Wie oft wünscht man sich, bestimmte wiederkehrende Routineaufgaben nicht mehr von Hand oder am liebsten gar nicht mehr selbst erledigen zu müssen? Warum sollte dies bei der Entwicklung von ETL-Prozessen in einem Data Warehouse anders sein?

Automatische Generierung der ETL-Prozesse: die Möglichkeiten von OWB und ODI

Irina Gotlibovych, MT AG

Anhand eines von der MT AG entwickelten Frameworks werden in diesem Artikel Möglichkeiten der automatischen Generierung von ETL-Prozessen im Oracle Warehouse Builder einerseits und im Oracle Data Integrator andererseits gegenübergestellt und verglichen.

Bei der Entwicklung der ETL-Prozesse in einem Data Warehouse sieht man sich wiederholt vor die Aufgabe gestellt, Prozess-Schritte aufbauen zu müssen, die einer gleichartigen Logik folgen. So werden in jedem Projekt viele Daten-Objekte auf die gleiche Weise aus Quellsystemen in den Arbeitsbereich geladen. Beim Transformationsschritt werden Daten in das einheitliche Format der Ziel-Datenbank überführt; gängige Verfahren dabei sind beispielsweise Datentyp-Konvertierung und Daten-Bereinigung. Anschließend werden Daten nach dem gleichen Prinzip – etwa mit Delta Load oder SCD – in das Data Warehouse eingebracht.

Wenn man mit dem OracleWarehouse Builder arbeitet, bedeutet dies in der Praxis oft, logisch identische Mappings in manueller Kleinarbeit anlegen zu müssen. In jedem dieser Mappings sind von Hand Operatoren anzulegen und zu verbinden. Für jedes Attribut eines Expression Operators muss der Ausdruck eingetragen werden. Eigen-

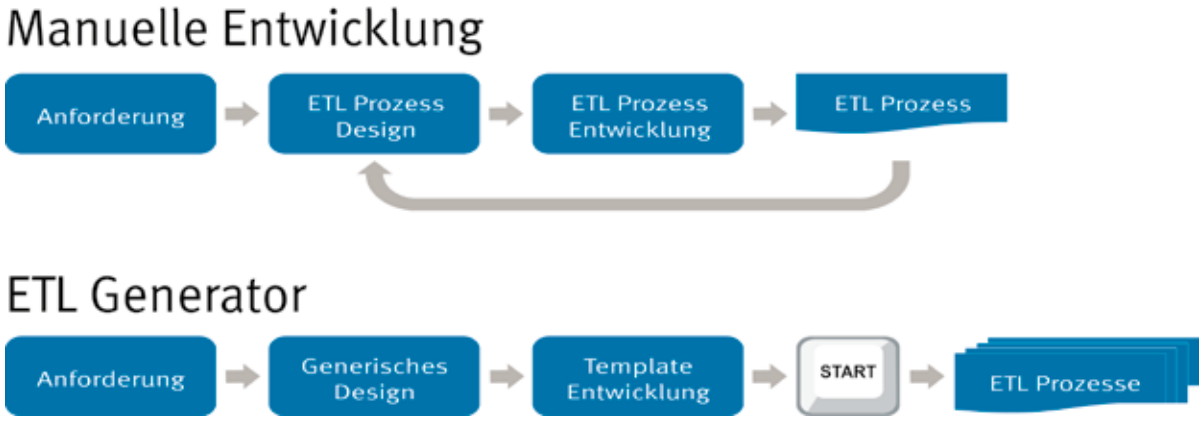

Abbildung 1: Prozesskette bei der Erstellung von Prozessen mit dem ETL-Generator im Vergleich zur manuellen Entwicklung

schaften von Operatoren und Attributen sind immer wieder neu zu setzen. Kommt in einer Quell-Tabelle später eine neue Spalte hinzu, muss sie in den meisten Fällen identisch zu den anderen Spalten geladen und verarbeitet werden. Um das zu erreichen, ist aber im entsprechenden Mapping jeder betroffene Operator manuell zu ändern. Besonders aufwändig wird es, wenn sich die grundlegende Logik ändert; im Falle von späteren Änderungsanforderungen ist zumeist jedes Mapping wieder anzupassen. Obwohl sie der gleichen Logik folgen, muss trotzdem jedes dieser Mappings einzeln getestet werden: Da sie unabhängig voneinander entwickelt wurden, können in jedem auch unterschiedliche Fehler auftreten. Bei der manuellen Entwicklung spielt der menschliche Faktor eine enorme Rolle. Repetitive Entwicklungsarbeiten wie das fünfzigfache Anfertigen eines Delta-Load-Mappings sind monoton und führen dadurch zu Fehlern.

Ganz ähnlich verhält es sich bei der Entwicklung mit dem Oracle Data Integrator. Interfaces müssen für jede Ziel-Tabelle einzeln angelegt werden. Trotz der gleichen Logik sind in jedem Interface Zuordnungen von Attributen unabhängig voneinander zu definieren und bei späteren Anforderungen auch einzeln zu ändern. Der Oracle Data Integrator beinhaltet durch die mitgelieferten Knowledge- Module bereits die Möglichkeit, generische Funktionalitäten zu nutzen beziehungsweise aufzubauen. Ähnlich wie Makros oder Templates beinhalten diese Knowledge-Module wiederverwendbare Logiken und nehmen dadurch dem Entwickler einen Teil der manuellen Arbeit ab. Sie können in mehrere Interfaces eingebunden werden, ersetzen aber nicht die manuelle Anlage jedes einzelnen.

Generische ETL-Entwicklung mit OWB und ODI

Die Frage stellt sich: „Warum entsteht so ein Mehraufwand bei der manuellen Entwicklung und wie kann man diesen vermeiden?“ Wäre es nicht schöner, die Logik nur einmal zu entwickeln und diese dann in weiteren Prozessen beziehungsweise Projekten mehrmals zu verwenden? Das Problem bei der Entwicklung sowohl im OWB als auch im ODI ist, dass es keine Möglichkeit gibt, Mappings beziehungsweise Interfaces ohne Bindung an konkrete Objekte (Tabellen, Spalten etc.) anzulegen. Die fachliche Logik eines Prozesses ist immer fest mit den Umgebungs-Informationen verbunden. Um die gleiche Logik nicht mehrfach neu erzeugen zu müssen, bräuchte man einen Weg, Prozesse generisch, also ohne Bezug zu den eigentlichen Objekten definieren zu können. Die Erzeugung der Mappings beziehungsweise Interfaces kann dann automatisch erfolgen, wobei Objekt-Namen als Parameter dem Generierungsprozess mitgegeben werden.

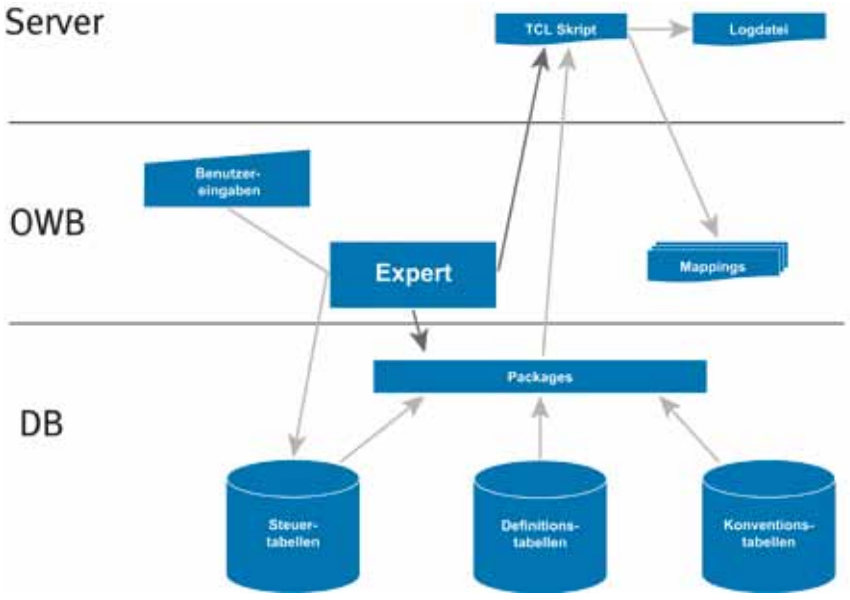

Abbildung 2: Architektur des OWB Mapping Generators

Dieser Ansatz wurde bei der Entwicklung des ETL-Generators zugrunde gelegt (siehe Abbildung 1).

Welche Vorteile bringt so ein generischer Ansatz? Angenommen, man möchte Slowly Changing Dimensions Typ 2 in seinem Data Warehouse implementieren. Bei der manuellen Entwicklung würde man für jede Ziel-Tabelle die komplexen Join- und Splitter-Bedingungen in Abhängigkeit von den jeweiligen Primärschlüsseln und Spalten einzeln implementieren. Wählt man den generischen Ansatz, wird die Logik unabhängig von Tabellen und Spalten einmal in Form eines Templates definiert. Der Generierungsprozess arbeitet nun nach allgemeinen Regeln, was identischen Code und gleiche Qualität für alle Prozesse garantiert. Und wenn später eine neue Spalte hinzukommt? Da das Template allgemein definiert wurde und die konkreten Primärschlüssel beziehungsweise Spalten erst bei der Verarbeitung aus der Datenbank ausgelesen werden, wird die neue Spalte bei einer erneuten Generierung des Mappings beziehungsweise Interface automatisch berücksichtigt. Es ist keine manuelle Anpassung des Prozesses im OWB beziehungsweise ODI notwendig. Doch wie sieht es mit Fehlerbehebung und Testen aus? Da man die Logik an einer Stelle entwickelt, müssen die Fehler auch nur an einer Stelle behoben werden – nämlich in dem zugrunde liegenden Template und nicht in jedem Prozess einzeln. Ist man einmal sicher, dass die Logik in dem Template richtig definiert ist, kann man auch sicher sein, dass jedes damit generierte Mapping beziehungsweise Interface korrekt laufen wird.

ETL-Generator

Das besagte Framework besteht aus zwei Komponenten – OWB Mapping Generator und ODI Interface Generator – und ermöglicht es, Prozesse auf Basis von mitgelieferten oder selbst entwickelten Templates automatisch zu generieren. Im OWB Mapping Generator werden Mappings mithilfe von OMB Plus beziehungsweise TCL erzeugt. ODI Interface Generator verwendet das bei ODI zur Verfügung stehende Java-API (siehe Tabelle 1). Im Gegensatz

Objekt	OWB	ODI
Prozess	LOOP insert_line (OMBCREATE MAPPING <name>); END LOOP;	LOOP odiInterface = new OdiInterface (…,<name>,…); END LOOP;
Operator	LOOP insert_line (OMBALTER MAPPING <name> ADD <operator type> OPERATOR <operator name>); END LOOP;	LOOP Source table sourceDatastore = …findByName (<name>,<model>); interfaceHelper.setSourceDataS- tore (sourceDatastore); Filter interfaceHelper.setSourceFilter (<filter condition>); END LOOP;

Tabelle 1: Automatische Generierung: OWB vs. ODI

Objekt	Eigenschaft	Wert
OWB Table Operator	Operator type	TABLE
OWB Set Operator	SET_OPERATION	MINUS
OWB Expression Operator	EXPRESSION	INGRP1.$attr_name
ODI Source Datastore	Model	STAGE
ODI DataSet	Operator	UNION
ODI Target Column	Mapping	SYSDATE
ODI Filter	Filter condition	$func_get_my_filter_cond

Tabelle 2: Definition von Templates

Schema	SOURCE	STAGE	CORE
Tabelle	SRC_PRODUCT	STG_PRODUCT	PRODUCT

Tabelle 3: Tabellenstamm „PRODUCT“ in den Schemata „Source“, „Stage“ und „Core“

zur manuellen Entwicklung setzt man beim Gebrauch des Frameworks nicht mehr jedes Mapping im OWB Design Center beziehungsweise jedes Interface im ODI Designer einzeln um, sondern (siehe Abbildung 1) definiert ein allgemeingültiges Template für eine Klasse von Prozessen. Anschließend generiert man die dazugehörigen Prozesse unter Einbeziehung seiner Projektvorgaben automatisch. An dieser Stelle ist anzumerken, dass es sich bei solch einem Template nicht um ein programmiertes Skript zur Generierung der Prozesse handelt, sondern genauso wie im OWB und ODI um eine deklarative Definition auf Basis von Metadaten.

Im OWB Mapping Generator wird die Generierung der Mappings über einen „OWB Expert“ aus der Oracle Warehouse Builder GUI angestoßen. Dieser leitet dialoggestützt durch die einzelnen Schritte. Nachdem man seine Auswahl bezüglich des zu generierenden Templates und der zu verwendenden Objekte getroffen hat, legt der OWB Mapping Generator die Eingaben in den Steuer-Tabellen auf der Datenbank ab. Danach werden PL/SQL-Prozesse gestartet (siehe Abbildung 2), die aus den Definitions-Tabellen das gewünschte Template auslesen und die darin gespeicherte Logik mit den Umgebungs-Informationen aus den Konventions-Tabellen anreichern. Damit wird nun ein TCL-(OMB-Plus)-Skript generiert, mit dem die Mappings anschließend automatisch erzeugt werden. Das Ergebnis ist sofort im Oracle Warehouse Builder sichtbar und kann weiterverwendet beziehungsweise eingesetzt werden. Während der Generierung der Mappings ist jeder Schritt in einer Logdatei auf dem Server protokolliert. Man hat so stets den vollen Überblick über die generierten Objekte im Data Warehouse.

Die Architektur des ODI Interface Generators ist der des OWB Mapping Generators sehr ähnlich (siehe Abbildung 3). Die Generierung der Prozesse funktioniert allerdings auf eine andere Weise. Statt der dynamischen PL/SQL-Prozesse kommt hier das statische ODI-Java-API zum Einsatz. Es stellt vordefinierte Methoden zur Verfügung, mit denen ODI-Interfaces direkt erzeugt werden können. Im Gegensatz dazu wird im OWB Mapping Generator das ausführbare Skript erst zur Laufzeit erstellt. Beim Ausführen eines OMB-Plus-Skripts wird das Logfile immer automatisch auf dem Server generiert. Im ODI Interface Generator ist das Logging ein Teil der implementierten Java-Schnittstelle.

Einer für alle

Das Kernstück der Architektur des ETL-Generators bilden die bereits mehrfach erwähnten generischen Templates. Der ETL-Generator stellt in der Datenbank einen Satz von Definitions-Tabellen bereit, in denen die Templates mithilfe der Metadaten beschrieben sind. In den Definitions-Tabellen findet man keine Tabellen- oder Attribut-Namen, die konkreten Objekte werden erst während der Generierung automatisch an die Templates gebunden. Um den Einstieg in das Framework zu erleichtern und die Anlage der Templates zu vereinfachen, besitzen OWB Mapping Generator und ODI Interface Generator jeweils eigene Definitions-Tabellen. Die Begrifflichkeiten beim OWB Mapping Generator sind die gleichen wie im Oracle Warehouse Builder und beim ODI Interface Generator wie im Oracle Data Integrator – man findet sich demnach schnell zurecht.

Im Datenmodell des OWB Mapping Generators sind Informationen über Operatoren, Attribute, Properties und Connections enthalten. Das Datenmodell des ODI Interface Generators beinhaltet unter anderem Data-Sets, Operatoren (wie Tabellen, Joiner oder Filter), Properties (wie Mappings oder Knowledge-Module) und Optionen. Bei Namen für Tabellen-Operatoren, die sich von Prozess zu Prozess unterscheiden

und von der zugehörigen Tabelle abhängen, werden Platzhalter verwendet. Die Property-Tabellen können je nach Anforderung oder Komplexität der umzusetzenden Logik sowohl statische als auch dynamische Werte enthalten (siehe Tabelle 2). Eine Eigenschaft lässt sich mithilfe vordefinierter dynamischer Parameter festlegen. Somit beschreibt man beispielsweise alle Attribute eines Operators zusammen, und nicht jedes Attribut einzeln. Erfordert die fachliche Logik eine umfassendere Berechnung der Werte, etwa abhängig vom Primärschlüssel der Tabelle oder von Datentypen der Spalten, hat man im ETL-Generator die Möglichkeit, eine benutzerdefinierte Funktion anzulegen, die dann in der Property-Tabelle verwendet werden kann.

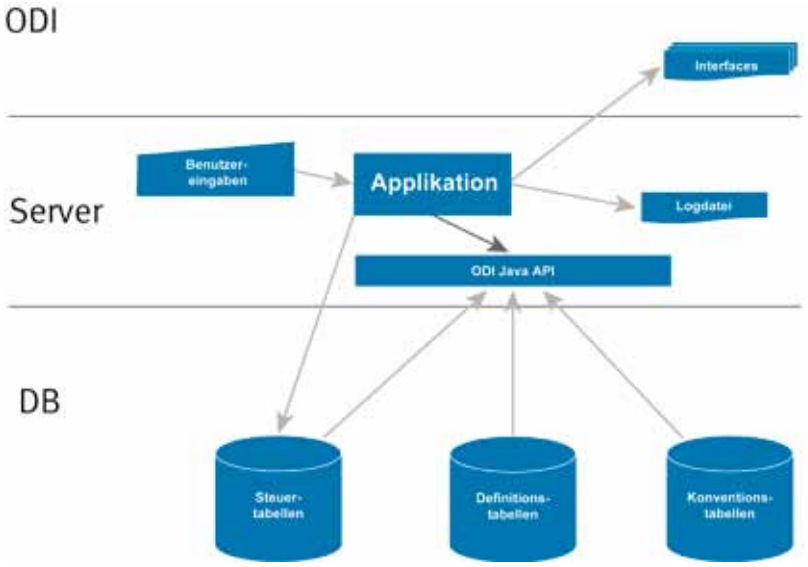

Abbildung 3: Architektur des ODI Interface Generators

Ohne Namenskonventionen läuft nichts

Da die Definition der Templates generisch erfolgt, braucht man nun einen Weg, diese mit den erforderlichen Umgebungsobjekten (Schemata, Tabellen etc.) zu verbinden. Um die Generierung einzelner Prozesse entsprechend den jeweiligen Anforderungen zu ermöglichen, sollte man im ETL-Generator Namenskonventionen und Umgebungsinformationen ablegen. Dabei spielt der Begriff „Tabellenstamm" (table radical) eine zentrale Rolle. Damit ist der gemeinsame Teil der Tabellen-Namen (siehe Tabelle 3) über alle im ETL-Prozess verwendeten Schemata hinweg gemeint.

Der Tabellenstamm wird bei der Generierung von ETL-Prozessen verwendet, um zusammengehörende Objekte in einem Prozess zu verbinden. Die Funktionsweise des Frameworks basiert auf der Annahme, dass alle verwendeten Datenbank-Objekte einer allgemeinen Namenskonvention folgen. In den bereitgestellten Konventions-Tabellen beschreibt man mithilfe der regulären Ausdrücke die Namenskonventionen der Datenbank-Objekte innerhalb der OWB-Module beziehungsweise ODI-Module und legt die Namenskonvention für die zu erzeugenden Mappings beziehungsweise Interfaces fest. Durch die einfache Erweiterbarkeit und Individualisierung des Frameworks können im ETL-Generator beliebige Namenskonventionen abgebildet werden.

Fazit

Der ETL-Generator ist ein kleines Framework, mit dem man die Entwicklung in OWB- und ODI-Projekten enorm beschleunigen kann. Wie in Abbildung 4 leicht zu erkennen ist, lohnt sich der Einsatz des ETL-Generators bei steigender Anzahl der Prozesse. Man profitiert dabei in allen Projektphasen:

- Die Entwicklung wird beschleunigt, da statt einer großen Menge von Prozessen nur noch ein Template entwickelt werden muss
- Vereinheitlichung und damit auch Qualitätsverbesserung des Codes ist ein unstrittiger Gewinn, den man in großen Projekten kaum noch erreichen kann
- Der Testaufwand wird dank des generischen Ansatzes ebenfalls deutlich reduziert
- Auf Neuanforderungen kann schnell reagiert und ein Data Warehouse in kurzer Zeit neu aufgebaut werden
- Die Wartungsaufwände werden bei der Verwendung des ETL-Generators deutlich minimiert, wodurch man mehr Zeit für konzeptionelle Aufgaben und neue Projekte gewinnt

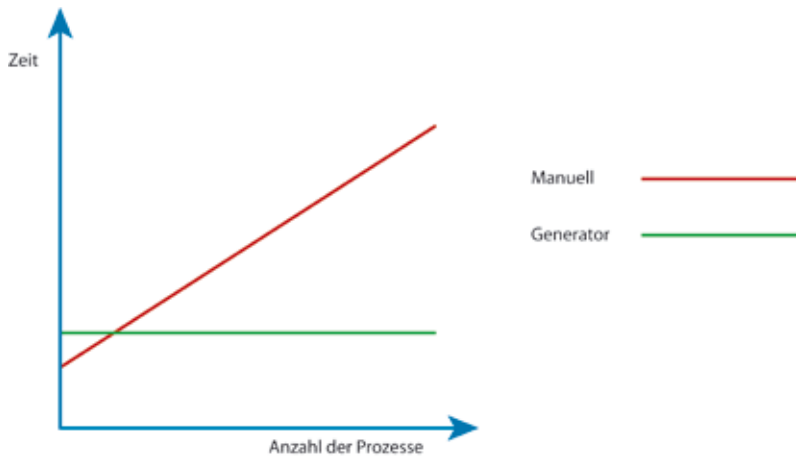

Abbildung 4: Zeitgewinn innerhalb aller Projekt-Phasen beim Einsatz des ETL-Generators

Irina Gotlibovych
irina.gotlibovych@mt-ag.com