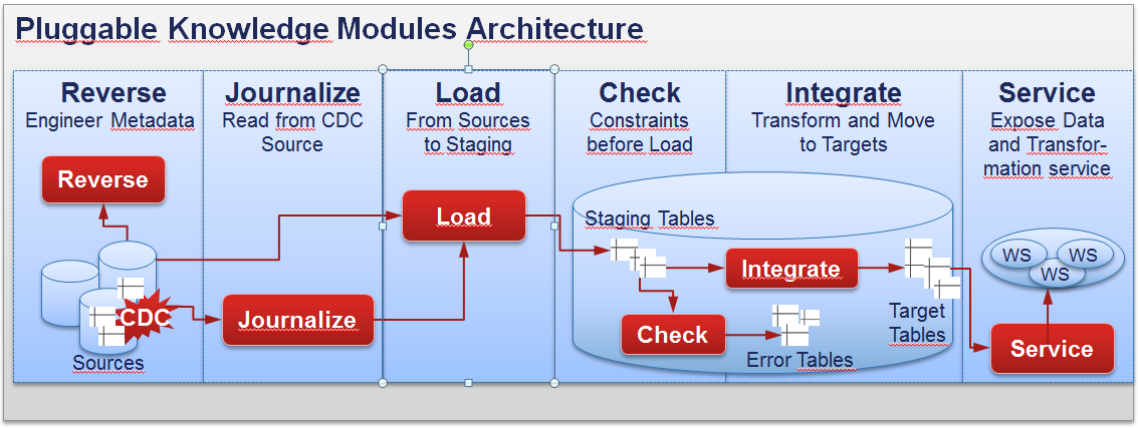

Abbildung 2: Einsatz der unterschiedlichen Arten von Knowledgemodulen im Integrationsprozess.

Viele Kunden fragen sie jetzt, wie die Zukunft beider Werkzeuge aussehen wird. Laut ETL-Roadmap von Oracle wird es mit der Version 12 nur noch ein (zu lizensierendes) ETL-Werkzeug geben. Dieses Werkzeug soll dann das Beste aus beiden Welten vereinen, denn nach wie vor sind OWB und ODI in ihren Funktionen nicht deckungsgleich.
Mit der Version 11.2 des OWBs wurden Funktionen des ODIs in den OWB übernommen. So können jetzt Knowledgemodule, im OWB "Code Templates" genutzt werden und ein nativer Zugriff über JDBC auf eine Vielzahl von Datenquellen ist jetzt möglich. Da hierfür aber die "Data Integrator Enterprise Edition" lizensiert werden muss, kann genauso gut der ODI genutzt werden. Damit ergeben sich interessante Szenarien, die nachfolgend betrachtet werden.

Erweiterung des OWBs um ODI-Funktionen
Beim Anlegen eines nicht-Oracle Datenbankmoduls fällt sofort auf, dass neben der Auswahl des "Transparent Gateways", eine "Native Database Connection" angeboten wird. Letzteres stellt die reizvolle Möglichkeit dar, statt eines teuren Gateways oder einer langsamen ODBC-Verbindung mittels JDBC nativ auf andere Datenquellen lesend und schreibend zuzugreifen. Weiterhin kann schon beim Anlegen des Datenbankmoduls das KM für CDC auf die Quelle angegeben und die zu journalisierenden Tabellen definiert werden.
Statt nun ein Mapping in der gewohnten Art und Weise zu erstellen, kann ein "Template Mapping" erzeugt werden. Im Gegensatz zum "normalen" Mapping, welches direkt einem Modul/DB-Schema zugeordnet ist, wird in dem Code Template Mapping durch Definition von Ausführungseinheiten ("Execution Units") bestimmt, welcher Teil des Mappings wo (Quellen und Zielen) ausgeführt wird (siehe Abbildung 3). Für jede Ausführungseinheit kann ein Integrations- und ein Check-KM angegeben werden. Die KMs können im OWB zwar nicht entwickelt, aber zumindest importiert werden, so dass auch hier individuelle Anpassungen möglich sind.

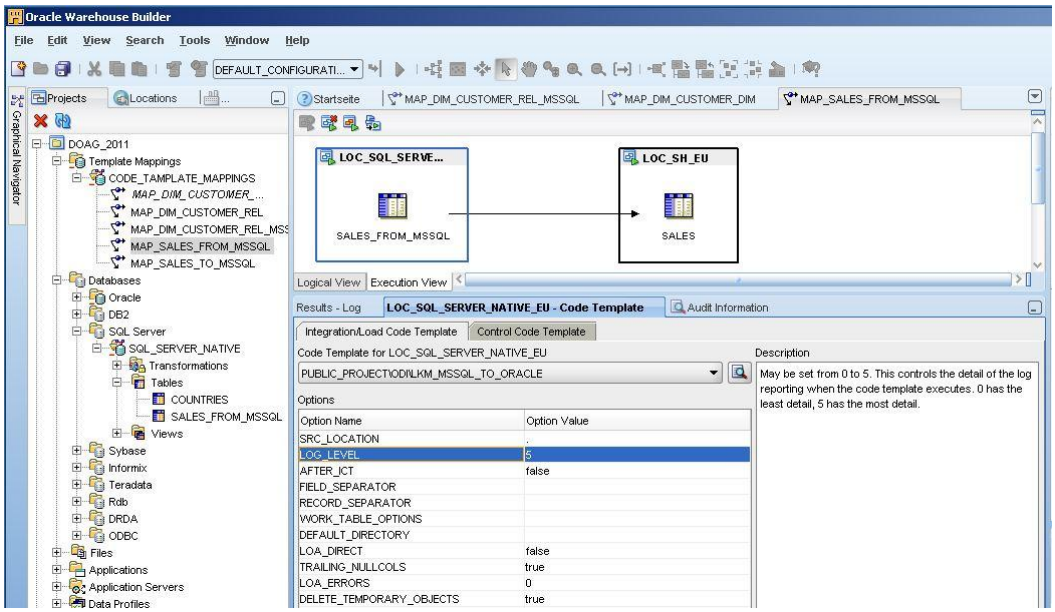

Abbildung 3: Auslesen einer Tabelle aus einem MS SQL Server und Speichern der Daten in einer Oracle Datenbank. Für die einzelnen "Execution Units" können KMs definiert und konfiguriert werden.

Richtig interessant wird diese Funktion dann, wenn man nun Ausführungseinheiten auf Basis der OWB Code Templates (KMs) mit den Mappingfunktionalitäten des OWBs kombiniert. Dazu kann man entweder direkt ein Code Template Mapping erstellen oder ein bereits vorhandenes Mapping per Copy &Paste in den Bereich der Code Template Mappings überführen. Damit können Daten unter der Verwendung von KMs aus den unterschiedlichsten Quellen ausgelesen und zeitgleich mit den mächtigen Möglichkeiten der OWB-Mappings weiterverarbeitet werden. Ein Beispiel ist Abbildung 4 dargestellt.

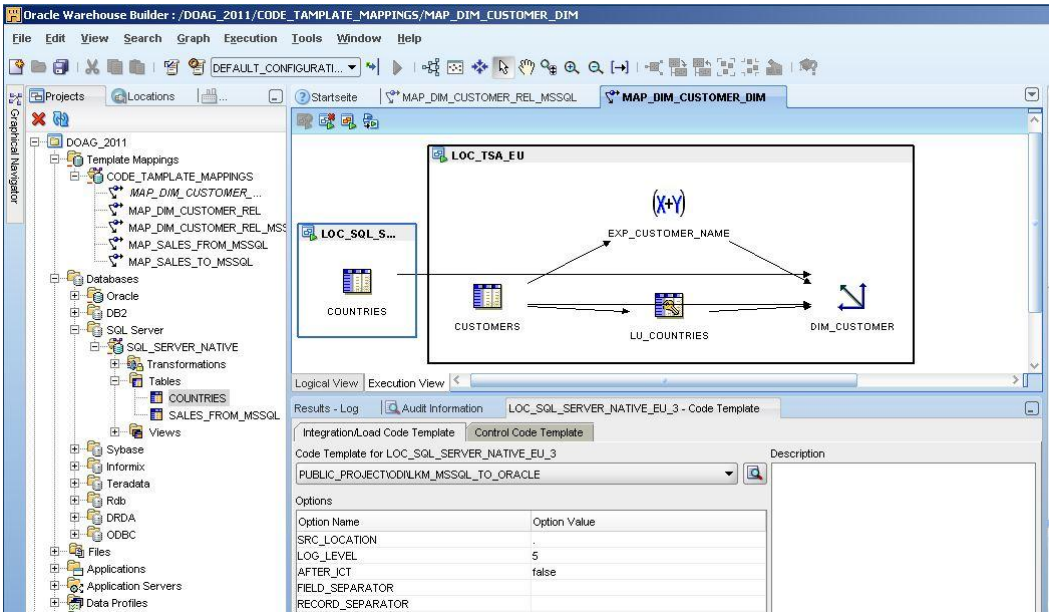

Abbildung 4: Erstellen eines Code Template Mappings – Verwendung von KM- und OWB Mapping-Funktionalitäten

Erweiterung des ODIs um OWB-Funktionen

Schaut man sich dann den ODI an und vergleicht die Interfaces mit den OWB Mappings, so ist man zunächst vom Funktionsumfang der "Interfaces" enttäuscht. Das Konzept des ODIs ist in diesem Bereich einfach nicht mit dem des OWB vergleichbar. Die Interfaces behandeln die Daten sehr viel "kleinteiliger", statt eines großen Mappings werden im ODI mehrere Interfaces verwendet. In Kombination mit den KMs können mit dem ODI die gleichen oder deutlich anspruchsvollere Aufgaben bewältigt werden.

Trotzdem gibt es Bereiche, in denen das "gute alte" OWB Mapping immer noch leistungsfähiger oder einfacher zu nutzen ist. Für diese Fälle wäre es wünschenswert, beide Werkzeuge miteinander zu kombinieren. Wie die Integration von ODI-Funktionalitäten in den OWB realisiert werden kann, ist im vorhergehenden Abschnitt bereits behandelt worden Nachfolgen wird beschrieben, wie OWB Mappings oder Prozesse in den ODI integriert werden können.

Schaut man sich den ODI näher an, so fallen spontan zwei Objekte auf, über die eine Integration möglich wäre: ODI Procedures und Knowledgemodule.

Die Nutzung von ODI Procedures kommt dem OWB- oder PL/SQL-Entwickler sehr entgegen. Alle Mappings und Prozesse lassen sich über eine PL/SQL-API aufrufen und genau das kann auch in einer ODI Procedure implementiert werden. Diese kann dann in den ODI Prozess (Package) aufgenommen und ausgeführt werden. Möchte man die Procedure noch etwas dynamischer gestalten, so können Variablen eingebunden werden. Abbildung 5 zeigt ein entsprechendes Beispiel.

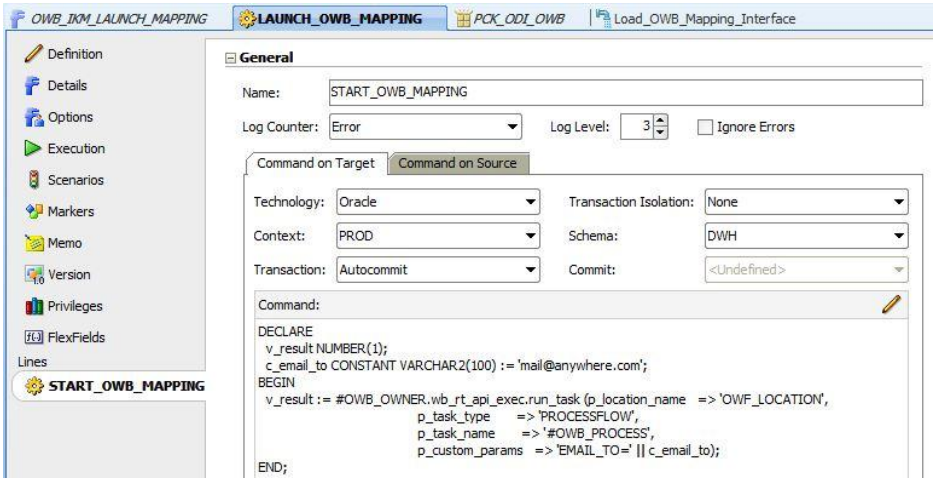

Abbildung 5: Aufruf eines OWB Mappings aus einer ODI Procedure heraus. Mit #%VARIABLE% können zur Laufzeit Ersetzungen vorgenommen werden.

Deutlich eleganter ist an dieser Stelle die Verwendung eines Knowledgemoduls. Vom Prinzip ist das Vorgehen her ähnlich, allerdings kann über die Erstellung eines spezifischen KMs streng noch ODI-Richtlinien vorgegangen werden. Der Aufruf ist fast identisch, allerdings wird für die Ersetzung des "Task Names" (Name für das OWB-Mapping) eine ODI Funktion verwendet:

```
DECLARE
  v_result NUMBER(1);
  c_email_to CONSTANT VARCHAR2(100) := 'mail@anywhere.com';
BEGIN
  v_result := #OWB_OWNER.wb_rt_api_exec.run_task
              (p_location_name => 'OWF_LOCATION',
               p_task_type     => 'PROCESSFLOW',
               p_task_name     => '<%=odiRef.getTargetTable("RES_NAME")%>',
               p_custom_params => 'EMAIL_TO=' || c_email_to);
END;
```

Damit diese Funktion die korrekten Werte ermitteln kann, werden in einem eigenen "Datastore" die Namen der OWB Mappings hinterlegt. Darauf aufbauend kann dann ein Interface erstellt werden, das dann das KM verwendet.

Fazit

Welchen Weg soll man jetzt beschreiten? Das hängt sicherlich von der Ist-Situation vor Ort ab. Wird der OWB bereits großflächig eingesetzt, so kommt eine vollkommende Migration auf den ODI sicherlich nicht in Frage. Trotzdem wird langfristig nur ein ETL-Werkzeug weiterentwickelt werden, und das wird mit großer Sicherheit näher am ODI als am OWB sein. Daher ist der aktuelle Ansatz, das Beste aus beiden Werkzeugen zu kombinieren, sicherlich valide.
Die Nutzung von Knowledgemodulen (Code Templates) im OWB ist eine gute Möglichkeit, den OWB auf die Datenbankwelt abseits von Oracle zu erweitern. Wehrmutstropfen ist derzeit, dass die Code Templates nicht im OWB entwickelt werden können, dafür wird in jedem Fall der ODI benötigt.
Im umgekehrten Fall können die OWB Mappings einfach in den ODI integriert werden. Diese Variante stellt eine gute Möglichkeit für eine schleichende Migration vom OWB in Richtung ODI dar. Es können vorhandene ETL-Strecken wiederverwendet werden, gleichzeitig werden Neuentwicklungen und Änderungen bereits im ODI umgesetzt.
Für diese Variante müssen beide Werkzeuge installiert und betrieben werden. Eine Entwicklung der OWB Mappings kann nur im OWB erfolgen. Dadurch können die Szenarien des ODI nur für den Aufruf des Mappings verwendet werden. Bei der internen Versionierung des ODIs sind die Mappings also nicht enthalten, was auch hier einen Medienbruch darstellt. Trotzdem stellt diese Variante eine für die Zukunft interessante Lösung dar.

Kontaktadresse:

Till Sander
OPITZ CONSULTING Hamburg GmbH
Butendeichsweg 2
D-21129 Hamburg

Telefon:	+49 (0) 173 5479331
Fax:	+49 (0) 40 741122-4300
E-Mail	Till.Sander@OPITZ-CONSULTING.com
Internet:	www.OPITZ-CONSULTING.com