noETL yesSQL in der Praxis

Data Analytics 2018

Alec Shalashou

20.03.2018

Buzzwords Explained

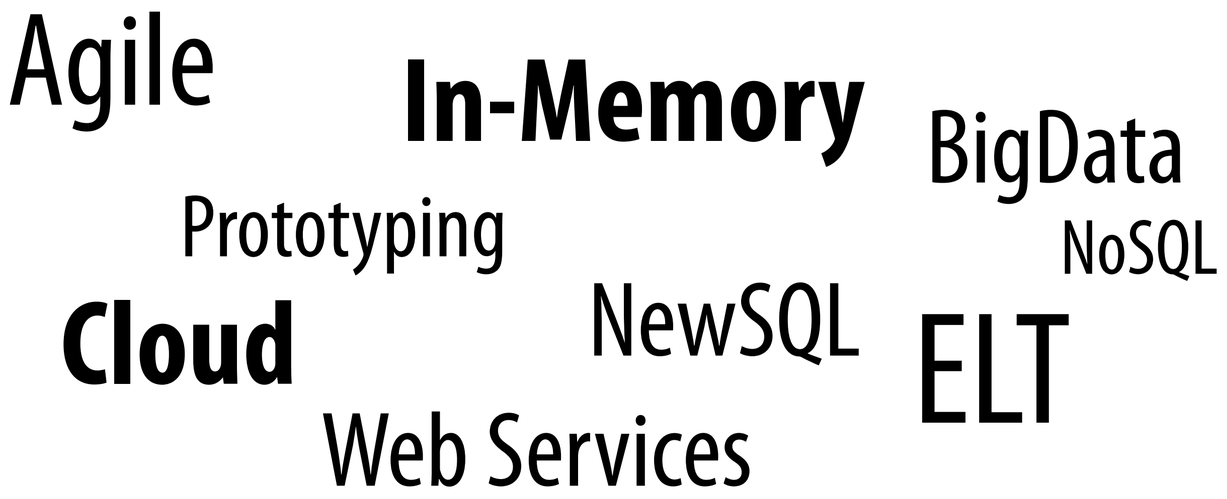
noETL = Ausführung von Transformationen dort, wo die Daten gespeichert sind

yesSQL = Entwicklung der Transformationslogik mit SQL

Inhalt

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

Aktuelle Trends



Aktuelle Trends

- 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

Warum noETL?

- Datenbank bietet sich als Transformationsplatform 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 Transformationssprache

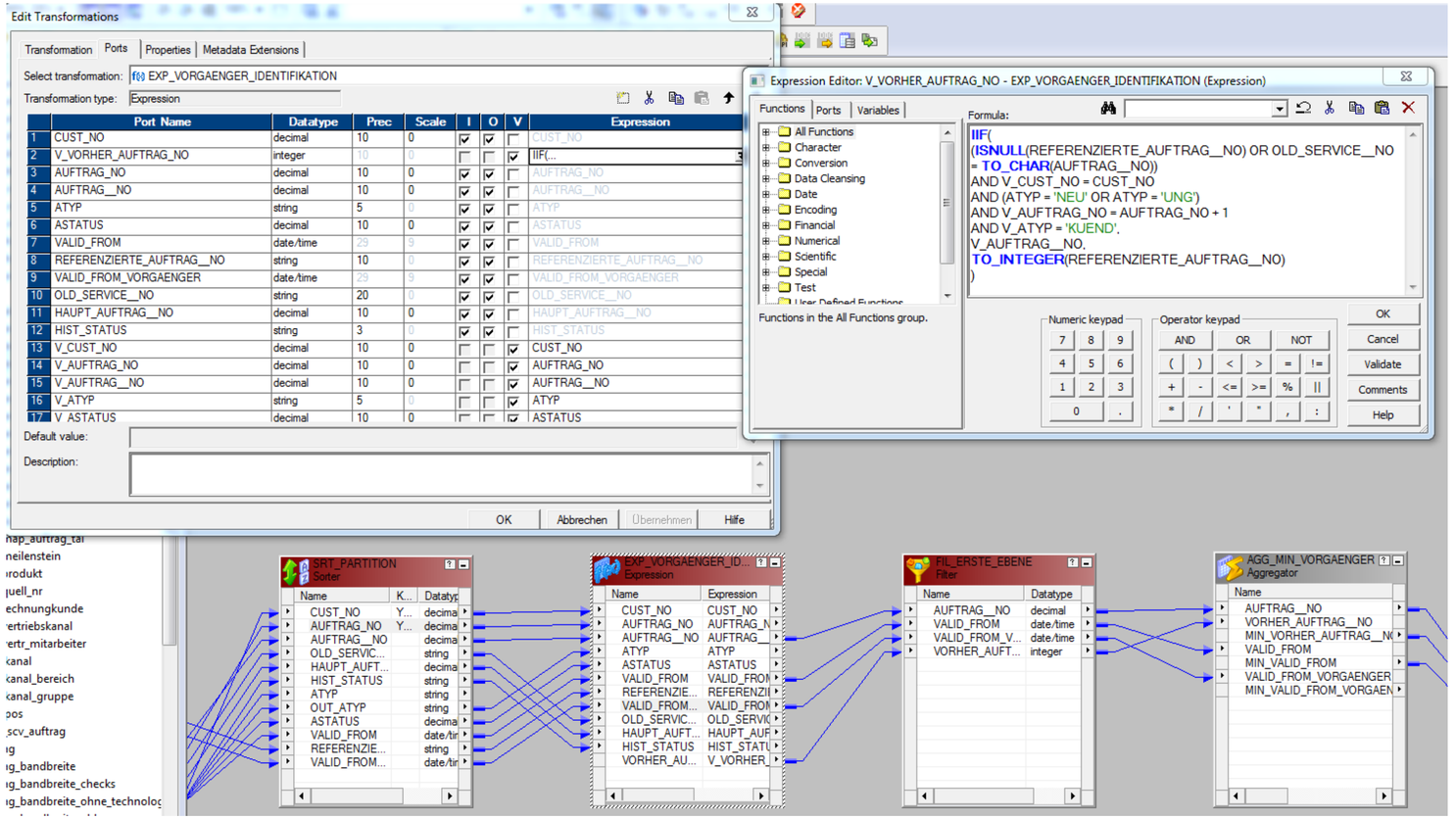
Warum yesSQL?

- und als Transformationsspache 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

yesSQL vs Point-and-Click Transformationsentwicklung

- 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



yesSQL vs Point-and-Click


```
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;
```

Kontext Data Warehouse

- 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

Wichtige Aspekte

- 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

Demo

Fazit

- Datenbank ist ein mächstiges Transformationsplatform
- Data Warehousing eignet sich besonders gut für In-Database Transformationen
- SQL ist eine flexible, standartisierte 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