Datenflüsse in der DWH-Landschaft: Einsatz von 3rd party Software

Dr. Gernot Schreib
b.telligent GmbH & Co.KG
Georg-Brauchle-Ring 54, 80992 München

Schlüsselworte
BI-Architecture, Business Intelligence, BI, ETL, ELT, data flow of BI-environment

Einleitung

ETL vs. ELT ist nach wie vor eines der Reizwörter bei der Diskussion von BI-Architekturen. Die Diskussionen beziehen sich dabei i. d. R. auf den Schritt der Beladung des DWHs (Staging) und der Weiterverarbeitung in das DWH-Modell (Integration-Layer). Bei der Erstellung z. B. der Data Marts für die BI-Applikationen wird dem Datenmanagement dagegen weniger Aufmerksamkeit geschenkt. Nicht selten sind Performanceprobleme die Folge, die mit viel Aufwand (und Hardware) wieder beseitigt werden müssen. Betrachtet man BI-Applikationen nur als ETL-Prozesse lassen, öffnen sich durch den veränderten Blickwickel auf das Datenmanagement unentdeckte Möglichkeiten zur Lösung von Problemen. Der Vortrag zeigt dies anhand eines Anwendungsbeispiels bei einer Direktbank.

1. BI-Architektur eines Data Warehouse (DWH)

Die Planung von Datenflüssen in DWH-Landschaften setzt ein Architekturprinzip von Schichten und Ebenen voraus. Aus den Erfahrungen der letzten 20 Jahre hat sich eine Standardarchitektur für den Bau von DWHs als praxistauglich ergeben. Der „Source Abstraction (Stage)"-, „Integration (Core)"- und „Presentation (KPI)"-Layer darf dabei in einer modernen DWH-Architektur nicht fehlen (siehe Abbildung 1).

1.1 3-Tier-Architektur

Der Source-Abstraction-Layer hat die Aufgabe, die Quelldaten aufzunehmen und in eine für die weiteren Verarbeitungsschritte einheitlich 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 (ggf. unter Berücksichtigung von Benutzer- und Rollenkonzepten) eine einfach handhabbare, performante Zugriffsschicht auf das DWH zur Verfügung. In diesem Vortrag wird v. a. dabei die Umsetzung der Datenflüsse (Nr. 1-4, Abbildung 1) in bzw. zwischen diesen Layern im Mittelpunkt stehen.

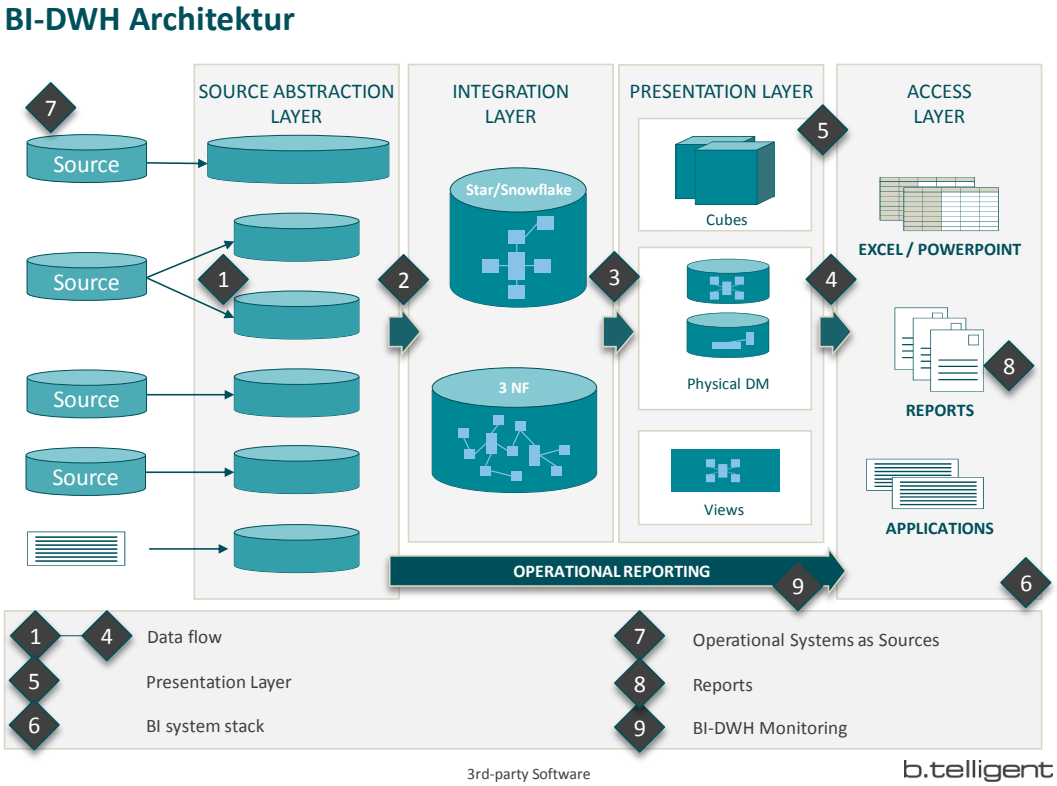

Abbildung 1: BI-DWH Architektur

1.2 ETL vs. ELT

Eine Google-Suche mit „ETL vs. ELT“ ergibt ca. 270 tsd. Treffer, davon eine ganze Reihe von Blog-Einträgen. Das Thema wurde in der Vergangenheit und wird aktuell nach wie vor intensiv diskutiert. Interessanterweise lässt sich bei den meisten der Beiträge eine klare Befürwortung bzw. Ablehnung einer der beiden Architekturen nachweisen. Dies ist insofern erstaunlich, da nahezu jeder der Artikel eine mehr oder weniger vollständige Vor-/Nachteile-Betrachtung enthält, die es erlauben würde, sich fallweise für die eine oder andere Architektur zu entscheiden. Es ist möglicherweise die Zu- bzw. Abneigung zum Einsatz von Tools, ggf. einem bestimmten Tool, durch die sich die Autoren dann doch zu einer Generalisierung hinreißen lassen und einer der beiden Architekturprinzipien grundsätzlich den Vorzug geben.

1.3 Definition ETL/ELT

Zur Unterscheidung der Reihenfolge **T**ransform-**L**oad bzw. **L**oad-**T**ransform wird in diesem Vortrag das Kriterium benutzt, ob die Daten die Datenbank verlassen müssen, bzw. ob die Datenbank während der Transformation noch die Kontrolle über das Datenmanagement (z. B. Möglichkeit der Parallelisierung) inne hat oder eben nicht. Während eines ETL-Prozesses dient die Datenbank ausschließlich als Datenspeicher. Das Datenmanagement der Transform-Operationen wird außerhalb, z. B. durch ein externes Tool durchgeführt (siehe Abbildung 2). Die grau hinterlegten Operationen, insbesondere Datenmanagement und Logik, finden außerhalb der Kontrolle der Datenbank statt.

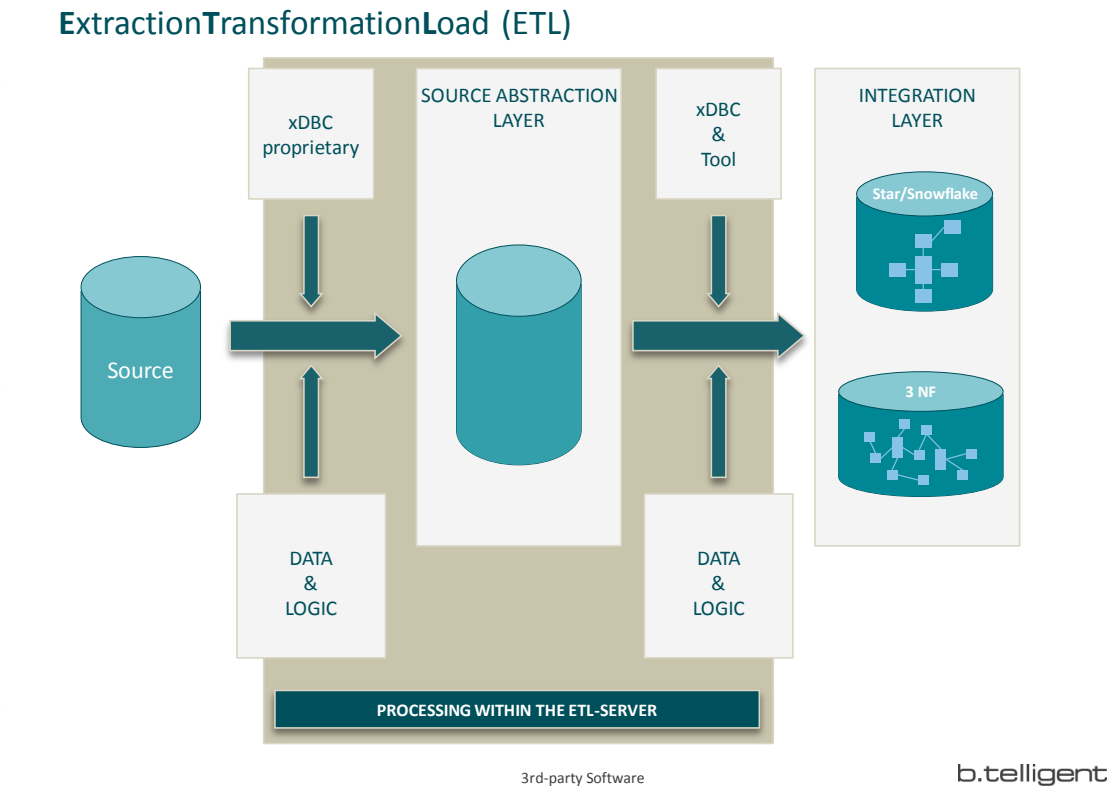

Abbildung 2: ETL-Datenfluss

Im ELT-Datenfluss dagegen ist die Datenbank das treibende System. Die Transform-Operation wird durch eine DML-/SQL-Operation innerhalb der Datenbank durchgeführt. Die notwendig – i. d. R. zeilenbasierte – Logik muss durch die Datenbank aufrufbar zur Verfügung stehen (siehe Abbildung 3). Die grau hinterlegten Operationen finden unter der Kontrolle der Datenbank statt. Die notwendigen logischen Operationen werden durch DB-interne Prozeduren (ggf. hinter Nutzung von external C/Java Funktionsaufrufe) realisiert.

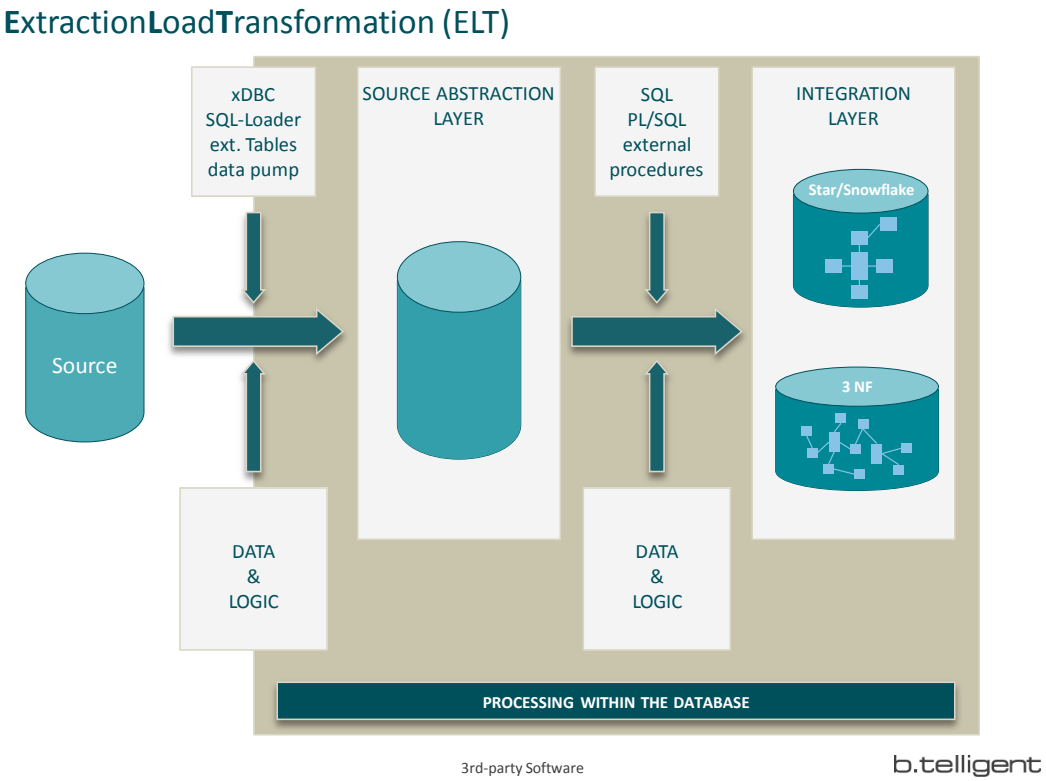

Abbildung 3: ELT-Datenfluss

1.4 Vor- und Nachteile von ETL bzw. ELT

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-Systemen zum DWH (Source-Abstraction-Layer) eine Grundsatzentscheidung getroffen wird (ETL-Tool vs. ELT-DB-Ladung), ist die Situation für die Herstellung des Presentation-Layers diffiziler. Eine Gegenüberstellung der Vor- und Nachteil speziell bei diesem Übergang folgt im nächsten Abschnitt.

2. Applikationen als ETL-Prozesse

Der Sprachgebrauch ETL bzw. ELT wird 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 aus Performance-, Prozess- und Kostengesichtspunkten sehr individuell die BI-Architektur den Erfordernissen angepasst werden. Beispiele für Anwendungen hier sind vor allem KPI-Berechnungen als Basis für

- Reporting
- analytisches CRM wie Segmentierung, Entscheidungsbäume, Regression, neuronale Netze, …
- operatives CRM wie Regelverarbeitungen von CRM-Logiken, …

2.1 Vor- und Nachteile von Applikationen als ETL-/ELT-Prozesse

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 (z. B. SAS-, SPSS-Server, visual rules, Talend, …) abzubilden. Die Vor- und Nachteile des ELT-Prozesses zeigt Abbildung 4.

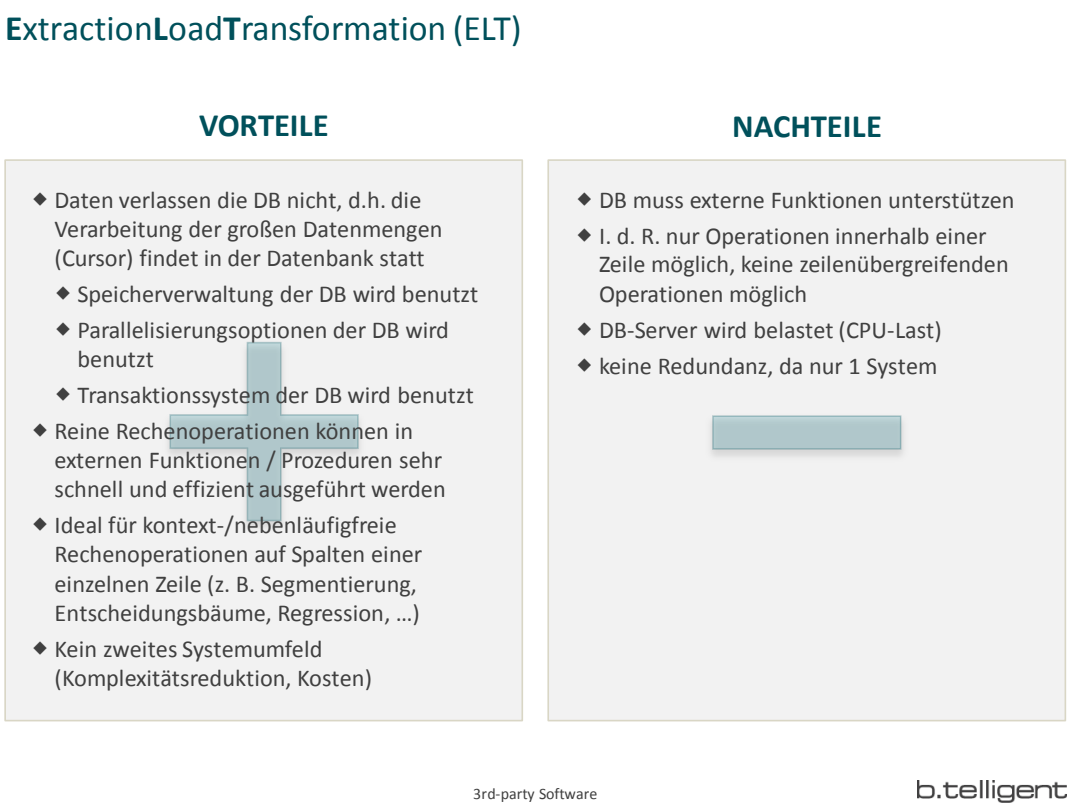

Abbildung 4: Vor- und Nachteile von ELT

Der Einsatz von externen Tools (wie z. B. SAS Enterprise Miner) bei komplexen Logiken ist insofern gerechtfertigt, da die komplexen Logiken dort auch entwickelt werden 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. Die Vor- und Nachteile dieses Ansatzes zeigt Abbildung 5.

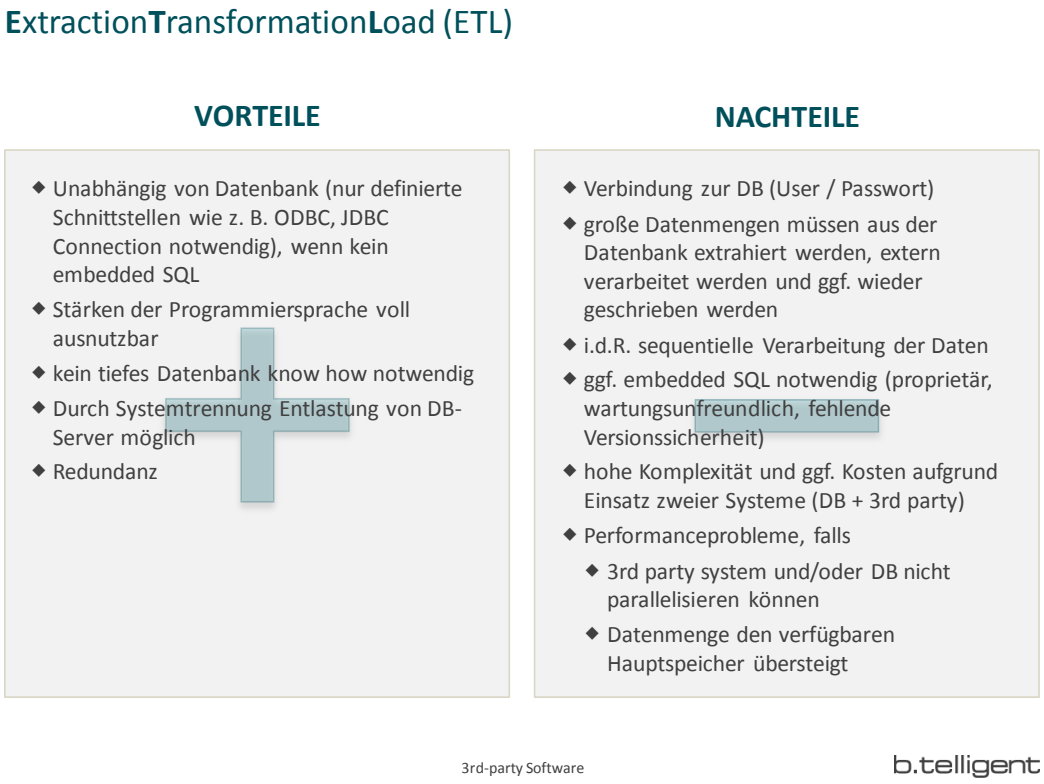

Abbildung 5: Vor- und Nachteile von ETL

2.2 Applikationen als ELT-Prozesse

Aufgrund der großen Bandbreite von Applikationen eines BI-DWHs ist eine generelle Entscheidung für eine ETL- bzw. ELT-Architektur beim Übergang von Integration- zum Presentation-Layer (Nr. 3 in Abbildung 1) noch schwieriger zu treffen, als bei der Datenladung des Source-Abstraction- (Stage) bzw. Integration-Layers (Nrn. 1 und 2 in Abbildung 1). Möglicherweise ist diese generelle Entscheidung gar nicht sinnvoll und sollte von Applikation zu Applikation separat getroffen werden, um die jeweiligen Vorteile zu nutzen bzw. Nachteile zu vermeiden. Leider wird gerade bei Einsatz eines externen Tools bzw. Servers oft die Möglichkeit, den (operativen) Prozess über einen ELT-Ansatz abzuwickeln gar nicht berücksichtigt. Warum nicht?
Damit eine Überführung der Logik in die Datenbank möglich ist, muss zum einen dies das externe Tool unterstützen. 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 werden 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 sei hier eine PL/SQL-Funktion für eine C-Funktion gezeigt:

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

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

Die Überführung eines ETL-Applikations-Prozesses 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 Architektursicht wäre hier ein ELT-Ansatz viel sinnvoller.

3. Das Anwendungsbeispiel bei einer Direktbank

Wie jedes Unternehmen, das vielen Endkunden bedient, ist es für eine Direktbank notwendig analytische und operative Berechnungen auf Kundendaten durchzuführen. Für bestimmte Prozesse wie der Risikovorsorge ist dies sogar gesetzlich vorgeschrieben. Im konkreten Fall werden mehrere Verfahren für unterschiedlich Anwendungen (Risikobewertung, Segmentierung, …) eingesetzt. Hierzu steht eine historisierte Tabelle mit KPIs als Basis der Anwendungen zur Verfügung. Die Modelle werden Hilfe des SAS Enterprise Miners (EM) von der Fachabteilung entwickeln. Aus Gründen der operativen Stabilität (hohe Verfügbarkeit des SAS-EM-Servers notwendig) wurde entschieden, den SAS EM-Server nicht in die operativen Prozesse mit einzubinden. Stattdessen wurde die Fähigkeit des EM genutzt, die entwickelten Modelle entweder als Java oder als C-Code (zusätzlich entsteht ein XML-File, dass die Metadatenbeschreibung enthält) zu exportieren und diese der IT-Abteilung zu übergeben. Diese hat nun die Aufgabe, täglich die Modellrechnungen für alle Kunden durchzuführen. Für diesen Prozess sind nun alle Voraussetzungen erfüllt, damit er als ELT-Applikations-Prozess implementiert werden kann:

- Es wird eine große Datenmenge (historisierte KPI-Tabelle) als Input benötigt
- Es wird eine große Datenmenge (Wert je Tag und Kunde) als Output generiert
- Das BI-Tool (hier SAS Enterprise Miner) ist in der Lage die Applikationslogik in einer gängigen Programmiersprache zur Verfügung zu stellen
- Die Logiken sind im Wesentlichen Zeilentransformationen.
- Die Datenbank, Oracle, ist in der Lage die externe Logik einzubinden.

Die IT-Abteilung hat nun zusätzlich das Interesse, den Ablauf stabil und wartungsarm zu realisieren. Dazu gehört auch, dass Änderungen am Modell schnell und unkompliziert operativ verfügbar gemacht werden können. Dies ist vor allem dann der Fall, wenn der generierte Code möglichst ohne Änderungen direkt verwendet werden kann. Leider ist dies selten – auch hier nicht – der Fall. Meist ist es notwendig den generierten Code nachzubearbeiten, was möglichst durch automatisierte Verfahren durchgeführt werden sollte. Dazu wurden zwei Lösungsvarianten, sowohl ein ETL- als auch ein ELT-Ablauf entwickeln.

3.1 Lösungsvarianten „embedded SQL“ (ETL-Prozess)

Für diese Lösungsvariante wurde ein Shell-Script entwickelt, dass den von SAS Enterprise Miner generierten C-Code mit Hilfe des XML-Files in ein C-Programm-Template mit embedded SQL-Befehle einbettet. Dieses C-Programm stellt eine Verbindung zur Datenbank her, öffnet einen Cursor für die Tabelle, ruft die Programmlogik des SAS-C-Programms auf, speichert die Ergebnisse in einer Hauptspeicher-Struktur und befüllt nach Abschluss mit einem bulk-insert die Zieltabelle. Da das Datenmanagement nicht durch Oracle durchgeführt wird, liegt hier ein ETL-Prozess vor. Die Applikationslogik (und das Datenmanagement) ist in einem C-Programm abgelegt, das vom DWH-Server ausgeführt wird. Damit verbleiben zwar die Daten auf demselben Host, das Datenmanagement wird allerdings nicht von Oracle überwacht. Dies hat einige Nachteile:

- Relativ aufwändiger Build-Prozess für die C-Programme, u.a. ist der Präprozessor für embedded SQL von Oracle notwendig.
- Hoher Hauptspeicherverbrauch während der Programmausführung auf dem DWH-Server.
- Das Passwort für den Login in Oracle muss auf dem Server hinterlegt werden, was wg. notwendiger Verschlüsselung zu einer weiteren Komplexität im C-Programm führt.
- Komplex Job-Steuerung, da parallel Ausführung von mehreren Modellen koordiniert werden muss.

Um diese Nachteile zu umgehen, wurde ein zweites Verfahren entwickelt, den Prozess als ELT-Prozess innerhalb Oracle abzubilden.

3.2 Lösungsvarianten „external function“ (ELT-Prozess)

Für diese Lösungsvariante wurden ein kleines Datenmodell und ein Code-Generator in PL/SQL entwickelt, die die notwendigen Anpassungen des von SAS-EM generierten C-/XML-Codes durchführt. Zunächst wird das XML in eine Datenbank-Tabelle geladen und mit Hilfe von in Views abgelegten XDB-Befehlen (XMLType von Oracle) als relationale Daten zur Verfügung gestellt. Der Code-Generator liefert nun hieraus ein C-Programm (ca. 20 Zeilen Code), den PL/SQL-Wrapper-Code (1 Zeile Code, siehe oben) und eine View (ca. 10 Zeilen Code) je Modell. Das C-Programm hat dabei die Aufgabe die Verbindung zwischen dem PL/SQL-Wrapper von Oracle und dem von SAS-EM generiertem C-Code herzustellen, da letzterer aufgrund der komplexen Signatur der Funktion des Modells nicht direkt von Oracle aufgerufen werden kann. Die View bildet (zunächst nur virtuell) den Aufruf der Logik ab, d. h. sie entspricht der Ergebnismenge der Berechnung aus der vorigen Lösungsvariante. Die tatsächliche Berechnung der Werte je Tag und Geschäftspartner reduziert sich auf die Materialisierung der View in Form von

```
INSERT INTO <fact_table>
   (SELECT * FROM <modell_view> WHERE DATE_ID = <id>);
```

Dieser Ansatz vermeidet die Nachteile der ersten Lösungsvariante:

- sehr einfacher Build-Prozess (< 15 Minuten) bei veränderten oder neuen Modellen
- keine Last auf dem DWH-Server durch Programme außerhalb von Oracle
- kein Passwort mehr notwendig
- sehr einfache Prozesssteuerung möglich

Da nun das Datenmanagement vollständig innerhalb Oracle stattfindet, sind alle gängigen Performancemaßnahmen (wie z. B. Parallelisierung) innerhalb Oracle anwendbar.

4. 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 DWHs (Source-Abstraction-Layer) ggf. noch für die Mappings in den Integration-Layer. Bei der Erstellung des Präsentation-Layers wird dem Datenmanagement weniger Aufmerksamkeit geschenkt. Da funktionale Anforderungen in Vordergrund stehen, „passiert“ die Implementierung dieses Prozesses vielmehr so nebenbei während der Umsetzung. Nicht selten sind Performanceprobleme die Folge, die mit viel Aufwand (oft auch Hardware) wieder beseitigt werden müssen. Eine Möglichkeit der Lösung des Problems ist es, eine Umstellung der Applikation von einem ETL-Prozess zu einem ELT-Prozess durchzuführen. Aus dem Blickwickel des Datenmanagement in der DWH-Landschaft eröffnen sich hier oft unentdeckte Möglichkeiten.

Kontaktadresse:

Dr. Gernot Schreib
b.telligent GmbH & Co.KG
Georg-Brauchle-Ring 54
D-80992 München

Telefon: +49 (0) 89-1222 81110
E-Mail gernot.schreib@btelligent.com
Internet: www.btelligent.com

Heiko Becker
comdirect bank AG
Pascalkehre 15
D-25543 Quickborn

Telefon: +49 (0) 4103-7041730
E-Mail: heiko.becker@comdirect.de
Internet: www.comdirect.de

b.telligent ist eine Unternehmensberatung, die auf Einführung und Weiterentwicklung von Business Intelligence, Customer Relationship Management und E-Commerce in Unternehmen in Massenmärkten spezialisiert ist.

Der Fokus liegt dabei auf der kontinuierlichen Optimierung von Geschäftsprozessen, Kunden- und Lieferantenbeziehungen durch den Erkenntnisgewinn aus der Verdichtung und Analyse von systemübergreifenden Geschäftsdaten. So lassen sich Margen erhöhen, Kosten senken und Risiken besser kontrollieren.

Kunden von b.telligent sind Branchenführer aus den Bereichen Telekommunikation, Finanzdienstleistung, Handel und Industrie.