12c Oracle Warehousing – voll Groovy; Ein Projektbericht

Dominic Ketteltasche und Bernhard Rosenberger, MT AG
Ratingen
Jörg Lang, GFKL
Essen

Schlüsselworte

Oracle, ODI, Data Integrator, Oracle Warehouse Builder, OWB, groovy, ETL, 12c, Datawarehouse, DWH, deklarativer Code, SQL, OMB, SQL Developer, APEX

Einleitung

Der ETL-Platzhirsch im Oracle DWH Umfeld war bislang der OWB. Hält der ODI in der neuen Version 12c den Nachfolgeversprechungen wirklich stand?

Ein aktuell gemeinsam mit dem Kunden GFKL auf der grünen Wiese aufgesetztes Data Warehouse von strategischer Bedeutung war die einmalige Chance dessen Tauglichkeit eingehend auf Herz und Nieren zu prüfen. Die Datenbank und ODI in12c, der jüngste SQL Developer mit Data Modeler und APEX zur Applikationsentwicklung sind dabei im Einsatz.

Die Zielarchitektur des neu zu entwickelnden DWH ist nachfolgend dargestellt:

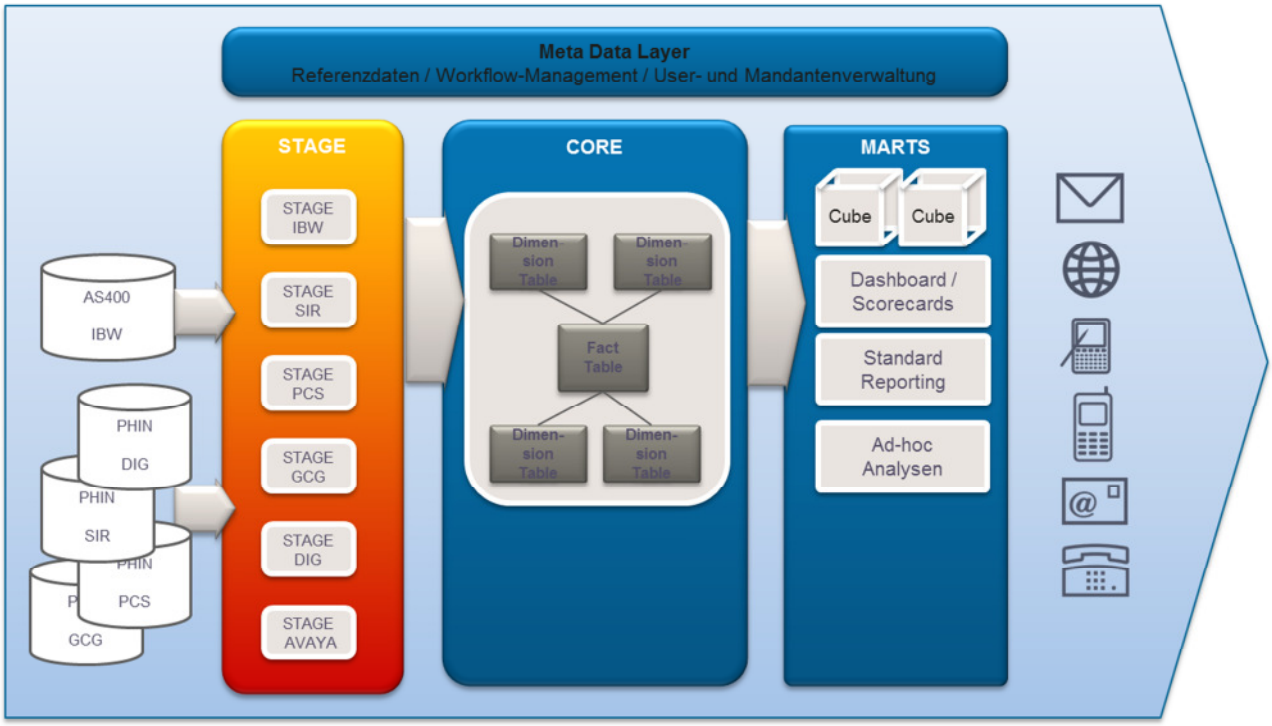

Abb. 1: DWH Zielarchitektur

DWH auf der grünen Wiese

Unser Kunde GFKL stand vor der Herausforderung, eine historisch gewachsene heterogene ETL-Landschaft abzulösen, um die IT-Infrastruktur auf lange Sicht auszubauen und eine flexible Erweiterbarkeit zu ermöglichen.

Die GFKL als Finanzdienstleister im Forderungsmanagement ist aufgrund einer großen Vielfalt zu verwaltender Einzeltransaktionen auf effizient strukturierte Datenhaltung angewiesen, um eine kundenorientierte schlanke Abwicklung Ihres Geschäfts sicherstellen zu können.

Wegen der im Hause fundiert vorhandenen technischen Kenntnisse fiel die Entscheidung, ein DWH mit der aktuellen Datenbankversion 12c, dem ODI 12c als ETL Werkzeug, dem SQL Developer und APEX zur ad hoc Berichtserstellung komplett neu aufzubauen. Zur Unterstützung in Architekturfragen und bei der Realisierung wurde die MT AG mit hinzugezogen.

Die fachliche Herausforderung bestand in der semantischen Datenintegration unterschiedlich ausgerichteter Tochtergesellschaften mit dem Ziel ein zukunftsfähiges vereinheitlichtes Berichtswesen mit Dashboards und KPI's für die Geschäftsleitung aufzubauen.

Bei der Realisierung des DWH kamen einige neue Features des ODI 12c und der 12c Datenbank zur Anwendung. Im Vortrag wird aufgezeigt, wie diese technischen Neuerungen zum Gelingen des Vorhabens beigetragen haben.

Einsatz von 12c Technologien

Ein prognostiziertes tägliches Datenwachstum von vielen Gigabytes motivierte das Team zur Anwendung der 12c Features Identity Columns, Advanced Compression und Intervall Reference-Partitionierung, die eine Minimierung des Speicherplatzes und reduzierten Wartungsaufwand mit dem Vorteil des optimierten Lesezugriffs versprachen.

Die Anwendung von Advanced Compression auf Tablespace-Ebene ermöglichte eine spürbare Reduktion des täglichen Datenwachstums. Eine Verwendung von Intervall-Reference Partitioning bot sich bei partitionierten Parent Child Beziehungen im Core an. Außerdem vereinfachten Identity Columns die Generierung technischer Keys, die für die Integrität im CORE gebraucht wurden.

Der Vortrag wird die dabei gemachten Erfahrungen aufzeigen.

Installation und Konfiguration der ODI 12 Umgebung

Da das Projekt auf der grünen Wiese gestartet wurde, standen zu Beginn Überlegungen, wie der ODI sinnvoll aufgesetzt werden kann. Wie soll die Trennung von Entwicklung und Produktion aussehen, welche Repositories sind erforderlich? Welche ODI Agents sollen zum Einsatz kommen? Zumal neben dem bewährten Standalone Agent seit 11c ein JEE Agent und mit der aktuellen 12c ein Collocated Agent zur Auswahl stehen.

Mehrere mögliche Repository Konfigurationen waren zu bewerten, um im Umfeld der GFKL ein Master Repository mit drei Work Repositories als die sinnvollste Konfiguration auszuarbeiten.

Implementierung mit dem ODI 12c

Gewählt wurde eine klassische Stage-Cleanse-Core-Mart Architektur. Im Staging erfolgt das Bereitstellen der Rohdaten aus den Vorsystemen mittels Datapump. Im Cleansing werden die Daten der Vorsysteme zusammengeführt und bereinigt bevor daraus ein integriertes Core beladen wird. Die Strukturierung im ODI Projektmanager ist in nachstehendem Screenshot zu sehen:

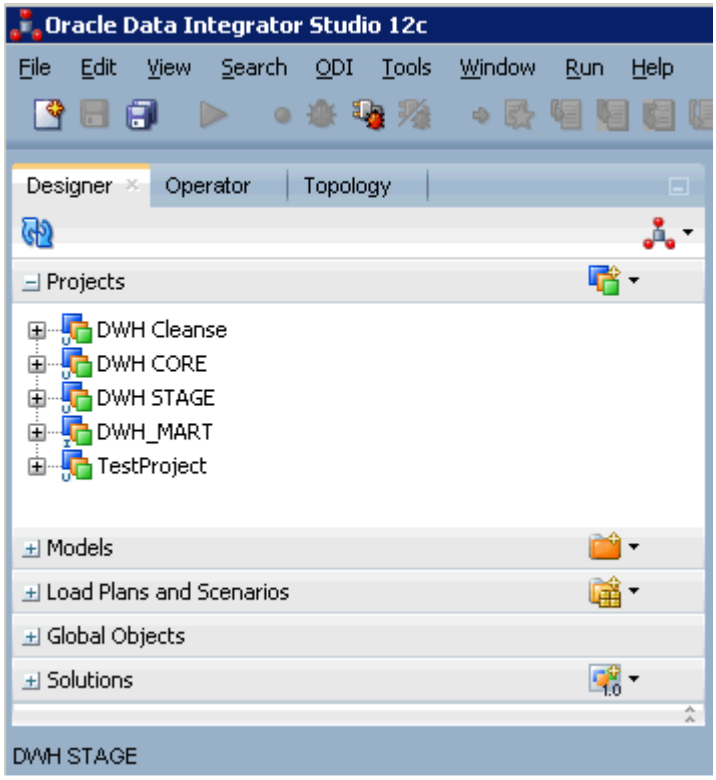

Abb. 1: ODI Projektstruktur

Aufbau der Staging Area

Zum Aufbau des Staging mussten vier Vorsysteme mit jeweils 42 Tabellen angebunden werden. Um die unvermeidliche Fleißarbeit bei der Implementierung gleichartiger Mappings zu vermeiden, konnte die "Groovy Skripting Language" erfolgreich eingesetzt werden. So genügt es eine Beladungsstrategie einmalig (z.B. Datapumpload oder Deltaabzug) zu skripten, um daraus ODI Ladestrecken für jede beliebige Tabelle automatisch generieren zu können. Das Vorgehen reduzierte die manuellen Aufwände erheblich und darüber hinaus die Fehlerquote bei der Stage Implementierung.

```
…
icps = comp_set.getInputConnectorPoints() //2 input connector points
i = icps.iterator()
icp1 = i.next() // First input connector point
icp2 = i.next() // Second input connector point
comp_1.connectTo(icp1)
comp_2.connectTo(icp2)
…
```

Abb. 2: Groovy Snippet – setzt im Mapping zwei Eingangsverbindungen eines ODI Set-Operators

Neben den klar auf der Hand liegenden Vorteilen, wie Zeitersparnis bei der Erstellung von gleichartigen Mappings, weist der Vortrag auf die Fallstricke hin, die während des Projekts umschifft werden mussten.

ODI Mappings im Core

Die komplexeren Mappings wurden aus Zeitgründen zu Beginn als einfache Mappings realisiert, welche auf Datenbank-Views als Datenquelle aufsetzen, in denen sich die komplexe SQL-Logik befindet. Diese Logik wurde im weiteren Projektverlauf aus der View in das eigentliche Mapping verlagert und die View durch die tatsächlichen Datenbanktabellen ersetzt. Der Vortrag zeigt die Vor- und Nachteile beider Ansätze auf.

ODI Check Constraints

Ebenfalls eingesetzt werden ODI Check Constraints, um die Datenintegrität nach einer Beladung hinsichtlich technischer (Unique Keys, Foreign keys) und fachlicher Regeln prüfen zu können. Es bestand kundenseitig der Wunsch fehlerhafte Datensätze nicht auszusteuern, sondern durchzuladen und als fehlerhaft zu markieren.

APEX dient hier als Reporting Werkzeug, wobei auf die E$_-Tabellen des ODI zugegriffen wird um ausgesteuerte Fehlersätze zu berichten.

Packages und LOAD Plans

Die Orchestrierung des Ladevorgangs im ODI wurde mit den ODI Elementen Globale Variablen, Packages, Procedures und Load Plans realisiert. Dabei konnte eine sinnvoll parallelisierte Anordnung von Prozessen zu einer insgesamt deutlich optimierten Auslastung der Rechnerressourcen und zu reduzierten Ladezeiten beitragen.

APEX zum Reporting und zur Metadatenpflege

Zur Vereinheitlichung der zum Teil semantisch unterschiedlich interpretierbaren Daten der Vorsysteme war eine Bereitstellung von Lookuptabellen erforderlich, um eine vereinheitlichte Darstellung im CORE ablegen zu können. Die Pflege der fachlichen Codes und deren vereinheitlichte Übersetzung wurde mit Hilfe einer APEX Anwendung realisiert und dem verantwortlichen Fachbereich zu Verfügung gestellt.

Als Interimswerkzeug für KPI Reports wird ebenfalls APEX eingesetzt. Darüber sind mit vertretbaren Aufwand Berichte realisiert, die sogar fest vordefinierte Drill Downs ermöglichen.

Aufgetretene Probleme

Konnte der Einsatz neuester Oracle Technologien zu einem reibungslosen Projektfortschritt beitragen? Die wesentlichen Projektziele wurden im Rahmen des selbst gesteckten Zeitrahmens erreicht. Trotzdem sei erwähnt, dass einige Schwierigkeiten zu meistern waren. Der ein oder anderen Service

Request musste gestellt werden und so waren auch neue Versionseinspielungen in Datenbank und ODI erforderlich, um die Probleme zu beheben.

So werden wir Herausforderungen beispielsweise beim Einsatz von Groovy, bei globalen Variablen und kaskadierender Parallelität schildern. Ein weiteres Beispiel ist SQL Code, welcher im SQL Developer lauffähig ist, beim Aufruf durch den ODI aber Abbrüche produziert.

Zusammenfassung

Wie sind die im Projekt gemachten Erfahrungen im Vergleich zum OWB einzuordnen?

- Komplexe Mappings mit dem ODI? Geht das?
  Das Projekt hat gezeigt, dass sich natürlich sehr aufwändige Abfragen mit Views kapseln lassen – was zu erwarten war. Darüber hinaus ist dem ODI in 12c aufgrund neu spendierter Component Style Knowledge Module die Fähigkeit gegeben worden, Mappings ähnlich zu strukturieren wie es im OWB möglich war.
- automatisierte Mapping Generierung mit Groovy.
  Der Einsatz der Groovy Scripting Language ermöglichte eine automatisierte Generierung von Mappings, wie es im OWB mit der Tcl Sprache OMB*Plus möglich ist. Es wurden alle Erwartungen erfüllt und teilweise auch übertroffen.
- Einsatz von Load Plans
  Mit den Elementen Variablen, Procedures und Packages lassen sich komplexe Load Plans implementieren. Parallele Prozessverarbeitungen sind möglich und so blieben im Projekt keine Wünsche im Vergleich zur Workflow-Technik des OWB offen.
- Performance mit dem ODI
  Ein direkter Vergleich ist hier schwer zu treffen. Beide Tools generieren SQL Code, welcher auf der Zieldatenbank ausgeführt wird. Schwierigkeiten sind hier mit ODI 12c lediglich im Zusammenhang mit Update-Statements zu erwähnen. Das generierte SQL mit umständlichen Subqueries führte zu nicht akzeptablen Laufzeiten. Ein SR bei Oracle führte zu der Empfehlung einen Patch einzuspielen
- Historisierung out of the box?
  Die in 12c überarbeitete SCD Funktionalität verspricht durch Konfiguration des Tabellen Modells den „Out oft he box"- Einsatz eines Knowledge Moduls mit SCD 2 Funktionalität. Wir zeigen den Weg auf.

In Summe wurde der OWB 11gR2 im Rahmen des Projekts nicht wirklich vermisst. Im Gegenteil sehen wir, wie der ODI komplexe Mechanismen, wie z.B. SCD 2, elegant in den Knowledge Modulen kapseln kann und damit die Entwicklungszeit zu reduzieren hilft.

Die Laufzeitinformationen des ODI Repositories die über den Operator Tabulator abgefragt werden können, lieferten ausreichend Informationen um Fehler rasch zu identifizieren und zu beheben. Im Gegensatz zum OWB war es im Projekt nicht erforderlich zusätzliche „Self-made" Repository Queries bereitzustellen.

Kontaktadresse:

Bernhard RosenbergerMT AG

Solmsstr. 10

D-60486 Frankfurt

Telefon:	+49 69 2649243-20
Fax:	+49 2102 309 61-10
E-Mail	bernhard.rosenberger@mt-ag.com
Internet:	www.mt-ag.com

Dominic Ketteltasche

MT AG

Balcke-Dürr-Alle 9

D-40882 Ratingen

Telefon:	+49 2102 309 61-0
Fax:	+49 2102 309 61-10
E-Mail	dominic.ketteltasche@mt-ag.com
Internet:	www.mt-ag.com