Best Practices im generierten Oracle Data Warehouse mit PL/SQL

Christoph Hisserich

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

03.04.2014

trivadis
makes IT easier.

Christoph Hisserich

- BI Consultant bei Trivadis in Stuttgart
- Seit 4 Jahren im Bereich DWH/BI tätig
- Kernthemen: OWB & ODI

trivadis
makes IT easier.

Mit über 600 IT- und Fachexperten bei Ihnen vor Ort

12 Trivadis Niederlassungen mit über 600 Mitarbeitenden

200 Service Level Agreements

Mehr als 4'000 Trainingsteilnehmer

Forschungs- und Entwicklungs-budget: CHF 5.0 Mio. / EUR 4.0 Mio.

Finanziell unabhängig und nachhaltig profitabel

Erfahrung aus mehr als 1'900 Projekten pro Jahr bei über 800 Kunden

trivadis
makes IT easier.

AGENDA

1. Übliche Data Warehouse Situationen
2. Das generierte Data Warehouse
3. Architektur Best Practices in DWHs mit PL/SQL

Übliche Data Warehouse Situationen

Data Warehouse Objekte

- Anzahl der Tabellen in Stage, Cleanse, Core und Data Mart kann schnell in den 4-stelligen Bereich gehen
- Laderoutinen für Stage sind zu 99% identisch
  - ETL-Tools (z.B. OWB): pro Stage Tabelle ein Mapping, kein Copy/Paste möglich
  - PL/SQL: Quell- und Zielattribute müssen explizit genannt werden, Aufwand selbst mit Data Dictionary recht hoch

Typische (Flüchtigkeits-)Fehler

- Stage oder Cleanse Tabellen werden nicht geleert

- Join Condition nicht komplett

```
CLS.MANDANT_NR          =   COR.MANDANT_NR AND
CLS.GESELLSCHAFT_NR     =   COR.GESELLSCHAFT_NR AND
CLS.PRODUKT_NR          =   COR.PRODUKT_NR
```

- LEFT bei Join Anweisung vergessen

Data Warehouse Objekte (2)

- Tabellen und Loader in Cleanse, Core oder Mart unterscheiden sich häufig durch spezielle Geschäftslogik
- Grundfunktionen sind jedoch identisch
  - Historisierung im Core
  - Delta- und Restartfähigkeit
  - „Overhead“ (Logging, Auditing, Jobsteuerung)
- Problem: Mehrere Entwickler haben nicht immer den gleichen Programmierstil

Das generierte Data Warehouse

Das generierte Data Warehouse

- Generierung von Objekten (Tabellen, Views, Loader etc) hilft Flüchtigkeitsfehler zu vermeiden
- Geschäftslogik von technischer Implementierung trennen
- Inhaltliches Design als Metadaten erfassen

Die einzelnen Schritte

- Anlegen der Quellsysteme
- Import der Quellobjektstrukturen
- Modellierung der Zielstruktur
- Erstellen von Templates
- Generierung der Scripts für Zieltechnologie (z.B. PL/SQL)
- Deployment auf Zielsystem

Quellsysteme anlegen & importieren

- Verbindungsinformationen hinterlegen
- Quellstrukturen einlesen
  - Tabellen und Views inkl. Attribute
- Quellobjekte auswählen und importieren

Modellierung der Zielstruktur

- Unterteilung nach Entitäten (Stammdaten), Dimensionen und Fakten (Bewegungsdaten)
- Entitäten werden aus Quellobjekten erzeugt oder manuell definiert
  - (= Zielstruktur im Core)
- Dimensionen setzen sich aus einer Hierarchie von Entitäten zusammen
  - (z.B. Product – Subcategory – Category)
- Fakten werden wie Entitäten erzeugt
  - Definition der Fremdschlüssel-Beziehungen über Dimensionen

Erstellen von Templates

- Simple Syntax für dynamische Inhalte (Variablen)
- foreach-Schleifen, z.B. um Attribute einer Entität auszulesen
- if-else, um verschiedene Fälle in einem Template abzubilden
- Namenskonventionen global definieren und in Templates wiederverwenden
- Erstellung wie Programmierung normaler PL/SQL Scripts
  - `CREATE TABLE <#= Instance.Name #>`
  -
```
<#
    if(column.IsBusinessKey)
            WriteLine("DWH_BK_"+col.Name+" NUMBER NOT NULL")
#>
```

Generierung und Deployment

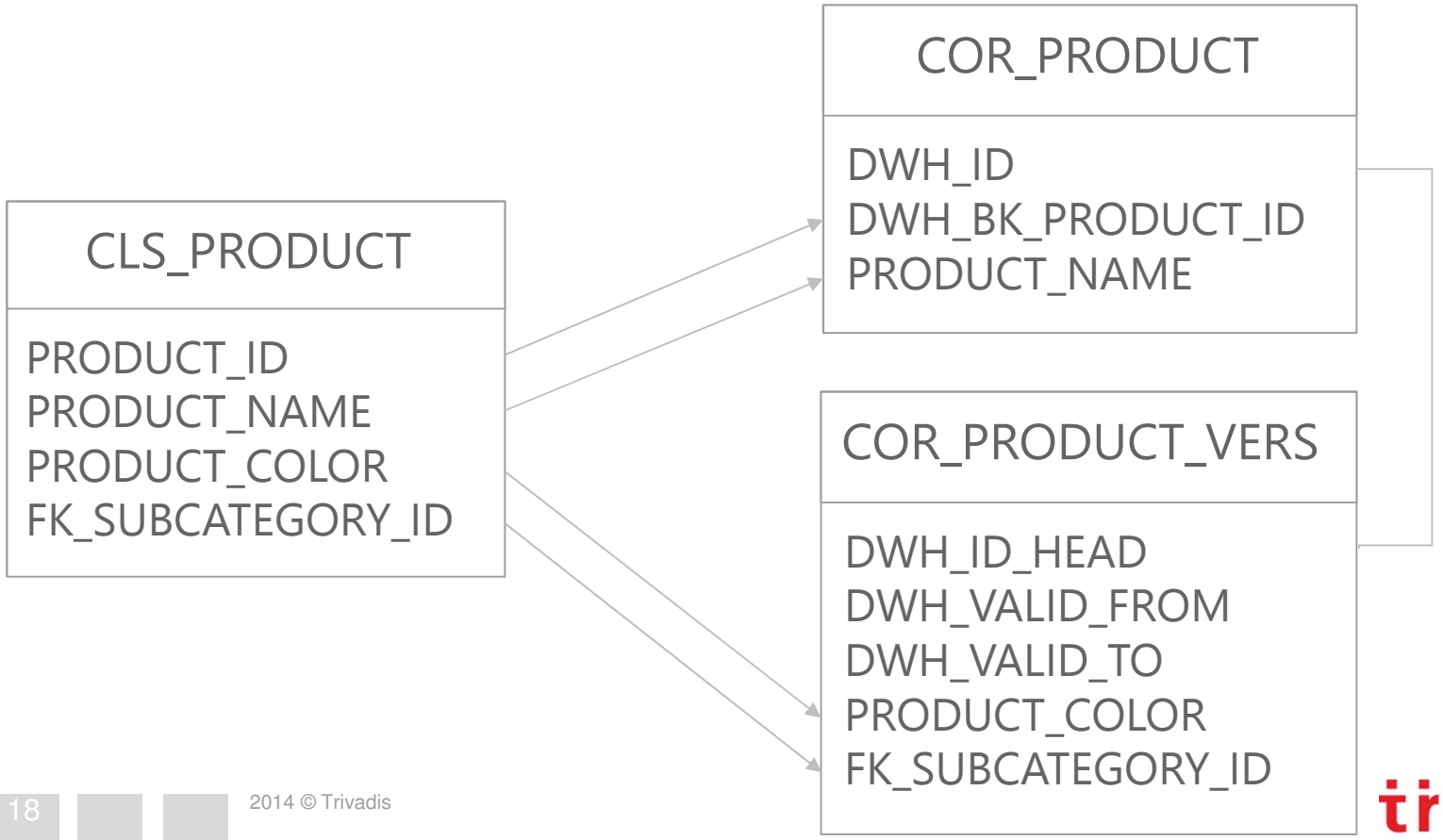
- Nach Definition der Templates können diese „kompiliert“ werden
- Resultat sind fertige Scripts, die auf dem Zielsystem ausgeführt werden können
  - PL/SQL Scripts
  - OMB*Plus (OWB)
  - Groovy (ODI)

Architektur Best Practices in DWHs mit PL/SQL

Mögliche Zielarchitektur

- Stage
  - Source View → Stage Loader → Stage Tabelle
- Cleanse
  - Join View → Cleanse Loader → Cleanse Tabelle
- Core
  - Kopf (SCD1-Attribute)
    - Lookup View → SCD1 View → Core Loader → Core Tabelle (Kopf)
  - Version (SCD2-Attribute)
    - Lookup View → SCD2 View → Core Loader → Core Tabelle (Version)
- …

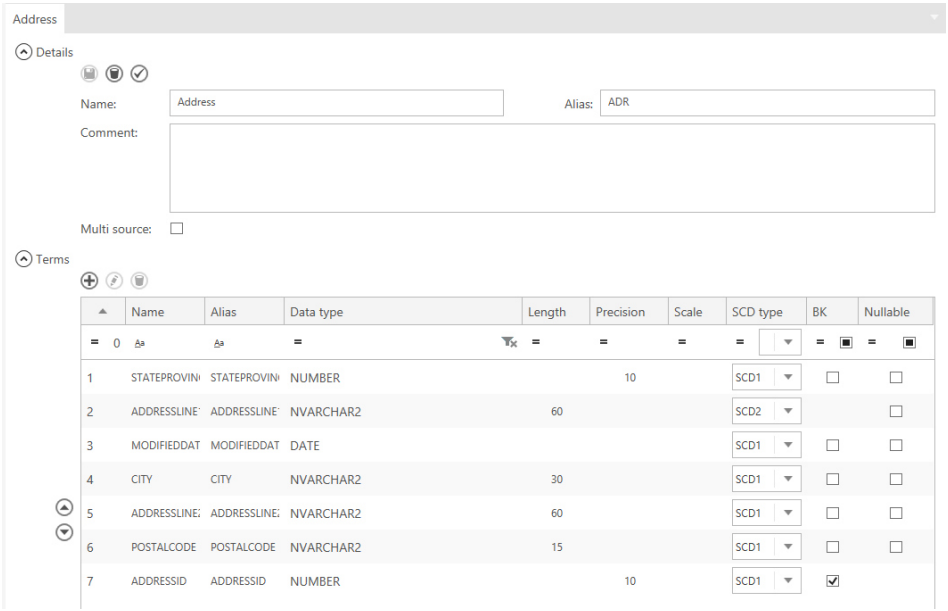
Kopf / Version Modellierung im Core



Praktisches Beispiel

- Neue Quelltabelle einbinden
- Anlegen als Entität
- Generieren & Deployen

Demo




```
<#@ include file="TemplateExtensions\TemplateExtensions.t4" #>
<#@ parameter type="Trivadis.biGenius.DAL.ExternalModel.SourceObject" name="Instance" #>

<#
    BuildTableHeader(Instance["STG_Table"], "yes");

    WriteLine("    {0,-1}{1,-50} {2,-20} {3}", " ", "DWH_LOAD_ID", "NUMBER", "NOT NULL");
    WriteLine("    {0,-1}{1,-50} {2,-20} {3}", ",", "DWH_JOB_ID", "NUMBER", "NOT NULL");
    WriteLine("    {0,-1}{1,-50} {2,-20} {3}", ",", "DWH_SRC_SYSTEM_ID", "VARCHAR2(50)", "");

    foreach(SourceField field in Instance.SourceFields)
    {
        WriteLine("    {0,-1}{1,-50} {2,-20} {3}", ",", field.Alias, field.DataType, "");
    }
#>
);

<#
    GrantSelect(Instance["STG_Table"],"DWH_CLEANSE");
#>
```

Vorteile der Generierung

- Schnelle Entwicklung
- Hohe Flexibilität bei neuen Anforderungen
- Kontinuität innerhalb der Architektur
- Vereinfachter Technologie-Wechsel

Fragen und Antworten...

Christoph Hisserich
Business Intelligence Consultant

christoph.hisserich@trivadis.com

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

trivadis
makes IT easier.