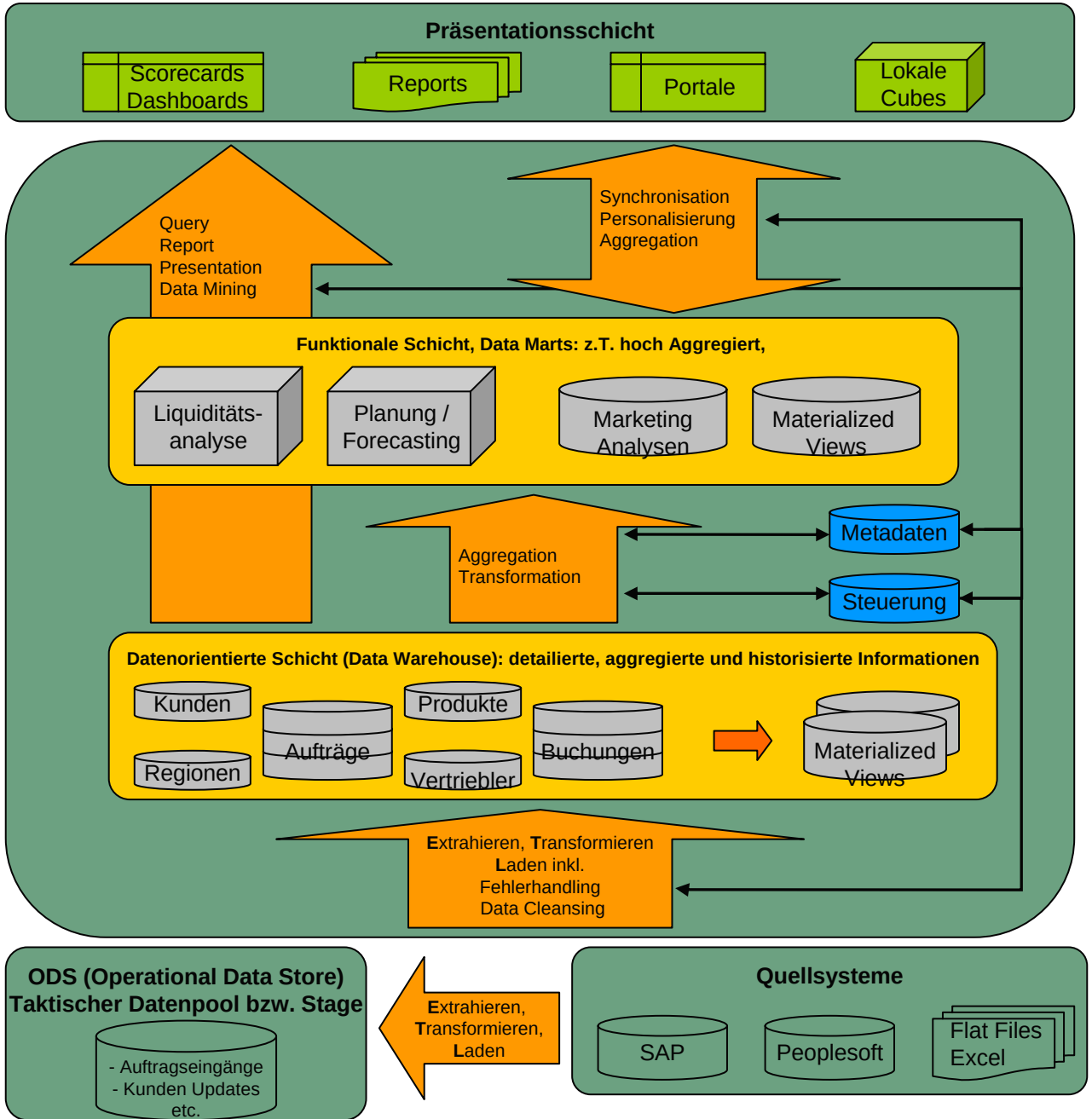
Generische Core Data Warehouse (DWH) - Layer

Dipl. Inf. Marc Werner
Business Intelligence & Data Warehouse Consultant
eMail: Marc-Werner@gmx.net
Mobile: +49 1721068289

Agenda

- Datawarehouse – Was kostet die Welt ?
- Komplexitätsreduzierung / Formalisierung
- Das Framework (Lösung)
- Demo

Data Warehouse: Individuell, komplex, teuer



Aufbau eines neuen DWH bzw. Anbindung neuer Quellsysteme – Ist Situation:

- Verschiedene DB-basierte OLTP Systeme anzubinden – möglichst zeitnah, um Datenverlust zu minimieren
- Noch keine klaren Analyse/Reporting – Anforderungen
- Viele Leute reden mit – Analyse der Quellsysteme, Lieferung neuer Anforderungen
- Kleine Budgets, dennoch Sicherstellung von Performance und Skalierbarkeit

Komplexitätsreduzierung → Kostenreduzierung und dennoch zukunftsfähiges, stabiles Design

- Welche Anforderungen an das DWH sind klar formulierbar ?:
  - Quell-Tabelle X,Y,Z anbinden und historisieren
  - ...
- Konzentration auf das sofort Umsetzbare / absolut Notwendige (Scope)
  - Aufbau einer zentralen Metadatenbasis von Beginn an
  - Datengetriebener Ansatz (von der Quellsystemen her)
  - Schaffung einer -für das/die zukünftige Reporting / Analyse- offenen Daten**BASIS (DWH Core Layer)**

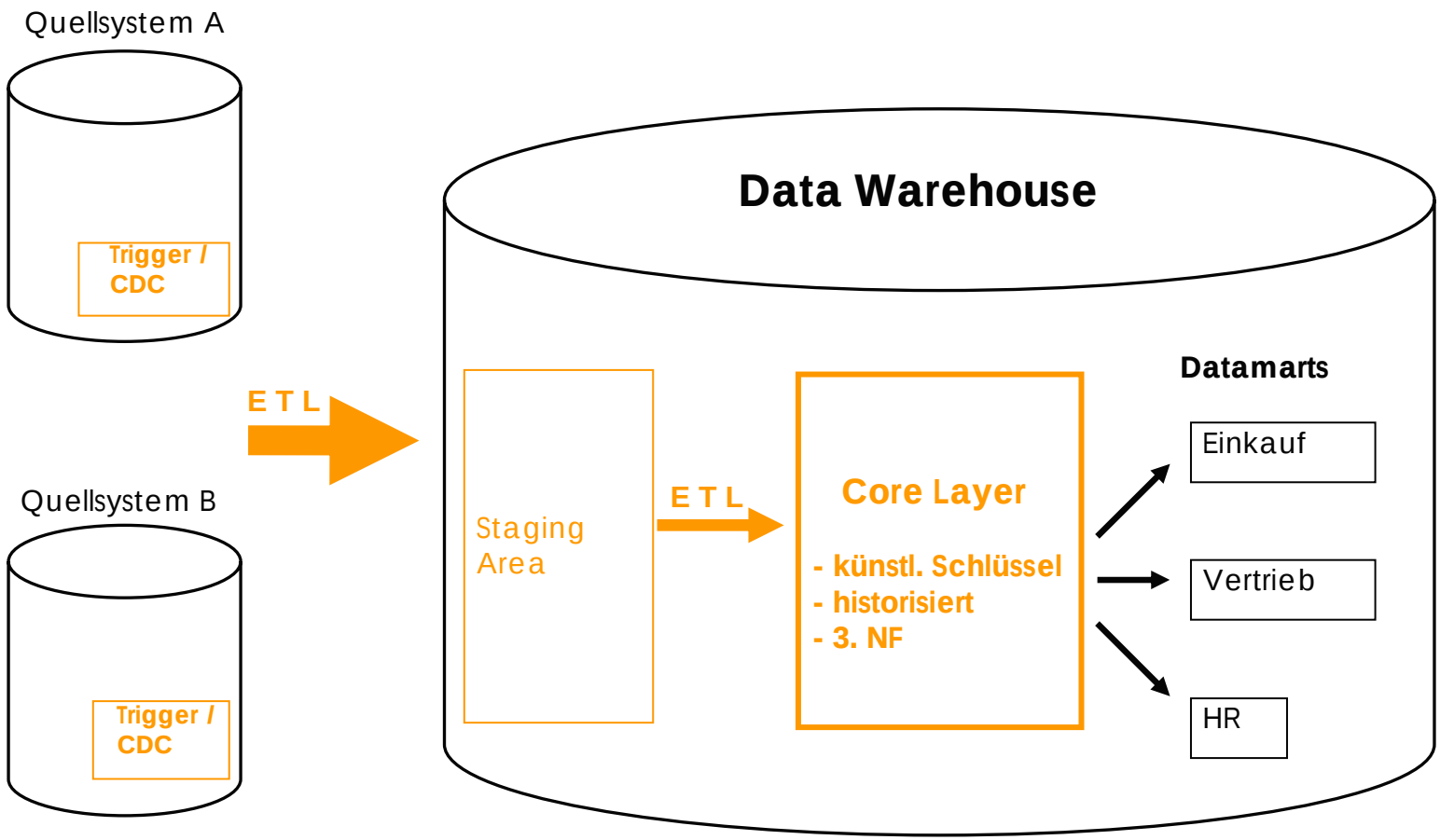
Vereinfachende „Designmaßnahmen“ und Annahmen

(Maxime: keine Einschränkungen für spätere Analyse)

- Alle Spalten der Quell-Tabellen werden 1:1 (Datentyp) ins DWH übernommen (wenn nicht muss Kunde Liste in DB ablegen)
- Es werden nie Spalten im DWH gelöscht (eigtl. DWH Definition)
- Quellsysteme halten Ihre Daten in RDBMS – Datenbanken (→ Zugriff via DB – Link) bzw. können Metadaten zu Feldern und Felddefinitionen liefern
- Für alle Quell-Tabellen kann ein PK / UK formuliert werden

Das Framwork (Scope)

(generierte Bereiche orange markiert)



...

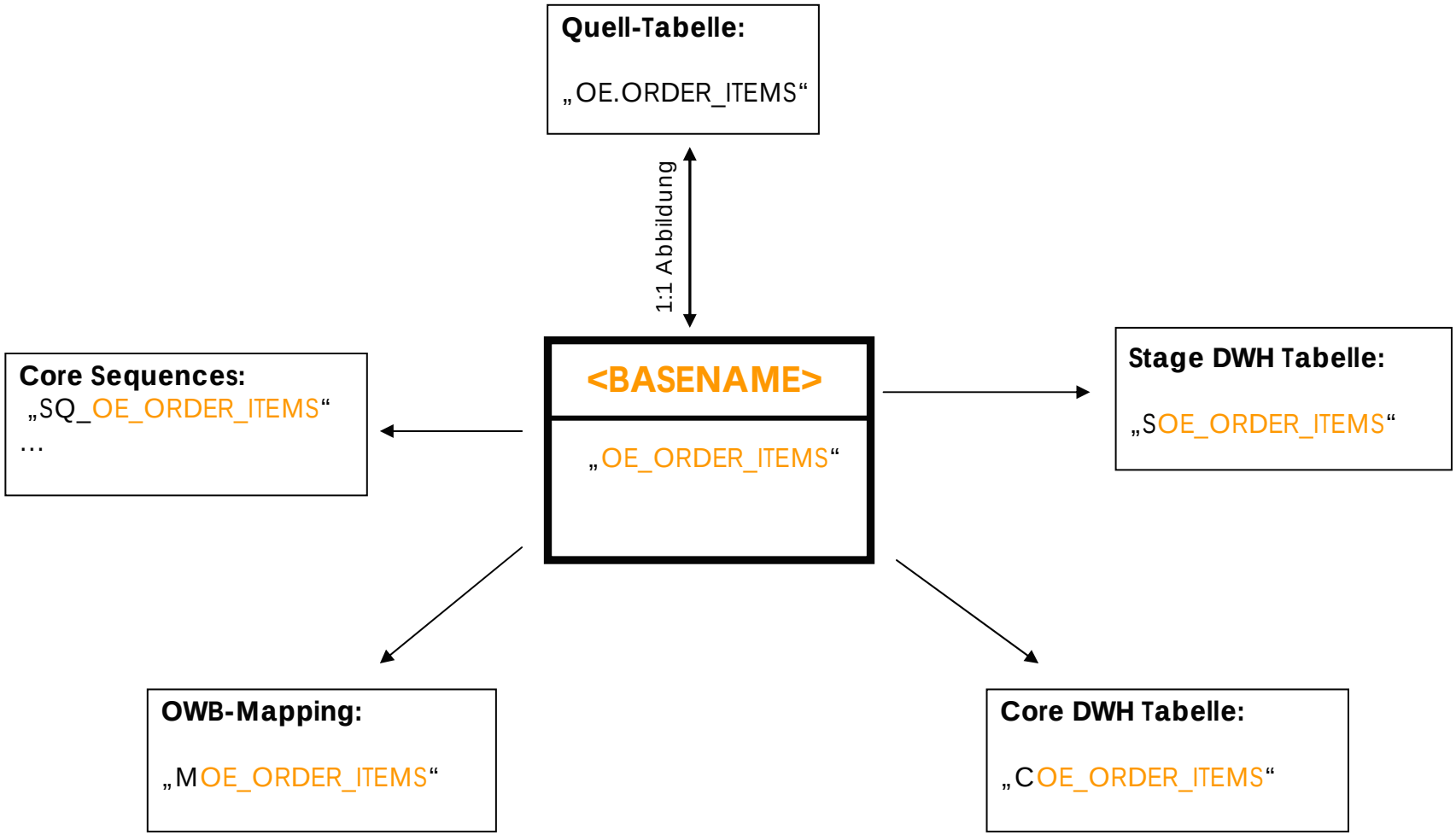
Ableitung aus den Quellsystemen je Tabelle, Anreicherung mit künstl. Schlüsseln, Historisierung

Quell-Tabelle: ORDER_ITEMS

Update am 20.07.2005

Primary Key							
ORDER_ID	LINE_ITEM_ID	PRODUCT_ID	QUANTITY	SID	OID	VALID_FROM	VALID_TO
1	1	1	100	1	1	01.05.2005	31.12.9999
2	2	10	42	2	2	01.05.2005	20.07.2005
2	*2*	*33*	*42*	3	2	21.07.2005	31.12.9999

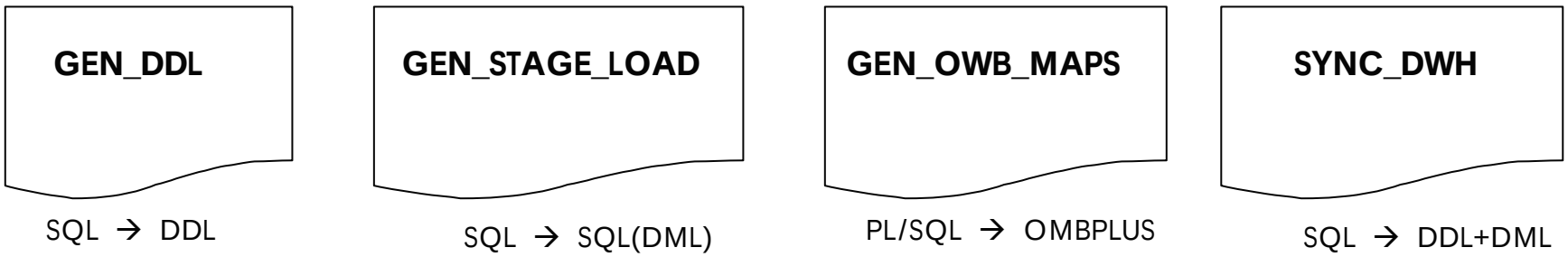
Schaffung eines global eindeutigen <BASENAME> , Ableitung aller DWH-Objektnamen davon



Funktionale Sicht auf das Framework

```
generate_Model_and_ETL (
                  quell_tabelle (basename)
                , delta_verfahren
                , historis_verfahren
                );
```

Generator Module:



Der Generierungsprozess

1. Quelldatenmodelle: Release X

2. Anpassung Metadaten (neue Tabellen, geänderte Lademodi, ...)

3. Generierung u. Deployment Zieldatenmodelle (Tabellen und Sequenzen) und CDC / Trigger Skripte
   a. Deployment in DWH DB
   b. Deployment (CDC, Trigger) in Quell-Datenbanken
   c. Ggfs. Merge von Werten hinzugekommener Spalten

4. (Re-)Import von Stage/Core Tables ins OWB Repository, Generierung des OMBPLUS Code und Deployment der Mappings (PL/SQL) ins Filesystem

5. Deployment der Mappings (PL/SQL) in DWH DB

Verwaltung der Codefragmente in 1 zentralen DB Tabelle

BASENAME	LOAD_ MECHAN	DELTA_ MECHAN	STA_CREATE	DWH_CREATE	SEQ_SID	SEQ_OID	TRIGG_OR_CDC	OMB_PLUS
AX_MARKUPAUTOLINE	SCD1D	CDC	CREATE TABLE STAGE.SAX_ MARKUPAUT OLINE (TABLETABLEI D NUMBER(10), TABLERECID NUMBER(20), LINENUM NUMBER(32,1 6), MARKUPCOD E	CREATE TABLE DWH.CAX_MA RKUPAUTOLIN E (DCLG_SID NUMBER(20) PRIMARY KEY NOT NULL, DCLG_LOADID NUMBER(10) NOT NULL, DCLG_CREAT ED DATE NOT NULL,	CREATE SEQUENCE DWH.SCAX _MARKUPA UTOLINE_S START WITH 1 MAXVALUE 9999999999 9999999999 9999999 MINVALUE 1 NOCYCLE CACHE 100000	CREATE SEQUENCE DWH.SCAX _MARKUPA UTOLINE_O START WITH 1 MAXVALUE 9999999999 9999999999 9999999 MINVALUE 1 NOCYCLE CACHE 100000	BEGIN DBMS_CDC_PUB LISH.CREATE_C HANGE_TABLE(owner => 'AX_CHANGES', change_table_na me => 'AX_MARKUPAUT OLINE_CT', change_set_name	set OMBCONTINUE_ON_ERROR TRUE OMBDROP MAPPING 'MAX_MARKUPAUTOLINE' set OMBCONTINUE_ON_ERROR FALSE OMBCREATE MAPPING 'MAX_MARKUPAUTOLINE' ADD TABLE OPERATOR 'SAX_MARKUPAUTOLINE' BOUND TO TABLE '../STAGING/SAX_MARKUPAUTOLIN E'\
AX_MARKUPTBL	SCD3	CDC	CREATE TABLE STAGE.SAX_ MARKUPTBL (TXT VARCHAR2(12 0), CUSTPOSTIN G	CREATE TABLE DWH.CAX_MA RKUPTBL (DCLG_SID NUMBER(20) PRIMARY KEY NOT NULL, DCLG_OID	CREATE SEQUENCE DWH.SCAX _MARKUPT BL_S START WITH 1 MAXVALUE 9999999999	CREATE SEQUENCE DWH.SCAX _MARKUPT BL_O START WITH 1 MAXVALUE 9999999999	BEGIN DBMS_CDC_PUB LISH.CREATE_C HANGE_TABLE(owner => 'AX_CHANGES',	set OMBCONTINUE_ON_ERROR TRUE OMBDROP MAPPING 'MAX_MARKUPTBL' set OMBCONTINUE_ON_ERROR FALSE OMBCREATE MAPPING
LI_EWB_KONTO	SCD3	FULL	CREATE TABLE STAGE.STA_L I_EWB_KONT	CREATE TABLE DWH.CORE_LI _EWB_KONTO	CREATE SEQUENCE DWH.SQ_LI _EWB_KON	CREATE SEQUENCE DWH.SQ_LI _EWB_KON	BEGIN DBMS_CDC_PUB LISH.CREATE_C	set OMBCONTINUE_ON_ERROR TRUE OMBDROP MAPPING
AX_PRICEDISCGROUP	SCD2	CDC	CREATE TABLE STAGE.SAX_ PRICEDISCGR	CREATE TABLE DWH.CAX_PRI CEDISCGROU	CREATE SEQUENCE DWH.SCAX _PRICEDIS	CREATE SEQUENCE DWH.SCAX _PRICEDIS	BEGIN DBMS_CDC_PUB LISH.CREATE_C	set OMBCONTINUE_ON_ERROR TRUE OMBDROP MAPPING
LO_LAX_ARTK	SCD1	TS	CREATE TABLE STAGE.SLO_L AX_ARTK (ARTKID NUMBER(12), ARTKBZNGID	CREATE TABLE DWH.CLO_LAX _ARTK (DCLG_SID NUMBER(20) PRIMARY KEY	CREATE SEQUENCE DWH.SCLO _LAX_ARTK _S START WITH 1 MAXVALUE	CREATE SEQUENCE DWH.SCLO _LAX_ARTK _O START WITH 1 MAXVALUE	CREATE OR REPLACE TRIGGER "OLTP_XY"."LO_L AX_ARTK_TTS" BEFORE INSERT OR UPDATE ON	set OMBCONTINUE_ON_ERROR TRUE OMBDROP MAPPING 'MLO_LAX_ARTK' set OMBCONTINUE_ON_ERROR FALSE

Demo

- 1:1 Tabellenanbindung (eigtl. 3.NF.: 1 Quelltabelle → 2 DWH Core Tabellen)
- 1 Quell - DB, Anbindung von 4 Tabellen aus Oracle DB Demo Schemata HR und Order Entry
- Präsentation der Ladeverfahren: Change Data Capture, Trigger+TS (Delta-Load), Trigger (Delta-Load) und Full Refresh
- Inbetriebnahme (… initial Load), Delta Load
- Simulation: neues Release wegen Datenmodelländerungen auf Quellsystem im laufenden Betrieb

Fazit

- Eine Data Warehouse Core Layer kann nach minimaler Analyse und Auswertung/Anreicherung von Metadaten Live ohne Implementationsaufwand gehen
- Wiederverwendung existierender / Integration zusätzlicher Metadaten und Repositories – ein Muss für jedes DWH
- Kontrollierte Komplexität durch Generatoren: Neue ETL - Funktionalität schnell „deployed“ durch Änderungen an zentraler Stelle

Fragen ?

Marc-Werner@gmx.net
01721068289

Vielen Dank für Ihre Aufmerksamkeit

BACKUP

Funktionalität

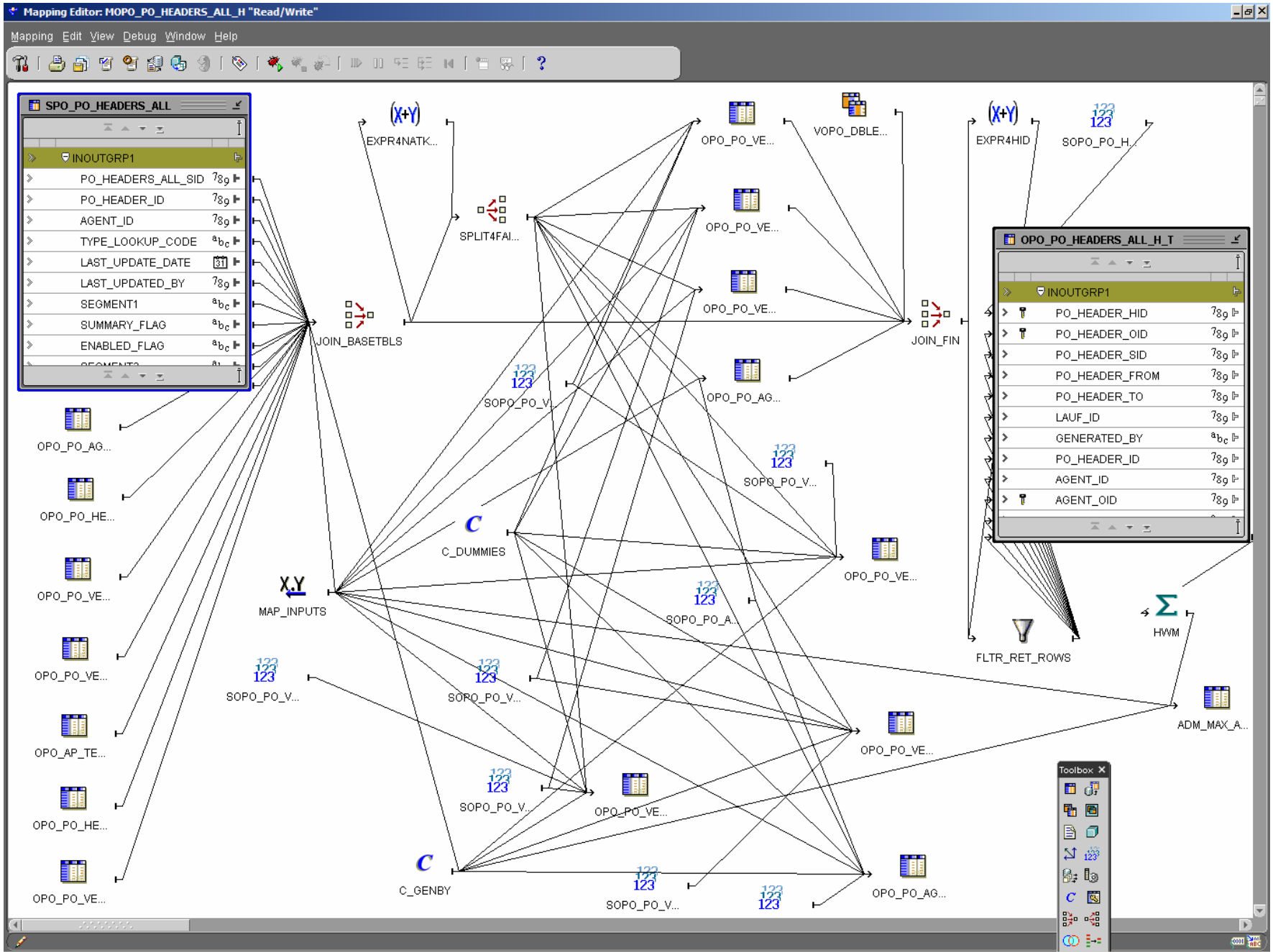
- Generierung von:
  - Change Data Capture, Trigger für Delta Load
  - Zieldatenmodellen (Staging Area und ODS)
  - Mapping Code:
    - SOURCE → Staging Area (wahlweise SQL o. OWB)
    - Staging Area → ODS

- Generierte ETL Mechanismen:
  - Surrogate Key - Erzeugung
  - Delta Abzug der Daten aus Quellsystem
  - SCD I , SCD II, Deletion - Tracking
  - Einfügen von Dummy-Sätzen bei fehlgeschlagenem Lookup

- Synchronisation bei Datenmodelländerung der Quellsysteme

- Protokollierung von Generierungskonflikten –und Fehlern

Komplexität



Warum metadatengetrieben Generieren ?

- Keine Kosten für Entwicklungstätigkeiten
- Gegen Datenmodelländerungen stabilerer ETL , da metadatengetrieben
- Einmaliger Änderungsaufwand Datenmodell oder ETL - Logik – Anpassungen, die für den gesamte Core Layer greifen sollen (Bsp.: Gültigkeitszeitraum bei SCD II)
- Quell - Datenmodell generiert Code → vollständiges System → keine Fehler, da keine Codierung

Voraussetzungen / Abgrenzung

- Entwicklungsphase eines DWH
- Eindeutige, über Metadaten formalisierbare, ETL Prozesse (SCD I , SCD II)
- Vorranging für Staging Area und ODS
- Zugriff auf Remote & Local Data Dictionary

Anreicherung der Metadaten

globale Konfiguration:

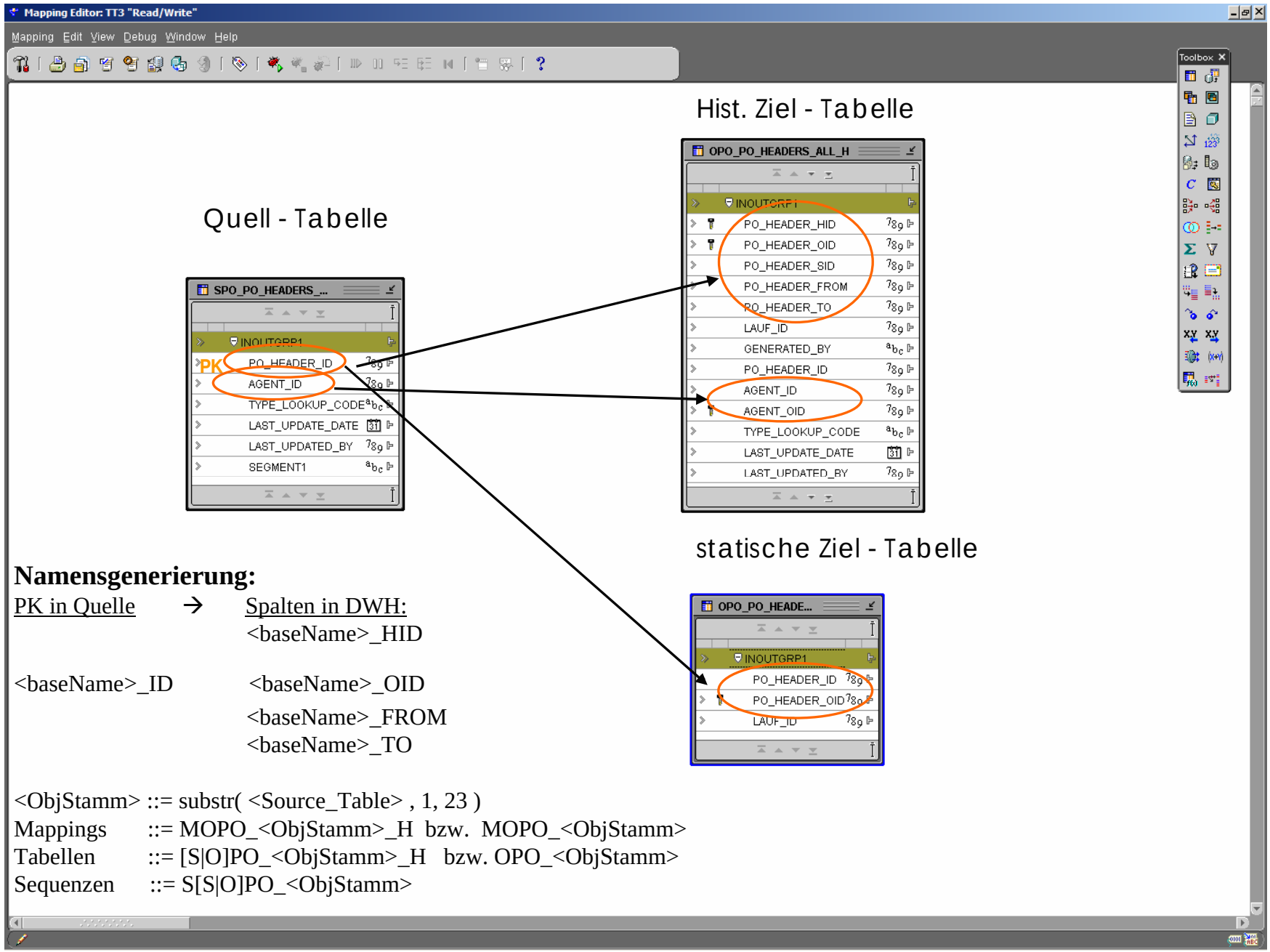
DWH_ID	SRC_SYSTEM	OPTION	DESC	DEFAULT_VALUE
HOCHTIEF_II	APPS	**TS_4DELTA**	Zeitstempel – Feld für Delta Abzug	**LAST_UPDATE_DATE**
		TS_4_HIST	Zeitstempel – Feld für Gültigkeitszeitraum bei SCD II	**LAST_UPDATE_DATE**
		LOAD_TYPE	Verfahren zur Ablage der Daten	**SCD II**
		ON_FAILED_LKUP	Aktion bei fehlgeschlagenem „Lookup for Surrogate Key“	**INS_INTO_LKUP_TBL**

Überschreiben der globalen Konfiguration auf Tabellen –bzw. Spaltenebene:

SRC_Table	*TABLE_ID*	*LOAD_TYPE*	*TS_4_DELTA*	*TS_4_HIST*
PO_HEADERS_ALL		SCD I		
PO_LINES_ALL				CLOSED_DATE
HR_ALL_ORGANIZATION_UNITS_TL	HR_ALL_ORG_UNIT_TL	SIMPLE_INS		

SRC_Table	*SRC_Column*	*TRGT_Column*	*SURROG_4_FK*	*ON_FAILED_LKUP*	HIST_MODE	Exclude
PO_HEADERS_ALL	AGENT_ID			REJECT		
	SEGMENT1	BELEGNR			STATIC	
	SEGMENT2					Y
PO_LINES_ALL	LINE_TYPE_ID		N			
	NOT_TO_EXCEED_PRICE				TRACK	

Transition in Zieldatenmodell



Namensgenerierung:

PK in Quelle	→	Spalten in DWH:
		<baseName>_HID
<baseName>_ID		<baseName>_OID
		<baseName>_FROM
		<baseName>_TO

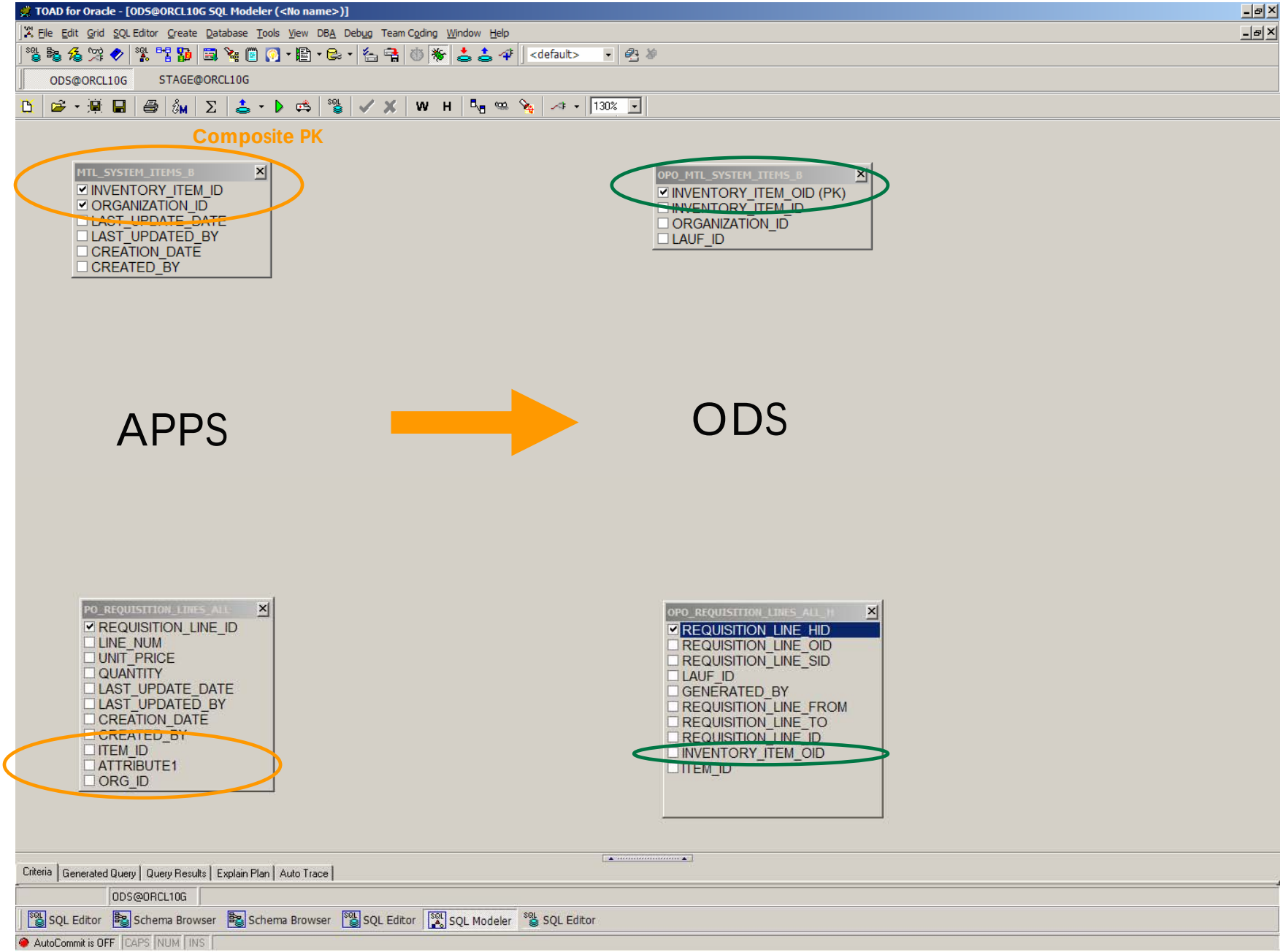
<ObjStamm> ::= substr(<Source_Table> , 1, 23)
Mappings ::= MOPO_<ObjStamm>_H bzw. MOPO_<ObjStamm>
Tabellen ::= [S|O]PO_<ObjStamm>_H bzw. OPO_<ObjStamm>
Sequenzen ::= S[S|O]PO_<ObjStamm>

Die FK Transition (1 / 2)

3 – stufig:

1. FK Definition aus Source DB
2. Namensgleichheit + PK Definition
3. Mapping Tabelle (Fallback)

Die FK Transition (2/2)



FK Mapping Tabelle (Fallback)

SURROG_KEY	*SRC_Table*	*SRC_EXPR*	*TRGT_Table*	*TRGT_Column*
INVENTORY_ITEM_OID	PO_REQUISITION_LINES_ALL	ITEM_ID	MTL_SYSTEM_ITEMS_B	INVENTORY_ITEM_ID
INVENTORY_ITEM_OID	PO_REQUISITION_LINES_ALL	Substr(ATTRIBUTE1, 5)	MTL_SYSTEM_ITEMS_B	ORGANIZATION_ID

Funktion:

- Definition von Relationships, die im DWH per Surrogate Key abgebildet werden sollen, aber nicht im Quellsystem Repository entdeckt werden könnnen

- Auflösung von Mehrdeutigkeiten bei Namen + PK – Match

Ausblick / Erweiterungen

- Einführung einer View – Schicht →
  - Abfangen von Generierungskonflikten
  - Laden von Dimensionen

- Erweiterung des Frameworks um „Ziel - Code Option“(OWB , PL/SQL) auch für ODS Schicht

- Generierung von Teilen des ETL‘s in die Data Marts möglich. Bsp.: „History – Merge“

Datenmodell - Ableitung aus den Quellsystemen, Anreicherung mit künstl. Schlüsseln, Historisierung

Tabelle: ORDER_ITEMS

Feld	Datentyp	
ORDER_ID	Number(10)	
LINE_ITEM_ID	Number(20)	
PRODUCT_ID	Number(5)	
QUANTITY	NUMBER(5)	