# ODI für OWB-Entwickler

**Dr. Holger Dresing**
**Oracle Deutschland B.V. & Co. KG**

**Schlüsselworte:**
ODI – Oracle Data Integrator, OWB – Oracle Warehouse Builder

**Einleitung**
OWB und ODI sind weit verbreitete ETL-/Datenintegrations-Werkzeuge. Oracle hat entschieden, bei zukünftigen Releases im Bereich der Datenintegration und somit auch im Bereich ETL nur noch den ODI funktional zu erweitern. Somit stehen viele OWB-Entwickler vor der Frage, wie sie ihr OWB Know How in die neue ODI-Welt übertragen können. Zunächst einmal sei gesagt, dass die Funktionalität beider Werkzeuge vergleichbar ist, aber einige Vorgehensweisen unterscheiden sich doch. Ziel ist es daher, einen Einblick in ODI zu geben, um den Umstieg nach ODI zu erleichtern. Die Vorgehensweise ist folgende:

- Architektur und Installation
- Entwickeln von Prozessen: Interfaces, Packages und Szenarien sowie Knowledge-Module
- Laufzeitumgebung
- Nutzen externer Funktionen: Java SDK und Open Tools

Die Lösungsansätze oder Best-Practices werden kurz skizziert. Es wurde versucht, an dieser Stelle mehr auf Masse der Features denn auf die Tiefe zu gehen.

**Architektur und Installation**
Der OWB und sein Repository kann mit der Datenbank installiert und aufgesetzt werden. Der ODI arbeitet ebenso Repository-basiert, hat aber einige Unterschiede: Es gibt ein Master- und ein oder mehrere Workrepositories, die in Datenbanken verschiedener Hersteller installiert werden können. Ein häufiges Anfängerproblem ist, dass man sich mit der internen ID-Vergabe in den Repositories vertraut machen sollte, bevor Daten zwischen Repositories ausgetauscht werden. Best-Practice ist, daß alle Workrepositories im gesamten Unternehmen unterschiedliche IDs besitzen.
Wie beim OWB ist auch beim ODI ein Agent aufzusetzen, der die Prozesse zur Laufzeit steuert. Der Scheduler ist beim ODI in den Agenten integriert. Nach einer Standardinstallation ist der Agent ohne Scheduling-Funktion aufgesetzt. Bei den Agenten gibt es seit ODI 11g zwei Varianten: den Stand-alone Agent und den JEE Agent. Der Standalone Agent benötigt ein JDK, um laufen zu können. Hier gilt: Zum Lastausgleich und zur Hochverfügbarkeit können mehrere Agenten manuell aufgesetzt werden. Der JEE-Agent läuft als Applikation im Oracle WebLogic Server (WLS). D.h., zunächst werden ODI und WLS installiert, anschließend wird eine WLS-Domäne für ODI erstellt und ODI wird im WLS integriert. Damit können die Funktionalitäten vom WLS benutzt werden. Einmal ODI installieren, über verschiedene Weblogic Server Instanzen ggf. im Cluster verteilen, und die Sicher-heitseinstellungen, Lastverteilung und Hochverfügbarkeit wird vom Weblogic Server übernommen (vgl. Abb. 1).
Aus Entwicklersicht stellt auch die Entwicklungsumgebung ODI Studio einen Agenten zur Verfügung, der aber nur für Entwicklungszwecke eingesetzt werden sollte. Nachdem die Installation abgeschlossen ist, können die Repositories aufgesetzt werden. Empfehlung ist, nicht das RCU (Repository-Creation-Utility) zu nutzen, sondern das Master Repository über den Menüpunkt Master Repository erstellen zu erzeugen. Anschließend wird mit dem ODI-Studio die erste Verbindung zum

Master-Repository hergestellt. Darauf aufbauend können anschließend die Workrepositories aufgesetzt werden. Wenn man sich mit dem ODI-Studio (genau gesagt mit dem Topology-Manager vom ODI-Studio) an das Master-Repository verbindet, kann man nur die Administrativen Komponenten benutzen. Um auch Prozesse erstellen zu können. muß man sich an ein Workrepository anmelden. Gegenüber dem OWB gibt es beim ODI eine logische Architektur, gegen die der Entwickler entwickelt und mehrere physische, die jeweils eine konkrete Datenbank-Instanz adressieren. Zwischen logischer und physischer Architektur gibt es den Context. Zur Laufzeit bzw. bei der Entwicklung wird dieser Kontext angegeben und daraus zieht ODI die Verbindungs- und gültigen Strukturdefinitionen.

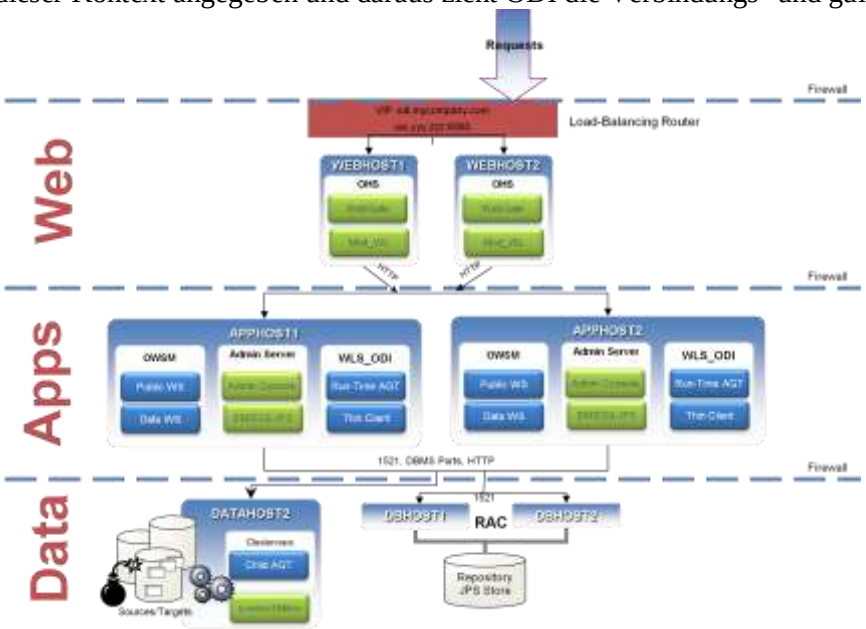


*Abb. 1: Hochverfügbarkeitsumgebung mit ODI*

Im Topology-Manager werden anschließend die Connections zu den Quell- und Zielstrukturen definiert. Diese Verbindungen nutzen i.d.R. immer JDBC. Zunächst ist eine Technology und anschließend ein Schema auszuwählen (siehe Abb. 2).

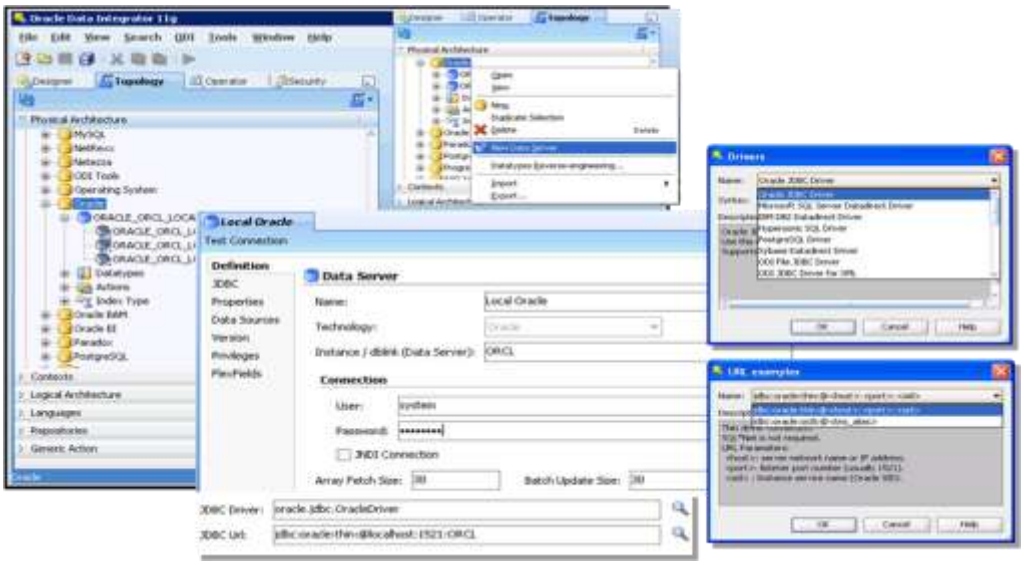


*Abb. 2: Erstellen der Verbindungen zu Datenquellen in ODI*

Jedes Werkzeug hat seine eigenen Begriffe. Bei den Datenquellen ist zu beachten, dass ODI immer von Data Servern und Schemata spricht, auch wenn die verschiedenen Technologien/Hersteller andere Begriffe dafür nutzen (siehe Abb. 3).

| Technology | Data server | Schema |
|---|---|---|
| Oracle | Instance | Schema |
| Microsoft SQL Server | Server | Database/Owner |
| Sybase ASE | Server | Database/Owner |
| DB2/400 | Server | Library |
| Teradata | Server | Schema |
| Microsoft Access | Database | (N/A) |
| JMS Topic | Router | Topic |
| File | File Server | Directory |

*Abb. 3: Begriff bei den Datenquellen abhängig von den Herstellern*

**Entwickeln von Prozessen**

Seit ODI 11g gibt es das ODI-Studio, mit dem entwickelt und administriert werden kann. ODI-Studio wurde wie der OWB aus dem gleichen GUI-Framework entwickelt. Daher ist das Look and Feel beider Werkzeuge vergleichbar (siehe Abb. 4).

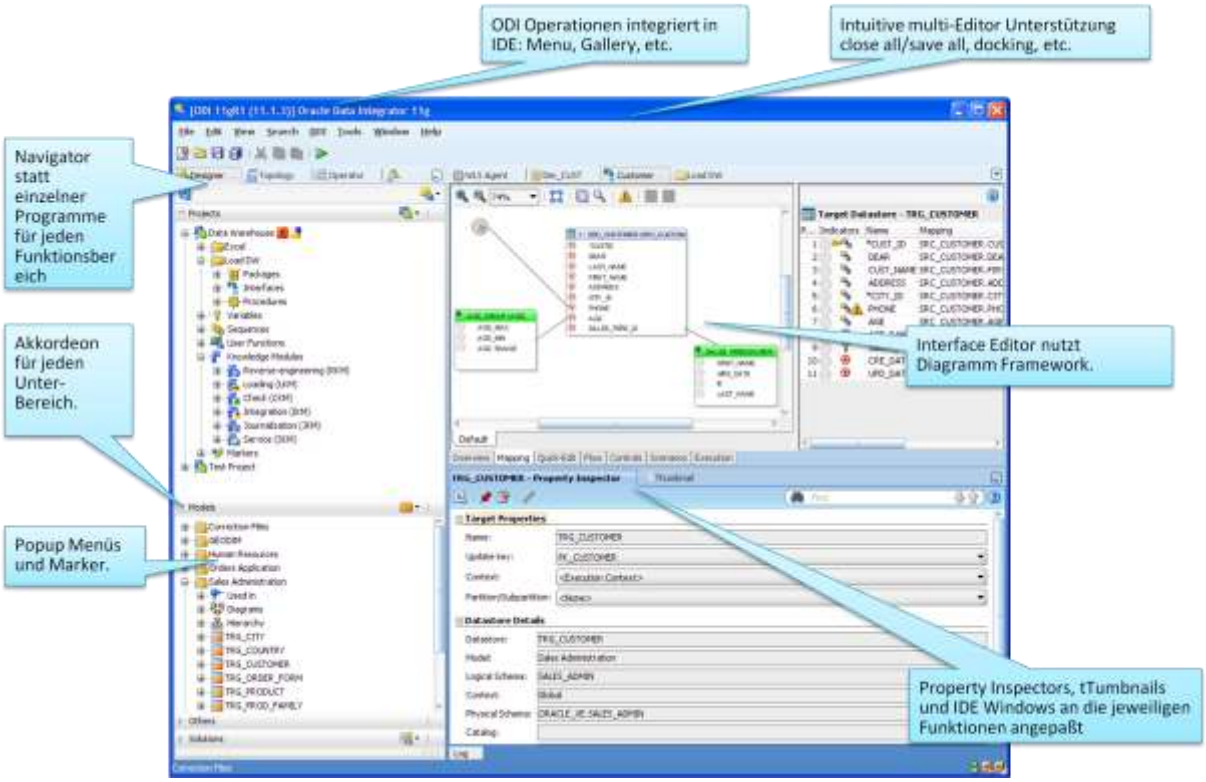


*Abb. 4: ODI-Studio*

Der erste Schritt ist, für die Quell- und Zielstrukturen ein Reverse Engineering durchzuführen. Basis dafür sind die Verbindungen zu den unter dem Reiter Topology definierten Data Servern. Das Reverse Engineering kann über JDBC- oder ein Knowledge Modul ausgeführt werden. Die Vorgehensweisen sind vergleichbar mit dem OWB. Unterschiede zum OWB liegen in den Details. Beispielsweise sind

die Verzeichnisse der Flat Files unter Topology zu bestimmen, die Dateistrukturen werden unter Models eingegeben. Seit ODI 11.1.1.5 können im ODI auch komplexe FlatFile-Strukturen wie im OWB definiert werden.

ODI 11.1.1.5 kennt verschiedene Arten von Transformationen:

- Transformationen in ODI werden in Objekten namens **Interfaces** erstellt – vergleichbar mit den Mappings in OWB
- Berechnungen oder konstante Werte werden in den **Mapping**s im Interface zugewiesen – vergleichbar den Konstanten oder Expresssions im OWB
- Interfaces werden in **Projekten** gespeichert – wie Projekte in OWB
- Interfaces werden in **Packages** in eine Ablauf-Reihenfolge gebracht, die übersetzt wird zu einem **Scenario** für die Ausführung in der Produktion. Zusätzlich gibt es **Load Plans**, um Interfaces oder Projekte parallel mit einem verbesserten Exception Handling und einer erweiterten Restart-Fähigkeit – vergleichbar zu den Business Flows im OWB.
- Falls Begriffe bei ODI und OWB gleich sind, wird das jeweilige Produkt in Klammern hinter den Begriff gesetzt, z. B. Mapping (ODI) oder Mapping (OWB).

In einem Interface wird definiert:

- Wohin die Daten gehen sollen (das Ziel/**Target**)
- Woher die Daten kommen (die Quellen/**Sources**)
- Wie die Daten vom Quell-Format in das Ziel-Format transformiert werden (die **Mappings**)
- Wohin die Daten transformiert werden von der Quelle zum Ziel (den "Daten-"**Flow**)

Mappings werden in SQL formuliert. Der Entwickler ist selbst dafür verantwortlich, ggf. das optimale SQL für das Ziel- und Quellsystem anzugeben. Es gibt wie beim OWB einen Expression Editor, der selbständig die SQL-Variante anpasst.
Flows werden definiert durch Templates, die Knowledge Module (KM) genannt werden.
Die Knowledge Modules (KMs)(ODI) oder Code Templates (OWB) gleichen sich, daher wird auf diesen Punkt hier nicht im Detail eingegangen.
Bei den Interfaces (ODI) und den Mappings (OWB) werden die Schwächen und Stärken immer wieder diskutiert. Festzuhalten ist, dass der ODI bis 11g nur jeweils in einen Ziel-Datastore schreiben kann. Die logische Struktur von ODI geht davon aus, dass es sich um eine Tabelle handelt. Es gibt eine eingeschränkte Menge von Operatoren: im wesentlichen Join, Filter, Lookup, Data Set und temporäre Interfaces. Ausdrücke für Berechnungen oder Konstanten werden im ODI in den Mappings definiert. Neue Tabellen können im Mapping direkt angegeben werden und beim Ausführen des Mappings wird dafür ein Parameter „Create Target Table" gesetzt. Ein weiterer Punkt ist, dass beim OWB immer „Row-based" und „Set-based" als Stärke angegeben wird. Beim ODI sind alle Knowledge Module „Set-based", es gibt jedoch ein spezielles Knowledge Module, dass immer „Row-based" die Daten lädt. Dieses KM stellt auch Debugging-Funktionen zur Verfügung. Dieses kann als Vorlage für andere KMs dienen. Der OWB kann wahlweise insert-, update- oder delete-Statements generieren. Beim ODI hängt auch das von den KMs ab. Alle KMs mit „incremental update" im Namen generieren abhängig vom Update Key insert- und update-Statements, alle KMs mit „insert append" nur inserts (ohne Update Key). Für delete-Statements sollen Procedures (ODI) geschrieben werden.
Wichtig ist, dass jedes Interface zwei Beschreibungsstrukturen hat. Das Diagramm in der Oberfläche beschreibt die Logik der Mappings und die Operatoren. Zusätzlich gibt es den Flow, in dem die KMs und dessen „Options" festgelegt werden. Der Flow ist der Pfad, den die Daten nutzen, um von der Quelle zum Ziel in einem ODI-Interface zu gelangen. Im Gegensatz zum OWB kann der Entwickler festlegen, ob die Prozesse oder Teile davon auf der Quelle, im Ziel oder in einer ODI-spezifischen Staging Area ablaufen.

Beispielhaft wird das in Abb. 5 skizziert, wobei die Berechnungen (Joins) in der Staging Area laufen. Im Standard laufen die Prozesse auf dem Zielsystem. Mit Hilfe des Kontextes ist es jedoch möglich, die Berechnungen und Joins auf dem Quellsystem durchzuführen.

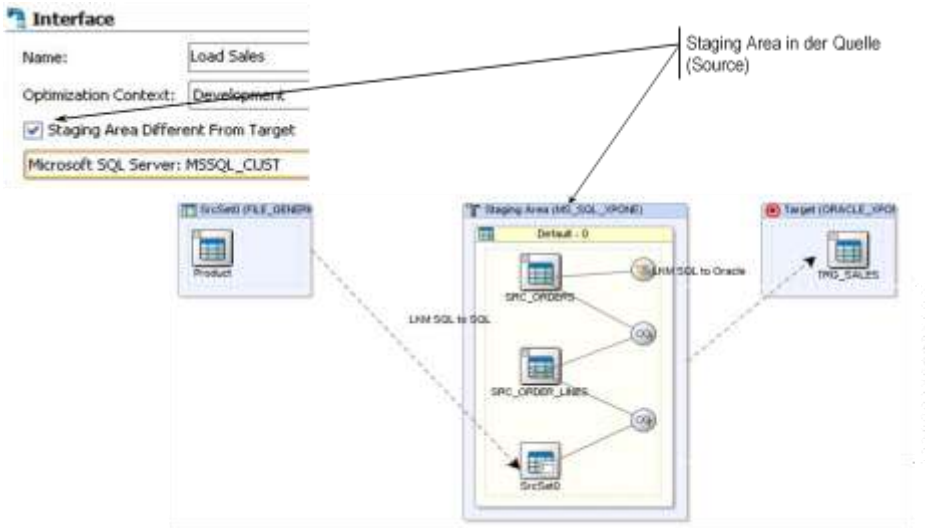


*Abb.5:. Ausführen von Berechnungen in der Staging Area*

Variablen gibt es nur in ODI. Bei ODI wird zwischen lokalen und globalen Variablen unterschieden. Auf globale Variablen kann projektübergreifend zugegriffen werden. Sie können überall eingefügt werden, selbst als Name für eine Tabelle oder als Dateiname bei einer Datei. Zur Laufzeit wird Ihnen als erstes ein Wert zugewiesen und danach wird die Anweisung ausgeführt. Best-Practice ist, alle Variablen zu deklarieren (notwendig für die Migration von ODI 10g nach 11g).
Ein Package ist eine vordefinierte Reihenfolgevon Schritten (steps), die den Workflow beschreiben. Jeder Schritt stellt die kleinste Einheit dar, die in einem Package zusammengestellt werden kann. Bei Packages gibt es ähnlich viele Operatoren wie beim OWB. Beim Arbeiten mit dem ODI Studio ist zu beachten, dass die Tools ausgewählt werden müssen und man anschließend ein Tool durch Klicken in das Diagramm einfügt. Im Gegensatz zum OWB kennt der ODI nur die Zustände „ok“ und „ko“. Der Ablauf von Packages lässt sich durch Variablen mittels Schleifen und Bedingungen steuern (siehe Abb. 6).

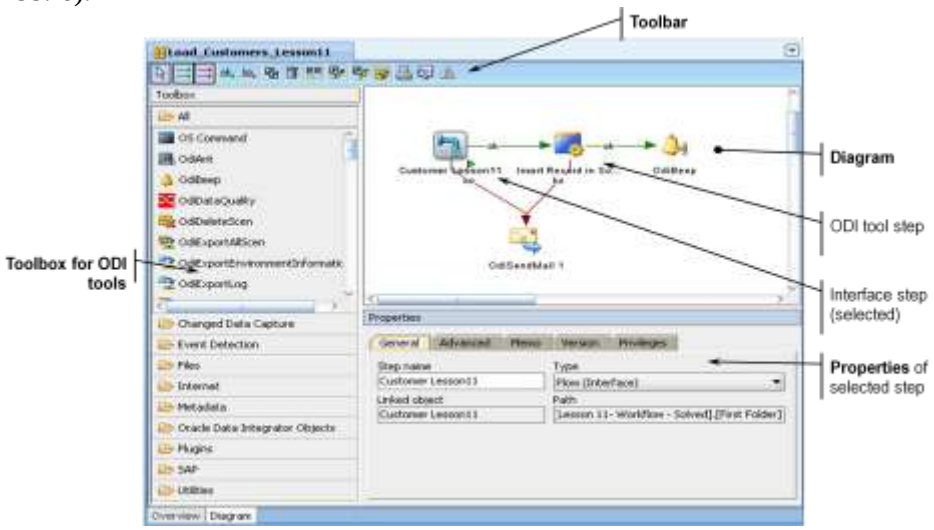


*Abb. 6: Das Package Diagramm*

Seit dem Patchset für ODI 11.1.1.5 gibt es die Load-Plans. Das sind hierarchische Objekte, die Szenarien enhalten und alle Schritte in diesen Szenarien können seriell oder parallel ausgeführt werden. Damit ist es beispielsweise möglich, zunächst alle Dimensionen in einem Szenario zu laden und anschließend seriell die Kennzahlen. Die GUI für Load-Plans unterscheidet sich erheblich von der der Packages.

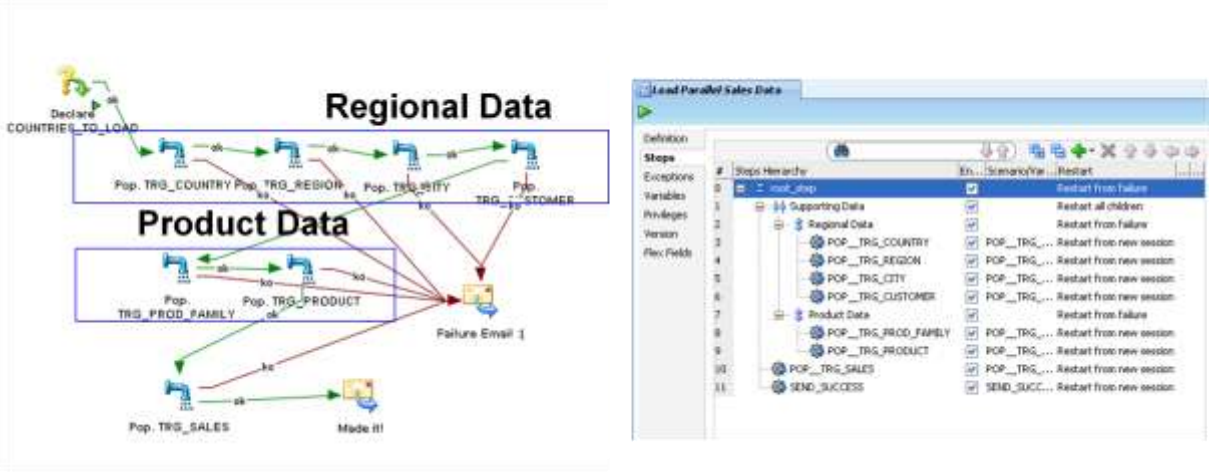


*Abb. 7: Beispiel für einen Load-Plan in ODI*

Eine Procedure ist eine Folge von Kommandos, die von der Datenbank-Engine, dem Betriebssystem oder den ODI-Werkzeugen ausgeführt wird. Eine Procedure kann Options besitzen, die ihr Verhalten beeinflussen. Procedures sind wiederverwendbare Komponenten, die in Packages eingesetzt werden können. Ihr Aussehen entspricht denen von Knowledge Modulen, wobei KMs projektübergreifend wiederverwendbar sind. Der OWB kennt eine solche Struktur nicht.

Um die Interfaces und Packages zur Laufzeit von einem Agenten aufrufen zu können, muß ein **Scenario** erstellt werden. Dazu reicht es aus, das Kommando „Create Scenario“ auszuführen. Es können nur Szenarien an den Scheduler von ODI übergeben werden. Von der Funktionalität ist der Scheduler mit dem vom OWB vergleichbar.

Für alle Objekte in ODI gibt es eine Versionsverwaltung. D. h., ein Interface oder ein KM kann mit einer neuen Version versehen werden. Beim Aufruf eines Interfaces wird gefragt, welche Version ausgeführt werden soll.

**Laufzeitumgebung**
Bevor ein Prozess in einer anderen Umgebung aufgerufen werden kann (siehe Abb. 8), muß er in diese Umgebung exportiert werden. Wie beim OWB gibt es in ODI die Möglichkeit, Projekte zu exportieren. In älteren Releases mußte immer das gesamte Projekt exportiert werden. Seit ODI 11.1.1.5 gibt es auch hier die Möglichkeit, Teile von Projekten oder Datastores mit allen dazugehörenden Prozessen zu exportieren. Beim Export/Import ist zu beachten, dass die Repository-internen IDs für die Objekte übernommen oder neu vergeben werden können (duplicate vs. sysnonym mode). Eine Objekt-ID setzt sich aus den Repository-IDs und einer fortlaufenden Zahl zusammen. Wenn Repository-IDs mehrfach vergeben wurden, kann es dadurch zu Fehlermeldungen bzw. Inkonsistenzen kommen. Diese Problematik tritt beim OWB nicht auf. Ein anderer Fall ist, dass beim OWB nach jeder Änderung eines Mapping der Deploy-Prozess neu laufen muß. Ein solcher Deploy-Prozess kennt der ODI nicht. Vorteil ist, wenn sich z. B. nur die Connect-Information geändert hat, daß nicht alle Prozesse neu deployed werden müssen.

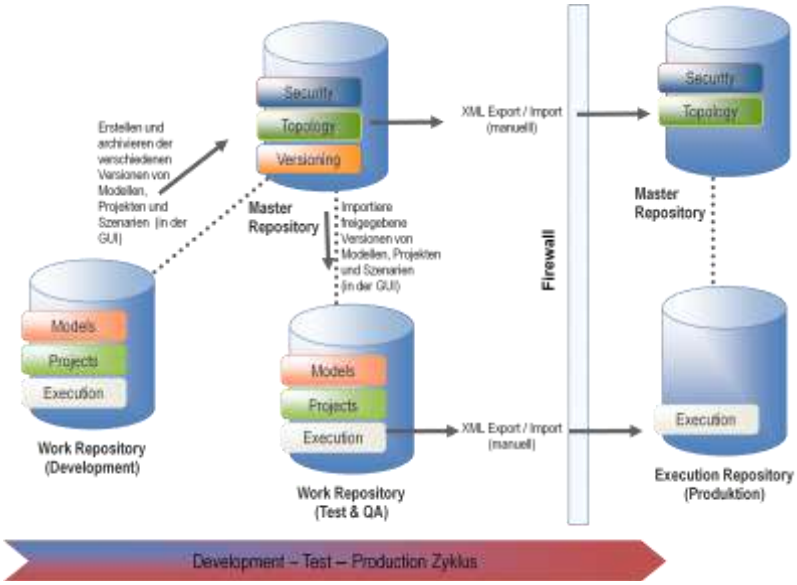


*Abb. 8: Beispiel für eine Repository-Struktur in ODI*

Beim Aufruf von Interfaces, Packages, Procedures oder Scenarios ist die Version und der Agent anzugeben. Im ODI-Studio kann auf die Daten der Laufzeit-Umgebung unter dem Operator-Tab zugegriffen werden. Bei Packages mit Verzweigungen werden die Einträge für die einzelnen Steps erst dann vorgenommen, wenn der jeweilige Step durchlaufen wird.
Daneben gibt es noch die ODI Console und den Oracle Enterprise Manager. Das ist der Browser-basierte Zugang zum ODI. In der ODI Console können alle Entwicklungs- und Laufzeit-Informationen angezeigt werden. Aus der ODI Console können Szenarien gestartet werden. Auch die Lineage Analyse, erfolgt aus der ODI Console heraus. Abhängigkeiten zwischen Interfaces und Tabellen werden auch im ODI-Studio angezeigt. Im Tab Designer unter Modelle kann für jede Tabelle in der Baumstruktur angesehen werden, für welche Interfaces die Tabelle benutzt wurde.
Seit ODI 11.1.1.5 können die Repository-Einträge von ODI an OBI EE übergeben werden. Zusätzlich gibt es vordefinierte Berichte in OBI EE um die Inhalte des Repositories anzuzeigen (siehe Abb. 9).

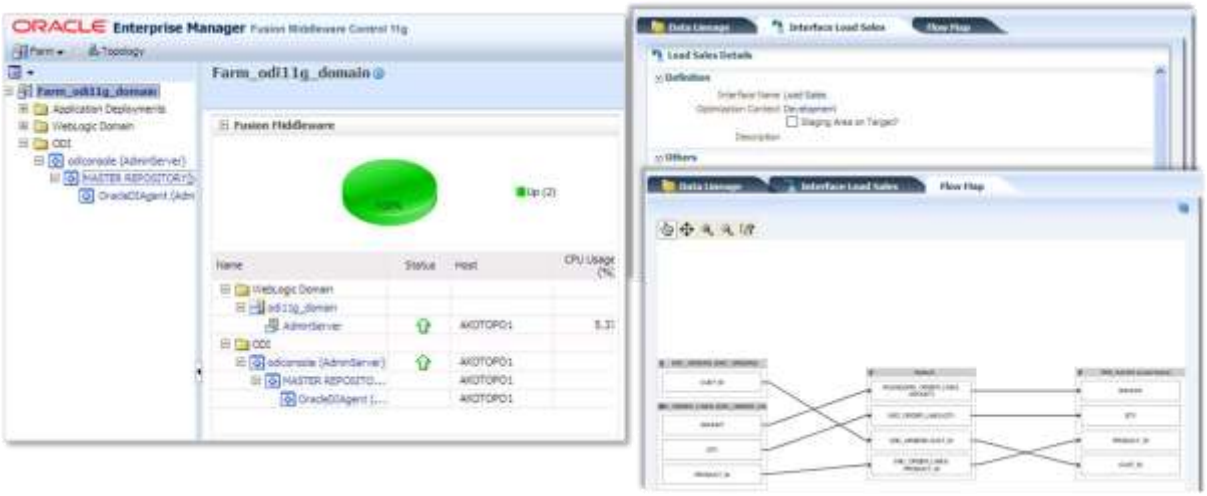


Abb.9: Enterprise Manager und Data Lineage mit der ODI Console

**Nutzen externer Funktionen**

Im OWB gibt es zum Programmieren eigener Funktionen OMBPlus. Bei ODI gibt es das ODI SDK, dass eine Java-Library bereitstellt, um damit Prozesse außerhalb des ODI Studios zu erstellen. Mit dem SDK besteht damit die Möglichkeit, eigene Programme zum Erstellen und Ausführen von Prozessen in ODI zu erstellen. Das SDK ermöglicht die programmatische Definition von Repositories, Modellen, Projekten, Interfaces, Variablen oder Packages. Im aktuellen Release gibt es noch Einschränkungen. Das Erstellen neuer Knowledge Module oder das Nutzen von Versionen ist via ODI SDK noch nicht möglich. Häufig wird das Java-SDK zusammen mit Groovy benutzt. Groovy ist eine Skriptsprache für die Java Virtual Machine und nutzt Java-Syntax.

Immer wieder wird auch die Frage gestellt, wie Scenarios aufgerufen werden können, z. B. als Kommando auf der Ebene des Betriebssystems. Dafür gibt es die Open Tools API von ODI. Aus dem Verzeichnis „bin“ kann das Kommando startscen mit den Parametern Name des Scenarios und Kontext aufgerufen werden:

```
startscen PKG_LD_ALL 001 DEV
```

Ebenso lassen sich auch alle Kommandos in den Packages über die Tools API aufrufen. Hier ein Beispiel für den Export aller Szenarien (Anführungsstriche notwendig unter Windows):

```
startcmd OdiExportAllScen "-FROM_PROJECT=2071" "-TODIR=c:\temp\" "RECURSIVE_EXPORT=yes"
```

Voraussetzung dafür ist, dass die ODIParams.bat/sh, wie beim Aufruf des Agenten gezeigt, manuell konfiguriert wird. In dieser Datei steht der Zugang zu dem Master-/Workrepository mit den Benutzer-/Passworten (letztere verschlüsselt).

**Zusammenfassung**

OWB und ODI sind zwei Werkzeuge, deren Funktionalität in vielen Bereichen vergleichbar ist. Jedoch ist die Vorgehensweise manchmal unterschiedlich. Es konnte hier nur ein kurzer Überblick gegeben werden, ohne auf alle Unterschiede im Detail einzugehen. Wichtig ist sicherlich, dass der OWB eng mit der Oracle Datenbank verknüpft ist. Der ODI als Alternative nutzt die Oracle Middleware mit dem WebLogic Server als Basis.

Die graphischen Werkzeuge und deren Unterschiede wurden erläutert, die Entwicklungsumgebungen mit OMBPlus bei OWB und dem Java SDK bei ODI wurden kurz erwähnt. Hinzu kommt, dass es unterschiedliche Data Quality-Optionen gibt. Die Data Quality Option von OWB und das Enterprise Data Quality (EDQ) für den ODI (ehemals Datanomic).

Zudem bleibt spannend, wie sich der ODI weiterentwickeln wird. Die Aussage von Oracle ist, daß zukünftige Entwicklungen auf Basis ODI laufen. Aber wie wird die Migration aussehen, was passiert mit den fehlenden Funktionalitäten hinsichtlich Modellierung und Operatoren in Mappings. Werden weitere Werkzeuge integriert (z. B. Oracle Golden Gate und EDQ)? Die Weiterentwicklung von ODI bleibt spannend.

Kontaktadresse:
Dr. Holger Dresing
Oracle Deutschland B.V. & Co. KG
Data Integration Solutions DIS EMEA
Thurnithistr. 2
D-30519 Hannover

Telefon/Fax: +49 (0) 511-95787 118
E-Mail holger.dresing@oracle.com
Internet: www.oracle.com