„ETL vs. ELT" ist nach wie vor eines der Reizthemen bei der Diskussion von BI-Architekturen. Die Diskussionen beziehen sich dabei in der Regel auf den Schritt der Beladung des DWH (Staging) und der Weiterverarbeitung in das DWH-Modell (Integration Layer). Etwa bei der Erstellung der Data Marts für die BI-Applikationen erhält das Daten-Management dagegen weniger Aufmerksamkeit. Nicht selten sind Performance-Probleme die Folge, die mit viel Aufwand (und Hardware) wieder beseitigt werden müssen. Betrachtet man BI-Applikationen nur als ETL-Prozesse, öffnen sich durch den veränderten Blickwickel auf das Daten-Management unentdeckte Möglichkeiten zur Lösung von Problemen.

Applikationsanbindung an das Data Warehouse: ETL vs. ELT

Dr. Gernot Schreib, b.telligent GmbH & Co. KG

Die Planung von Datenflüssen in DWH-Landschaften setzt ein Architektur-Prinzip von Schichten und Ebenen voraus. Aus den Erfahrungen der letzten zwanzig Jahre hat sich eine Standard-Architektur als praxistauglich erwiesen. Die Layer „Source Abstraction (Stage)", „Integration (Core)" und „Presentation (KPI)" dürfen in einer modernen DWH-Architektur nicht fehlen (siehe Abbildung 1: BI-DWH-Architektur).

Der Source-Abstraction-Layer hat die Aufgabe, die Quelldaten aufzunehmen und in eine für die weiteren Verarbeitungsschritte einheitliche Schnittstelle zu überführen. Der Integration-Layer bildet den fachlichen Kern des Datenmodells ab und modelliert idealerweise unabhängig von den Datenquellen (und Anwendungen) die fachlichen Entitäten auf granularer Ebene. Der Presentation-Layer schließlich stellt für die Applikationen (gegebenenfalls unter Berücksichtigung von Benutzer- und Rollen-Konzepten) eine einfach handhabbare, performante Zugriffsschicht auf das DWH zur Verfügung. In diesem Artikel steht die Umsetzung der Datenflüsse zwischen diesen Layern im Mittelpunkt (siehe Abbildung 1, Nr. 1 bis 4).

ETL vs. ELT

Die Google-Suche mit „ETL vs. ELT" ergibt ca. 270.000 Treffer, davon eine ganze Reihe von Blog-Einträgen. Das Thema wurde in der Vergangenheit und wird aktuell nach wie vor intensiv diskutiert. Interessanterweise ist bei den meisten der Beiträge eine klare Befürwortung beziehungsweise Ablehnung einer der beiden Architekturen zu erkennen. Dies ist insofern erstaunlich, als nahezu jeder der Artikel eine mehr oder weniger vollständige Betrachtung der Vor- und Nachteile enthält, die es erlauben würde, sich fallweise für die eine oder andere Architektur zu entscheiden. Es ist möglicherweise die Zubeziehungsweise Abneigung gegenüber dem Einsatz von Tools, gegebenenfalls auch einem bestimmten Tool, durch die sich die Autoren dann doch zu einer Generalisierung hinreißen lassen und einer der beiden Architekturprinzipien grundsätzlich den Vorzug geben.

Zur Unterscheidung der Reihenfolge Transform/Load beziehungsweise Load/Transform wird in diesem Artikel das Kriterium verwendet, ob die Daten die Datenbank verlassen müssen beziehungsweise ob die Datenbank während der Transformation noch die Kontrolle über das Daten-Management (wie Möglichkeit der Parallelisierung) innehat oder nicht. Während eines ETL-Prozesses dient die Datenbank ausschließlich als Datenspeicher. Das Daten-Management der Transform-

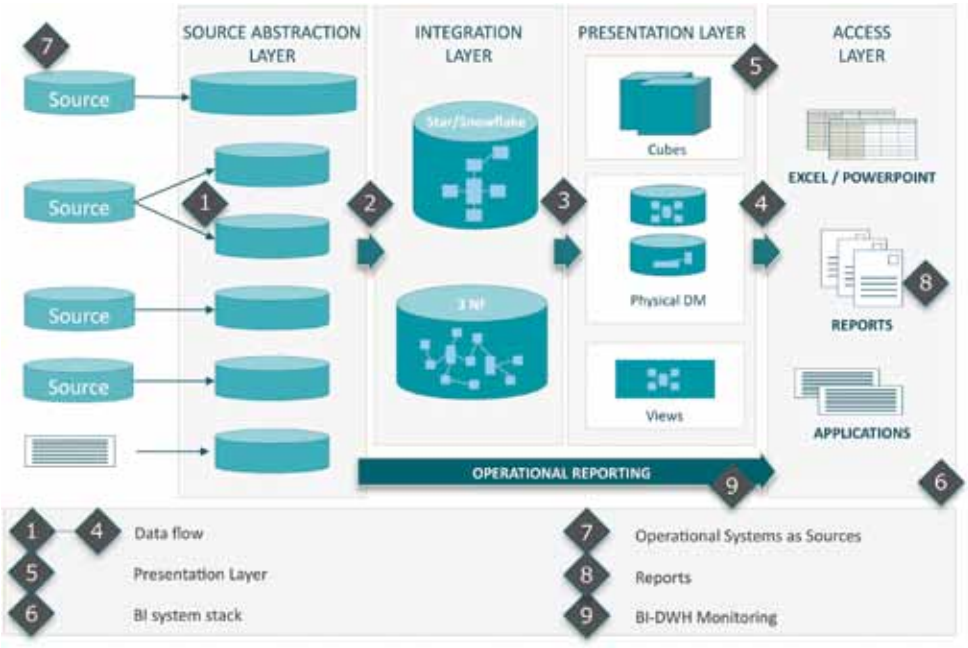

Abbildung 1: BI-DWH-Architektur

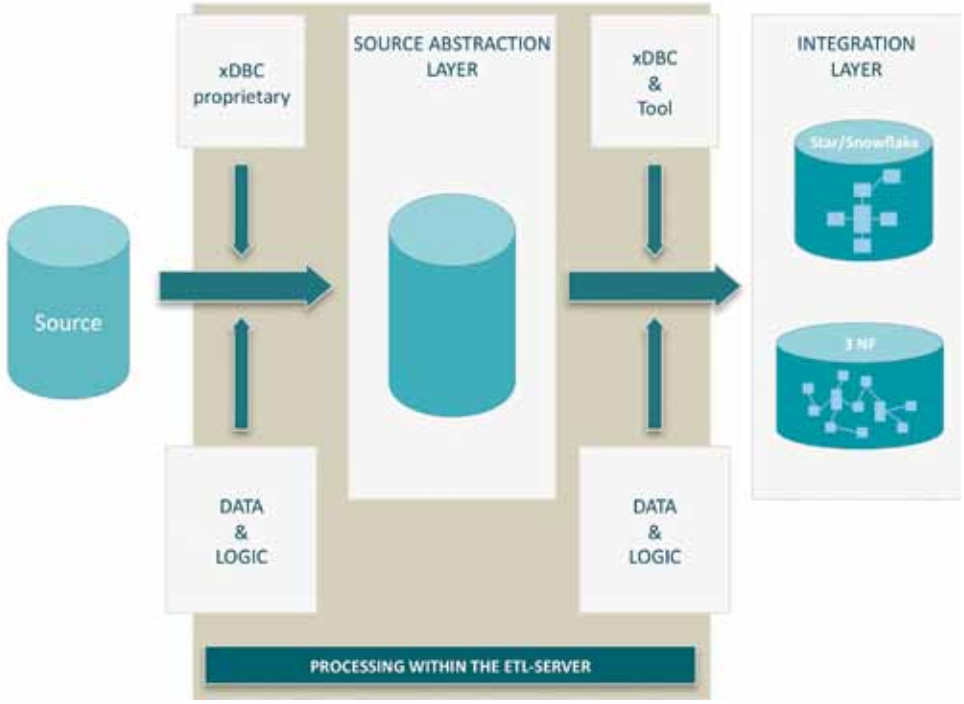

Abbildung 2: ETL-Datenfluss

Operationen wird außerhalb, etwa durch ein externes Tool durchgeführt (siehe Abbildung 2). Die grau hinterlegten Operationen, insbesondere Daten-Management und Logik, liegen außerhalb der Kontrolle der Datenbank.

Im ELT-Datenfluss ist hingegen die Datenbank das treibende System. Die Transform-Operation wird durch eine DML-/SQL-Operation innerhalb der Datenbank durchgeführt. Die notwendige (in der Regel zeilenbasierte) Logik muss durch die Datenbank aufrufbar zur Verfügung stehen (siehe Abbildung 3: ELT-Datenfluss). Die grau hinterlegten Operationen finden unter der Kontrolle der Datenbank statt. Die notwendigen logischen Operationen sind durch Datenbank-interne Prozeduren (gegebenenfalls nach der Nutzung von External-C-/Java-Funktionsaufrufen) realisiert.

Sowohl ETL- als auch ELT-Architekturen haben ihre Berechtigung, je nachdem, welche der Vorteile jeweils für die konkrete Aufgabenstellung günstiger sind. Während oft für die Datenbewirtschaftung der Schnittstelle „Source-Systeme“ zum DWH (Source Abstraction Layer) eine Grundsatzentscheidung getroffen wird (ETL-Tool vs. ELT-Datenbank-Ladung), ist die Situation für die Herstellung des Presentation-Layers differenziert. Eine Gegenüberstellung der Vor- und Nachteile speziell bei diesem Übergang folgt im nächsten Abschnitt.

Applikationen als ETL-Prozesse

Die Begriffe ETL beziehungsweise ELT werden in der Regel für den Prozess des Ladens der Sourcen in das DWH verwendet. Seltener wird der Prozess der Informationsbereitstellung für BI-Applikationen – also der Übergang in den Presentation-Layer – auch als ETL-Prozess bezeichnet, obwohl hier meist genauso viele Daten unter Einsatz von noch mehr Logik bewegt werden. Gerade hier sollte die BI-Architektur aus Performance-, Prozess- und Kosten-Aspekten sehr individuell den Erfordernissen angepasst sein. Beispiele für Anwendungen sind vor allem KPI-Berechnungen als Basis für Reporting, analytisches CRM (wie Segmentierung, Entscheidungsbäume, Regression, neuronale Netze etc.) und operatives CRM (wie Regelverarbeitungen von CRM-Logiken etc.).

Während die Bereitstellung von Basis-KPIs meist selbstverständlich direkt in SQL (und damit in einem ELT-Prozess) realisiert wird, ist es bei komplexen Logiken durchaus üblich, diese in einem externen System (wie SAS-, SPSS-Server, Visual Rules, Talend etc.) abzubilden. Tabelle 1 zeigt die Vor- und Nachteile des ELT-Prozesses.

Der Einsatz von externen Tools (wie SPSS Modeler oder SAS Enterprise Miner) bei komplexen Logiken ist insofern gerechtfertigt, als die komplexen Logiken dort auch entwickelt und selten direkt als SQL implementiert werden. Dies führt dann in der Nomenklatur dieses Vortrags zu einem ETL-Prozess, da die Datenbank als reiner Datenspeicher dient und das Datenmanagement im externen Tool liegt. Tabelle 2 zeigt die Vor- und Nachteile dieses Ansatzes.

Applikationen als ELT-Prozesse

Aufgrund der großen Bandbreite von Applikationen eines BI-DWH ist eine generelle Entscheidung für eine ETL- beziehungsweise ELT-Architektur beim Übergang vom Integration- zum Pre-

VORTEILE	NACHTEILE
• Daten verlassen die DB nicht, d.h. die Verarbeitung der großen Datenmengen (Cursor) findet in der Datenbank statt • Speicherverwaltung der DB wird benutzt • Parallelisierungsoptionen der DB wird benutzt • Transaktionssystem der DB wird benutzt • Reine Rechenoperationen können in (externen) Funktionen / Prozeduren sehr schnell und effizient ausgeführt werden • Ideal für kontext-/nebenläufigfreie Rechenoperationen auf Spalten einer einzelnen Zeile (z. B. Segmentierung, Scoring, decision tree, regression, ...) • Kein zweites Systemumfeld (Komplexitätsreduktion, Kosten)	• Logik muss in SQL / PLSQL codiert werden • DB muss externe Funktionen unterstützen • Bei externen Funktionen i. d. R. nur Operationen innerhalb einer Zeile möglich; keine zeilenübergreifenden Operationen • DB-Server wird belastet (CPU-Last) • keine Redundanz, da nur 1 System

Tabelle 1: Vor- und Nachteile von ELT

sentation-Layer (Nr. 3 in Abbildung 1: BI-DWH-Architektur) noch schwieriger zu treffen als bei der Datenladung des Source-Abstraction- (Stage) beziehungsweise Integration-Layers (Nr. 1 und 2 in Abbildung 1: BI-DWH-Architektur). Möglicherweise ist diese generelle Entscheidung gar nicht sinnvoll und sollte von Applikation zu Applikation separat getroffen werden, um die jeweiligen Vorteile zu nutzen beziehungsweise Nachteile zu vermeiden. Leider wird gerade beim Einsatz eines externen Tools beziehungsweise Servers oft die Möglichkeit, den (operativen) Prozess über einen ELT-Ansatz abzuwickeln, gar nicht berücksichtigt. Warum eigentlich nicht?

Damit eine Überführung der Logik in die Datenbank möglich ist, muss sie zum einen durch das externe Tool unterstützt werden. Dies wird dadurch realisiert, dass die Applikationslogik entweder als SQL oder in einer gängigen Programmiersprache wie Java oder C exportiert werden kann. Zum anderen muss dann innerhalb der Datenbank die Programmlogik des Exports performant zur Verfügung gestellt werden können. Für Oracle ist letztere Bedingung durchweg erfüllt. Ein generiertes SQL ist (meist) kein Problem, aber auch Java-Code kann direkt in Oracle geladen und durch die integrierte Java-Umgebung ausgeführt werden. Schließlich lassen sich dynamische Libraries (C-Code) über PL/SQL-Wrapper mit wenigen Handgriffen einbinden. Als einfachstes Beispiel dient hier eine PL/SQL-Funktion für eine C-Funktion (siehe Listing 1).

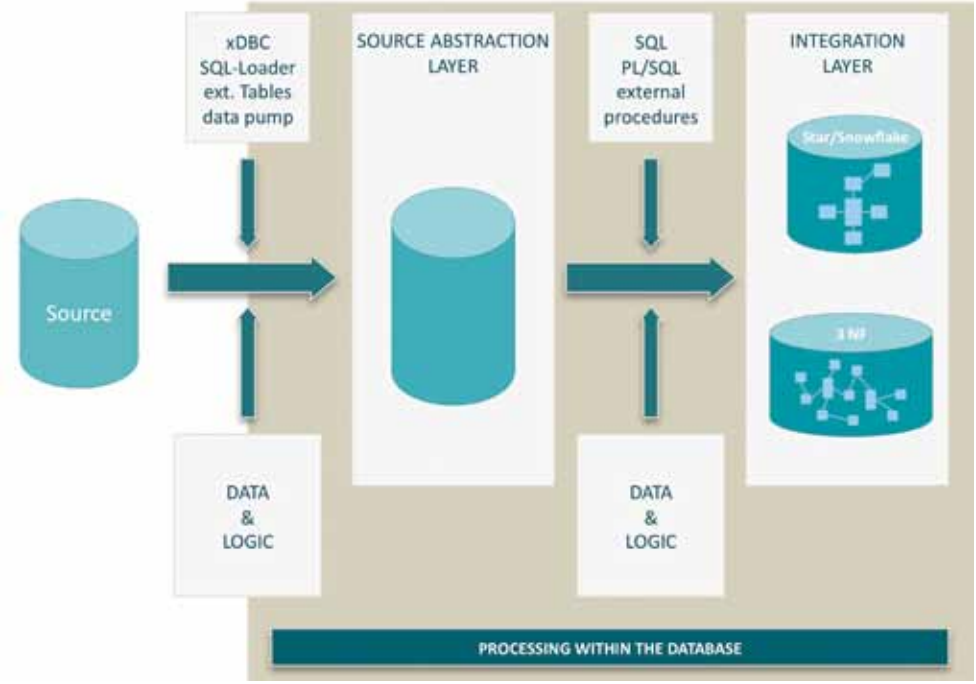

Abbildung 3: ELT-Datenfluss

Dabei ist „<libname>“ ein mit „create library“ angelegtes Oracle-Library-Objekt, das den Pfad und den Namen der Library auf Betriebssystem-Ebene enthält, und „<c-fct-name>“ der Name der C-Funktion innerhalb dieser Library. Die Syntax für das Wrapping von C-Funktionen innerhalb von Oracle ist sehr mächtig und kann deutlich mehr. Dieses Beispiel hier ist der einfachste Fall, zeigt aber auch, dass das Einbinden von C-Funktionen innerhalb von Oracle (mit den Standard-Datentypen „number“ und „varchar2“) sehr einfach umgesetzt werden könnte.

Die Überführung eines ETL-Applikationsprozesses in einen ELT-Prozess scheitert daher meist an der fehlenden Unterstützung durch die Applikations-Tools. Dies ist nicht weiter verwunderlich, da alle Hersteller aus nachvollziehbaren Gründen daran interessiert sind, mit ihren Tools einen möglichst großen Teil der Prozesskette abzudecken. Deshalb werden Funktionalitäten zum Export der Logiken entweder gar nicht oder nur rudimentär und mit Einschränkungen implementiert. Vor allem bei datenintensiven Anwendungen steht dieses Vorgehen klar im Konflikt zum Interesse des BI-Architekten. Aus Architektur-Sicht wäre hier ein ELT-Ansatz viel sinnvoller.

VORTEILE	NACHTEILE

• Unabhängig von Datenbank (nur definierte Schnittstellen wie z. B. ODBC, JDBC Connection notwendig), wenn kein embedded SQL
• Stärken der Programmiersprache im externen Tool voll ausnutzbar
• kein tiefes Datenbank know how notwendig
• Durch Systemtrennung Entlastung von DB-Server möglich
• Redundanz
	• Verbindung zur DB (User / Passwort)
	• größere Datenmengen müssen aus der Datenbank extrahiert werden, extern verarbeitet werden und ggf. wieder geschrieben werden
	• i.d.R. sequentielle Verarbeitung der Daten
	• ggf. embedded SQL notwendig (proprietär, wartungsunfreundlich, fehlende Versionssicherheit)
	• hohe Komplexität und ggf. Kosten aufgrund Einsatz zweier Systeme (DB + 3rd party)
	• Performanceprobleme, falls
	  • 3rd party System und/oder DB nicht parallelisieren können
	  • Datenmenge den verfügbaren Hauptspeicher übersteigt

Tabelle 1: Vor- und Nachteile von ETL

Anwendungsbeispiele

BI-Prozesse finden sich in jedem Unternehmen; die Haupt-Anwendungsgebiete sind Management-Informationssysteme, Analytik sowie Kampagnen-Management und Planung. Erfahrungsgemäß entwickelt sich ein BI-Umfeld auch entlang dieser Themen. Sind diese Haupt-Anwendungen etabliert, benötigen sie dann in etwa die gleichen Ressourcen des BI-DWH.

Unternehmungen mit ausgeprägter B2C-Geschäftstätigkeit betreiben

```
FUNCTION c_fct_wrapper (<var-list>)
  RETURN BINARY_INTEGER
AS LANGUAGE C LIBRARY <libname> NAME "<c-fct-name>";
```

Listing 1

(neben dem Klassiker „Reporting“) intensiv Anwendungen zur Pflege des Kundenstamms. Ohne Anspruch auf Vollständigkeit sind im Folgenden drei wichtige Beispiele beschrieben.

- *Segmentierung*
  Zur Bewertung des Kundenstamms, der Darstellung der Entwicklung der Zusammensetzung sowie der Allokation von Angeboten und Produkten erfolgt eine Segmentierung des Kundenbestands. Dabei wird jedem Kunden gemäß der Segmentierungsregel genau eine Klasse zugewiesen. Dieses Attribut bleibt nun über einen vorgegebenen Zeitraum – etwa Quartal, Halbjahr oder Jahr – für alle Kunden konstant, sodass mithilfe dieser Klassifikation Bewegung zwischen den Segmenten beobachtet, im günstigen Fall aktiv beeinflusst werden kann. Die Anzahl und Beschreibung der Klassen ist – mit Ausnahme der Klasse der „Neukunden“ – sehr individuell und wird den Zielen und Bedürfnissen des jeweiligen Unternehmens angepasst.
- *Analytik und Prognose*
  Der Begriff „Scoring“ ist in der Öffentlichkeit vor allem aus dem Bankensektor bekannt und dort (leider) negativ besetzt. Ganz anders sieht die Situation im BI-Umfeld aus. Scoring bezeichnet konkret die Anwendung statistischer Verfahren zur Prognose des Kundenverhaltens aus bekannten Daten, etwa darüber, ob ein Kunde ein Angebot aus der Werbung (nicht) annehmen wird, oder – in einigen Branchen sogar gesetzlich vorgeschrieben – die Vorhersage des Risikos eines Zahlungsausfalls. Mit Scoring-Verfahren werden also die Eintrittswahrscheinlichkeiten für das betrachtete Ereignis pro Kunde auf Basis von bereits bekannten Kundendaten ermittelt.
- *Kampagnen und Selektion*
  Ein weiteres wichtiges Anwendungsfeld ist das operative CRM. Basierend auf Prognosen über das Kauf- und Reaktionsverhalten von Kunden auf eine werbliche Ansprache werden die Kunden selektiert, die die höchste Affinität zur Kampagne aufweisen. Bei vorher festgelegter Größe der benötigten Kundengruppe kann so ein optimales Ergebnis erzielt werden.

Die genannten und viele weitere BI-Anwendungen erlangen im Laufe ihres Reifungsprozesses nahezu den Status operativer Anwendungen. Umso wichtiger ist es, diese Prozesse entsprechend zu automatisieren und in einem geordneten Regelbetrieb abzubilden. Besonderes Augenmerk ist dabei auf den Übergang von der Entwicklung (wie analytisches CRM) in den operativen Einsatz (wie operatives CRM) zu legen, da gerade das Operationalisieren komplexer analytischer Algorithmen aufwändig werden kann.

ETL- und ELT-Anbindung eines „SAS Enterprise Miner“-Modells

Obwohl die Verzahnung der operativen System- und BI-Landschaft in den letzten Jahren zugenommen hat, stehen zwischen beiden Welten sehr häufig, meist aus Sicherheits- und Stabilitätsgründen, hohe Mauern. In der täglichen Praxis bedeutet dies, dass die BI-Abteilung zwar eine agile und schnelle Analyse von geschäftskritischen Prozessen zu leisten vermag, die entsprechend zeitnahe Umsetzung im operativen System dann jedoch an der Ressourcen-Knappheit der IT-Abteilung scheitert. Dies liegt vor allem daran, dass die zur Analyse verwendeten Tools nicht gleichzeitig für die operative Umsetzung verwendet werden oder werden können. Wie diese Hürde dennoch überwunden werden kann, soll nun am Beispiel des SAS Enterprise Miner (EM) gezeigt werden.

Mit diesem oft genutzten und mächtigen Analyse-Tool können in einer entsprechenden Umgebung Analysen und Modellentwicklungen auf Basis eines Analyse-Datensatzes leicht aufgebaut werden (siehe Abbildung 4). Die Architektur der operativen Umgebung – für jeden Kundendatensatz muss regelmäßig die Modellberechnung durchgeführt werden – ist alles andere als trivial. Eine dedizierte Instanz des EM als Teil der operativen System-Landschaft (mit entsprechenden Release-Zyklen) ist aus Architektursicht denkbar. Die Anbindung erfolgt klassisch als ETL-Prozess.

Allein aus Kostengründen fällt diese Variante jedoch häufig aus. Neben den reinen Lizenzkosten sind auch Prozesskosten zu betrachten. So muss der IT-Betrieb die Wartung und den Betrieb zur Verfügung stellen und das durchaus komplexe Deployment der BI-Modelle in der Produktivumgebung durchführen. Letzteres erfordert spezielles SAS-EM-Wissen, das selten zum Portfolio eines IT-Betriebs gehört. Nun ist ein komplexer Deployment-Prozess in ein operatives System schlichtweg inkompatibel mit der agilen Arbeitsweise einer BI-Analytik-Abteilung. Andere Lösungen müssen gefunden werden.

Dafür wird die Fähigkeit des EM genutzt, die entwickelten Modelle entweder als Java- oder als C-Code (zusätzlich entsteht ein XML-File, das die Metadatenbeschreibung enthält) zu exportieren. Java- und C-Code sind für die IT leicht zu verarbeiten. Der vom SAS-EM generierte Code kann in ein C-/Java-/Embedded-SQL-Framework mit einer Oracle-Datenbank als Datenquelle eingefügt werden. Für die IT ist es nun ein Leichtes, einen schnellen und schlanken Deployment-Prozess zu etablieren, sodass der Agilität der BI-Analytik Rechnung getragen werden kann. Die Verwendung von Programmen mit Embedded SQL führt zu ETL-Prozessen.

Vom ETL zum ELT: Lösungsvariante „external function“

Falls die Berechnung der im BI-Modell enthaltenen Logik vor allem aus Zeilentransformationen besteht, lässt sich der vom SAS-EM generierte Code auch gut als „external function“ innerhalb der Oracle-Datenbank zur Verfügung

stellen. Da das Daten-Management dann vollständig innerhalb von Oracle stattfindet, sind alle gängigen Performance-Maßnahmen (wie Parallelisierung) innerhalb der Datenbank anwendbar. Darüber hinaus lässt sich die Automatisierung der Codeerstellung sogar so weiterentwickeln, dass etwa über ein in der Oracle-Datenbank (PL/SQL) hinterlegtes Framework die XML-Metadaten geladen und ausgewertet werden.

Es ist kein Problem für die BI-Abteilung, in der ihr gut bekannten PL/SQL-Umgebung einen Generator zu implementieren, der die notwendigen Header-/Wrapper-Funktionen in PL/SQL und gegebenenfalls C beziehungsweise Java erzeugt. Dieser Code kann in einer Test- oder Integrations-Umgebung auf syntaktische und semantische Fehler hin durch die BI-Abteilung selbst geprüft werden. Der durch die IT durchzuführende Deployment-Vorgang in der Produktionsumgebung kann auf ein absolut notwendiges Minimum an Tätigkeiten – hier den Austausch einer Library auf Betriebssystemebene – reduziert werden. Damit ist die Agilität auch in der Produktionsumgebung angekommen. Durch Verwendung der „external function" innerhalb der DB ist nun der Wechsel vom ETL- zum ELT-Prozess vollzogen. Während ETL-Prozesse alle Basisdaten extrahieren und dann das Ergebnis für den gesamten Datenbestand zurückliefern müssen, kann etwa durch selektive Auswertung von Views, die die Logik mithilfe der „external function" abbilden, die notwendige Materialisierung auf ein Minimum – beispielsweise pro Tag oder einen Teil der Kundengruppe – reduziert werden. Nur der Teil, der tatsächlich benötigt wird, muss materialisiert werden.

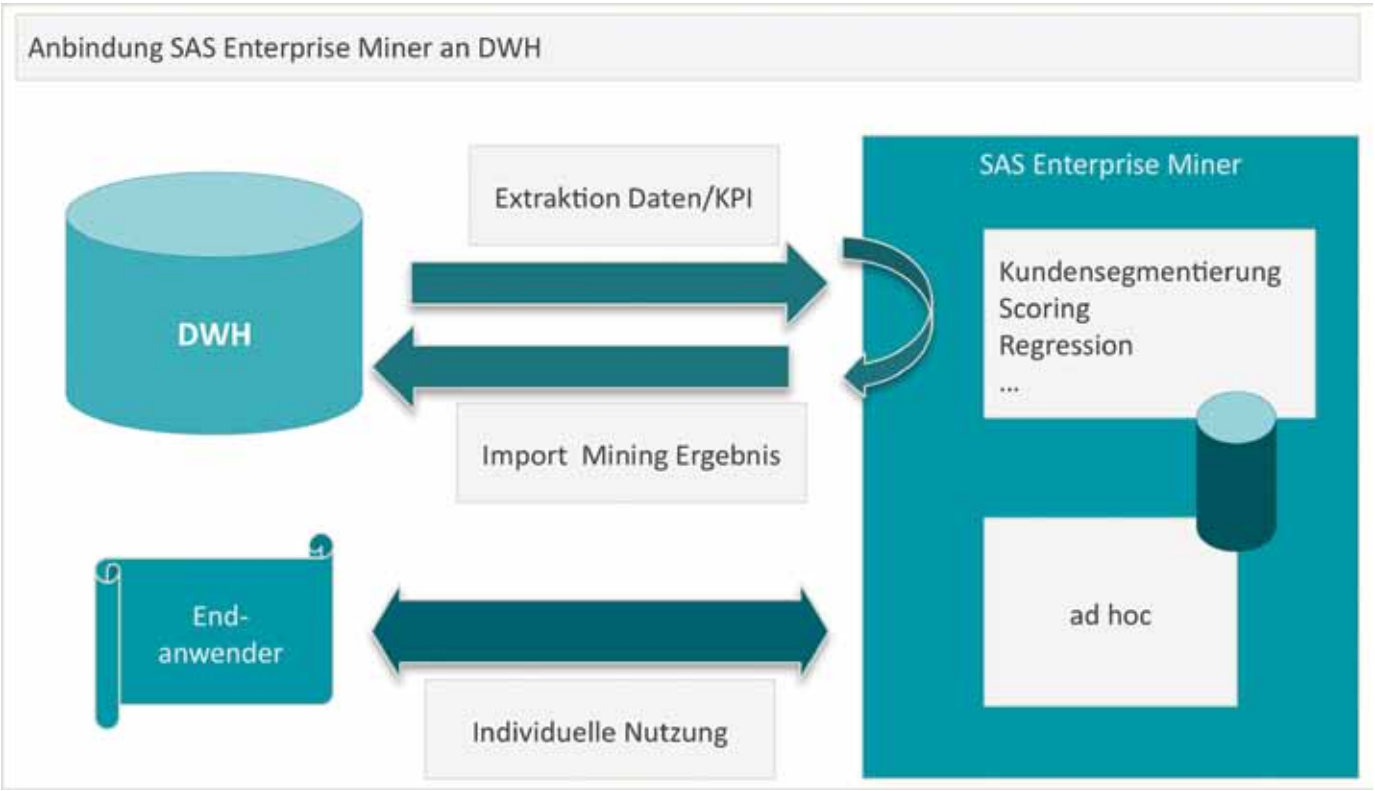

Abbildung 4: Beispiel ETL mit SAS-EM

Fazit

Bei der Erstellung der BI-Architektur von DWH-Landschaften wird oft ein erbitterter Streit zwischen ETL- und ELT-Anhängern geführt. Dies bezieht sich aber meist auf den Schritt der Beladung des DWH (Source-Abstraction-Layer), gegebenenfalls noch auf das der Mappings in den Integration-Layer. Bei der Erstellung des Presentation-Layers wird dem Daten-Management weniger Aufmerksamkeit geschenkt. Da funktionale Anforderungen im Vordergrund stehen, erfolgt die Implementierung dieses Prozesses vielmehr so nebenbei während der Umsetzung. Nicht selten sind Performance-Probleme die Folge, die mit viel Aufwand (oft auch Hardware) wieder beseitigt werden müssen. Eine mögliche Lösung des Problems besteht darin, eine Umstellung der Applikation von einem ETL-Prozess zu einem ELT-Prozess durchzuführen. Aus dem Blickwinkel des Datenmanagements in der DWH-Landschaft eröffnen sich hier oft ungeahnte Möglichkeiten.

Dr. Gernot Schreib
gernot.schreib@btelligent.com