Automatisierte Modellierung?
SQL Developer Data Modeler!

Dr.-Ing. Holger Friedrich
sumIT AG
Baden - Schweiz

Schlüsselworte
SQL Developer Data Modeler, Datenmodellierung Data Vault, Automatisierung

Einleitung
Automatisierung ist einer der Mega-Trends im Bereich des Data-Warehousing. Zumeist bezieht er sich auf die automatisierte Erstellung von Ladelogik. Wie Datenmodelle automatisch erstellt werden können wird seltener diskutiert. Programmatische Erstellung von DDL-Dateien oder direkt von Zielstrukturen wird zuweilen angewendet, verzichtet aber als Lösung auf die Vorteile von Enterprise Data Modeling Werkzeugen. Diese sind, unter Anderem, Wartbarkeit der Modelle, Dokumentation, Versionierung und automatische Change-Skript-Erstellung. Leider sind die Klassiker des Genres, z.B. CA Erwin kaum automatisierbar. Arbeiten mit diesen Werkzeugen ist daher langsam und kostenintensiv.
Oracle SQL Developer Data Modeler (SDDM) dagegen enthält Features, mittels derer komplexe Modelle automatisch erstellt, geändert und gewartet werden können. Gleichzeitig bietet er alle Vorteile anderer Enterprise Modellierungswerkzeuge. SDDM erlaubt so die günstige und einfache Erstellung und Wartung grosser logischer und relationaler DWH-Modelle.
Diese Präsentation konzentriet sich auf die Automatisierungseigenschaften des SDDM. Die Bestandteile von Datenmodellen und ihre Repräsentation in SDDM werden im Folgenden angesprochen. Danach wird die prinzipielle Umsetzung von Automatisierung mittels SDDM erklärt. Schliesslich wird gezeigt, wie die Erstellung von Aspekten eines relationalen Modells automatisiert werden kann und was es dazu bedarf.

SDDM – eine kurze Historie
Bis vor einigen Jahren bot Oracle das Werkzeug ‚Oracle Designer' als Enterprise Data Modelling Tool an. Als dieses Produkt auslief, waren Oracles Kunden f:ur einige Jahre gezwungen auf die Produkte anderer Hersteller auszuweichen, wollten sie Datenmodellierung für Oracle Datenbanken durchführen. Einzig für den Bereich des Data Warehousing bot Oracle Modellierungsfunktionalität im Rahmen des, mittlerweile ebenfalls ausgelaufenen ‚Oracle Warehouse Builder' an.

Doch dann, circa 2009 wurde Sql Developer Release 2 um ein Modul zur Datenmodellierung erweitert. Das Modul gab und gibt es auch als alleinstehendes Produkt. Der Sql Developer Data Modeller, kurz SDDM, war geboren. Im Verlaufe der letzten sechs Jahre hat sich das anfänglich recht einfach gehaltene Werkzeug zu einem ausgewachsenen Enterprise-Modellierungs-Werkzeug gemausert, welches zudem vollkommen kostenfrei von Oracle bereitgestellt wird.

Der gegenwärtige Release trägt die Nummer 4.1. Er bietet eine Vielzahl von Modellierungsebenen und –möglichkeiten, von denen im folgenden Abschnitt einige vorgestellt werden.

Modelle, deren Bestandteile und Weiteres
SDDM organisiert alle Projektaspekte in einer klassischen Baumansicht, wie Abbildung 1 sie zeigt.

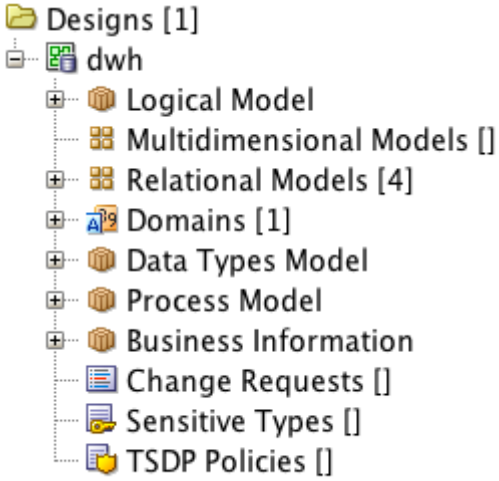

Abb. 1: Projektbaum in SDDM

Man kann sehen, dass alle traditionellen Aspekte der Enterprise-Modellierung vorhanden sind.

- Logische Datenmodelle,
- relationale Modelle,
- Definition von Domänen und
- Datentypen.

Darüber hinaus können aber auch

- Business-Informationen,
- Prozessmodelle
- und beliebige nutzerdefinierte Attribute

in SDDMs dateibasiertem Repository abgelegt werden.

Schliesslich bietet SDDM zusätzliche Leistungen an, darunter

- DDL-Codegenerierung und Synchronisation von Modellen mit Datenbanken
- Integration mit Subversion Version Management
- Report-Generierung für alle Modellaspekte in vielen Formaten (PDF, HTML, XLS etc)

SDDM bietet also alles, was man von einem Enterprise-Modellierungswerkzeug erwarten kann und das noch zu einem unschlagbaren Preis, nämlich kostenfrei. Die meisten Kosten entstehen bei Werkzeugen dieser Art jedoch nicht bei den Lizenzen, sondern im Verlaufe der jahrelangen Nutzung durch die Arbeitszeit, welche die Mitarbeiter aufwenden, um Modelle zu erstellen und zu warten.

Automatisierungsmöglichkeiten

Modellierungswerkzeuge haben bezüglich der laufenden Nutzungskosten in der Regel einen gravierenden Nachteil. Es ist dies die Begrenzung der Produktivität auf die manuelle Arbeitsgeschwindigkeit der Entwickler. Wie schnell und damit wie teuer oder preiswert ein Modell erstellt oder geändert werden kann, hängt in der Regel von der Point- und-Klick-Geschwindigkeit des Benutzers ab. Natürlich gibt es in allen Werkzeugen Wizards und Makrofunktionen, diese sind jedoch in der Regel auf spezielle Anwendungsfälle und –szenarien beschränkt.

SDDM bietet hier weitaus mehr als andere Werkzeuge auf dem Markt. SDDM erlaubt es eine Fülle von Arbeiten zu automatisieren und auf Teilmodelle und Einstellungen komfortabel mehrfach anzuwenden. Das mehrfache anwenden einer einmal geleisteten Arbeit wird zum Beispiel durch die Verwendung von Storage-Templates ermöglicht. Diese können einmal definiert und einer beliebigen Menge von Tabellen zugewiesen werden. Ein Beispiel dafür zeigt Abbildung 2.

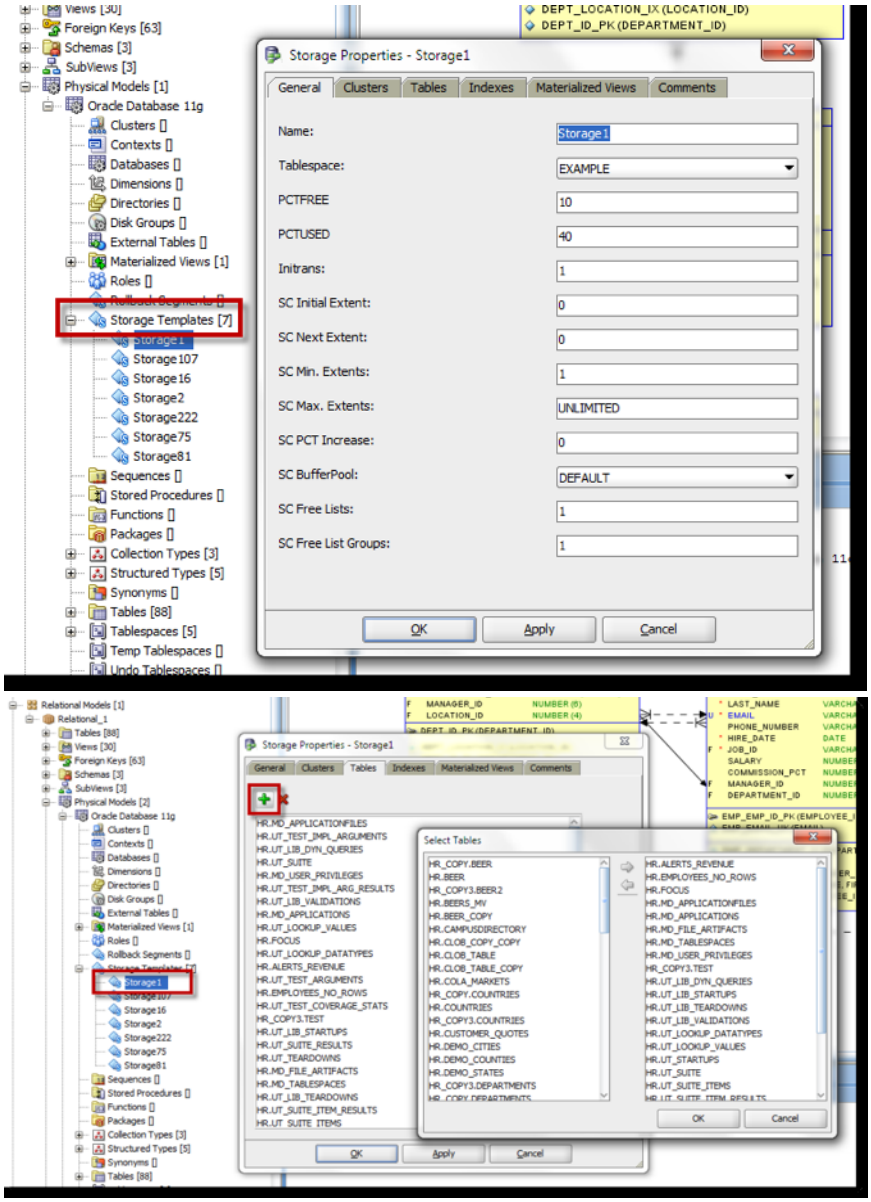

Abb.2: Definition und wiederholte Anwendung von Storage Templates

Auch können auch Benennungs- und andere Regeln formuliert und auf Mengen von Objekten angewendet werden. Massenänderungen sind auf diese Weise schnell und effizient durchführbar, ohne dass Entwickler sich viele Stunden durch Menüs und Attributlisten klicken und die Metadaten von Hand editieren müssten.

Abbildung 3 zeigt das Auswahlmenü und eine Auswahl benutzerdefinierter Automatisierungsskripte.

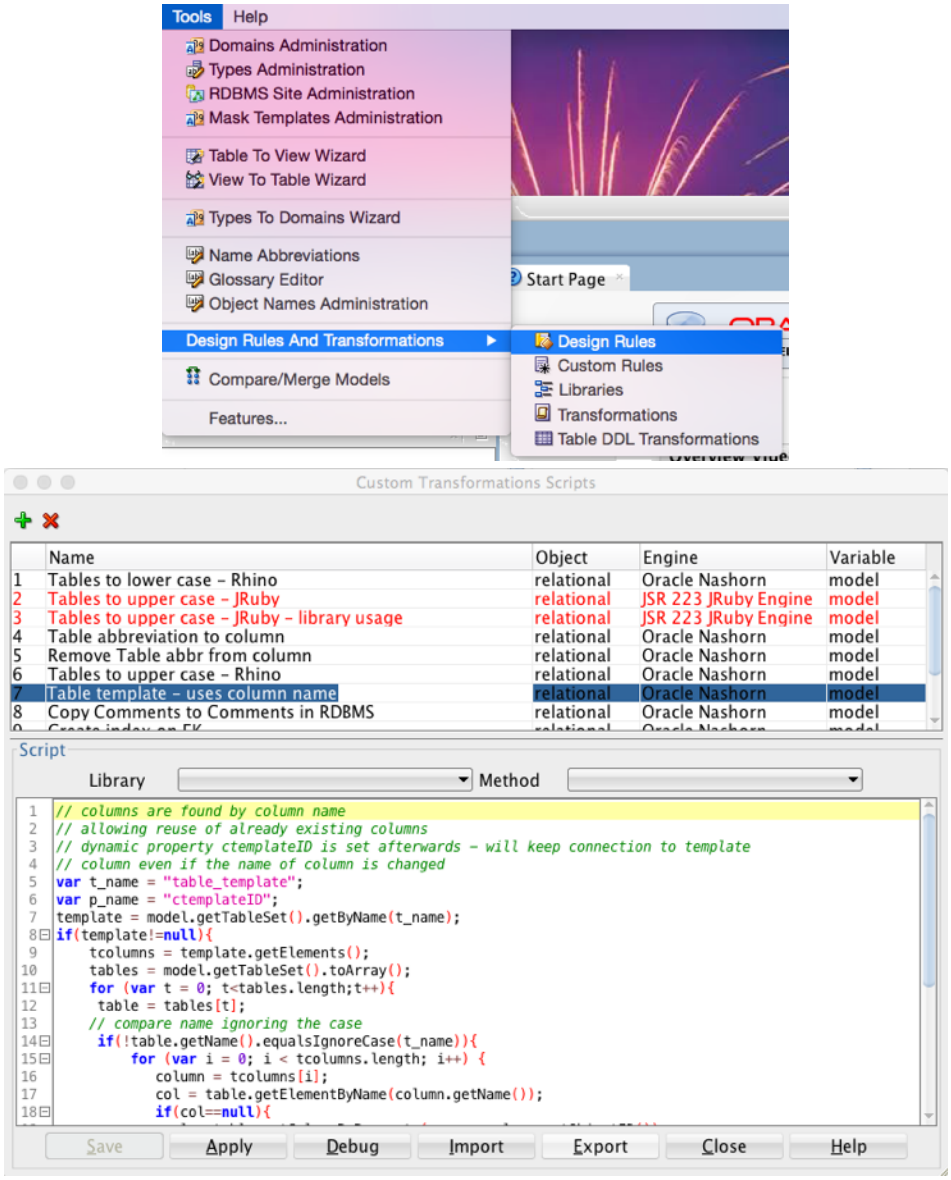

Abb.3: Automatisierungseinträge im Tools-Menu und Beispiele

In der Spalte Engine des unteren Screenshots der Abbildung 3 ist deutlich zu sehen, dass SDDM die Verwendung verschiedener Scripting-Engines und damit Programmiersprachen zur Task-Automatisierung gestattet. Dies gibt den Entwicklern sogar einen zusätzlichen Freiheitsgrad, nämlich eine ihnen bereits vertraute Programmiersprache zur Automatisierung zu verwenden.

Let's get Groovy

Die Anwendung von Templates, einfacher Regeln oder kurzer Skripten zur Durchführung von Massenänderungen bietet bereits mächtige Möglichkeiten. Um das Potential des SDDM voll auszuschöpfen kann man die Automatisierung aber noch einen grossen Schritt weiter treiben. Es lassen sich gesamte Datenmodelle mittels geeigneter Skripte aus Konfigurationsdaten erstellen und auch warten.

Hierzu wird dreierlei gebraucht:

1. Gute Kenntnisse einer Scripting-Sprache, deren Engine von SDDM unterstützt wird.

2. Kenntnis des Java-Objektmodells und der Methoden, welche innerhalb SDDMs verwendet werden.
3. Design-Metadaten, die in geeigneter, maschinell lesbarer Form vorliegen.

Eine hervorragend geeignete Sktiptsprache ist Groovy. Diese gescriptete Java-Variante ermöglicht den Zugriff auf beliebige Java-Libraries innerhalb der SDDM-Automatisierung. Zur Verwendung von Groovy müssen deren Jar-Archive SDDM kenntlich gemacht und so die Engine verfügbar gemacht werden. Danach können in den SDDM Transformations-Libraries eigene Groovy-Skripte hinterlegt und auf Modelle angewendet werden.

Schwieriger zu erreichen ist die vertiefte Kenntnis des SDDM-internen Objektmodells, die darin vorhandenen Methoden und wie diese anzuwenden sind. Dieses Wissen ist absolut notwendig, um innerhalb eines SDDM-Modells navigieren und es geeignet manipulieren zu können. Hierfür gibt es leider bislang keine Entwickler-Dokumentation seitens Oracle. Was es gibt sind

- eine Javadoc-artige HTML-Zusammenstellung der Klassen und Methoden im Installationsbaum des SDDM (`$SDDM_OME/datamodeler/datamodeler/xmlmetadata/doc/index.html`).
- und verschiedenste Snippets im WWW.

Aus diesen Inputs muss man sich die Information zusammensuchen. Dies ist zwar mühsam, aber es lohnt sich.

Schliesslich werden Designmetadaten für das zu erstellende Modell benötigt. Sie müssen in einem Format vorliegen, welches aus Java/Groovy heraus gelesen werden kann. Es bieten sich hier Designtabellen in einem Datenbankschema an, da die eigenen Groovy-Skripte auf diese mittels JDBC-Verbindung zugreifen können. Eine Alternative sind beispielsweise JSON-Dateien. Auch für den Zugriff auf solche Dateien und die Interpretation deren Inhalts, gibt in der Java-Welt viele Libraries, die von Groovy-Skripten eingesetzt werden können.

Aus all diesen Inputs und daraufhin selbstgeschriebenem Code lassen sich Skripte, Klassen und gar ganze Applikation bauen, welche komplexe Modelle in SDDM aufbauen. Abbildung 4 zeigt einen Ausschnitt aus einer Groovy-Applikation mittels derer sumIT für ihre Kunden komplette Basislayer für Data Warehouses nach dem Data-Vault-Modellierungsparadigma automatisch erstellt. Abbildung 4 zeigt einen Ausschnitt aus der komplexen Java-Klasse, welche hierfür implementiert wurde.

```
/* ============================================================
   # Add a sub-partition template to the storage model of the table
   # Prerequisite: table with physical model exists
   #============================================================ */
private void sddmAddSubpartitionTemplate(String p_name, String p_valueListString) {
    def storageObject = this.sddmStorageDesign.getStorageObject(sddmTable.getObjectID());
    def subTmplList = storageObject.getListStorageTemplates();
    def subTmpl = subTmplList.createElement();
    subTmpl.setName(p_name);
    subTmpl.setValueList(p_valueListString);

    // finalize
    this.sddmTable.setDirty(true);
}
```

Abb.4: Ausschnitt aus Java-Klasse zur automatischen Modellierung

Fazit
SQL Developer Data Modeler ist nicht nur kostenfrei, sondern auch in Sachen Effizienz und Entwicklerproduktivität den Mitbewerbern voraus. Die Vielzahl vorhandener Automatisierungsmöglichkeiten sind der Schlüssel für diese Leistungsfähigkeit. Insbesondere die Scripting-Schnittstelle, welche das anbinden komplexer eigener Entwicklungen erlaubt, ist äusserst nützlich. Mit ihr lassen sich umfangreiche Aufgaben, bis zur Erstellung und Wartung von Modellen mit hunderten Objekten automatisieren.

Kontaktadresse:
Dr.-Ing. Holger Friedrich
sumIT AG
Täfernstrasse 28
CH-5405 Baden-Dättwill

Telefon:	+41 (0) 56 – 470 2500
Fax:	+41 (0) 79 – 320 8179
E-Mail	holger.friedrich@sumit.ch
Internet:	www.sumit.ch