Effiziente Mandantisierung eines bestehenden Data Warehouse mit Hilfe von OMB*Plus

Thomas Thies
sumIT AG
Baden - Dättwil, Schweiz

Schlüsselworte:

OWB, OMB+, DWH, Mandantisierung, parallele Entwicklung

Einleitung

Im Zuge des bisher grössten Bankenmigrationsprojektes der Schweiz wurde ein mit Oracle Warehouse Builder entwickeltes Data Warehouse für den mandantenfähigen Betrieb bei der schweizer RBA Bankengruppe im Auftrag des Entris-Banking Servicecenters erweitert. Während der Implementationsphase für die Mandantisierung des Datenmodels wurde die Entwicklung neuer Komponenten auf Basis der alten Logik uneingeschränkt fortgeführt. Sämtliche Änderungen für die Funktionsfähigkeit der mandantisierten Logik des Models wurden durch intensive Nutzung der OMB-Skripting Engine innerhalb des Oracle Warehouse Builders realisiert. Dabei sind knapp 2000 Komponenten des Data Warehouses durch 1,5 Personenäquivalente innerhalb eines halben Jahres auf mandantisierte Verarbeitung inklusive Testing umgestellt wurden. Objekte die während dieser Zeit auf der nicht mandantisierten Logik entwickelt wurden konnten durch den generischen Ansatz ohne Mehraufwand in die mandantisierte Logik überführt werden. Der Kunde bekam auf diese Weise Ressourcensparend eine mandantisierte Version des sich bei der Migrosbank, eine schweizer Retailbank, im Betrieb befindenden Data Warehouses. Aufgrund der generischen Implementierung der Logik konnten individuelle Fehler einzelner Entwickler vollkommen ausgeschlossen und die Risiken minimiert werden. Dieser Vortrag verdeutlicht das Vorgehen im Projekt und zeigt welche Techniken eingesetzt wurden.

Grundlegendes

Die sumIT entwickelte in den Jahren 2008 und 2009 ein neues Data Warehouse für die Migrosbank, eine grosse Retailbank, in der Schweiz. Bei diesem Migrationsprojekt wurde die Core-Banking Lösung komplett ausgetaucht und es wurde neu das Core-Bankingsystem von FINNOVA eingesetzt. Das neue Data Warehouse konnte sozusagen auf der grünen Wiese erstellt werden, da keine Rücksicht auf in der Vergangenheit benutzte Lösungen und deren Fortbestand genommen werden musste.
Die Migrosbank entschied sich für den Vorschlag der sumIT das Data Warehouse mit Hilfe von ORACLE Technologien zu entwickeln. Dabei stand bereits eine effiziente Implementierung mit möglichst wenig Personaleinsatz im Vordergrund. Um in der gegebenen Zeit und mit den gegebenen Ressourcen eine möglichst umfassende, flexible und robuste Lösung zu implementieren und da wir uns im Bereich Warehousing in einer reinen Oracle Landschaft bewegten, wurde der Oracle Warehouse Builder eingesetzt. Hier wurde von Beginn an massgeblich auf die eingebaute TCL-

Skriptingfunktionalität zurückgegriffen um die Entwickler bei der Implementierung des ETL-Models möglichst effizient zu unterstützen.
Es entstand eine Lösung in der weite Teile des Models generisch durch sogenannte Experten generiert werden konnten. Dadurch konnten die Entwickler mehr Zeit in die Business Analyse der einzelnen Entitäten investieren, ausserdem wurde in grossen Teilen des Models die Gefahr von individuellen Entwicklerfehlern minimiert beziehungsweise sogar ganz ausgeschlossen.
Der Output eines Entwicklers (Businessobjekt / Zeit) konnte zusätzlich erhöht werden im Vergleich zu einer Lösung in der jeder Entwickler die Logik einzelner Entitäten vollständig von Extraktion über Transformation bis hin zum Load hätte händisch implementieren müssen.

Das Model

In diesem Kapitel wird grob das bestehende Model gezeigt um leichter den notwendigen Aufgaben für einen Umbau zur Mandantenfähigkeit folgen zu können.

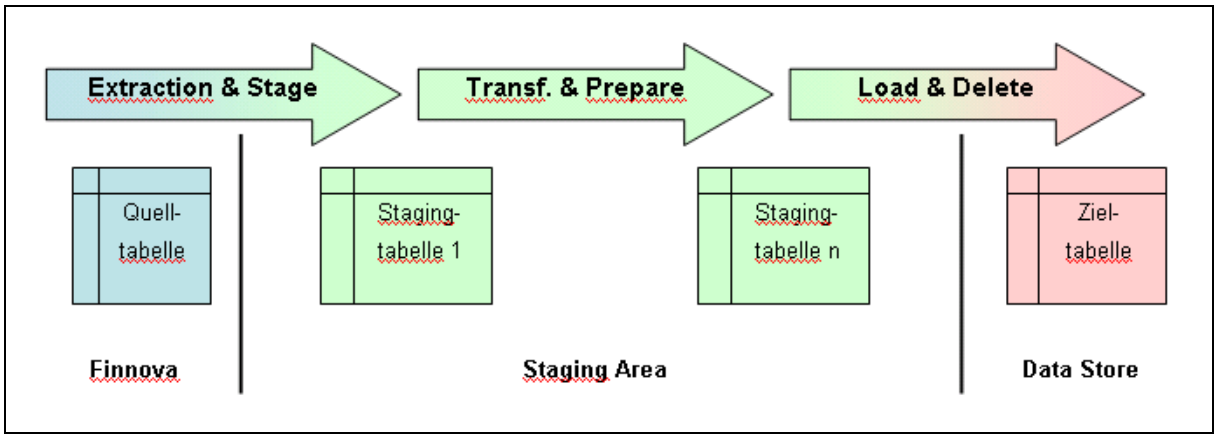

Abb.1: Schematische Skizze des Models

In Abbildung 1 ist der schematische Aufbau der Ladereihenfolge dargestellt. In der Staging Area werden die aus dem Quellsystem abgezogenen Daten konsolidiert sowie teilweise denormalisiert und zur Weiterverarbeitung innerhalb der Instanz in sogenannten IN-Tabellen abgelegt. Das hat den Vorteil das die Quellsysteme beim nächtlichen Staging der Daten lediglich für eine möglichst kurze Zeit belastet werden. Im nächsten Schritt werden die Daten aus den IN-Tabellen transformiert und in TR-Tabellen abgelegt. In diesem Schritt werden Foreignkeyrelationen aufgelöst, die Benamung der Attribute standardisiert und das Change Data Capturing wird vorbereitet. Die in dieser Form aufbereiteten Daten durchlaufen nun in mehreren parallel laufenden Verarbeitungsschritten verschiedene Arten von Mappings, die feststellen welche Art der Änderung an den Daten stattgefunden hat. Handelt es sich also um Updates, Deletes bzw. Inserts innerhalb der Quellsysteme. Ausserdem werden Besonderheiten des Quellsystems (wie z.B. Reaktivierte Datensätze, Selbst- und Kreuzreferenzen) abgefangen und korrekt verarbeitet. Die so aufbereiteten und mit der Art der Verarbeitung getaggten Daten werden in sogenannten PR-Tabellen abgelegt die als letzte Stufe vor dem Load in den Data Store stehen. Im letzten Schritt werden die Daten aus den PR-Tabellen in Historisierte bzw. Nicht Historisierte Business Objekte des Data Stores geladen.

Der Umbau

Das bestehende Model wurde nun Entris-Banking, die in Zukunft das Data Warehouse der RBA-Bankengruppe betreiben, zur Verfügung gestellt und musste dort für die Anforderungen einer mandantenfähigen Data Warehouse Lösung umgebaut werden. Die beiden Parteien (Migrosbank, Entris-Banking) vereinbarten eine Zusammenarbeit im Bereich Data Warehousing um von den Synergieeffekten bei der Weiterentwicklung des in Zukunft gemeinsamen Data Warehouse Models gegenseitig zu profitieren.
Für die sumIT bedeutete dies: Lediglich eine gemeinsame Codebasis zu führen, die Weiterentwicklung sowie der Betrieb des bestehenden Data Warehouses musste uneingeschränkt gewährleistet sein und die weiter beziehungsweise neu entwickelten Objekte mussten für die Zeit des Umbaus für einen Mandanten und nach Fertigstellung des Modelumbaus ebenfalls in einem mandantisierten Umfeld korrekt funktionieren. Ein temporärer Stop der Weiterentwicklung für die Zeit des Umbaus war ausgeschlossen.
Somit mussten die knapp 1200 in OWB implementierten Mappings und c.a. 700 Tabellen des Metadatenmodels auf mandantisierte Logik umgestellt werden während das Betriebs- und Entwicklungsprojekt der Migrosbank ohne mandantisierte Logik weiterhin für die Dauer der Entwicklung der erweiterten Logik funktionieren musste und zu einem bestimmten Zeitpunkt innerhalb kürzester Zeit auf Mandantisierung umgestellt werden sollte. Ausserdem waren für die Analyse der durchzuführenden Arbeiten sowie der Implementierung der neuen Logik nur 1,5 Personenäquivalente vorgesehen.

Herausforderungen

Es mussten verschiedene Herausforderungen bewältigt werden. Zum einen musste bei forlaufendem Betrieb der alten Logik die Entwicklung der mandantisierten Logik vorangetrieben werden. Da es sich um ein Metadatenmodel handelt wäre es möglich gewesen im Repository des laufenden Systems stückweise die neue Logik zu etablieren. Werden allerdings Bugs im alten Model entdeckt die beseitigt und auf die physische Instanz ausgebracht werden müssen, könnte es passieren das der entsprechende Teil schon auf mandantisierte Logik umgebaut wurde und ein Deployment wäre in diesem Fall nicht mehr möglich gewesen, da zu diesem Zeitpunkt bereits ein Mix aus alter und neuer Logik im Repository vorhanden gewesen wäre.
Eine weitere Herausforderung war die Weiterentwicklung des Models auf Grundlage der alten Logik parallel zur Entwicklung der neuen Logik. Während dem halben Jahr der Mandantisierungs-entwicklung wurden permanent neue Objekte in der alten Umgebung auf Grundlage der alten Logik entwickelt und produktiv gesetzt. Eine parallele Entwicklung der mandantisierten Logik im gleichen Metadatenrepository hätte zwangsläufig dazu geführt das Teile des Models und der neu implementierten Objekte bereits mandantisierte Logik enthielten und somit nicht für das laufende System benutzbar gewesen wären.

Lösungen

Wir entschieden uns für ein eigenes OWB Repository in dem mit jedem Release des Data Warehouse Models ein vollständiger MDL-Import des aktuellen Models durchgeführt wurde. Der Vorteil war, daß das Betriebsprojekt völlig frei ihre Weiterentwicklung betreiben konnte und durch den Umbau der Mandantisierung in keinster Weise beeinträchtigt wurde.
Nachdem die Analyse der durchzuführenden Arbeiten für den Umbau des Models abgeschlossen war, wurde für jede Mappingart (IN, TR, PR, Load) ein Migrationsskript erstellt. Zusätzlich wurde ein

weiteres Skript angefertigt das alle nötigen Umbauten an den Tabellenmetadaten durchführt. Die Skripte für den Umbau der Mappings teilten diese anhand gewisser Kriterien in Gruppen ein. Somit war es möglich für jeden neuen Release des Models sämtliche Skripte auf dem Entwicklungsrepository anzuwenden und die umgebaute Logik auf Plausibilität zu testen. Durch die in grossen Teilen generische Generierung neuer Objekte des ursprünglichen Models war es mit den Mandantisierungsskripten möglich auch diese neuen Objekte ohne weiteren Mehraufwand in die mandantenfähige Logik zu überführen.

Abschliessende Feststellung

Trotz des hohen Grades an generisch generierten Business Objekten im Model waren viele Ausnahmen in der Logik zu finden. Gerade im Bereich der TR-Mappings in denen der wesentliche Transformationsschritt vom Quellobjekt zum Datastoreobjekt implementiert ist dominierten die Ausnahmen von der standardisierten Logik da viele Quellobjekte unterschiedliche Transformations-schritte benötigten.
Wir hatten bei unserer Herangehensweise die Mandantisierung des Models zu implementieren natürlich den Vorteil das wir das zugrunde liegende Model bereits hochgradig generisch und auf Standardisierung ausgelegt entwickelt hatten. Allerdings sollte, bei gutem Design, jedes grössere Projekt in dem mehrere Entwickler beteiligt sind einen gewissen Grad an Standardisierung aufweisen. Daher sollte ein vollumfänglicher Umbau bestehender Logik durch OMB*Plus in den meisten grösseren Projekten effizient möglich sein. Auch wenn es ein wenig mehr Aufwand bei der Analyse der bestehenden Logik erfordert so liegen die Vorteile einer skriptbasierten Lösung klar auf der Hand.
Bei der skriptbasierten anpassung von Entitäten werden individuelle Entwicklerfehler, die in der Regel beim Testing schwer zu finden sind, faktisch ausgeschlossen. Gibt es einen Bug in einem Skript, zieht sich dieser durch alle von diesem Skript bearbeiteten Entitäten und ist grundsätzlich an der selben Stelle im Mapping zu finden. Das macht es sehr leicht solche Fehler zu identifizieren und zu korrigieren.
Wird ein Model skriptbasiert umgebaut bleibt das grundlegende Model so lange bestehen und funktionsfähig bis die Skripte endgültig auf den produktiven Metadaten angewandt werden. Bevor dies geschieht besteht in häufigen, iterativen Anwendungen der Skripte die Möglichkeit potenzielle Fehler in der neuen Logik zu eliminieren ohne dabei das produktive Metadatenrepository zu modifizieren.
Letztendlich entfällt auch das Zeitaufwändige „zeichnen“ der Logik auf dem Oracle Warehouse Builder Canvas und damit vermindert sich automatisch der Personalbedarf für die Entwicklung der Skripte.

Kontaktadresse:

Thomas Thies
sumIT AG
Täfernstrasse, 28
CH-5405 Baden-Dättwil

Telefon:	+41 (0) 76-440 3700
Fax:	+41 (0) 56-470 2505
E-Mail	post@thomasthies.eu, thomas.thies@sumit.ch
Internet:	www.sumit.ch