

noETL yesSQL in der Praxis

Data Analytics 2018

Alec Shalashou
20.03.2018



noETL = Ausführung von Transformationen dort, wo die Daten gespeichert sind

yesSQL = Entwicklung der Transformationslogik mit SQL



- Aktuelle Trends
- Motivation für noETL
- Motivation für yesSQL
- SQL vs Point-and-Click
- Kontext DWH
- Wichtige Aspekte
- Demo



Agile In-Memory
Prototyping
Cloud NewSQL
Web Services
BigData
NoSQL
ELT



- Entwicklung wird agiler: Kürzere Entwicklungszyklen, Prototyping
- Datenbanken gehen In Memory und in die Cloud
- Datenmengen wachsen und ELT Verfahren sind oft gewünscht
- Datenbanken bieten viele Integrationskomponenten an: DB-Link, External Tables, Flat File Loaders, CSV, XML und JSON-Parser
- Neue SQL-basierte Technologien wie SQL-On-Hadoop, NewSQL



- Datenbank bietet sich als Transformationsplattform an
- Datenverarbeitung findet dort statt, wo die Daten gespeichert sind (native)
- Bessere Ausnutzung der DB Features (auch zukünftigen)
- Transformationen sind in der Zielumgebung direkt ausführbar
- Keine neue Transformationsprache



- und als Transformationssprache bietet sich SQL an
- Ermöglicht natives Transformieren in der Datenbank
- Universelle Schnittstellensprache (Datenbanken, SQL-on-Hadoop)
- Schnelle Entwicklung, da die Analyse, der Prototyp und die Implementierung in der gleichen Sprache erfolgen
- Einfachere Fehleranalyse, da der Transformations-Code direkt ausführbar ist



- Unabhängigkeit vom Tool, da der Inhalt der Transformationen durch SQL portabel ist
- Größeres Angebot an Mitarbeitern mit SQL Kenntnissen
- Schnellere Einarbeitung, da weniger spezial-Kenntnissen benötigt wird
- SQL ist (in der Regel) übersichtlicher als Point-and-Click Mapping und deshalb besser für komplexere Logik geeignet



yesSQL vs Point-and-Click

The screenshot displays the 'Edit Transformations' window in DataSquill. The 'Select transformation' is 'EXP_VORGAENGER_IDENTIFIKATION' of type 'Expression'. A table lists 17 ports with their names, datatypes, and precision/scale values. An 'Expression Editor' window is open, showing a complex SQL-like formula for the 'V_VORHER_AUFTRAG_NO' port. The formula uses the IIF function to return either the previous order number or the current one based on several conditions. Below the editors, a data flow diagram shows the transformation being applied to a data stream, with arrows indicating the flow between various components like 'SRT_PARTITION', 'EXP_VORGAENGER_ID...', 'FIL_ERSTE_EBENE', and 'AGG_MIN_VORGAENGER'.

Port Name	Datatype	Prec	Scale	I	O	V	Expression
1 CUST_NO	decimal	10	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	CUST_NO
2 V_VORHER_AUFTRAG_NO	integer	10	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	IIF(...
3 AUFTRAG_NO	decimal	10	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	AUFTRAG_NO
4 AUFTRAG_NO	decimal	10	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	AUFTRAG_NO
5 ATYP	string	5	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	ATYP
6 ASTATUS	decimal	10	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	ASTATUS
7 VALID_FROM	date/time	29	9	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	VALID_FROM
8 REFERENZIERT_AUFTRAG_NO	string	10	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	REFERENZIERT_AUFTRAG_NO
9 VALID_FROM_VORGAENGER	date/time	29	9	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	VALID_FROM_VORGAENGER
10 OLD_SERVICE_NO	string	20	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	OLD_SERVICE_NO
11 HAUPT_AUFTRAG_NO	decimal	10	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	HAUPT_AUFTRAG_NO
12 HIST_STATUS	string	3	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	HIST_STATUS
13 V_CUST_NO	decimal	10	0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	CUST_NO
14 V_AUFTRAG_NO	decimal	10	0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	AUFTRAG_NO
15 V_AUFTRAG_NO	decimal	10	0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	AUFTRAG_NO
16 V_ATYP	string	5	0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ATYP
17 V_ASTATUS	decimal	10	0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ASTATUS

```
IIF(
  (ISNULL(REFERENZIERT_AUFTRAG_NO) OR OLD_SERVICE_NO
  = TO_CHAR(AUFTRAG_NO))
  AND V_CUST_NO = CUST_NO
  AND (ATYP = 'NEU' OR ATYP = 'UNG')
  AND V_AUFTRAG_NO = AUFTRAG_NO + 1
  AND V_ATYP = 'KUEND',
  V_AUFTRAG_NO,
  TO_INTEGER(REFERENZIERT_AUFTRAG_NO)
)
```



```
SELECT auftrag__no
      , MIN(ref_auftrag__no) vorher_auftrag__no
      , MIN(valid_from) gueltig_von
FROM (SELECT auf.auftrag__no
      , auf.auftrag__no
      , auf.valid_from
      , auf.hist_status
      , TRIM(auf.atyp) atyp
      , auf.astatus
      , auf.cust_no
      , auf.old_service__no
      , ref_auf.min_valid_from
      , ref_auf.max_valid_to
      , ref_auf.ref_atyp
      , CASE
          WHEN NVL(ref_auf.ref_auftrag__no, auf.auftrag__no) = auf.auftrag__no
            AND TRIM(auf.atyp) IN ('NEU', 'UNG')
            AND LEAD(TRIM(auf.atyp)) OVER (PARTITION BY auf.cust_no ORDER BY auf.auftrag__no) = 'KUEND'
            AND LEAD(auf.auftrag__no) OVER (PARTITION BY auf.cust_no ORDER BY auf.auftrag__no) = auf.auftrag__no + 1
          THEN LEAD(auf.auftrag__no) OVER (PARTITION BY auf.cust_no ORDER BY auf.auftrag__no)
          ELSE NVL(ref_auf.ref_auftrag__no, auf.auftrag__no)
        END ref_auftrag__no
      FROM vv_auftrag auf
      , tmp_ref_auftrag ref_auf
      WHERE auf.old_service__no = TO_CHAR(ref_auf.ref_auftrag__no(+))
WHERE ref_auftrag__no != auftrag__no
GROUP BY auftrag__no;
```



- Die meisten DWHs halten sich in einer oder wenigen DBs auf
- 3 NF Layer
 - ▶ komplexe Transformationen
 - ▶ größte Entwicklungsaufwand
 - ▶ wiederverwendbare Komponenten z.B. Historisierung
- Data Mart
 - ▶ viele einfache Transformationen
 - ▶ wiederkehrende Operationen
- Staging
 - ▶ sehr einfache Transformationen
 - ▶ Toolgestützte Anbindung von Quellen



- Modularisierung
 - ▶ Plain SQL für die fachliche Logik verwenden
 - ▶ Wiederkehrende Operationen modularisieren (z.B. mit Scripting Sprache oder PL/SQL)
 - ▶ Kein Templating!
- Ablaufsteuerung
- Scheduling
- Externe Quellen / Ziele anbinden





- Datenbank ist ein mächtiges Transformationsplattform
- Data Warehousing eignet sich besonders gut für In-Database Transformationen
- SQL ist eine flexible, standardisierte und zukunftssichere Sprache für Datenverarbeitung
- Einstieg in die ‚noETL yesSQL‘ Welt auch schrittweise ohne ‚Big Bang‘ möglich



Vielen Dank!

Alec Shalashou

Consultant Data Warehousing

alec@datasqill.de

datasqill.de

