„noETL, yesSQL“ – warum ELT und SQL die optimale Wahl für ein modernes Data Warehouse sind

Alec Shalashou, datasqill

Dieser Artikel beschäftigt sich mit der Analyse unter Verwendung des ELT-Verfahrens (extract, load, transform) und des Einsatzes von SQL als Transformationssprache im Kontext moderner DWHs. Der Artikel stellt das ELT-Verfahren dem gängigen ETL-Verfahren (extract, transform, load) gegenüber und vergleicht die Transformationsentwicklung mit SQL mit der grafischen Transformationsentwicklung. Er stellt Beispiele für ELT-Transformationslandschaften vor und macht Vorschläge für die sinnvolle Umsetzung des ELT-Verfahrens sowie der Entwicklung mit SQL in einem Data Warehouse (DWH).

Die Welt des modernen Data Warehousing ist dynamisch wie noch nie zuvor. Die Datenmengen steigen und die Komplexität der analytischen Fragestellungen wächst kontinuierlich weiter an. Selbst kleine und mittelständische Unternehmen stehen inzwischen vor anspruchsvollen analytischen Anforderungen.

Viele Unternehmen setzen heute auf agile Entwicklungsmethoden, um die schneller wechselnden und wachsenden Kunden-Anforderungen bedienen zu können. Dies verkürzt Entwicklungs- und Release-Zyklen, verringert Risiken durch Prototyping der Lösungsansätze und stellt Ergebnisse wie Reports und Analysen für Endanwender schneller bereit. Das Ganze wird unterstützt durch die kontinuierliche Verbesserung und Spezialisierung der verfügbaren Werkzeuge für die verschiedenen Einsatzgebiete in DWH-Entwicklungsprojekten.

In solch einem dynamischen und datenintensiven Umfeld sieht sich ein Entwicklungsteam häufig mit der Frage konfrontiert, mit welchen Ansätzen und Tools sich der Datenaufbereitungs-Prozess in einem DWH am effizientesten gestalten lässt. Eine mögliche Antwort sind die Konzepte, die sich unter dem von uns vorgeschlagenen Begriff „noETL, yesSQL“ verbergen:

- *noETL*
  Ausführung von Datentransformationen dort, wo die Daten gespeichert sind
- *yesSQL*
  Programmierung der Transformationslogik in SQL

noETL

Mit dem Aufkommen der Data-Warehouse-Lösungen kommen ETL-Prozesse für ihre Befüllung und Bewirtschaftung zum Einsatz. Dabei überträgt ein Integrationsserver die Daten aus den Quellsystemen in die Datenbank des DWH. Da der Datenspeicher begrenzt und teuer ist, werden die Daten gleich bereinigt, transformiert und in geeigneter Form gespeichert *(siehe Abbildung 1)*.

Mit der Zeit wuchsen die Datenbestände in Data-Warehouse-Systemen immer weiter an. Zur logischen Strukturierung der Daten entstanden die heute verbreiteten Schichten für Staging, 3NF-Layer und Data Marts. Die Entwicklung und Ausführung der Transformationen erfolgt nach wie vor mit den etablierten ETL-Werkzeugen. Dabei kommt folgendes Verfahren zum Einsatz *(siehe Abbildung 2)*:

- Die Daten werden aus den Quellsystemen extrahiert („extract“)
- Die Daten werden auf dem Application-Server des ETL-Tools transformiert („transform“)
- Die Daten werden in das Zielsystem geschrieben („load“)

Bei dem beschriebenen Verfahren ist die DWH-Datenbank sowohl Quell- als auch Ziel-System für die Mehrzahl der Transformationen. Gerade bei einer mehrstufigen Verarbeitungslogik kann das zum wiederholten Lesen von Daten aus dem DWH in den Application-Server führen, die in vorherigen Schritten von dort in die Datenbank geschrieben wurden.

Viele ETL-Tools bieten eine grafische Transformationssprache, die allerdings proprietär ist und oft mit weiteren proprietären Programmiersprachen kombiniert wird. Der ausgeführte Transformationscode ist für den Entwickler nicht immer transparent und nachvollziehbar. Einige Tools generieren Code für Standard-Programmiersprachen wie zum Beispiel Java. Um die übersetzte Transformationslogik nachzuvollziehen und im Fehlerfalle analysieren zu können, müssen viele Datenbank-Entwickler daher zunächst eine neue Technologie verstehen und sicher beherrschen lernen.

Wie bewährt sich die Datentransformation mit ETL unter modernen Aspekten? Zunächst einmal arbeiten viele ETL-Tools zeilenorientiert, was im Zusammenspiel mit dem wiederholten Lesen und Schreiben von Daten zu entsprechenden Einbußen bei der Performance führt und sich spätestens bei der Verarbeitung von Massendaten als unüberwindbarer Hemmschuh erweist. Zweitens ist der Lösungsansatz, bei dem die Daten aus dem Datenspeicher ausgelesen werden, oft schwerfällig oder ungeeignet für neue Technologien wie Cloud-, In-Memory-, MPP-Datenbanken oder Big Data. Zum Dritten werden die vorhandenen Ressourcen und das Potenzial des Datenbank-Servers nicht optimal genutzt. Viertens entsteht durch die proprietäre Transformationsumgebung und aufgrund fehlender Standardisierungen ein starkes Vendor-Lock-in zu oft hochpreisigen ETL-Produkten. Die Entwickler müssen neben SQL auch die grafische Sprache des ETL-Tools beherrschen, was den Entwicklungsprozess ebenfalls verteuert und die Suche nach geeigneten Mitarbeitern erschwert.

Als Alternative zu ETL bietet sich ELT als Datenverarbeitungsverfahren für Data Warehousing an. Dabei werden die Daten aus den Quellsystemen extrahiert und ohne Transformation ins Data Warehouse geladen. Dort finden alle weiteren Transformationsschritte im Datenspeicher statt *(siehe Abbildung 3)*.

Das Performance-Verhalten bei ELT ist für Massendaten besser ausgelegt, da die komplexe Datenverarbeitung dort stattfindet, wo die Daten gespeichert sind. Es vermeidet Netzwerk-Verkehr und die bereits beschriebenen Schreib-/Lese-Vorgänge bei Transformationen – was gerade für Cloud-Lösungen, Massendaten-Verarbeitung und Big-Data-Lösungen für entsprechende Performance sorgt und ideale Voraussetzungen zur Skalierung mit sich bringt. Auch bei Architekturen mit In-Memory- und MPP-Datenbanken nutzt ELT die Features und Optionen der Datenspeicher optimaler aus, was zugleich zum Schutz der getätigten Investitionen in die Datenbank-Technologie beiträgt. Eine für Datenspeicher native Transformationssprache (wie SQL bei Datenbanken) erlaubt die einfachere Entwicklung und Fehler-Analyse, da der Quellcode der Transformationen direkt in der Zielumgebung ausführbar ist. Das minimiert zudem das Vendor-Lock-in, da der Transformationscode standardisiert und portabel ist.

Moderne Datenbanken stellen gute ELT-Plattformen dar, sie sind ausgelegt für die Verarbeitung von Massendaten und bieten Algorithmen zur Optimierung der Aus-

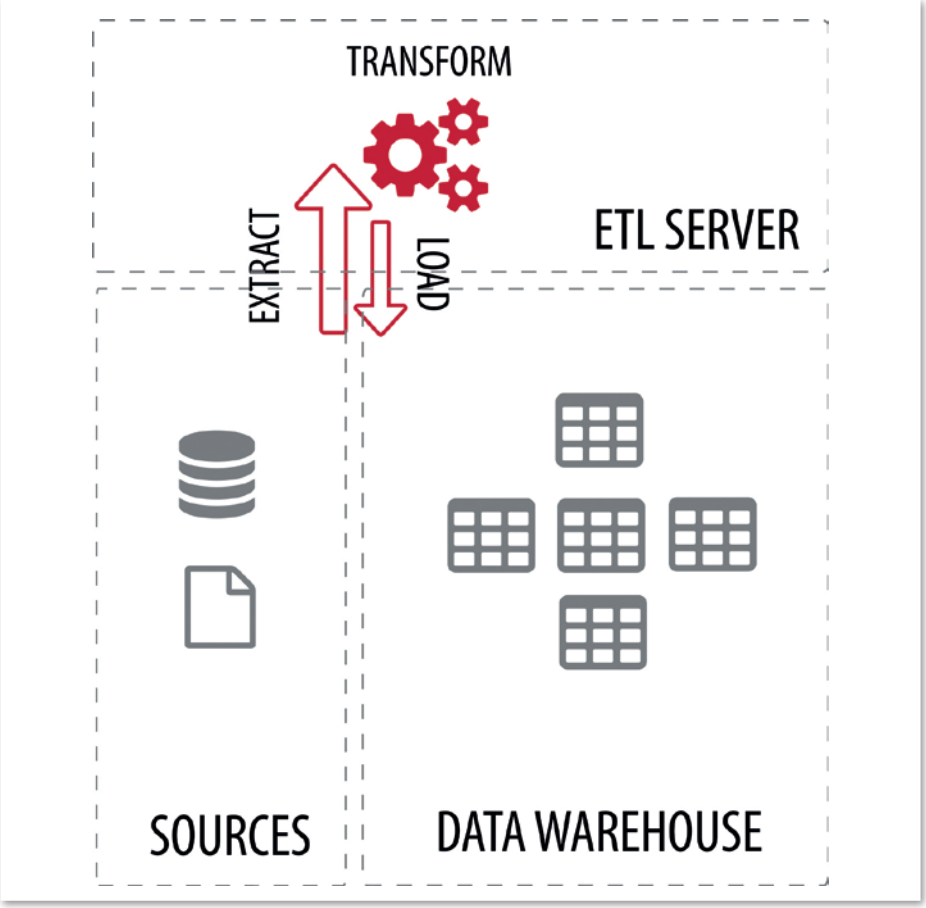

Abbildung 1: Erster DWH-Ansatz

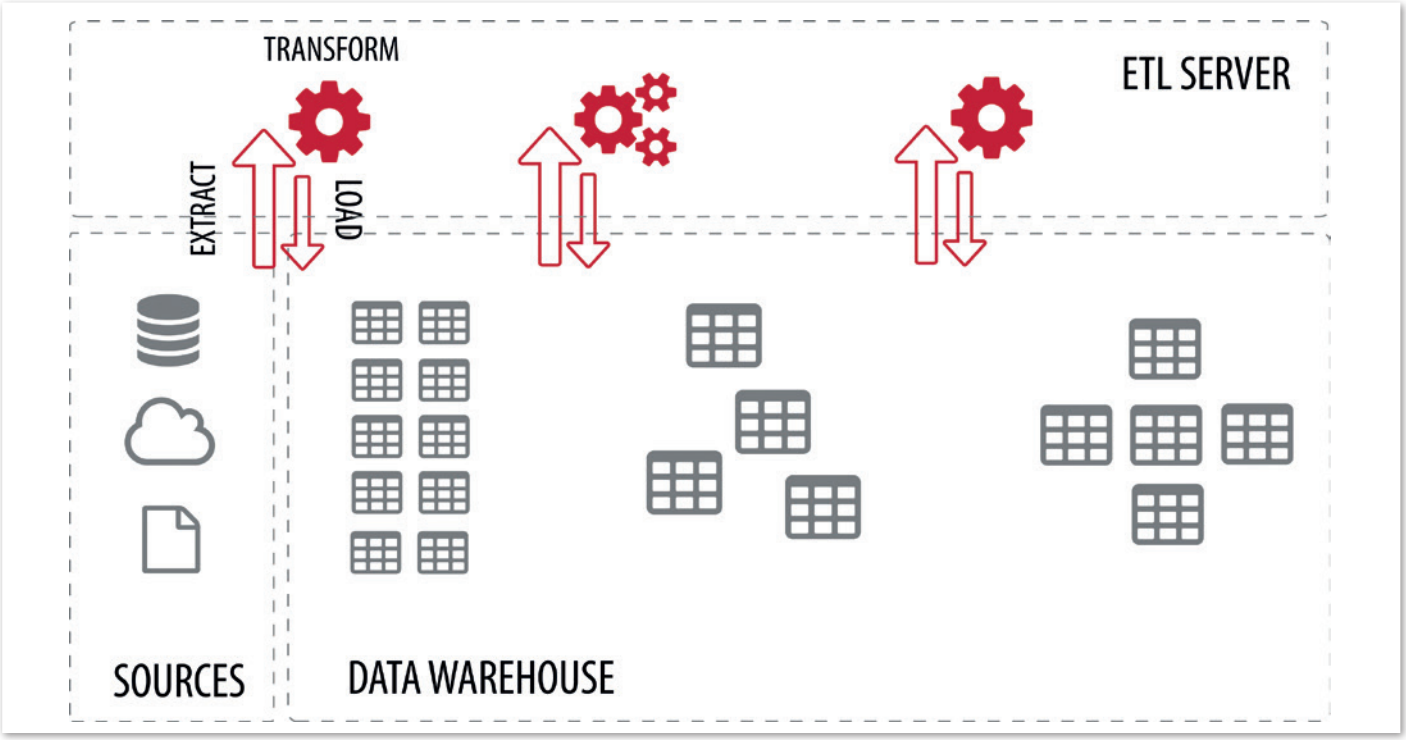

Abbildung 2: ETL in einem mehrschichtigen Data Warehouse

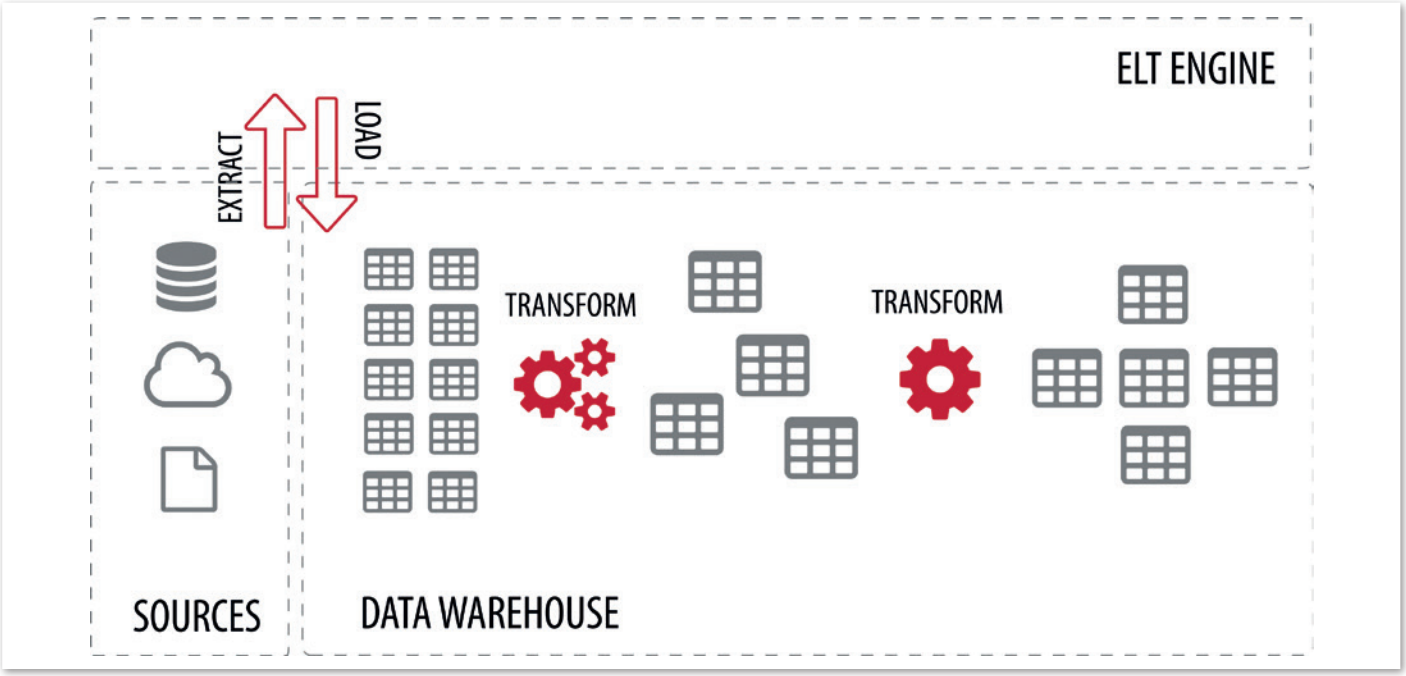

Abbildung 3: ELT in einem mehrschichtigen Data Warehouse

führungspläne für größere Datenmengen. Speicherplatz ist heutzutage keine teure Ressource mehr. Column-based-, In-Memory- und MPP-Datenbanken sind speziell für Data Warehousing und Daten-Analysen ausgelegt. Viele Datenbanken bieten inzwischen Integrationskomponenten wie Bulk-Loader und CSV-, XML- oder JSON-Parser.

Datenbanken können in der Cloud betrieben werden und skalieren dort nach Bedarf. Dadurch haben sich neue, auf Data Warehousing spezialisierte Cloud-Datenbanken (wie Autonomous Data Warehouse von Oracle, Amazon Redshift, Snowflake) auf dem Markt etabliert. Die Entwicklung der Datenbank-Technologie ist also sehr lebendig und noch lange nicht abgeschlossen, die bestehenden Datenbanken werden kontinuierlich weiterentwickelt, daneben kommen immer wieder neue Technologien und Produkte hinzu.

yesSQL

Wenn man die Datenbank als ELT-Plattform betrachtet, bietet sich SQL als Transformationssprache ganz natürlich an. Es ist eine verbreitete Sprache, die mächtige Transformationsmöglichkeiten (wie analytische Funktionen) mitbringt. ANSI SQL ist als Standard etabliert, was SQL zu einer universellen Abfragesprache für heterogene Datenbanken macht. Nicht nur Datenbanken

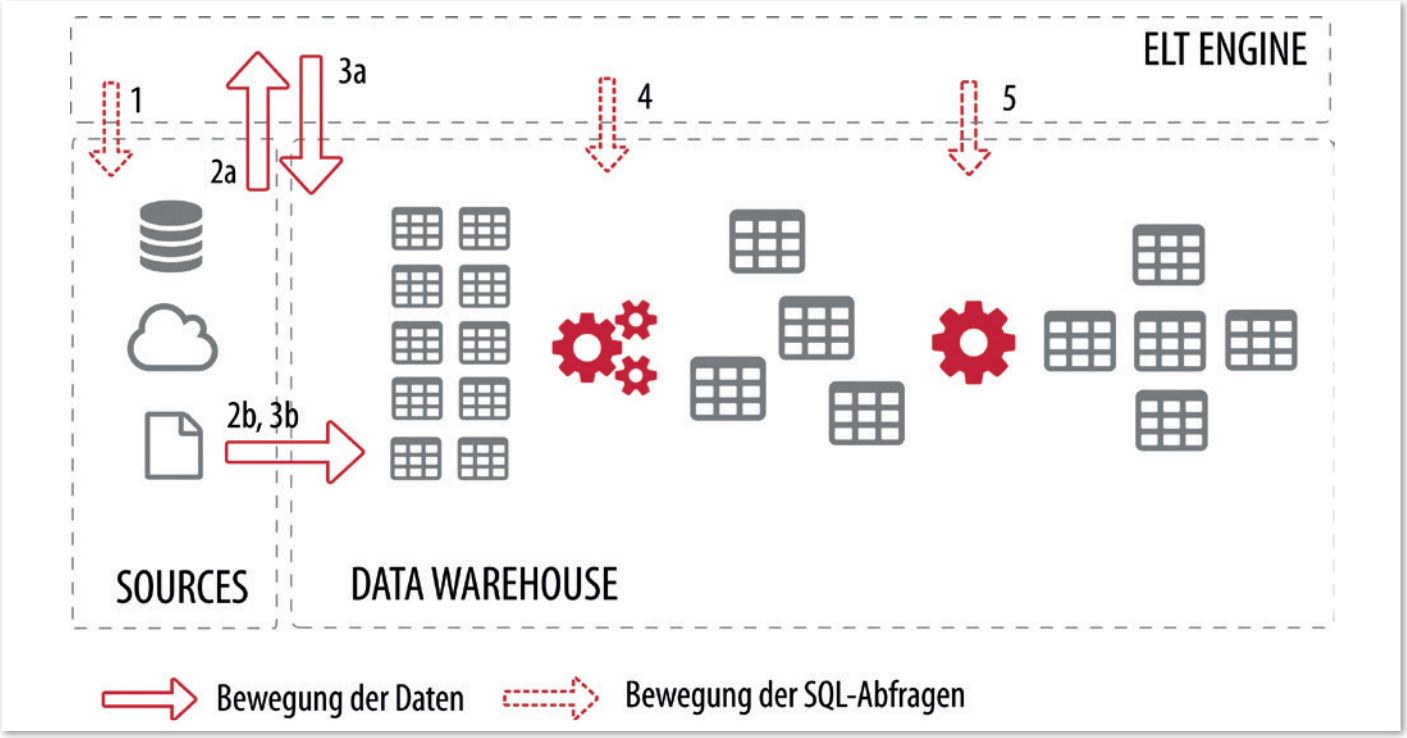

Abbildung 4: „noETL, yesSQL" im Data Warehouse

verstehen SQL, es gibt viele Big-Data-SQL-Tools (Oracle Big Data SQL, Hive, Impala, Spark SQL), SQL-basierte Flat-File-Integrationstools (Apache Drill) und die Mehrzahl der BI-Tools unterstützt die Verwendung von SQL.

SQL ist eine sehr verbreitete Sprache, was für ein größeres Angebot an Mitarbeitern mit SQL-Kenntnissen im Gegensatz zu proprietärem ETL-Tool-Know-how sorgt.

Die Transformationsentwicklung mit SQL verläuft schneller und agiler als die Entwicklung mit grafischen Tools, da die Analyse, der Prototyp und die Implementierung in der gleichen Sprache erfolgen. So entsteht kein Medienbruch zwischen dem während der Analyse entwickelten Code (SQL) und dem produktiv laufenden Code (proprietärer Code bei Verwendung eines ETL-Tools). Der Transformation-Code in SQL ist portabel und in der Regel kompakter und übersichtlicher als grafische Mappings. Damit eignet er sich auch besser für komplexere Logik.

„noETL, yesSQL" im Data-Warehousing-Kontext

Data-Warehouse-Lösungen bieten einen perfekten Rahmen für das „noETL, yesSQL"-Konzept, weil die Daten in einer oder wenigen Datenbanken gespeichert sind. Solche Lösungen sind Massendaten-orientiert und benötigen viele komplexe Transformationen für die Datenbewirtschaftung.

Für die Umsetzung des „noETL, yesSQL"-Verfahrens im Data Warehouse wird eine ELT-Engine benötigt. Das ist ein Tool oder ein Set von Tools, um Integrations-, Orchestrierungs- und Scheduling-Aufgaben in einem ELT-Prozess zu übernehmen. Dabei kann ein spezielles ELT-Tool zum Einsatz kommen oder ein Standard-ETL-Tool, das ELT-Verfahren unterstützt, oder es wird eine Kombination aus einem Datenintegrations-Tool und einem eigenen SQL-Framework verwendet *(siehe Abbildung 4)*.

Man unterscheidet grundsätzlich zwei Schritte in der Datenaufbereitung: Daten-Integration (EL) und Daten-Transformation (T). Im ersten Schritt werden die Daten aus den Quellen extrahiert und ins Data Warehouse geladen. Die Übertragung der Daten erfolgt „1:1" mit Datentyp-Konvertierung ins Zielformat. Für die Abfrage der Quellsysteme bieten sich SQL oder eine andere native Sprache des Quellsystems an (*Abbildung 4*, Punkt 1). Die Gestaltung des Daten-Integrationsschritts hängt von der jeweiligen Landschaft ab; die folgenden Optionen könnten einzeln oder in Kombination verwendet werden:

- Laden mit einem Integrations- oder Standard-ETL-Tool (*Abbildung 4*, Punkte 2a und 3a)
- Laden mit den nativen Integrations-Features der Datenbank (*Abbildung 4*, Punkte 2b und 3b)

Im zweiten Schritt werden die Daten im Data Warehouse mit SQL-Abfragen transformiert (*Abbildung 4*, Punkte 4 und 5). Bei der Entwicklung der Transformationen empfiehlt es sich, Plain-SQL für fachliche Transformationslogik zu verwenden. Einerseits ist die SQL-Ausführung in der Datenbank performant, weil sie für Massendaten-Operationen entworfen und optimiert wurde, andererseits vereinfacht sich Data Lineage, weil Plain-SQL besser geparst und analysiert werden kann. Um Performance und Wartbarkeit zu gewährleisten, sind die Transformationen möglichst granular zu halten. PL/SQL oder Scripting könnten zur Automatisierung der wiederkehrenden oder technischen Operationen eingesetzt werden. Um SQL-Wildwuchs zu vermeiden, bieten sich drei Optionen an:

- Beim Templating wird eine Vorlage erstellt, die die Entwickler verwenden können. Diese Templates können beispielsweise technische Routinen wie Logging beinhalten, damit sie nicht von jedem Entwickler neu ausprogrammiert werden müssen. Der Template-basierte Code der Entwickler wird anschließend ausgerollt. Die Nachteile des Templating bestehen darin, dass die falsche oder fehlerhafte Verwendung eines Templates schwer zu erkennen und zu verhindern ist. Zudem führen Änderungen an den Templates

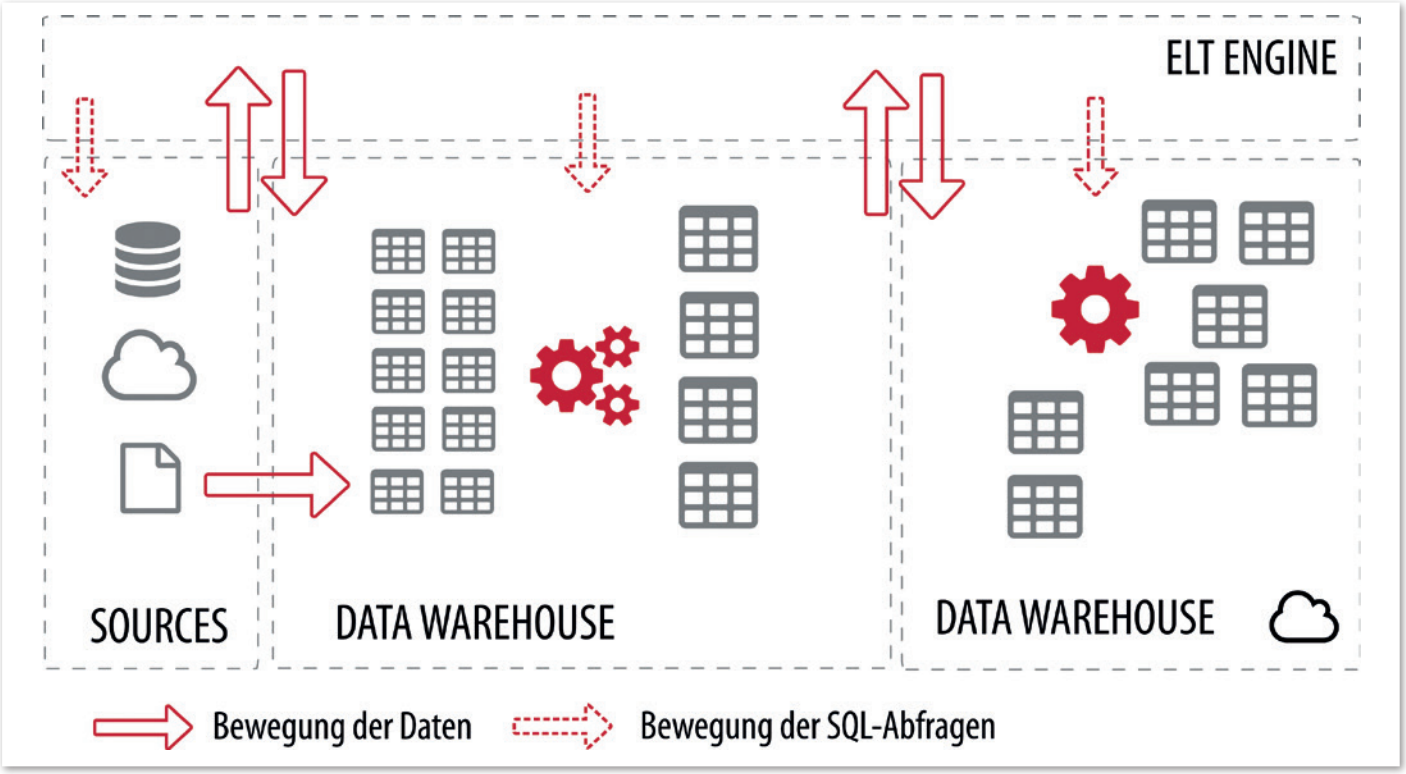

Abbildung 5: „noETL, yesSQL"-hybride Architektur im Data Warehouse

dazu, dass alle bereits laufenden Instanzen gegebenenfalls angepasst werden müssen.

- Code-Generierung erzeugt Instanzen anhand der Template-Vorlage und der Ergänzungen des Entwicklers. Die Generierung verhindert, dass der Entwickler das Template falsch oder unvollständig verwendet. Bei Änderungen des Templates ist eventuell das Re-Deployment aller bereits erzeugten Instanzen notwendig.
- Bei der Modularisierung wird der effektive Code zur Laufzeit anhand der im Modul hinterlegten Logik (Template) und des vom Entwickler eingegebenen Codes erzeugt. Bei Änderung der Modullogik ist lediglich das Re-Deployment des Moduls erforderlich. Das macht die Stärke dieser Methode bei der SQL-Entwicklung aus.

Idealerweise unterstützt die ELT-Engine eines dieser drei Verfahren. Der Aufbau einer zukunftssicheren Transformations-Architektur sollte von folgenden Überlegungen geleitet werden: Wo immer möglich, werden die Stärken der beteiligten Systeme ausgenutzt und die Transformationen an diejenige Umgebung delegiert (sogenanntes „Push-Down"), in der die Ausführung am effizientesten durchgeführt werden kann. Die „noETL, yesSQL"-Transformations-Architektur trägt genau diesem Kalkül Rechnung; dadurch ist sie transparent, flexibel und unabhängig von der transformierten Datenmenge, indem sie sich auf die Orchestrierung der Transformationen beschränkt. Auch in komplexeren Landschaften, in denen mehrere Systeme am Transformationsprozess beteiligt sind, bleibt der Grundgedanke der Transformations-Architektur derselbe, nur dass weitere Transformations-Plattformen hinzukommen. Dazu ein Beispiel, bei dem die Landschaft um eine weitere Cloud-Datenbank erweitert wird *(siehe Abbildung 5)*.

Bei diesem Aufbau kommt eine neue Integrationsstelle zwischen der On-Premises- und der Cloud-DWH-Datenbank dazu. Manche Cloud-Datenbanken bieten auch native Integrations-Komponenten an, die aus SQL heraus gesteuert werden können. Alles andere bleibt aus der Sicht der Transformations-Architektur identisch, die SQL-Transformationsabfragen werden lediglich an eine weitere Datenbank gesendet.

Fazit

In der modernen Data-Warehousing-Welt mit steigender Komplexität und wachsendem Daten-Aufkommen sind transparentere und flexiblere Datentransformations-Lösungen erforderlich. ELT ist ein alternatives Datentransformations-Verfahren, das sich besonders gut im Bereich von Massendaten-Verarbeitung für Data Warehousing bewährt hat.

Durch Anwendung des ELT-Verfahrens werden im Gegensatz zu ETL die Stärken der jeweiligen Datenspeicher-Technologien wie Cloud-, In-Memory-, MPP-Datenbanken, Hadoop oder Spark ausgespielt. Durch zahlreiche Features wie auf Massendaten optimierte Ausführungspläne, hohe Skalierbarkeit und reiche Integrationsfeatures bieten moderne Datenbanken durchaus eine passende Plattform für ELT.

SQL stellt dabei eine universelle, standardisierte und zukunftssichere Sprache für die Datenverarbeitung und Transformationsentwicklung dar. Die Kombination aus ELT und der Transformationsentwicklung mit SQL bietet eine sogenannte „noETL, yesSQL"-Transformationslandschaft. Diese benötigt nicht immer ein anderes Tooling, sondern vor allem eine andere Herangehensweise. Aus diesem Grund ist der Einstieg in die „noETL, yesSQL"-Welt auch schrittweise und ohne Big Bang möglich und sinnvoll.

Alec Shalashou
alec@datasqill.de