JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

Entwicklung einer Business Intelligence-Lösung zur Reduzierung des Bullwhip-Effektes

am Beispiel der Sony DADC Austria AG

Diplomarbeit

zur Erlangung des akademischen Grades

Magister der Sozial- und Wirtschaftswissenschaften

im Diplomstudium Wirtschaftsinformatik

Eingereicht am

Institut für Wirtschaftsinformatik

Data and Knowledge Engineering

Eingereicht von

Martin Ibinger

Begutachter

o. Univ.-Prof. Dr. Michael Schrefl

Mitbetreuer

Dr. Stefan Berger

Linz, im Juni 2010

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit mit dem Titel „Entwicklung einer Business Intelligence-Lösung zur Reduzierung des Bullwhip-Effektes am Beispiel der Sony DADC Austria AG“ selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Linz, im Juni 2010

_____

(Martin Ibinger)

Zusammenfassung

Moderne Versorgungsketten bestehen aufgrund der immer komplexeren Produktions- und Distributionsprozesse aus vielen unterschiedlichen Akteuren. Der Informationsfluss zwischen den Akteuren der Versorgungskette ist jedoch oft auf die Tätigung von Bestellungen beschränkt. Dies kann zum Auftreten des Bullwhip-Effektes führen, der eine Aufschaukelung der Bestellmengen entlang der Versorgungskette vom Einzelhändler bis zum Rohstofflieferanten bezeichnet. Somit führt der Bullwhip-Effekt zu unnötigen Produktions- und Lagerkosten bei den Akteuren der Versorgungskette, da diese ihre Produktions- und Lagerkapazitäten auch bei annähernd gleichmäßiger Endkonsumentennachfrage an erhebliche Schwankungen anpassen müssen. Ist die Endkonsumentennachfrage allen Akteuren der Versorgungskette bekannt kann der auftretende Bullwhip-Effekt reduziert werden, da die Akteure der Versorgungskette den weiteren Bedarfsverlauf ihres jeweiligen Abnehmers anhand des Verlaufes der Endkonsumentennachfrage besser abschätzen können.

Ziel dieser Diplomarbeit ist daher die prototypische Entwicklung einer Business Intelligence-Lösung, die den Verlauf der Endkonsumentennachfrage direkt allen Akteuren der Versorgungskette zugänglich macht.

Die Umsetzung der Business Intelligence-Lösung erfolgt basierend auf einer Data Warehouse-Architektur, um eine integrierte Sicht auf die Endverkaufsdaten unterschiedlicher Handelsketten zu ermöglichen. Weiters ist das Design der entwickelten Business Intelligence-Lösung so gestaltet, dass eine dynamische Anzahl heterogener Datenquellen an das Data Warehouse gebunden werden kann. Die Analyseergebnisse der Business Intelligence-Lösung zeigen, dass der Bullwhip-Effekt auch in der Versorgungskette der Sony DADC Austria AG auftritt. Eine Anpassung der Produktions- und Lagermengen an den Verlauf der Endkonsumentennachfrage ermöglicht somit die Reduktion des auftretenden Bullwhip-Effektes.

Abstract

Due to sophisticated production and distribution processes, modern supply chains consist of many different actors. Albeit, the flow of information among the supply chain actors is often limited to simple purchase orders that do not always reveal the consumer demand. This behaviour can lead to the bullwhip effect that describes the magnification of order sizes moving up the supply chain from retailers to raw material suppliers. As a result, each supply chain actor faces an even greater fluctuation of order sizes moving up the supply chain. Consequently, the bullwhip effect causes unnecessary production and storage expenses because the supply chain actors have to fit their output and safety stock to significant demand fluctuations even if the consumer demand is nearly steady. A starting point to reduce the bullwhip effect in the supply chain is to make the customer demand available to each supply chain actor. This can lead to a reduction of the bullwhip effect because the supply chain actors are able to predict the upcoming demand size according to the customer demand process.

Therefore, the aim of this diploma thesis is the prototypical development of a business intelligence solution that makes the consumer demand available to each actor in the supply chain.

The implementation of the business intelligence solution is based on data warehouse architecture. Therefore, the business intelligence solution is able to provide an integrated view to the consumer demand process of different retail chains. Moreover, the design of the business intelligence solution allows using a dynamic number of heterogeneous data sources. The analytical results of the business intelligence solution show that the bullwhip effect occurs in the supply chain of Sony DADC Austria AG. Thus, an adjustment of the production and inventory amount based on the consumer demand enables the reduction of the bullwhip effect.

INHALTSVERZEICHNIS

1 Einleitung

Um im globalen Wettbewerb Schritt halten zu können, ist es für Unternehmen unabdingbar, ihre Produktions- und Logistikprozesse möglichst weitgehend auch an inhomogene Endkonsumentenbedürfnisse anzupassen. Um trotz dieser Rahmenbedingungen eine hohe Kosteneffizienz und Lieferzuverlässigkeit gewährleisten zu können, reicht es nicht aus, unternehmensinterne Abläufe zu optimieren. Daher rückt die unternehmensübergreifende Zusammenarbeit von Zulieferern, Produzenten, Logistik-Dienstleistern und Handel zur Optimierung der gesamten Lieferkette in den Vordergrund. (Beckmann 2004, S. 1)

Die Mehrgliedrigkeit von Logistikketten führt jedoch zu Problemen beim Informationsaustausch, da dieser gewöhnlich nur zwischen benachbarten Gliedern der Logistikkette stattfindet. Diese für jeden Akteur der Logistikkette lokal begrenzte Information führt dazu, dass sich eine kleine Schwankung der Endkonsumentennachfrage auf den vorgelagerten Stufen der Logistikkette überproportional fortsetzt und aufschaukelt. (Heusler 2004, S. 25)

Dieses Phänomen wird auch als Bullwhip-Effekt bezeichnet. Um den Bullwhip-Effekt in Lieferketten möglichst gering zu halten, gibt es verschiedene Möglichkeiten, wie Verkürzung der Reaktions- und Lieferzeiten zwischen den Gliedern der Lieferkette. Außerdem wird die Steuerung des Bestandes des Einzelhändlers direkt durch Lieferanten vorgeschlagen. Vor allem wird aber der Aufbau eines integrierten Informationssystems, das allen Gliedern der Lieferkette Einblick in die Nachfragesituation am Endkonsumentenmarkt bietet, gefordert. (Heusler 2004, S. 26 f.)

In dieser Arbeit wird zunächst näher auf den Bullwhip-Effekt eingegangen, bevor die Auswirkungen eines derartigen Informationssystems auf den Bullwhip-Effekt untersucht werden. Anschließend wird der Aufbau eines Informationssystems zur Widerspiegelung der Situation am Absatzmarkt am konkreten Beispiel der Lieferkette der Firma *Sony DADC Austria AG* vorgestellt.

1.1 Rahmenbedingungen

Aufgrund der Durchführung der Diplomarbeit in Zusammenarbeit mit einem Unternehmen werden in diesem Abschnitt die Rahmenbedingungen erläutert, die sich aus dieser Zusammenarbeit ergeben haben.

1.1.1 Unternehmensbeschreibung

Abbildung 1: Logo der Sony DADC Austria AG

Das Unternehmen *Sony DADC* (*Digital Audio Disc Corporation*) *Austria AG* (im Folgenden kurz *Sony DADC*) wurde 1986 mit den beiden Werken in Anif und Thalgau bei Salzburg gegründet (vgl. Logo Abbildung 1). Neben der Replikation von optischen Datenträgern (Discs) wie CD, DVD und BluRay bietet *Sony DADC* auch Softwarelösungen, etwa für die Online Distribution von Unterhaltungsmedien, an.

Die Werke in Salzburg sowie eine Produktionsstätte in Southwater in Großbritannien produzieren jährlich knapp 700 Millionen Discs bei einem Gesamtumsatz von etwa 500 Millionen Euro (April 2008 - März 2009). Das Unternehmen beschäftigt derzeit etwa 1600 fest angestellte Mitarbeiter.

Das Distributionsnetzwerk in Europa besteht neben den drei Produktionsstätten aus fünf Distributionszentren in Deutschland, Frankreich, Großbritannien, Schweden und Großbritannien. Die Logistikkette besteht daher in den meisten Fällen ohne Betrachtung von Lieferanten aus drei Akteuren. Diese sind Produktion, Distributionszentrum und Einzelhändler.

1.1.2 Aufgabenstellung und Zielsetzung

Der Informationsfluss der in Abschnitt 1.1.1 erläuterten Logistikkette beschränkte sich aktuell auf einzelne, meist größere Bestellungen von Einzelhändlern, die von den Distributionszentren bearbeitet werden. Diese wiederum übermitteln ihre Bedarfe an die Produktionswerke, die auf Auftrag der Distributionszentren tätig werden. Dabei wird jedoch keine Information über die tatsächlichen Absatzzahlen bei den Einzelhändlern übermittelt.

Ziel dieser Diplomarbeit ist daher die Bereitstellung von aggregierter und detaillierter Information zu Verkaufszahlen der von *Sony DADC* produzierten Discs im Einzelhandel um auf Schwankungen bei den Absatzzahlen besser reagieren zu können.

Um diese Anforderung prototypisch umsetzen zu können, sind im Rahmen der Diplomarbeit folgende Teilaufgaben zu lösen:

- Suche nach geeigneten Datenquellen für das Informationssystem sowohl unternehmensintern als auch unternehmensextern
- Entwicklung einer Software-Lösung, die es ermöglicht, die relevanten Daten geeigneter Datenquellen zu homogenisieren und in ein Data Warehouse zu übertragen. Diese Softwarelösung sollte unabhängig von der aktuellen Situation bei *Sony DADC* anwendbar sein. Dies impliziert, dass es möglich sein soll, die erstellte Softwarelösung mit geringem Aufwand an vergleichbare Situationen (etwa in anderen Supply Chains) anzupassen. Außerdem sollte die Software auch das Hinzufügen oder Entfernen von Datenquellen mit geringem Aufwand ermöglichen.
- Die Daten des Data Warehouse sollen in Form von Dashboards mit Übersichtsdiagrammen und Tabellen, sowohl für das operative Management der Fachabteilungen Produktion- und Logistik als auch für die strategische Führungsebene, aufbereitet werden. Dazu ist das Werkzeug *Oracle Business Intelligence Enterprise Edition* (OBIEE) zu verwenden.

1.2 Vorgehensmodell

Als Vorgehensmodell bei der Entwicklung der BI-Lösung wurde ein Prototyping-orientiertes Prozessmodell gewählt. Dies bedeutet, dass bei der Entwicklung Zwischenprodukte, wie die Systemspezifikation oder das Datenmodell, noch nicht vollständig vorliegen müssen, bevor mit der nächsten Phase, also etwa der Implementierung, begonnen werden kann. Die Entscheidung für diese Vorgehensweise wurde getroffen, da bei Beginn der Entwicklung noch keine exakte Spezifikation vorhanden war. Darüber hinaus gab es für die gestellten Anforderungen mehrere Implementierungsmöglichkeiten. Die Entscheidung, welche Realisierungsmöglichkeit am besten geeignet war, konnte jedoch nicht immer auf analytischem Wege, sondern nur auf Experimenten gestützt, getroffen werden.

Beim Prototyping werden drei unterschiedliche Ansätze unterschieden (Floyd 1984):

- **Exploratives Prototyping**: Ziel des explorativen Prototypings ist, von einer zunächst ungenauen Systemspezifikation, durch die Erstellung von Anwendungsbeispielen den Endbenutzer einzubeziehen, um die letztendlich gewünschte Funktionalität zu ermitteln. Hauptzweck des explorativen Prototypings ist also die Problemanalyse und Systemspezifikation.
- **Experimentelles Prototyping:** Ziel des experimentellen Prototypings ist es, zu untersuchen, ob die vorgenommene Teilspezifikation technisch umsetzbar ist. Im Gegensatz zum explorativen Prototyping wird nicht der Endbenutzer sondern der Entwickler einbezogen, um den System- und Komponentenentwurf zu konkretisieren.
- **Evolutionäres Prototyping:** Ziel des evolutionären Prototypen ist eine schrittweise Annäherung an die gestellten Benutzeranforderungen. Man könnte evolutionäres Prototyping auch als Entwicklung in Versionen bezeichnen. Ebenso ist die Grenze zwischen Prototyp und fertigem Produkt unklar. Dieses Vorgehensmodell ist somit auch für Situationen geeignet, in denen sich die Anforderungen häufig ändern. Als wesentlicher Unterschied zu den anderen beiden Prototyping-Modellen ist festzuhalten, dass die Bestandteile des evolutionären Prototyps im Gegensatz zum experimentellen und explorativen Prototypen auch in der Endversion enthalten sind.

In der vorliegenden Arbeit wurde der Ansatz des evolutionären Prototypings herangezogen, da die erstellte Business Intelligence (BI)-Lösung kein Endprodukt darstellt, sondern aufgrund ihrer Erweiterbarkeit immer wieder ausgebaut werden kann. Festzuhalten ist aber, dass im Vorgehensmodell aufgrund der Einbeziehung der Benutzer auch explorative sowie aufgrund des Tests der technischen Realisierbarkeit auch experimentelle Elemente enthalten sind.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in acht Kapitel, die, nachdem die Ziele und Rahmenbedingungen der Arbeit abgeklärt wurden, die theoretischen Grundlagen für die praktische Umsetzung der Problemstellung schaffen. Danach wird der Umsetzungsweg aufgezeigt, um anschließend die Ergebnisse der entwickelten Business Intelligence (BI)-Lösung testen und diskutieren zu können.

Kapitel 1 - Einleitung vermittelt dem Leser einen Überblick über den Grund der Durchführung der Arbeit, sowie über die genaue Aufgabenstellung, die mit der Arbeit gelöst

werden sollte. Außerdem werden die Vorgehensweise und die Rahmenbedingungen, in denen die Umsetzung erfolgte, genauer erläutert.

Kapitel 2 - Grundlagen führt in die wissenschaftlichen Erkenntnisse, auf denen diese Arbeit aufbaut, ein. Besonders wird auf die Erkenntnisse aus dem Supply Chain-Management im Bezug auf den Informationsaustausch in der Supply Chain eingegangen. Darüber hinaus werden die Begriffe Business Intelligence und Data Warehousing abgegrenzt und mit der Aufgabenstellung in Zusammenhang gebracht, um anschließend näher auf die informationstechnischen Voraussetzungen zur Entwicklung eines Data Warehouse eingehen zu können. Weiters werden die Zusammenhänge zwischen den erwähnten wissenschaftlichen Disziplinen aufgezeigt.

Kapitel 3 - Anforderungen und Rahmenbedingungen beschreibt die funktionalen und nicht funktionalen Anforderungen an die Business Intelligence-Lösung sowie die Rahmenbedingungen, die bei der Entwicklung der Business Intelligence-Lösung im Unternehmensumfeld berücksichtigt werden mussten.

Kapitel 4 - Architektur beschreibt, unter Beachtung der Rahmenbedingungen, den Beladungsprozess, vom Zugriff auf die Datenquellen bis zur Integration der Daten in ein Data Warehouse. Die Vorgehensweise stützt sich auf die in den vorhergehenden Kapiteln ausgearbeiteten Erkenntnisse und Anforderungen. Kapitel 4 gibt somit den Rahmen für das im folgenden Abschnitt erläuterte Design wieder.

Kapitel 5 - Design erläutert, basierend auf der in Kapitel 4 beschriebenen Architektur, das Design der entwickelten BI-Lösung sowohl aus Perspektive der Datenbankschemata als auch aus Perspektive der zu entwickelnden Softwarelösung.

Kapitel 6 - Implementierung geht auf die verwendeten Technologien zur Umsetzung des in Kapitel 5 vorgestellten Designs ein. Darüber hinaus werden charakteristische Fragmente der Implementierung vorgestellt, die die technologiespezifische Umsetzung des Designs zeigen.

In *Kapitel 7 - Test* wird die erstellte BI-Lösung exemplarischen Tests unterzogen, um die Funktionstüchtigkeit der BI-Lösung unter Beweis zu stellen. Darüber hinaus werden die Analyseergebnisse auf ihre Wirksamkeit im Hinblick auf die Reduktion des Bullwhip-Effektes untersucht.

Kapitel 8 - Fazit und Ausblick diskutiert die Entwicklung und das Ergebnis der BI-Lösung. Dabei wird sowohl auf den Entwicklungsprozess als auch auf die erzielten Analyseergebnisse eingegangen. Darüber hinaus werden die wichtigsten Erkenntnisse der Arbeit zusammengefasst und es wird ein Ausblick auf weitere Anwendungs- und Erweiterungsmöglichkeiten der BI-Lösung geboten.

2 Grundlagen

In diesem Kapitel werden die wissenschaftlichen Erkenntnisse, auf denen diese Arbeit aufbaut, erläutert. Dazu wird zunächst in Abschnitt 2.1, der Bullwhip-Effekt eingeordnet und aus betriebswirtschaftlicher Sicht begründet. In Abschnitt 2.2 wird der Begriff Informationssystem im Zusammenhang mit dem Supply Chain-Management definiert, bevor in Abschnitt 2.3 Fallstudien angeführt werden, die die Auswirkungen von Informationssystemen auf den Bullwhip-Effekt untersuchen. In Abschnitt 2.4 wird der Begriff „Business Intelligence“ eingeführt um in Abschnitt 2.5 auf die Grundlagen des Data Warehousing einzugehen. Abschnitt 2.6 bezieht sich schließlich auf die Eigenschaften der Datenquellen während Abschnitt 2.7 auf die Phasen des Data Warehousing eingeht.

2.1 Der Bullwhip-Effekt in Supply Chains

Zum Verständnis der Entstehungsgründe und der Auswirkungen des Bullwhip-Effektes ist es zunächst notwendig, das Logistikkonzept von Supply Chains zu erläutern. Daher wird in Abschnitt 2.1.1 der Begriff Supply Chain definiert um in Abschnitt 2.1.2 und 2.1.3 auf die Ziele des Supply Chain-Managements und die unterschiedlichen Supply Chain-Typen eingehen zu können. Abschnitt 2.1.4 geht auf Probleme ein, die sich aus der Zusammenarbeit der Supply Chain-Akteure ergeben. In Abschnitt 2.1.5 wird der Bullwhip-Effekt, basierend auf den Erkenntnissen der vorhergehenden Abschnitte, erläutert. In Abschnitt 2.1.6 werden darauf aufbauend die Ursachen des Bullwhip-Effektes angeführt. In Abschnitt 2.1.7 werden schließlich Maßnahmen vorgeschlagen um den Bullwhip-Effekt in Supply Chains reduzieren zu können.

2.1.1 Abgrenzung des Begriffs Supply Chain

Der Begriff Supply Chain (Versorgungskette) ist eng verwandt mit dem Begriff Logistikkette. Eine Logistikkette bezeichnet die Prozesskette vom Lieferanten bis zum Abnehmer. Eine Logistikkette wird in Akteure und Prozesse unterteilt. In Abbildung 2 ist eine Logistikkette ersichtlich, die aus den drei Akteuren Lieferant, Produzent und Abnehmer besteht. Daneben existiert aus Sicht des Produzenten ein Beschaffungs-, Produktions- und Distributionsprozess. Die einzelnen Akteure treffen in der Logistikkette ihre Entscheidungen gegenüber den

jeweiligen Lieferanten isoliert also ohne Abstimmung mit anderen Akteuren. (Corsten & Gössinger 2001, S. 81 f.)

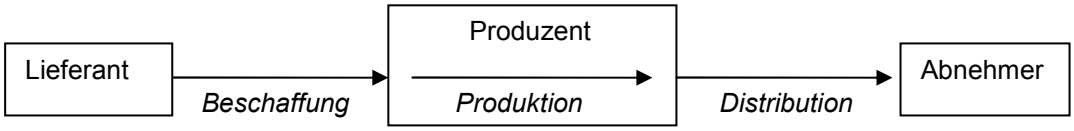


Abbildung 2: Darstellung einer Logistikkette (nach Corsten & Gössinger 2001, S. 82)

Eine Supply Chain betrachtet hingegen über den Beschaffungs- Produktions- und Distributionsprozess hinaus die Zusammenarbeit aller Akteure einer Logistikkette. Das bedeutet die in der Supply Chain getroffenen Entscheidungen werden nicht isoliert, sondern unter Einbeziehung der anderen Akteure der Supply Chain getroffen. Der wesentliche Unterschied liegt daher in der unternehmensübergreifenden Betrachtung der gesamten Wertschöpfungskette, die nicht das lokale Optimum des einzelnen Akteurs sondern das globale Optimum aller Akteure der Supply Chain in den Vordergrund stellt. Diese Betrachtungsweise schließt neben dem Warenfluss auch den Informationsfluss in die Betrachtung ein. (Corsten & Gössinger 2001, S. 84 f.)

2.1.2 Ziele des Supply Chain-Management

Um darzulegen, weshalb die Etablierung eines Supply Chain-Konzeptes im Vergleich zur Sichtweise als reine Logistikkette vorteilhaft sein kann, werden hier die wichtigsten Ziele, die mit dem Supply Chain-Management verfolgt werden, angeführt (Heusler 2004, S. 17 f.):

- **Realisierung von Zeitvorteilen**: Die Einbindung von Lieferanten in den Entwicklungsprozess neuer Produkte und die Einbindung von Abnehmern bei der Distribution von Produkten ermöglicht die Realisierung von Zeitvorteilen. Das bedeutet sowohl die Zeit bis zur Neueinführung eines Produktes auf dem Markt, als auch die Reaktionszeit vom Eingang einer Bestellung bis zur Auslieferung verkürzt sich.
- **Verbesserung der Qualität**: Durch die Entwicklung von unternehmens-übergreifenden Qualitätssicherungskonzepten kann sowohl die Produkt- als auch Prozessqualität verbessert werden.
- **Kostensenkung**: Durch die Reduzierung der Unsicherheiten über die Nachfragemenge, Lieferzeiten und Produktqualität können Sicherheitsbestände

reduziert werden. Außerdem wird eine Senkung der Transaktionskosten aufgrund der verbesserten Zusammenarbeit der Akteure erwartet.

- **Steigerung des Endkonsumentennutzens**: Als Hauptziel des Supply Chain-Managements wird die Steigerung des Endkonsumentennutzens genannt. Die im Mittelpunkt stehende Endkonsumentenorientierung sollte zu höherer Wettbewerbsfähigkeit aufgrund der gesteigerten Endkonsumentenzufriedenheit beitragen. Diese Steigerung der Endkonsumentenzufriedenheit soll vor allem durch verbesserte Lieferzeiten, erhöhte Termintreue und verbesserte Lieferflexibilität erreicht werden.

2.1.3 Supply Chain-Typen

Um den Grad der Kooperation der Supply Chain-Akteure bestimmen zu können, werden vier unterschiedliche Typen von Supply Chains, die nach Intensität der Kommunikation der beteiligten Partner unterteilt sind, unterschieden. Die Unterteilung wird in Abbildung 3 anhand der beiden folgenden Kriterien getroffen (Holweg u. a. 2005):

- **Gemeinsame Endverkaufsdaten**: Dieses Kriterium ist erfüllt, wenn die Absatzzahlen der Einzelhändler und somit die Information über die Endkonsumentennachfrage, allen Akteuren der Supply Chain zur Verfügung steht.
- **Lagerbestand vom Lieferanten verwaltet**: Dieses Kriterium ist erfüllt, wenn der Lieferant selbst die Verwaltung des Lagerbestandes seiner Kunden übernimmt. Das bedeutet der Lieferant entscheidet auf Basis einer Vereinbarung zwischen Lieferant und Abnehmer selbst, zu welchem Zeitpunkt und mit welcher Menge er den Lagerbestand des Abnehmers nachfüllt.

Gemeinsame Endverkaufsdaten?			
	Ja	Typ 1 **Bedarfstransparente Supply Chain**	Typ 3 **Integrierte Supply Chain**
	Nein	Typ 0 **Herkömmliche Supply Chain**	Typ 2 **Vendor Managed Replenishment**
		Nein	Ja
		Lagerbestand vom Lieferanten verwaltet?	

Abbildung 3: Supply Chain-Typen, festgelegt nach dem Grad der Kommunikation der Akteure (nach Holweg u. a. 2005, S. 172)

2.1.3.1 Typ 0 - Herkömmliche Supply Chain

Bei einer Supply Chain vom *Typ 0* spricht man auch von *einer herkömmlichen Supply Chain,* da dies die am häufigsten realisierte Form einer Supply Chain ist (vgl. Abbildung 4). In einer Typ 0-Supply Chain ist die einzige Informationsquelle für die vorgelagerten Akteure der Logistikkette die Bestellung. Eine formalisierte, über die reinen Bestelldaten hinausgehende Zusammenarbeit, findet nicht statt. Eine derartige Supply Chain ist daher auch mit der einfachen Sichtweise der Supply Chain als Lieferkette konform (vgl. 2.1.1).

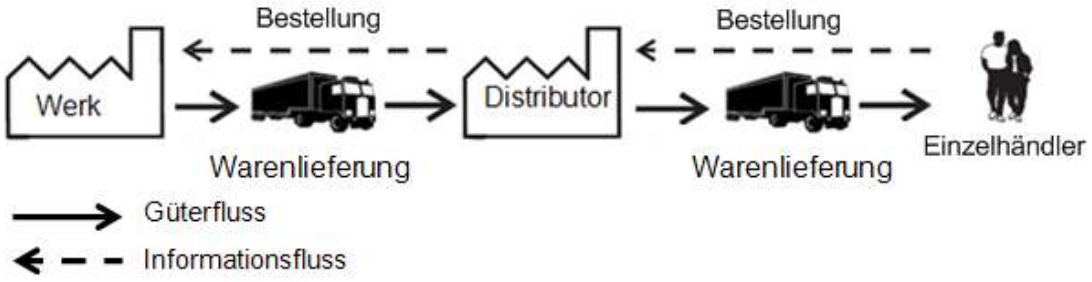


Abbildung 4: Typ 0 - Herkömmliche Supply Chain (nach S Disney 2003, S. 200)

2.1.3.2 Typ 1 - Bedarfstransparente Supply Chain

Von einer *Typ 1*-Supply Chain spricht man wenn, wie bei *Typ 0*, jeder Akteur seinen Bestand und seine Bestellungen unabhängig verwaltet. Es existiert allerdings ein Informationssystem für die gesamte Supply Chain (vgl. Abbildung 5). Somit ist es den einzelnen Akteuren der Supply Chain möglich, die Endkonsumentennachfrage direkt einzusehen und die eigene

Planung darauf abzustimmen. Eine Abstimmung lediglich auf die Kundenbestellungen ist nicht mehr notwendig. Die Implementierung einer *Typ 1*-Supply Chain ist vor allem bei Supply Chains, bei denen die Kooperationsbereitschaft unter den Akteuren nur bedingt gegeben ist, sinnvoll, da noch keine direkten Eingriffe in die Entscheidungsprozesse der Akteure stattfinden.

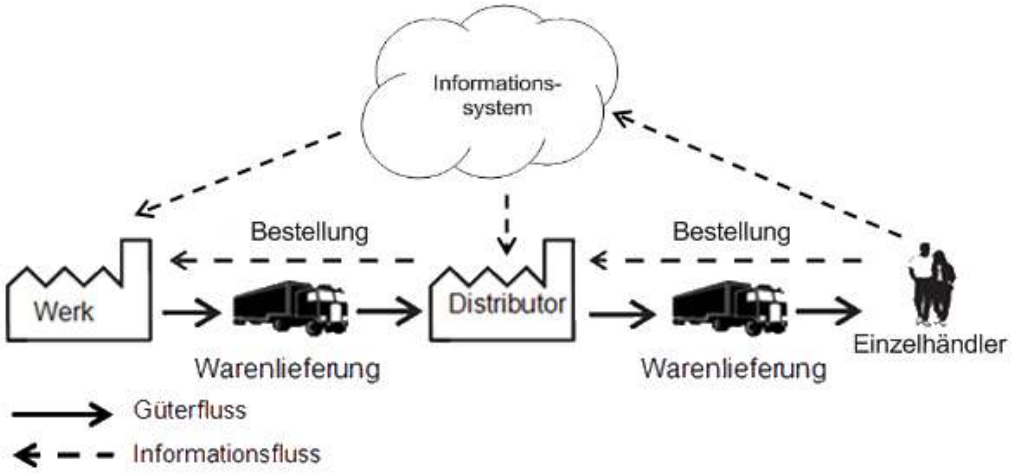


Abbildung 5: Typ 1 - Bedarfstransparente Supply Chain (nach S Disney 2003, S. 200)
Die Information über das Konsumentenverhalten wird über ein Informationssystem allen Akteuren zugänglich gemacht.

2.1.3.3 Typ 2 - Vendor Managed Replenishment

Von einer Supply Chain vom *Typ 2* spricht man wenn der Bestand etwa des Einzelhändlers direkt vom Lieferanten verwaltet wird. Es existiert allerdings kein Informationssystem zur Übermittlung der Endverkaufsdaten wie bei *Typ 1*. Aus diesem Grund entspricht eine *Typ 2*-Supply Chain einer *Typ 0*-Supply Chain mit dem Unterschied, dass die Entscheidung über den Bestand beim Kunden, also etwa einem Einzelhändler, vom Lieferanten getroffen wird.

2.1.3.4 Typ 3 - Integrierte Supply Chain

Eine *Integrierte Supply Chain* vom *Typ 3* (vgl. Abbildung 6) kombiniert den Informationsaustausch innerhalb der *Typ 1*-Supply Chain mit der Übertragung der Verwaltung des Kundenlagerbestandes an den Lieferanten, die für *Typ 2*-Supply Chains charakteristisch ist. Eine derartige Beziehung zwischen zwei Supply Chain-Akteuren wird auch als *Vendor Managed Inventory* bezeichnet. Um zu verdeutlichen, dass in einer

integrierten Supply Chain nicht unbedingt zwischen allen Akteuren ein *Vendor Managed Inventory* eingerichtet sein muss, wurde die Bezeichnung *Integrierte Supply Chain* gewählt.

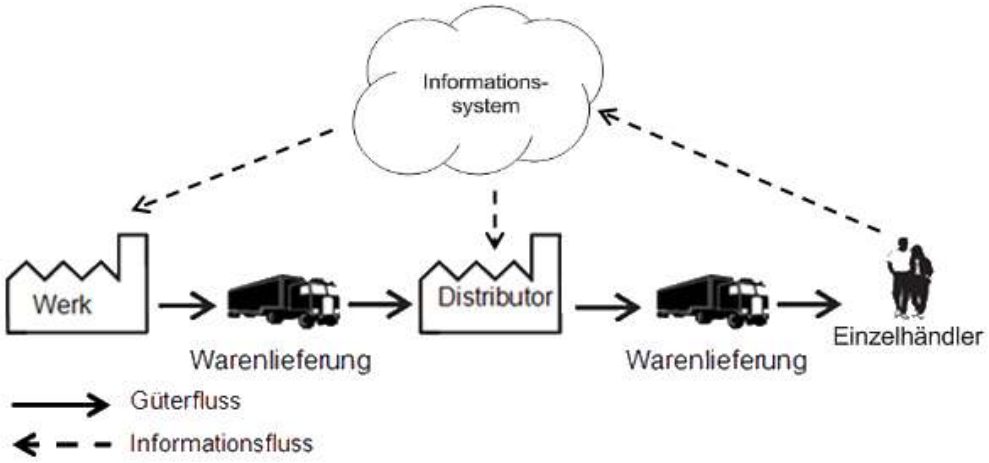


Abbildung 6 : Typ 3 - Integrierte Supply Chain (nach Zäpfel & Wasner 1999, S. 305)
Ein Informationssystem stellt die nötige Information zur Lagerbestandsverwaltung des Kunden bereit; der Kunde selbst tätigt keine Bestellungen mehr.

2.1.4 Gegenläufigkeit lokaler und globaler Ziele

Die ganzheitliche Betrachtungsweise von Supply Chains bringt es mit sich, dass die beteiligten Unternehmen ihre Entscheidungsprozesse an die anderen Akteure anpassen müssen. Dieser Anpassungsprozess ist bereits bei einer Typ 1-Supply Chain erforderlich. In den Vordergrund tritt er jedoch besonders bei einer Typ 3-Supply Chain, da diese eine weitgehende Zusammenarbeit und somit auch weitgehendes gegenseitiges Vertrauen der Supply Chain-Akteure erfordert. Oft kommt es allerdings dazu, dass das globale Optimum der Supply Chain nicht dem lokalen Optimum der einzelnen Supply Chain-Akteure entspricht.

So kommt es zwischen zwei Akteuren einer Supply Chain vor allem bei Typ 3-Supply Chains zu Zielkonflikten, wenn die gleichmäßige Kapazitätsauslastung eines Lieferanten im Widerspruch mit einer kurzen Wiederbeschaffungszeit für den Einzelhändler steht. Dieser Widerspruch kann besonders dann auftreten, wenn der Absatz im Einzelhandel starken Schwankungen im Zeitablauf unterworfen ist. (Kuhn & Hellingrath 2002, S. 16)

Die Gegenläufigkeit von lokalem und globalem Optimum führt zu der Notwendigkeit, Kosteneinsparungen entlang der gesamten Supply Chain so aufzuteilen, dass letztlich alle Akteure davon profitieren. Die Aufteilung der Einsparungen ist deshalb nötig, da einzelne Akteure bei der Teilnahme an einer Supply Chain, wie im vorigen Absatz genannt, ihre lokal

optimalen Prozesse abändern müssen. Da dieses Zurechnungsproblem formal nicht lösbar ist, müssen Maßnahmen in Verhandlungen der Akteure festgelegt werden, um letztendlich eine Situation herzustellen, von der alle Akteure der Supply Chain profitieren. (Corsten & Gössinger 2001, S. 84 f.)

Geeignete Maßnahmen zur Reduzierung von Zielkonflikten zwischen Einzelhändler und vorgelagerten Akteuren der Supply Chain könnten zum Beispiel eine Reduktion der variablen Stückpreise oder eine einfachere Rücknahmepolitik sein. Außerdem kann bei Interesse des Einzelhändlers die Implementierung eines Vendor Managed Inventory (vgl. 2.1.3.4) angedacht werden. Dadurch würde sich der Einzelhändler die Prüfung und Absendung von Bestellungen ersparen. Als dritte mögliche Maßnahme wird die Reduktion der Wiederbeschaffungszeit für vom Einzelhändler bestellte Artikel gesehen. (Lee u. a. 2000, S. 638)

2.1.5 Der Bullwhip-Effekt

Die in Abschnitt 2.1.4 angeführten Zielkonflikte und die damit einhergehende ungenügende Informationsweitergabe in der Supply Chain führen dazu, dass *„die Güterströme nicht in der tatsächlich benötigten Höhe durch die Lieferkette fließen“* (Keller 2004, S. 15). Die tatsächlich benötigte Höhe bezieht sich dabei immer auf die Endkonsumentennachfrage. Die Abweichung der Bestellmengen von der Endkonsumentennachfrage in der Supply Chain wird auch als Bullwhip-Effekt bezeichnet. (Kuhn & Hellingrath 2002, S. 17)

2.1.5.1 Historie

Entdeckt wurde der damals noch nicht mit einem eigenen Namen versehene Effekt von Forrester (1961). Er erkennt in seinem Modell (vgl. Abbildung 8), dass Zeitverzögerungen sowohl beim Güter- als auch beim Informationsfluss berücksichtigt, dass bereits ein unterstellter sprunghafter Anstieg von zehn Prozent der Verkaufsmenge beim Einzelhändler einen 18-prozentigen Anstieg der Bestellungen beim Großhändler zur Folge haben kann. Beim Produktionswerk schließlich beobachtet Forrester einen Anstieg von 51 Prozent im Vergleich zur vorhergegangenen Produktionsmenge. Die Produktionsmenge liegt also um etwa 37 Prozent über der tatsächlich abgesetzten Menge. Erklärt wird dieser Anstieg hier mit der Struktur der Bestellmenge, die, wie in Abschnitt 2.1.6 erläutert, nicht immer die tatsächliche Endkonsumentennachfrage wiedergibt. (Forrester 1961, S. 25)

Der Begriff „Bullwhip-Effekt“ wurde schließlich in den 90er Jahren vom Unternehmen Procter & Gamble geprägt. Das Unternehmen entdeckte, dass die Absatzzahlen für Babywindeln relativ gleichmäßig waren; trotzdem schwankten vor allem die Bestellungen von Procter & Gamble beim eigenen Zulieferer stark. Auch das Unternehmen Hewlett-Packard erkennte ein vergleichbares Problem beim Absatz von Druckern. Hewlett-Packard versuchte, zunächst vor allem durch Steigerung von Produktions- und Lagerkapazitäten zu reagieren. Dies führte allerdings zu hohen Kostensteigerungen durch unausgelastete Produktionsstätten und zu hohen Lagerbeständen. (Lee u. a. 1997, S. 97)

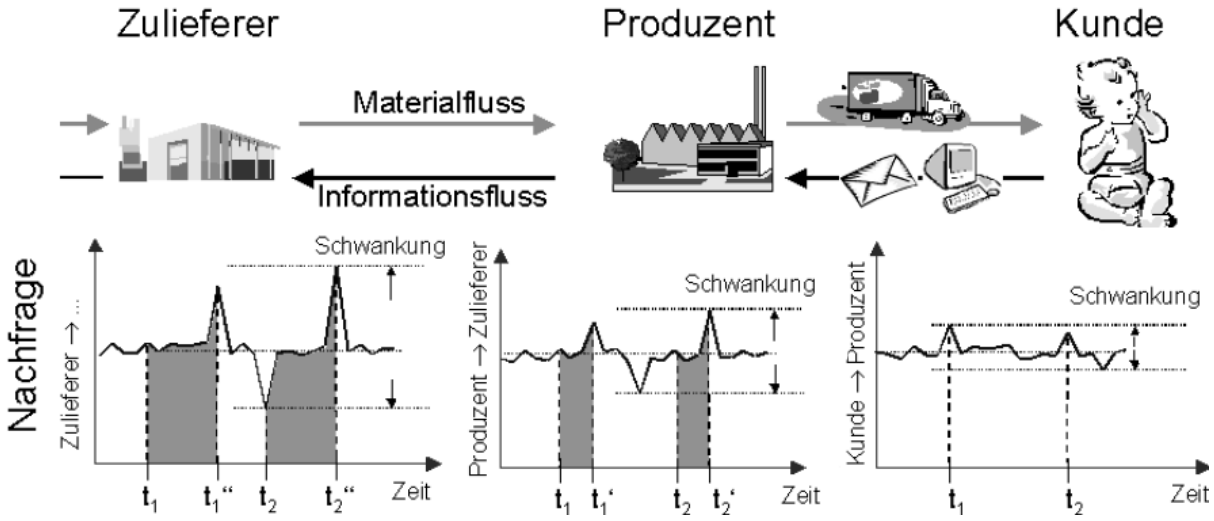


Abbildung 7: Bullwhip-Effekt am Beispiel von Procter & Gamble (Alicke 2005, S. 99)
Entlang der Supply Chain schaukeln sich geringe Änderungen der Endkonsumentennachfrage zeitverzögert auf. Die zunehmende zeitliche Verzögerung der Reaktion von Zulieferer und Produzent wie in Abbildung 8 dargestellt ist anhand der Vergleichszeitpunkte t1 und t2 erkennbar.

2.1.5.2 Zeitverzug

Wie in Abbildung 7 und der Definition ersichtlich, tritt der Bullwhip-Effekt zeitlich verzögert ein. Auf diesen, für den Bullwhip-Effekt charakteristischen Zeitverzug wird hier näher eingegangen.

Um den in Abbildung 7 dargestellten Zeitverzug, der in der Supply Chain entsteht, besser verstehen zu können, wurde das Modell von Corsten & Gössinger (2001) herangezogen, das sich an Forrester (1961, S. 22) anlehnt (vgl. Abbildung 8). Erkennbar ist, dass allein der Informationsfluss vom Endkonsumenten bis zum Produktionswerk im dargestellten Fall zehn Wochen dauert. Zusätzlich ist zu beachten, dass auch der interne Informationsfluss, etwa des Großhändlers, drei Wochen beträgt. Auch wenn die konkreten Zeitangaben für moderne

Logistikketten überholt sind, ist eine erhebliche Zeitverzögerung beim Informations- und Güterfluss über mehrere Akteure nicht zu verhindern.

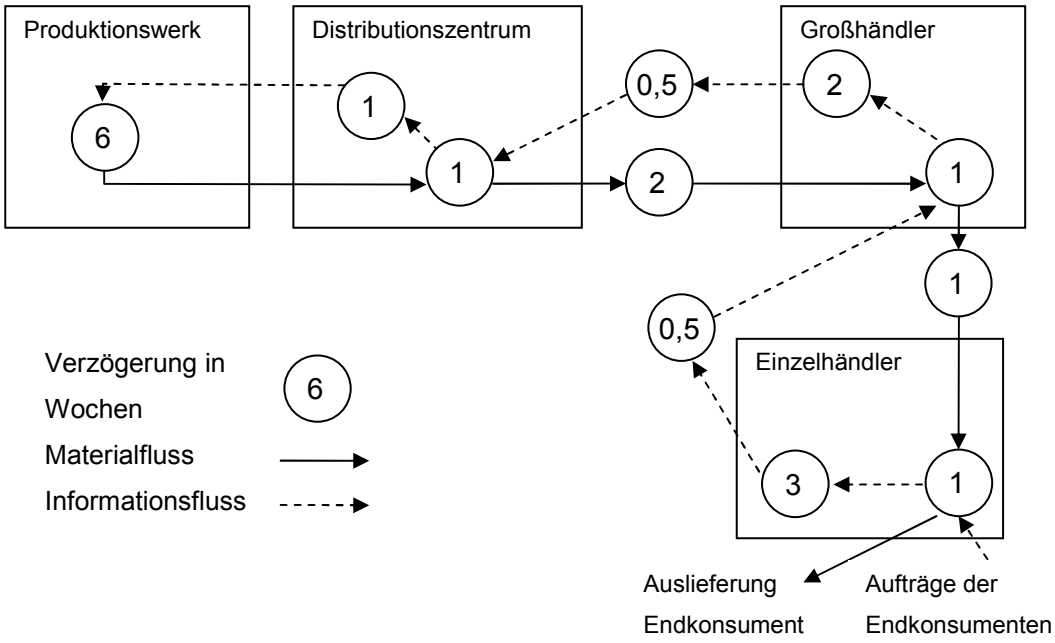


Abbildung 8: Zeitverzögerung des Informations- und Güterfluss in einer viergliedrigen Supply Chain (nach Corsten & Gössinger 2001, S. 88)

Die schrittweise Informationsweitergabe in der Supply Chain mit lokaler Information kann auch veranschaulicht werden, indem man sich eine vor einer roten Ampel wartende Autokolonne vorstellt, in der nur der erste wartende Autofahrer auf die Ampel achtet. Schaltet die Ampel auf Grün, setzt sich zunächst das erste wartende Auto in Bewegung. Das nächste Auto erkennt, dass sich das erste Auto in Bewegung gesetzt hat, und nimmt nun ebenfalls Fahrt auf, erst anschließend das nächste. Die Information, dass die Ampel grün ist, kann verglichen werden mit einer steigenden oder fallenden Endkonsumentennachfrage. Die Information darüber wird aber nur Schritt für Schritt mit entsprechender Zeitverzögerung weitergegeben. (Alicke 2005, S. 103)

2.1.5.3 Auswirkungen

Der Zusammenhang zwischen dem Auftreten des Bullwhip-Effektes und dem damit einhergehenden Zeitverzug ist in Abbildung 9 erkennbar. Die vertikalen Pfeile stellen den Zeitverzug dar, die horizontalen Pfeile den Bullwhip-Effekt. Sowohl der Zeitverzug als auch der Bullwhip-Effekt sind immer am Endkonsumenten orientiert. Die Höhe des Bullwhip-

Effektes ergibt sich daher aus den Abweichungen der Bestellmenge von der Endkonsumentennachfrage.

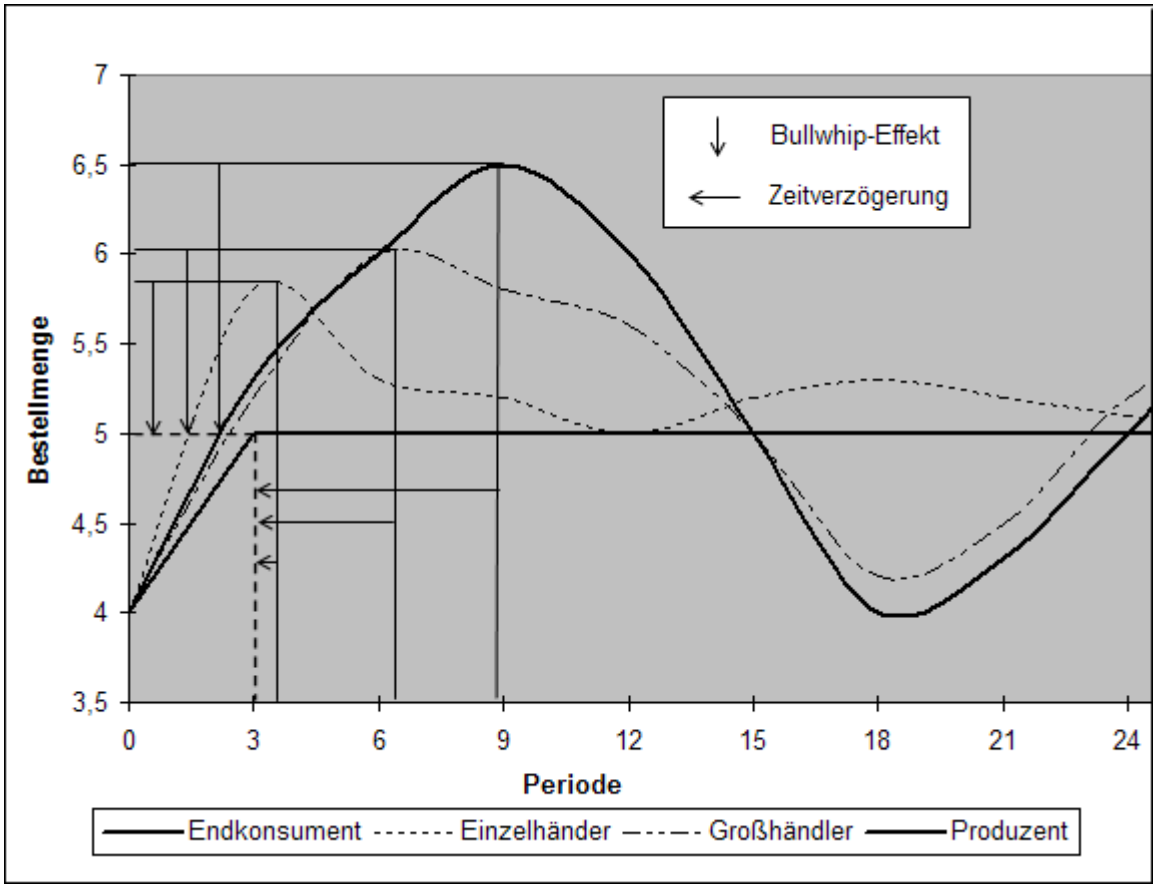


Abbildung 9: Darstellung des Bullwhip-Effektes in einer dreistufigen Supply Chain (nach Keller 2004, S. 33)
Die Endkonsumentennachfrage von Periode null zu Periode drei steigt von 4 auf 5 Einheiten an und bleibt dann konstant. Die eingezeichneten Pfeile stellen den Bullwhip-Effekt oder die Zeitverzögerung dar. Der jeweils kürzeste Pfeil stellt den Bullwhip-Effekt oder die Verzögerung beim Einzelhändler, der längste den Bullwhip-Effekt oder die Verzögerung beim Produzenten dar.

Letztlich führt das Auftreten des Bullwhip-Effektes in der Supply Chain zu überhöhten Beständen und Durchlaufzeiten und somit unnötig hohen Kosten. Darüber hinaus kann der Bullwhip-Effekt auch zu Umsatzverlusten aufgrund von mangelndem Angebot für den Endkonsumenten führen. (Zäpfel & Wasner 1999, S. 309)

Um die in Abschnitt 2.1.2 genannten Ziele des Supply Chain-Managements erreichen zu können, ist es notwendig, den Bullwhip-Effekt zu minimieren. Dazu ist es zunächst unabdingbar, die Ursachen des Bullwhip-Effektes näher zu beleuchten, um anschließend Maßnahmen zur Reduzierung des Bullwhip-Effektes diskutieren zu können.

2.1.6 Ursachen des Bullwhip-Effektes

Im Folgenden werden die vier wesentlichsten Ursachen für den Bullwhip-Effekt angeführt. Zusätzlich werden die Entstehungsgründe für jede Ursache des Bullwhip-Effektes erläutert. Die Zuordnung der Ursachen zu Entstehungsgründen erfolgt im Hinblick auf Abschnitt 2.1.7, in dem Maßnahmen zur Reduzierung des Bullwhip-Effektes, basierend auf den jeweiligen Entstehungsgründen, diskutiert werden. Tabelle 1 bietet eine Übersicht über die Ursachen des Bullwhip-Effektes und deren Entstehungsgründe.

Ursache / **Entstehungsgrund**	Falsche Wahrnehmung	Bündelung	Rabatt-aktionen	Engpasspoker
Lokale Information	✓	✓	✓	
Struktur der Bestellmenge	✓			
Prognosemethode	✓			
Entkoppelung der Nachfrage von der Endkonsumentennachfrage		✓	✓	✓
Entkoppelung der Nachfrage vom eigenen Bedarf			✓	✓
Knappheit des Angebotes				✓

Tabelle 1: Übersicht der Entstehungsgründe der Ursachen des Bullwhip-Effektes (nach Keller 2004, S. 146) [1]

Tabelle 1 zeigt, dass unterschiedliche Ursachen des Bullwhip-Effektes oft auf denselben Entstehungsgründen beruhen. Dies wird besonders beim Entstehungsgrund *Lokale Information* deutlich.

2.1.6.1 Falsche Wahrnehmung der Teilnehmer

Ohne Information über die Endkonsumentennachfrage (vgl. 2.1.3) wird jedes Unternehmen in der Supply Chain versuchen, die Nachfrage der Endkonsumenten mithilfe eines Prognosesystems einzuschätzen. Zur Erstellung eines Prognosesystems kann jedoch nur die jeweils vorhandene Information über die Bestellmengen des nächsten Gliedes der Supply Chain und nicht die tatsächliche Endkonsumentennachfrage herangezogen werden. Da die Bestellmenge, wie im Folgenden erläutert, nicht nur von einer veränderten

[1] Eine Zuordnung (✓) bedeutet, dass der jeweilige Entstehungsgrund für die Ursache vorhanden ist.

Endkonsumentennachfrage abhängt, weichen die tatsächlichen Bestellmengen von der Endkonsumentennachfrage ab. Daraus ergeben sich entlang der Supply Chain erhebliche, zunehmend verstärkte Schwankungen der Bestellmengen (vgl. Abbildung 7).

Folgende drei Entstehungsgründe werden für die falsche Wahrnehmung der Teilnehmer ins Treffen geführt (Keller 2004, S. 157 ff.):

a. ***Lokale Information***: Aufgrund der lokalen Information ist es nicht möglich, Einblick in die Struktur der Bestellmenge zu erhalten. Dies bedeutet, dass das Unternehmen allein aus der Bestellmenge nicht erkennen kann, ob eine veränderte Bestellmenge aufgrund einer Schwankung der Endkonsumentennachfrage eingetreten ist. Die Struktur der Bestellmenge ist dabei der ursächliche Entstehungsgrund, der aufgrund des Vorliegens nur lokaler Information zum Bullwhip-Effekt führt. Punkt b geht daher näher auf die Struktur der Bestellmenge ein.
b. ***Struktur der Bestellmenge***: Gemäß Forrester (1961, S. 23) setzt sich die Bestellmenge eines Unternehmens aus folgenden drei Teilen zusammen:

    - Bestellungen zum Ersatz verkaufter Güter
    - Bestellungen, um den Lagerbestand an die gewünschte Höhe anzupassen
    - Bestellungen, um die Vorlaufzeiten in der Supply Chain einzubeziehen

    Daraus wird deutlich, dass die Bestellmenge nur zu einem Teil die tatsächliche Anzahl an verkauften Gütern widerspiegelt. Jener Teil der Bestellmenge, der nicht der Befriedigung der Nachfrage dient, ist jedoch nicht ohne weitere Information erkennbar. Erhöht sich daher die Bestellmenge eines Akteurs etwa um zehn Prozent, so kann es sein, dass die Erhöhung lediglich der einmaligen Anpassung des Lagerbestandes des Abnehmers dient. Es ist aber auch möglich, dass die erhöhte Menge vollständig dem Ersatz verkaufter Güter dient.

    In der Praxis führt dies dazu, dass ein Großhändler in der Supply Chain, der mit gestiegenen Bestellmengen vom Einzelhandel konfrontiert ist, seine Bestellmengen ebenfalls erhöhen wird. Bei derselben Bestellung wird der Großhändler meist seinen Sicherheitsbestand erhöhen, um auf etwaige zukünftige Schwankungen besser reagieren zu können. Somit ist die Bestellung, die der Großhändler etwa an ein vorgelagertes Distributionszentrum weiterreicht, bereits höher, als der tatsächliche Anstieg der Nachfrage beim Einzelhändler. Das Distributionszentrum kennt ebenfalls

den Anteil der tatsächlich verkauften Güter an der Bestellmenge nicht. Daher nimmt das Distributionszentrum an, dass die gesamte Bestellmenge, die nun auch den gestiegenen Lagerbestand beim Einzelhändler umfasst, auf einer gestiegenen Endkonsumentennachfrage beruht. Dies wiederholt sich auch bei den weiteren Gliedern der Supply Chain, sodass die beim Produktionswerk bestellte Menge aufgrund der angepassten Sicherheitsbestände wesentlich über der tatsächlich gestiegenen Nachfrage liegen wird. (Keller 2004, S. 21 ff.)

c. ***Prognosemethode:*** Eine Prognosemethode dient dazu, zu erkennen, welcher Teil der Bestellmenge tatsächlich aufgrund von höherer Endkonsumentennachfrage entstanden ist. Somit wirkt sich die Qualität dieser Prognosemethode direkt auf den Bullwhip-Effekt aus, da eine wenig zutreffende Prognose die Situation am Endkonsumentenmarkt falsch widerspiegelt.

2.1.6.2 Bündelung von Aufträgen

Von der Bündelung von Bestellmengen spricht man, wenn Aufträge an den Lieferanten aufgrund der Fixkosten, die bei der Tätigung einer Bestellung, sei es durch Papier-, Personal- oder Transportkosten, entstehen, über bestimmte Perioden, etwa Wochen oder Monate, zusammengefasst werden. Erfolgen die Bestellungen beispielsweise immer am Monatsersten, so ergibt sich eine für den Lieferanten verzerrte Nachfragesituation, die eine Einschätzung der tatsächlichen Endkonsumentenbedarfe erschwert. (Alicke 2005, S. 101 f.)

Außerdem führt die Zusammenfassung von Bestellungen zu unnötig hohen Lagerbeständen beim jeweiligen Vorgänger in der Lieferkette, da dieser bestrebt ist, die Nachfrage seines Kunden jederzeit befriedigen zu können. (Keller 2004, S. 171)

Die Zusammenfassung von Bestellungen erklärt damit folgende zwei Entstehungsgründe des Bullwhip-Effektes (Keller 2004, S. 171 ff.):

a. ***Lokale Information***: Da keine zentrale Information über die Endkonsumentennachfrage vorliegt, führt die Zusammenfassung von Bestellungen zu einer Informationsverzerrung bezüglich der Endkonsumentennachfrage und somit zu einer ungleichen Auslastung. Gelingt es, die Ursache *lokale Information* zu beseitigen, so ist die zweite angeführte Ursache nicht mehr für den Bullwhip-Effekt relevant, da

dieser bereits durch die zentrale Verfügbarkeit des Endkonsumentenverhaltens abgefangen ist.

b. ***Entkopplung der Nachfrage von der Endkonsumentennachfrage***: Aufgrund der Fixkosten, die durch eine Bestellung entstehen, neigen die vorgelagerten Akteure der Supply Chain dazu, mehrere Bestellungen zusammenzufassen, um Kosten, die direkt mit einer Bestellung verbunden sind, einzusparen. Dies führt ebenfalls zu einer Informationsverzerrung bezüglich der Endkonsumentennachfrage.

2.1.6.3 Rabatte und Promotionen

Maßnahmen bei Promotionen, wie etwa vorübergehende Preisreduktionen, führen zu einer unnatürlichen Schwankung der Nachfrage, da bei Rabattaktionen der Kunde dazu neigt, mehr Waren zu bestellen, als er benötigt. Dies führt dazu, dass die Nachfrage während der Rabattphase ansteigt und nach Ende der Rabattphase einbricht. Somit entstehen unnatürliche Nachfrageschwankungen.

Folgende Gründe sind bei Rabatten und Promotionen für das Entstehen des Bullwhip-Effektes verantwortlich (Keller 2004, S. 175 ff.):

a. ***Lokale Information***: Liegt in einer Supply Chain nur lokale Information vor, so ist es nicht möglich etwaige Preisschwankungen oder Marketingaktionen zu erkennen. Somit führen Preisschwankungen, ähnlich wie die Bündelung von Bestellungen, zu Informationsverzerrungen bezüglich der tatsächlichen Endkonsumentennachfrage.

b. Entkopplung der Nachfrage von der Endkonsumentennachfrage: Akteure der Logistikkette, die nicht über eine Rabattaktion oder Promotion informiert sind, also Akteure, die nur über lokale Information verfügen, werden vermuten, die Endkonsumentennachfrage werde dauerhaft ansteigen. Daher werden die Akteure, die nur über lokale Information verfügen, ihre Kapazitäten steigern, obwohl dies aufgrund des nur kurzfristigen Anstieges der Nachfrage nicht notwendig wäre. Würde ein Informationssystem in der Supply Chain vorliegen, so wäre die Rabattaktion für einzelne Akteure der Logistikkette bereits anhand des Verlaufs der Endverkaufsdaten erkennbar. Somit würde ein Informationssystem die Entkopplung der Nachfrage der Akteure von der Endkonsumentennachfrage verhindern.

c. Entkoppelung der Nachfrage vom eigenen Bedarf: Niedrigpreisaktionen führen dazu, dass die Akteure der Supply Chain aufgrund des kurzfristig niedrigen Preises

dazu neigen, wesentlich mehr Waren zu bestellen als benötigt werden. Im Gegensatz zur Bündelung von Aufträgen über mehrere Perioden ist hier die Bestellmenge keine bloße Zusammenfassung der Bedarfe der folgenden Perioden. Vielmehr ist die Bestellmenge vollkommen vom eigenen Bedarf entkoppelt, um einen möglichst großen Vorteil aus der Rabattaktion zu ziehen.

2.1.6.4 Engpasspoker

Im Fall eines Engpasses, etwa bei einem Lieferanten, kann es dazu kommen, dass der Lieferant seine tatsächlich ausgelieferte Menge kontingentiert. Das heißt, alle Kunden des Lieferanten erhalten beispielsweise nur 60 Prozent der bestellten Menge. Geht der Kunde davon aus, dass ein derartiger Engpass vorliegt, so vervielfacht der Kunde, der versucht seinen Bedarf unbedingt zu decken, seine Bestellmenge, um letztlich die benötigte Stückzahl zu erhalten. Dies führt zu scheinbar gestiegenen Bedarfen auf Seite des Lieferanten, die tatsächlich nicht existieren. (Alicke 2005, S. 101 f.)

Das Auftreten des Engpasspokers lässt sich auf drei Entstehungsgründe zurückführen:

a. ***Knappheit des Angebots***: Eine von Gliedern der Supply Chain wahrgenommene Angebotsknappheit ist die Grundvoraussetzung für das Entstehen des Engpasspokers als Ursache des Bullwhip-Effektes. Die beiden weiteren Gründe treten daher nur auf, wenn Knappheit des Angebotes besteht.
b. ***Entkoppelung der Nachfrage von der Endkonsumentennachfrage:*** Beim Engpasspoker ist die Endkonsumentennachfrage in den Bestellmengen nicht mehr erkennbar, da sich die Bestellmengen primär am Grad der Knappheit und an dem gewählten Zuteilungsverfahren orientieren.
c. ***Entkoppelung der Nachfrage vom eigenen Bedarf:*** Aufgrund des Engpasspokers werden die eigenen Bedarfe nur mehr als Grundlage für die Bestellmengenberechnung beim Engpasspoker herangezogen. Die tatsächliche Bestellmenge liegt jedoch, wie bei der Entkoppelung der Nachfrage von der Endkonsumentennachfrage, am Grad der Knappheit und am gewählten Zuteilungsverfahren.

   In der Praxis war etwa das Unternehmen Motorola vom Engpasspoker betroffen. In den frühen 90er Jahren trat bei Motorola ein Engpass auf, dem von den Abnehmern mit drastisch überhöhten Bestellmengen entgegengetreten wurde. Dies führte bei Motorola zunächst zu steigenden Umsatzprognosen. Als sich die überhöhten

> Bestellmengen jedoch als Phantombestellungen entpuppten, führte dies zu einem plötzlichen Absturz des Aktienkurses um beinahe zehn Prozent. (Lee u. a. 1997, S. 98)

2.1.7 Maßnahmen zur Verringerung des Bullwhip-Effektes

Um den Bullwhip-Effekt in Supply Chains möglichst gering zu halten, müssen Maßnahmen ergriffen werden, die direkt an die in Abschnitt 2.1.6 genannten Entstehungsgründe des Bullwhip-Effektes ansetzen. Der Entstehungsgrund *Knappheit des Angebotes* ist von den Akteuren in der Supply Chain nicht behebbar und wird daher an dieser Stelle nicht näher behandelt (Keller 2004, S. 179).

2.1.7.1 Lokale Information

Der Entstehungsgrund *lokale Information* nimmt eine Sonderstellung bei den Entstehungsgründen des Bullwhip-Effektes ein, da dieser Entstehungsgrund, wie in Tabelle 1 dargestellt, ein Entstehungsgrund bei allen angeführten Ursachen des Bullwhip-Effektes, abgesehen vom Engpasspoker, ist.

Um Maßnahmen gegen den Entstehungsgrund *lokale Information* ergreifen zu können ist es notwendig, dass Information, die bisher nur lokal verfügbar war, möglichst vielen Akteuren der Supply Chain zur Verfügung gestellt wird. Dazu wird im Rahmen dieser Arbeit ein Informationssystem entworfen, dessen Eigenschaften ab Abschnitt 2.2 näher erläutert werden.

Im Hinblick auf der in Abschnitt 2.1.2 angeführten Endkonsumentenorientierung liegt es nahe, zunächst die Endkonsumentennachfrage im neu geschaffenen Informationssystem verfügbar zu machen. Aufgrund von flächendeckend vorhandenen elektronischen Scannerkassen ist die Erfassung dieser Information für den Einzelhändler einfach möglich. Darüber hinaus ist es möglich, weitere Kennzahlen, wie Prognose-, Lagerbestands und Auftragsdaten an dieses Informationssystem zu binden. (Alicke 2005, S. 110)

Die im Informationssystem zur Verfügung gestellten Informationsarten müssen auf den Supply Chain-Typ und den damit zusammenhängenden Grad der Zusammenarbeit in der Supply Chain abgestimmt werden (vgl. 2.1.3).

Eine erfolgreiche Umsetzung der Maßnahmen gegen *lokale Information* hilft, wie in Abschnitt 2.1.6 dargestellt, auch gegen folgende Ursachen des Bullwhip-Effektes vorzugehen:

- ***Prognosemethode***: Sind im Informationssystem neben zentral vorhandenen Endverkaufsdaten auch Prognoseverfahren, die die Produktions- und Lagerbestandsplanung für alle Glieder der Supply Chain vornehmen, vorhanden, so ist eine individuelle Prognose der Akteure der Supply Chain nicht mehr notwendig, sofern alle Akteure das Informationssystem nutzen. Ein Informationssystem als Maßnahme gegen *lokale Information* kann also ein lokales Prognosesystem ersetzen.
- ***Struktur der Bestellmenge***: Dieser Entstehungsgrund kann durch ein zentrales Informationssystem nur abgeschwächt, nicht aber völlig neutralisiert werden. Endverkaufsdaten helfen zwar jene Anteile an der Bestellmenge zu erkennen, die tatsächlich an die Endkonsumentennachfrage geknüpft sind. Ein Informationssystem alleine ist jedoch nicht in der Lage die Komplexität, die sich im Hinblick auf die in Abschnitt 2.1.6.1 erläuterte Struktur der Bestellmenge ergibt, vollständig zu lösen. Dies bedeutet im Umkehrschluss, dass allein ein Informationssystem den Bullwhip-Effekt nicht gänzlich vermeiden kann. (Keller 2004, S. 167)
- ***Entkoppelung der Nachfrage von der Endkonsumentennachfrage***: Dieser Entstehungsgrund, der sowohl bei Auftragsbündelung (vgl. 2.1.6.2) als auch bei Preisschwankungen (vgl. 2.1.6.3) auftritt, lässt sich durch ein Informationssystem vollkommen beseitigen, da ein Informationssystem mithilfe der Endverkaufsdaten die Endkonsumentennachfrage widerspiegelt. Somit kommt es zu keinen Informationsverzerrungen.

All diese Maßnahmen führen letztlich dazu, dass in der Supply Chain unnötige Lagerbestände abgebaut und Kosten für ungleichmäßige Auslastung abgewendet werden.

Bildlich vergleichen lässt sich eine bedarfstransparente Supply Chain vom Typ 1 auch mit einem Zug, bei dem alle Wagen direkt miteinander verbunden sind. Setzt sich die Lokomotive in Bewegung, so ziehen alle Glieder, im Gegensatz zum in Abschnitt 2.1.6 angeführten Beispiel im Straßenverkehr, ohne Zeitversatz mit. (Alicke 2005, S. 112)

Tabelle 2 zeigt welche Entstehungsgründe des Bullwhip-Effektes mit einem Informationssystem beseitigt werden können:

Ursache / Entstehungsgrund	Falsche Wahrnehmung	Bündelung	Rabatt-aktionen	Engpasspoker
Lokale Information	✓	✓	✓	
Struktur der Bestellmenge	✓			
Prognosemethode	✓			
Entkoppelung der Nachfrage von der Endkonsumentennachfrage		✓	✓	✓
Entkoppelung der Nachfrage vom eigenen Bedarf			✓	✓
Knappheit des Angebotes				✓

Tabelle 2: Ursache und Entstehungsgründe des Bullwhip-Effektes nach Implementierung eines Informationssystems (nach Keller 2004, S. 146)
Der kursiv angeführte Entstehungsgrund *Lokale Information* wurde durch das Informationssystem beseitigt. Die grau dargestellten Entstehungsgründe wurden durch das Informationssystem neutralisiert und sind somit für den Bullwhip-Effekt nicht mehr relevant.

2.1.7.2 Struktur der Bestellmenge

Neben dem Entstehungsgrund *lokale Information* kann auch an den anderen Entstehungsgründen zur Reduktion des Bullwhip-Effektes angesetzt werden. Für den Entstehungsgrund *Struktur der Bestellmenge* stellen sich folgende zwei Maßnahmen dar:

Bestellpolitiken: Mithilfe von Bestellpolitiken, die nicht sofort auf Abweichungen des Ist-Lagerbestandes vom Soll-Lagerbestand reagieren, ist es möglich, einen stabilisierenden Effekt auf die Supply Chain auszuüben. Eine derartige Bestellpolitik ermöglicht somit geringere Schwankungen in der Produktion, da Nachfrageschwankungen weitestgehend durch den Lagerbestand ausgeglichen werden. (Keller 2004, S. 168 ff.)

Beschaffungs- und Lagerhaltungsinformationssystem: Ein derartiges Informationssystem könnte etwa als ERP System implementiert sein. Entscheidend ist, dass das Informationssystem neben dem aktuellen Lagerbestand auch die bereits bestellten Mengen erfasst. Somit werden Doppelbestellungen aufgrund falscher Wahrnehmung der Teilnehmer vermieden.

2.1.7.3 Prognosemethode

Die hier vorgeschlagenen Maßnahmen sind nur wirksam, wenn es sich um eine Supply Chain vom *Typ 0* handelt, das heißt, in der Supply Chain ist kein globales Informationssystem vorhanden.

Wahl einer geeigneten Prognosemethode: Für die Aufstellung eines Prognosesystems werden zahlreiche Daten benötigt. Darunter fallen meist die Bedarfe der vorangegangenen Perioden, sowie, falls vorhanden, die für die vergangenen Perioden errechneten Prognosewerte. Zusätzlich wird meist ein konstanter Glättungsfaktor α verwendet, der vor Durchführung der Prognose festgelegt wird, um kürzlich zurückliegende Bedarfe je nach Absatzverlauf für die Prognose höher oder geringer zu gewichten. Ist der Absatzverlauf meist konstant, so kann ein angepasster α–Wert verhindern, dass ein einmaliger starker Anstieg der Bedarfe die Prognosewerte der nächsten Perioden zu stark beeinflusst (Davis & Heineke 2004, S. 441). Liegen etwa über saisonale Schwankungen der Bestellmenge nur eingeschränkte oder zweifelhafte Daten vor so, sollte ein Prognosesystem gewählt werden, das lediglich mit gesicherten Daten, wie etwa den Bestellmengen der vergangenen Perioden das Auslangen findet, damit die Parameter nicht geschätzt werden müssen.

Verbesserung der Güte der Daten: Die mit der angewendeten Prognosemethode ermittelten Absatzzahlen sollten mit den tatsächlichen Absatzzahlen verglichen werden. Somit werden Rückschlüsse auf die Qualität der Prognosemethode möglich. Dazu eignet sich etwa das *Abweichungssignal nach Trigg und Leach*, das auf der mittleren absoluten Abweichung von tatsächlichem Bedarf und dem davor berechneten Prognosewert basiert (Schönsleben 2007, S. 503 ff.).

2.1.7.4 Entkoppelung der Nachfrage von der Endkonsumentennachfrage

Die hier genannten Maßnahmen sind hinsichtlich des Bullwhip-Effektes bei Implementierung eines Informationssystems nur mehr eingeschränkt notwendig, da die Endkonsumentennachfrage durch das Informationssystem trotz der Entkoppelung der Nachfrage sichtbar bleibt. In allen anderen Fällen sind die angeführten Maßnahmen aber jedenfalls wirksam.

Senkung der bestellfixen Kosten: Um die Bündelung der Bestellmengen zu vermeiden, ist es notwendig, die Fixkosten, die bei jeder Bestellung anfallen, möglichst gering zu halten.

Dies wird zum einen durch ausschließlich elektronische Bestellsysteme ermöglicht, die im Vergleich zu Bestellungen mit Papierdokumenten nur etwa ein Zehntel der Kosten verursachen. Zum anderen wird das Problem der vergleichsweise hohen Transportkosten für geringe Mengen gelöst, indem einzelne Transportmittel, wie Lastkraftwägen, auch unterschiedliche Produktarten transportieren. Dadurch ist es möglich, dass ein einzelner Lastkraftwagen mit unterschiedlichen Produkttypen voll beladen werden kann. (Lee u. a. 1997, S. 100)

Sind die bestellfixen Kosten letztlich geringer als die daraus entstehenden Lagerkosten, so besteht kein Anreiz mehr, Bündelungen durchzuführen.

Offenlegung der Bündelungsstrategie: Hier wird dem jeweiligen Lieferanten die eigene Bündelungsstrategie offengelegt. Somit kann der Lieferant erkennen, dass die bei ihm eingegangenen Bestellungen auf einer Bündelungsstrategie beruhen. Damit ist die Informationsverzerrung zwar aufgehoben, die Nachfrage ist allerdings nach wie vor von der Endkonsumentennachfrage entkoppelt. Aufgrund der Informationsweitergabe ist, ähnlich wie beim Entstehungsgrund *lokale Information,* der Entstehungsgrund *Entkoppelung der Nachfrage von der Endkonsumentennachfrage* neutralisiert. (Keller 2004, S. 173 f.)

2.1.7.5 Entkoppelung der Nachfrage vom eigenen Bedarf

Die *Entkoppelung der Nachfrage vom eigenen Bedarf* in Supply Chains kann auf zwei unterschiedliche Arten bekämpft werden. Eine erfolgreiche *Beseitigung von Preisschwankungen* würde zusätzlich die Maßnahme *Information bezüglich Rabattaktionen* überflüssig machen. Die Implementierung eines Informationssystems ist nicht in der Lage die *Entkoppelung der Nachfrage vom eigenen Bedarf* zu bekämpfen, da hier Bestellmengen von der aktuellen Bedarfssituation völlig entkoppelt sind. Ist die Nachfrage vom eigenen Bedarf aufgrund eines Engpasses entkoppelt, so ist die Beseitigung der Ursache schwieriger als die *Beseitigung von Preisschwankungen*. In diesem Fall kann nur die Anwendung geeigneter Zuteilungsverfahren die Entkoppelung der Nachfrage vom eigenen Bedarf einschränken. (Keller 2004, S. 179).

Beseitigung der Preisschwankungen: Anstatt periodisch Promotions- und Rabattaktionen durchzuführen, sollte der Preis generell gesenkt werden. Eine derartige Dauerniedrigpreisstrategie (Every-Day-Low-Price-Strategy) verringert Absatzschwankungen aufgrund von Niedrigpreisaktionen. Der Entstehungsgrund *Entkoppelung der Nachfrage vom*

eigenen Bedarf ist somit beseitigt und kann nur mehr bei Engpassspekulationen auftreten (Lee u. a. 1997, S. 101).

Information bezüglich Rabattaktionen: Können Preisschwankungen, die durch Sonderangebote verursacht werden, nicht vermieden werden, so stellt diese Maßnahme die Information der Supply Chain-Akteure über das Stattfinden einer Rabattaktion sicher. Im Gegensatz zum Grund *Entkoppelung der Nachfrage von der Endkonsumentennachfrage* wird der Entstehungsgrund *Entkoppelung der Nachfrage vom eigenen Bedarf* durch die *Information bezüglich Rabattaktionen* aber nicht vollständig vermieden, da die Bestellmenge im Fall von Rabattaktionen trotz Information über den Umfang und die Dauer der Rabattaktion kaum abgeschätzt werden kann.

Um bei Rabattaktionen Bestellungen von mehr Einheiten, als tatsächlich nötig, zu vermeiden, ist es notwendig, dass der Lieferant Daten über seinen aktuellen Lagerstand an den Abnehmer übermittelt. Nur so kann verhindert werden, dass Abnehmer spekulative Annahmen über die Kapazitätssituation des Lieferanten treffen, um selbst einen Engpass zu vermeiden. (Lee u. a. 1997, S. 101)

Anwendung von Zuteilungsverfahren: Liegt eine Knappheitssituation vor, ist es sehr wahrscheinlich, dass die Akteure der Supply Chain, die mit einem Engpass konfrontiert sind, zur Engpassspekulation neigen. Das Zuteilungsverfahren des Anbieters muss daher in der Lage sein, die tatsächlich benötigte Bestellmenge möglichst genau zu ermitteln. Auf Seite des Lieferanten ist es sinnvoll, eine Kontingentierung nicht basierend auf der aktuellen, aufgrund des Engpasspokers vermutlich nicht bedarfsgerecht erhöhten Bestellmengen, sondern basierend auf den Bestellmengen von vergangenen Perioden, durchzuführen. Wird die Kontingentierung basierend auf Vergangenheitswerten vorgenommen, kann nicht auf tatsächlich gestiegene Bedarfe eingegangen werden. Dieses Zuteilungsverfahren trifft allerdings den tatsächlichen Bedarf des Abnehmers am ehesten, da es kaum möglich ist, das Ausmaß der Engpassspekulation bei den Teilnehmern einzuschätzen. (Keller 2004, S. 179 ff.)

2.2 Informationssysteme in der Supply Chain

Wie in Abschnitt 2.1 erläutert, gibt es eine Vielzahl von Maßnahmen, die gesetzt werden können, um den Bullwhip-Effekt in der Supply Chain zu reduzieren. Am vielversprechendsten erscheint der Ansatz, Maßnahmen gegen den Entstehungsgrund *lokale*

Information zu setzen, da dieser, wie in Abschnitt 2.1.7.1 erläutert, auch weitere Entstehungsgründe hinsichtlich des Bullwhip-Effektes reduziert oder gänzlich neutralisiert.

Aufgrund der Aufgabenstellung der Diplomarbeit, die vorsieht, eine Business Intelligence-Lösung zu erstellen, wird im Folgenden die Einrichtung eines Informationssystems zur Bekämpfung des Bullwhip-Effektes näher untersucht. Dazu wird in Abschnitt 2.2.1 zunächst definiert was ein Informationssystem leisten soll um in Abschnitt 2.2.2 festzulegen wer an dieses Informationssystem angebunden sein soll. Daraufhin wird in Abschnitt 2.2.3 auf die möglichen Koordinationsformen eines Informationssystems in einer Supply Chain eingegangen bevor in Abschnitt 2.2.4 auf die Informationsarten eines Informationssystems eingegangen wird.

2.2.1 Abgrenzung Informationssystem

"*Zweck eines Informationssystems ist es, Information zu produzieren, mit denen die Informationsnachfrage gedeckt wird*" (Heinrich 2001, S. 177). Ziel eines Informationssystems ist daher die Deckung der Informationsnachfrage für konkrete betriebswirtschaftliche Aufgaben. Ein Informationssystem verwendet als Ausgangsobjekt Daten und transformiert diese zum Endobjekt Information. Diese Transformation geschieht durch das Hinzufügen von Semantik, also dem Sachbezug und Zweck zu den Daten (vgl. Abbildung 10). (Heinrich 2001, S. 177)

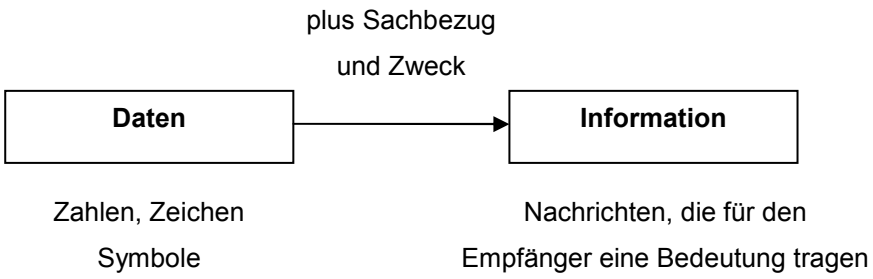


Abbildung 10: Transformationsprozess von Daten zu Information (nach Lutz J. Heinrich 2001, S. 132)

Im vorliegenden Fall eines Informationssystems für das Supply Chain-Management wäre dies etwa der Fall, wenn alle Verkaufszahlen aggregiert zu einer Summe aller Verkaufszahlen dargestellt werden. Ein anderes Beispiel wäre, die Zeit, zu der ein Kundenauftrag im Unternehmen vorliegt, bis zur Auslieferung des Produktes als Lieferzeit darzustellen. Dabei ist zu beachten, dass der Begriff Lieferzeit in verschiedenen Unternehmen unterschiedlich definiert werden kann und damit je nach Sachbezug variiert.

Das in Abschnitt 2.1.7 geforderte Informationssystem ist als überbetriebliches Informationssystem einzuordnen, da es sich um ein Informationssystem handelt, das für mehrere Unternehmen geschaffen und von mehreren Unternehmen benutzt wird (Heinrich 2001, S. 187). Besonders hervorzuheben sind die technologischen Anforderungen, die sich aus unterschiedlichen Datenformaten ergeben.

Die Entwicklung eines derartigen Informationssystems erfordert zunächst die Festlegung der Informationsbeteiligten und der vom Informationssystem zur Verfügung gestellten Informationsarten. Diese wird in den nächsten Abschnitten vorgenommen.

2.2.2 Informationsbeteiligte

Nachdem im vorhergehenden Abschnitt die Ziele und Eigenschaften eines Informationssystems abgegrenzt wurden, ist es nun möglich, darauf aufbauende Konzepte für ein Informationssystem vorzustellen. Dazu ist es zunächst notwendig, den Begriff Informationsbeteiligte zu definieren.

Aufgrund von mangelndem Vertrauen zwischen den Unternehmen und der in Abschnitt 2.1.4 dargestellten Problematik, die besagt, dass nicht jeder Akteur der Supply Chain gleichermaßen von einem Informationssystem profitiert, ist es nicht immer möglich, alle Akteure der Supply Chain zur Beteiligung an einem gemeinsamen Informationssystem zu bewegen. Daher ist die Teilung des Profites, der aus dem Informationssystem entsteht, ein wichtiger Bestandteil als Anreiz zur Gewinnung von Beteiligten am Informationssystem (Corsten & Gössinger 2001, S. 84 f.). Nur, wenn etwa der Einzelhändler finanziell an den Vorteilen eines Informationssystems beteiligt wird, wird er zustimmen, seine Absatzzahlen den anderen Beteiligten des Informationssystems zur Verfügung zu stellen. (Keller 2004, S. 160)

Bei eingeschränktem gegenseitigem Vertrauen der Supply Chain-Akteure ist es auch möglich, dass ein Informationssystem nicht alle Akteure der Supply Chain berücksichtigt. Es besteht daher die Möglichkeit, ein Informationssystem für alle Akteure oder nur für gewisse Akteure der Supply Chain, etwa Produzent und Einzelhändler, einzurichten.

2.2.3 Koordination

Neben der Anzahl der Beteiligten an einem Informationssystem ist der Grad der Zusammenarbeit unter den Informationsbeteiligten für die Koordination eines Informationssystems maßgeblich. Im Folgenden wird daher zwischen dezentraler und zentraler Koordination der Supply Chain unterschieden.

2.2.3.1 Dezentrale Koordination

Bei dezentraler Koordination wird die Endkonsumentennachfrage zentral in einem Informationssystem gesammelt. Im Gegensatz zu einem Informations- und Planungssystem existiert jedoch kein Entscheidungssystem. Darüber hinaus werden die Lagerbestandsdaten der einzelnen Akteure der Supply Chain in einem derartigen Informationssystem nicht erfasst. Das bedeutet, die Entscheidung über die Bestellmenge wird weiterhin dezentral getroffen. Damit ist für ein Informationssystem mit dezentraler Koordination nur ein relativ geringer Grad der Zusammenarbeit erforderlich. Ein dezentral koordiniertes Informationssystem entspricht somit einer *bedarfstransparenten Supply Chain* vom *Typ 1* (vgl. 2.1.3.2). (Zäpfel & Wasner 1999, S. 304 f.)

2.2.3.2 Zentrale Koordination

Zentrale Koordination bedeutet, das Informationssystem so zu implementieren, dass es als zentrale Planungsstelle der Supply Chain dient. Dazu muss das Informationssystem neben den Daten über die Endkonsumentennachfrage auch über die Lagerbestände, Transportzeiten und ausständige Lieferungen von allen Akteuren der Supply Chain verfügen. Mithilfe eines Entscheidungsmodells ist ein derartiges Informationssystem in der Lage, die Bestellmengen für jeden Akteur der Supply Chain selbstständig festzulegen. Ein Informationssystem mit zentraler Koordination erfordert einen hohen Grad der Zusammenarbeit der Informationsbeteiligten. Ein derartiges zentrales Informations- und Planungssystem entspricht einer *integrierten Supply Chain* vom *Typ 3* (vgl. 2.1.3.4) ermöglichen. (Zäpfel & Wasner 1999, S. 304 f.)

2.2.4 Informationsarten

Unter Informationsarten wird eine Kategorisierung jener Informationen verstanden, die in einem Informationssystem verfügbar gemacht werden. So stellt etwa die Information über die

Höhe der Endkonsumentennachfrage eine andere Informationsart, als die Information über die Höhe des Lagerbestandes, dar.

Die im Informationssystem dargestellte Information muss eine Nachricht darstellen, die für die an der Supply Chain beteiligten Unternehmen bisher nicht zugänglich war, jedoch dazu dient, den Bullwhip-Effekt zu reduzieren (Arentzen 2006, S. 171). Im Folgenden wird ein kurzer Abriss über Literatur gegeben, in der der Informationsaustausch in Supply Chains gefordert wird. Dabei werden besonders die geforderten Informationsarten berücksichtigt.

2.3 Auswirkung von Informationssystemen auf den Bullwhip-Effekt

In diesem Abschnitt werden die Auswirkungen von Informationssystemen auf den Bullwhip-Effekt anhand von Fallstudien untersucht. Aufgrund der Erkenntnisse von Abschnitt 2.1.7 ist in allen Fällen eine Reduktion des Bullwhip-Effektes zu erwarten. Die angeführten Fallstudien sollen diese Reduktion beispielhaft aufzeigen um somit die Wirksamkeit des in Abschnitt 2.1.7.1 geforderten Informationssystems zu unterstreichen. Nach der Vorstellung der Fallstudien in den Abschnitten 2.3.1 bis 2.3.5 werden in Abschnitt 2.3.6 die erzielten Ergebnisse zusammengefasst.

2.3.1 Fallstudie von Milling und Größler

Die Fallstudie von Milling und Größler (2001) modelliert eine vierstufige Supply Chain, die aus Einzelhändler, Großhändler, Distributionszentrum und einer Fabrik besteht. Beim Einzelhändler wird eine konstante Nachfrage von 1000 Stück pro Periode, die sprunghaft in Periode eins auf 1100 Stück pro Periode ansteigt und anschließend konstant bei 1100 Stück pro Periode verbleibt, modelliert. Abbildung 11 zeigt die von Milling und Größler (2001) modellierte Supply Chain. Deutlich zu erkennen ist, dass der Informationslauf aufgrund der dezentralen Koordination des Informationssystems nach wie vor direkt zwischen den Akteuren der Supply Chain erfolgt. Zusätzlich dazu wird ein Informationssystem eingeführt, das in Abbildung 11 als Infosicht bezeichnet wird. Diese Sichtweise entspricht einer *bedarfstransparenten Supply Chain* vom *Typ 1* (vgl. 2.1.3.2).

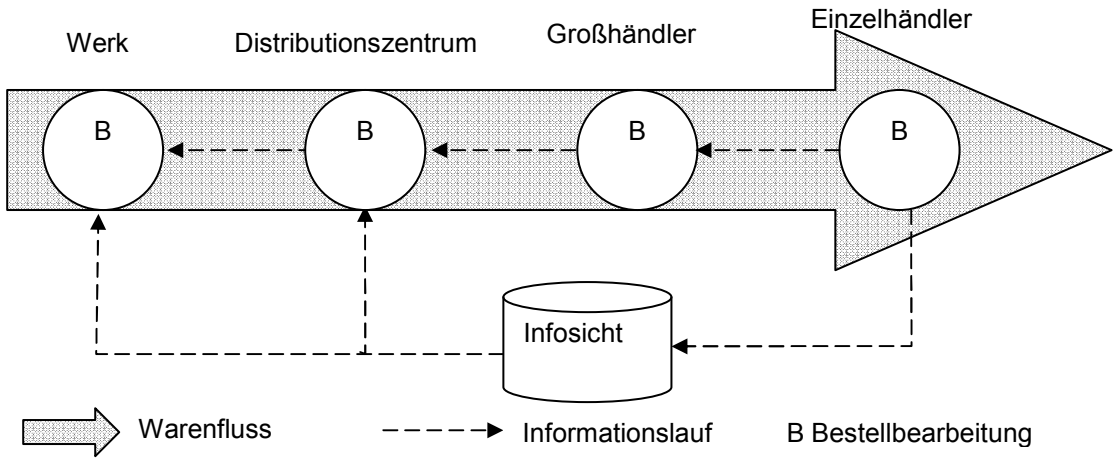


Abbildung 11: Bedarfstransparente Supply Chain mit Informationssystem (nach Milling & Größler 2001, S. 14)

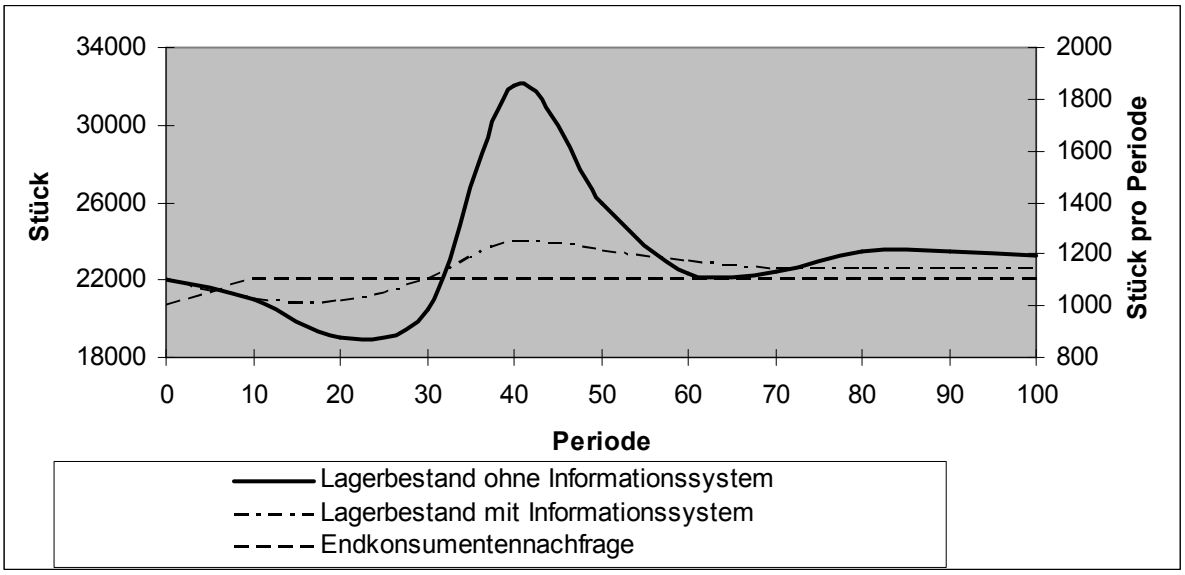


Abbildung 12: Lagerbestand bei allen Akteuren der Supply Chain mit und ohne Informationssystem (nach Milling & Größler 2001, S. 12 ff.)

Die Auswirkungen auf den Lagerbestand in der Supply Chain in diesem Modell zeigt Abbildung 12. Man erkennt, dass ein Informationssystem zu einer erheblichen Verringerung des Bullwhip-Effektes führt. Der verhältnismäßig geringe Anstieg der Nachfrage um 100 Stück zwischen Periode 0 und Periode 10 führt zunächst zu einem fallenden Lagerbestand in der Supply Chain. Aufgrund der überproportional angestiegenen Bestellmengen steigt der Lagerbestand aber bis in Periode 40 auf über 30 000 Stück. Selbst nach 100 Perioden liegt der Lagerbestand überproportional über dem Ausgangslagerbestand. Im Vergleich dazu zeigt sich ein wesentlich verringerter Lagerbestand nach der Einführung eines Informationssystems. Vor

allem in Periode 40 ist der Anstieg des Lagerbestandes im Vergleich zum Anstieg ohne Informationssystem vernachlässigbar niedrig. Auch ab Periode 80 ist erkennbar, dass sich der Lagerbestand bereits wieder an den Ausgangslagerbestand angepasst hat. (Milling & Größler 2001, S. 12 ff.)

2.3.2 Fallstudie von Lee, So und Tang

Die Fallstudie von Lee u. a. (2000) untersucht den Austausch von POS (*Point of Sale*)-Daten in einer zweistufigen Supply Chain, die aus einem Produzenten und einem Händler besteht. Als wesentliche Faktoren zur Bewertung der POS-Daten werden die Art des Nachfrageverlaufes beim Einzelhändler und die Wiederbeschaffungszeit angeführt.

2.3.2.1 Nachfrageschwankungen

Ist die Nachfrage großen Schwankungen unterworfen so erzielt im vorliegenden Modell der Einzelhändler keinen Vorteil aus einem Informationssystem, da keine weiteren Konzepte wie *Vendor Managed Inventory* oder *Vendor Managed Replenishment* (vgl. 2.1.3.4) verfolgt werden. Der Produzent kann jedoch, abhängig vom tatsächlichen Informationsgewinn durch die Absatzzahlen, Einsparungen erzielen. Der Informationsgewinn für den Produzenten ist dann hoch, wenn die Endkonsumentennachfrage starken Schwankungen unterworfen ist (vgl. Abbildung 13). Ist die Endkonsumentennachfrage hingegen konstant, so bringt ein Informationssystem keine Kosteneinsparungen.

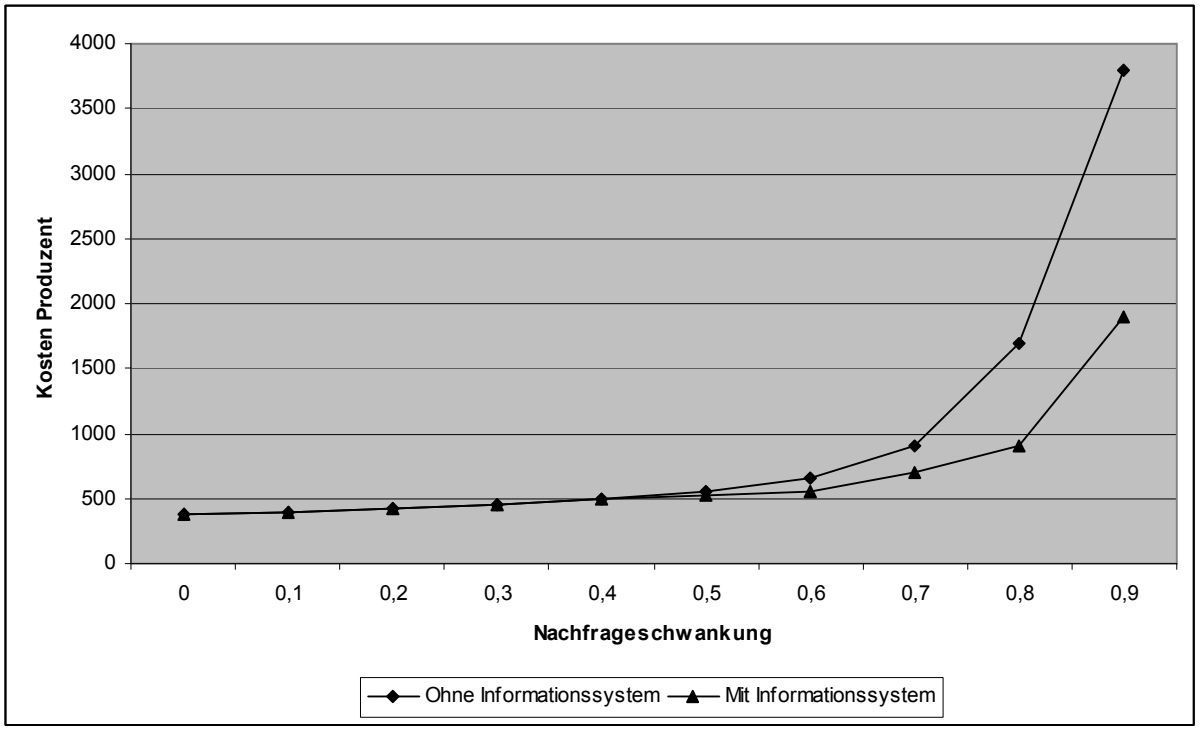


Abbildung 13: Auswirkungen eines Informationssystems auf die Kosten des Produzenten (nach Lee u. a. 2000, S. 636)
Eine Nachfrageschwankung von 0 bedeutet die Nachfrage ist konstant, 1 bedeutet die Nachfrage ist unendlich hohen Schwankungen unterworfen. Es ist zu erkennen, dass ein Informationssystem bei größeren Nachfrageschwankungen eine größere Kostenersparnis ermöglicht.

2.3.2.2 Wiederbeschaffungszeit

Als ebenso relevant wird im Modell die Wiederbeschaffungszeit betrachtet. Ist die Wiederbeschaffungszeit für den Einzelhändler lang, so ist das Einsparungspotenzial, das aus dem Austausch der POS-Daten erzielt wird, höher als bei kurzer Wiederbeschaffungszeit. Eine kurze Wiederbeschaffungszeit nützt vor allem dem Einzelhändler, da in diesem Fall nur ein geringer Sicherheitsbestand beim Einzelhändler notwendig ist.

Im Hinblick auf die in Abschnitt 2.1.4 erwähnte Gegenläufigkeit der Ziele der einzelnen Supply Chain-Partner würde die Realisierung eines Informationssystems Kosteneinsparungen beim Produzenten und die Reduktion der Wiederbeschaffungszeit Kosteneinsparungen beim Lieferanten ermöglichen. Die Kostenersparnis beim Produzenten beträgt im Modell bis zu 45 Prozent. Ein Beispiel für Produkte die hohen Nachfrageschwankungen unterworfen sind, sind Erzeugnisse der Hightech-Industrie. (Lee u. a. 2000)

2.3.3 Fallstudie von Hosoda, Naim, Disney und Potter

Die Fallstudie von Hosoda u. a. (2008) basiert auf einer realen Supply Chain für den Getränkemarkt. In dieser Supply Chain ist ein Informationssystem implementiert, das auch ein Prognosesystem enthält. Außerdem wird das Lager des Einzelhändlers direkt vom Produzenten befüllt. Es handelt sich daher um eine Supply Chain vom *Typ 3*. Zusätzlich wird in der genannten Supply Chain eine *Every-day-low-price-Strategy* verfolgt (vgl. 2.1.7.5), die zur Reduzierung des Bullwhip-Effektes beiträgt. (Hosoda u. a. 2008)

Im vorliegenden Szenario kommt man zu dem Schluss, dass in der vorliegenden Supply Chain ein Informationssystem zu einer Verbesserung der Prognosegenauigkeit von 8 bis 19 Prozent beim zweiten Akteur in der Supply Chain führt. Eine weitere Untersuchung hinsichtlich möglicher Kostenersparnisse wurde nicht durchgeführt. (Hosoda u. a. 2008)

2.3.4 Fallstudie von Zäpfel und Wasner

In der Fallstudie von Zäpfel und Wasner (1999) wird eine Supply Chain mit den drei Akteuren Produzent, Großhändler und Einzelhändler modelliert. Die Endkonsumentennachfrage beträgt im Modell zu Beginn des Beobachtungszeitraumes 4 Stück pro Periode, dementsprechend bestellt und liefert jeder Akteur der Supply Chain genau diese Anzahl. Ähnlich wie bei der *Fallstudie von Milling und Größler* (vgl. 2.3.1) tritt in Periode 1 ein Anstieg der Endkonsumentennachfrage von 4 auf 5 Stück ein. Anschließend stagniert die Nachfrage bis zum Ende des Beobachtungszeitraumes bei 5 Stück. Daneben findet im Modell auch ein als Lagerabweichung bezeichnetes Verhalten Berücksichtigung. Das bedeutet, dass bereits getätigte aber noch nicht im Lager eingetroffene Bestellungen vom jeweiligen Akteur nicht voll berücksichtigt werden (vgl. 2.1.7.2). Weiters wird eine Zeitverzögerung von jeweils 3 Perioden für den Informations- und Materialfluss unterstellt. Es treten Lagerkosten von 1 GE (Geldeinheiten) pro Periode und Verspätungskosten von 3 GE pro Periode auf.

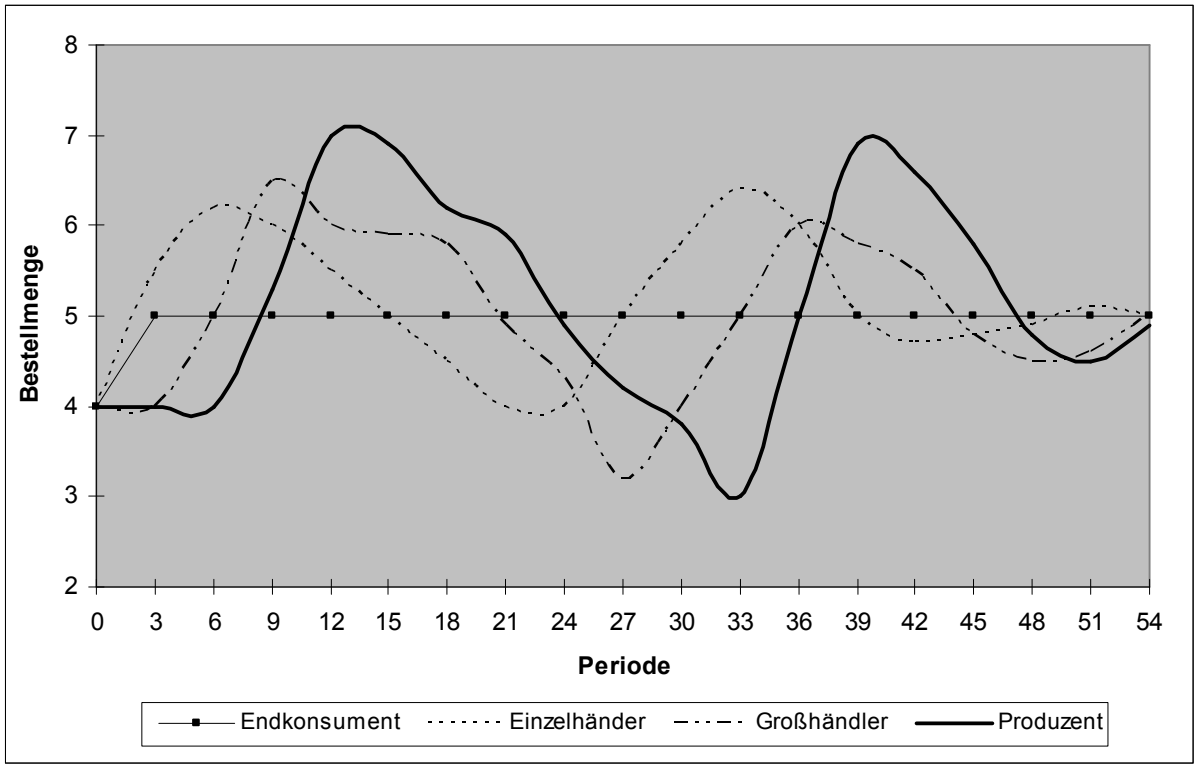


Abbildung 14: Bestellmengen in einer dreistufigen Supply Chain ohne Informationssystem (nach Zäpfel & Wasner 1999, S. 304)[2]

Wie in Abbildung 14 zu erkennen, stellt sich der Bullwhip-Effekt aufgrund der dreiwöchigen Verzögerung im Informationsfluss ohne zentrales Informationssystem beim Großhändler ab Periode 3 und beim Produzent ab Periode 6 ein. Selbst 45 Perioden nach dem Anstieg der Endkonsumentennachfrage sind die Bestellmengen in der Supply Chain noch nicht vollständig stabilisiert. Im untersuchten Szenario ohne Informationssystem entstehen Gesamtkosten von 758 GE.

[2] Da die Linie des Bedarfsverlaufes des Einzelhändlers im Original nicht mehr erkennbar ist, wurde ihr Verlauf gemäß den Annahmen im Modell nachempfunden.

2.3.4.1 Auswirkung einer informationstransparenten Supply Chain

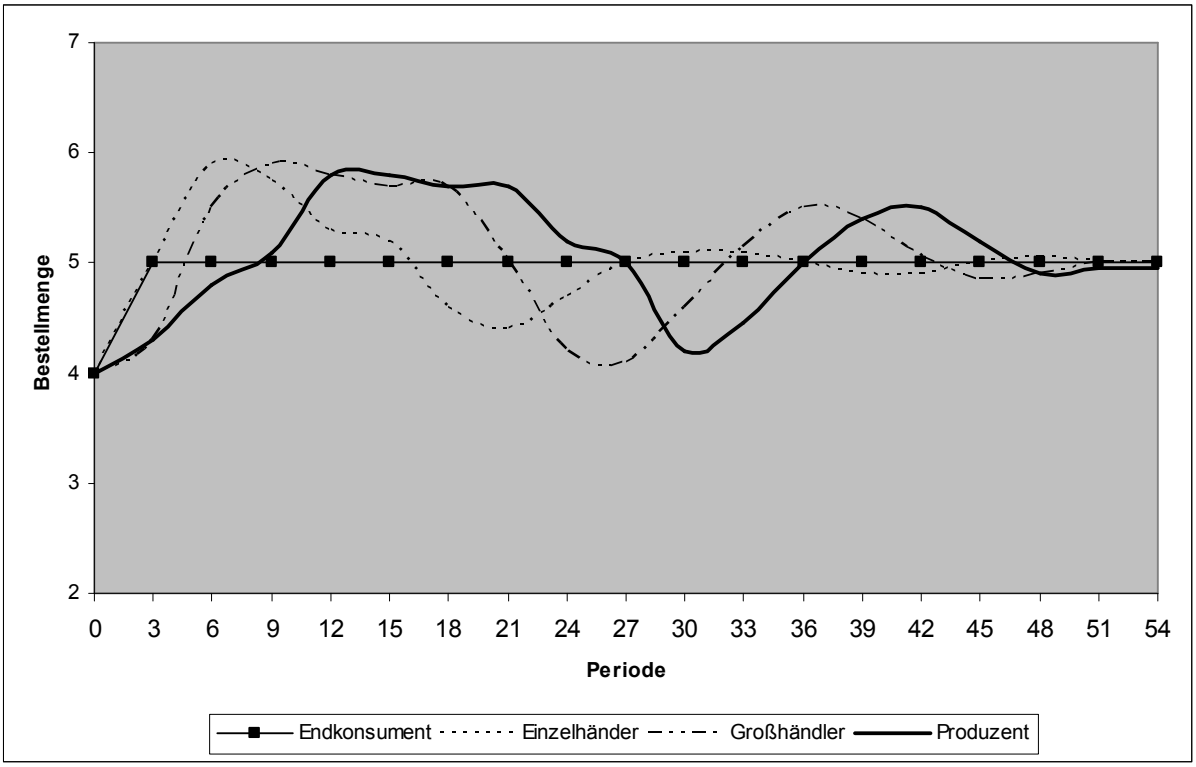


Abbildung 15: Bestellmengen in einer dreistufigen Supply Chain mit Informationssystem ohne zentrale Koordination (nach Zäpfel & Wasner 1999, S. 304)

In Abbildung 15 wird das gleiche Modell wie in Abbildung 14 unterstellt, nun steht allerdings allen Akteuren der Supply Chain ein Informationssystem zur Verfügung, das über POS-Daten verfügt. Es ist erkennbar, dass im Vergleich zur Supply Chain ohne Informationssystem sowohl die Amplitude der Bestellmengenschwankung als auch die Reaktionszeit auf die Veränderung der Endkonsumentennachfrage wesentlich geringer ist. Auch das Einschwingverhalten stellt sich früher als im Modell ohne Informationssystem ein, da es ab Woche 45 zu keinem wesentlichen Über- oder Unterschwingen der Bestellmengen mehr kommt. Dies schlägt sich auch in den Kosten nieder, die im Vergleich zum Modell ohne Informationssystem um etwa 13 Prozent auf 652 GE sinken.

2.3.4.2 Auswirkung einer integrierten Supply Chain

Zusätzlich wurde in der *Fallstudie von Zäpfel und Wasner* auch ein Informationssystem mit einem zentralen Planungssystem modelliert. Das bedeutet, die Akteure der Supply Chain tätigen ihre Bestellungen nicht mehr selbst, da alle Bestellungen direkt von einem Planungssystem zentral koordiniert werden. Das nun angenommene Szenario stellt eine

Integrierte Supply Chain dar (vgl. Abbildung 6). Zu beachten ist, dass der Warenfluss ausschließlich zwischen den Akteuren der Supply Chain stattfindet. Der Informationsaustausch wird über die zentrale Planstelle abgewickelt, wie auch in Abbildung 16 ersichtlich ist. Die zentrale Planstelle geht von einem Soll-Lagerbestand beim Einzelhändler von 6 Stück pro Periode aus.

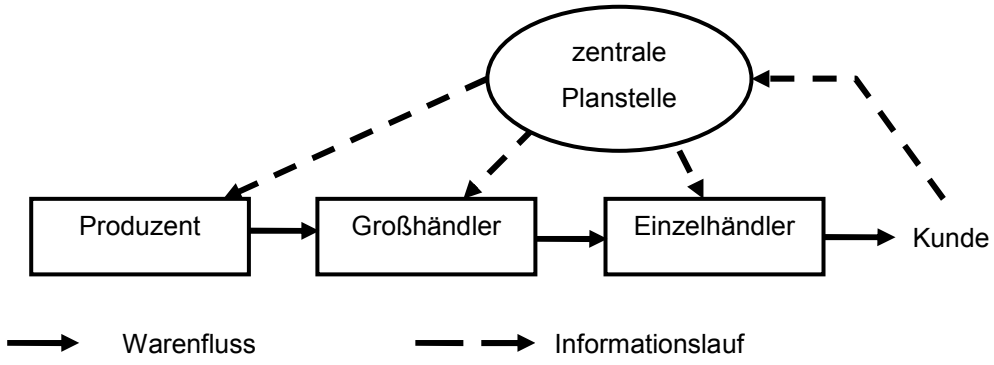


Abbildung 16: Zentrale koordiniertes Informationssystem (nach Zäpfel & Wasner 1999, S. 305)

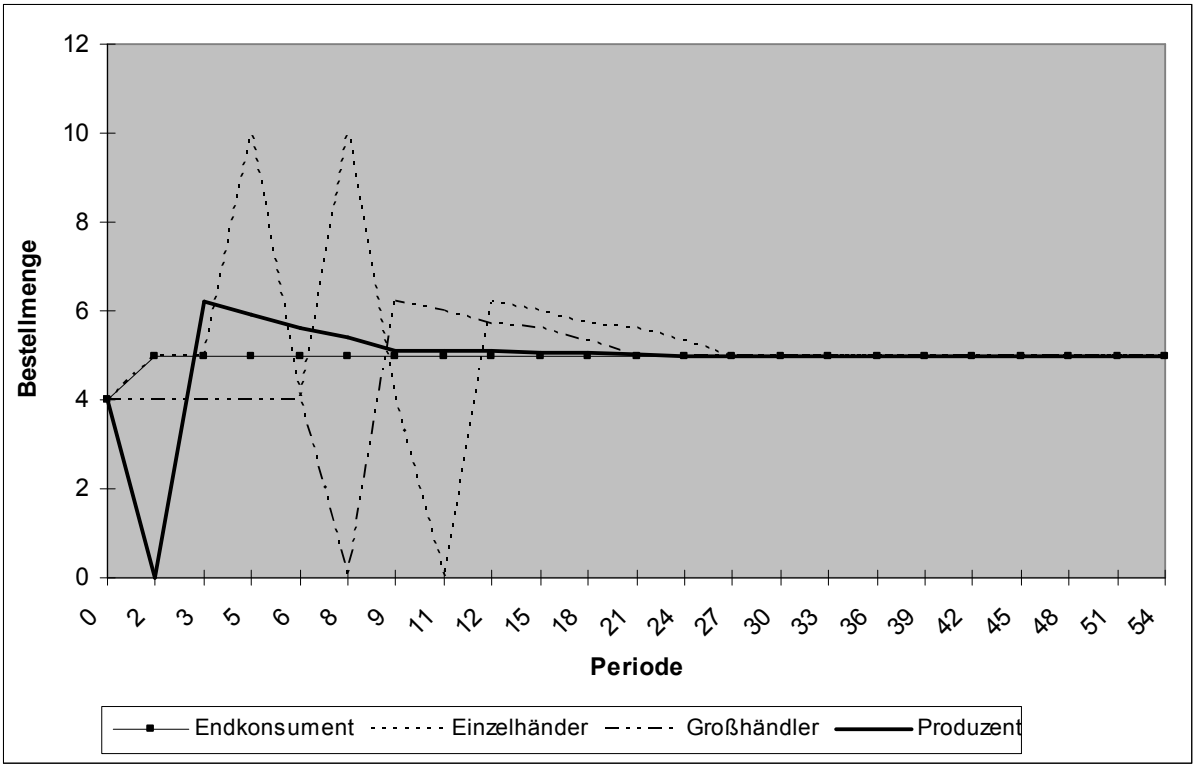


Abbildung 17: Bestellmengen in einer dreistufigen Supply Chain mit zentraler Koordination (nach Zäpfel & Wasner 1999, S. 307)

Aufgrund des zentralen Planungssystems ist es nun möglich, dass sowohl Produzent als auch Großhändler auf einen Sicherheitsbestand verzichten. Dieser ist nur mehr beim Einzelhändler vorhanden, um Schwankungen der Endkonsumentennachfrage absorbieren zu können. Die hohen Ausschläge zu Beginn des Beobachtungszeitraumes in Abbildung 17 sind damit zu erklären, dass die Lagerbestände von Produzent und Großhändler sogleich auf 0 reduziert werden. Die Ausschläge nach oben bei den Bestellmengen des Einzelhändlers erklären sich mit dem Entscheidungsmodell des Planungssystems, das sofort veranlasst, dass alle übrigen Mengen zum Einzelhändler gelangen müssen.

Zu beachten ist auch die Zeitverzögerung von 3 Einheiten zwischen den Akteuren. Diese lässt sich besonders gut an dem Verhältnis der Bestellmengen zwischen Einzelhändler und Großhändler erkennen, deren Kurven unter Beachtung der Verzögerung von 3 Zeiteinheiten weitgehend übereinstimmen. Ebenso ist die Verbindung zum Produzenten erkennbar, die aufgrund der Transport- und Aufenthaltszeit beim Großhändler zu einer Zeitverzögerung von insgesamt 9 Zeiteinheiten führt. Zu bemerken ist also, dass in diesem System die Nachfrage eines Akteurs immer dem Abgang des vorgelagerten Akteurs unter Berücksichtigung der Transportzeit von drei Zeiteinheiten entspricht.

In diesem Modell, mit einem Informationssystem und zentraler Koordination, ergeben sich Gesamtkosten von 315 Geldeinheiten. Daraus resultiert eine mögliche Gesamtersparnis von etwa 50 Prozent gegenüber einer *informationstransparenten Supply Chain* vom *Typ 1* (vgl. 2.1.3.2) und eine Ersparnis von etwa 60 Prozent gegenüber einer *herkömmlichen Supply Chain* vom *Typ 0* (vgl. 2.1.3.1). (Zäpfel & Wasner 1999, S. 304 ff.)

2.3.5 Fallstudie von Keller

Die Fallstudie von Keller (2004) basiert auf der *Fallstudie von Zäpfel und Wasner*. Die Berechnung der entstehenden Gesamtkosten wurde jedoch modifiziert. Außerdem wird bei Existenz eines Informationssystems eine Zeitverzögerung des Informationsflusses über die Absatzdaten unterstellt. Daher sind die erzielten Ergebnisse der *Fallstudie von Keller* nicht direkt mit den Ergebnissen der *Fallstudie von Zäpfel und Wasner* vergleichbar.

Zusätzlich wird neben der Auswirkung von zentraler und dezentraler Koordination auch die Auswirkung von eingeschränkter Informationsbeteiligung untersucht. Außerdem werden die Auswirkungen eines Prognoseverfahrens, das sowohl aktuelle Absatzzahlen beim

Einzelhändler als auch vergangene Absatzzahlen berücksichtigt, untersucht. Dabei ergeben sich beim Einsatz der verschiedenen Arten von Informationssystemen folgende Ergebnisse.

2.3.5.1 Auswirkung von Informationsbeteiligung

Im modellierten Szenario wurden die POS-Daten allen Akteuren der Supply Chain, außer dem Produzenten, zugänglich gemacht. Es zeigt sich, dass ein Informationssystem, das nicht für alle Akteure der Supply Chain zugänglich ist, in der Lage ist, den Bullwhip-Effekt sowie die Gesamtkosten um das Fünf- bis Zehnfache zu reduzieren. (Keller 2004, S. 342 ff.)

Wird nun auch der Produzent in das Informationssystem einbezogen, steigt der maximale Bullwhip-Effekt aufgrund des im Modell unterstellten verzögerten Informationsflusses zum Produzenten geringfügig an. Die Gesamtkosten sinken allerdings um weitere 30 Prozent, da Kosten für unnötige Transporte unter den Supply Chain-Akteuren vermieden werden. (Keller 2004, S. 343 ff.)

2.3.5.2 Auswirkung von Prognoseverfahren

Die *Fallstudie von Keller* untersucht darüber hinaus, wie sich die Existenz eines Prognosesystems auf den Bullwhip-Effekt auswirkt. Das bedeutet, im Modell ist es sowohl möglich, dass die aktuellen Bedarfe allein aus der aktuellen Endkonsumentennachfrage ermittelt werden, als auch, dass ein Prognosesystem herangezogen wird, das die Endkonsumentennachfrage der vergangenen Perioden einbezieht.

Es zeigt sich, dass der Einsatz von Prognoseverfahren nur in Supply Chains mit allgemein geringer Informationssicherheit über die Endkonsumentennachfrage zur Reduzierung des Bullwhip-Effektes geeignet ist. Eine Supply Chain mit geringer Informationssicherheit wird meist dezentral koordiniert, außerdem wird Information über die Endkonsumentennachfrage nicht allen Akteuren verfügbar gemacht. In diesem Fall führt ein Prognosesystem mit hoher Gewichtung der Prognosedaten und geringer Gewichtung der unsicheren Information über die Endkonsumentennachfrage zu einer Reduzierung des Bullwhip-Effektes.

Umgekehrt ist bei Supply Chains mit sicherer Information über die Endkonsumentennachfrage ein Prognosesystem zur Reduktion des Bullwhip-Effektes nur bedingt sinnvoll. Eine derartige Supply Chain mit sicherer Information ist durch zentrale Koordination und Informationsbeteiligung aller Akteure gekennzeichnet. Eine reduzierte

Gewichtung der Bestellmengen vergangener Perioden und eine erhöhte Gewichtung der Absatzdaten aus dem Informationssystem führt in einer Supply Chain mit sicherer Information über die Endkonsumentennachfrage zu einem geringeren Bullwhip-Effekt, da in diesem Fall die sichere Information über die aktuelle Endkonsumentennachfrage besser zur Vorhersage geeignet ist als Prognosewerte. (Keller 2004, S. 460 f.)

2.3.6 Zusammenfassung der Fallstudien

Zusammenfassend ist festzustellen, dass alle Modelle eine Reduktion des Bullwhip-Effektes bei geteilter Information über die Endkonsumentennachfrage zeigen. Je nach Typ der Supply Chain und dem Grad der Kooperation in der Supply Chain können die positiven Auswirkungen auf den Bullwhip-Effekt erheblich abweichen (vgl. Tabelle 3).

Fallstudie von	**Informations-beteiligte**		**Koordination**		**Informationsarten**		**Festgestellte Auswirkung**
	einige	alle	dezentral	zentral	POS	Prognose	
Milling & Größler	✓				✓		max. Lagerbestand minus 30 %
Lee u.a.		✓[3]	✓		✓		Kostensenkung bis zu 45 %
Hosoda u.a.		✓	✓		✓	✓	Prognosegenauigkeit um 8 bis 19 % gesteigert
Zäpfel & Wasner 2.3.4.1		✓	✓		✓		Kostensenkung 13 %
Zäpfel & Wasner 2.3.4.2		✓		✓	✓	✓	Kostensenkung 60 %
Keller 2.3.5.1 - Var. 1	✓		✓		✓		Kostensenkung um das 5-10fache
Keller 2.3.5.1 - Var. 2		✓	✓		✓		Weitere 30 % Kosten-senkung zu 2.3.5.1 Var. 1

Tabelle 3: Übersicht über die Ergebnisse der Fallstudien.
Eine Zuordnung (✓) bedeutet, dass das untersuchte Informationssystem die jeweilige Eigenschaft aufweist.

[3] Die unterstellte Supply Chain besteht nur aus zwei Gliedern, daher sind alle Glieder informationsbeteiligt.

2.4 Business Intelligence im Supply Chain-Management

Im vorhergehenden Abschnitt wurde anhand ausgewählter Fallstudien dargelegt, dass Informationssysteme in vielen Fällen in der Lage sind, den Bullwhip-Effekt in Supply Chains zu reduzieren. Ziel dieses Abschnitts ist es, zu erläutern, warum Business Intelligence den Anforderungen eines Informationssystems gerecht wird. Dazu wird in Abschnitt 2.4.1 die Bedeutung des Begriffs Business Intelligence näher erläutert. In Abschnitt 2.4.2 erfolgt die Einordnung als Informationssystem um in Abschnitt 2.4.3 auf die Architektur, und in Abschnitt 2.4.4 auf mögliche Datenquellen von Business Intelligence-Systemen einzugehen.

2.4.1 Definition Business Intelligence

Der Begriff „Intelligence“ ist im Zusammenhang mit Business Intelligence nicht mit “Intelligenz” im herkömmlichen Sinne, also der menschlichen Fähigkeit abstrakte Beziehungen zu erfassen und sie an neuartige Situationen anzupassen, zu übersetzen. Vielmehr bedeutet der Begriff „Intelligence“ im betriebswirtschaftlichen Kontext eine gezielte Auswahl und Sammlung von Daten und deren Transformation in Information durch die Verknüpfung mit Sachbezug und Zweck. (Mertens 2002, S. 2 f.)

Angelehnt an diese Grunddefinition kann der Begriff Business Intelligence auch noch erweitert werden und als

- Fortsetzung der Daten- und Informationsverarbeitung für die Unternehmensleitung,
- Filter für die Informationsflut,
- besonders schnelles und flexibles Auswertungssystem oder
- Frühwarnsystem

gesehen werden.

2.4.2 Business Intelligence als analytisches Informationssystem

Die Definition des Begriffes Business Intelligence ist eng an die Definition eines Informationssystems angelehnt. Business Intelligence steht daher als ein Synonym für ein analytisches Informationssystem, bei dem die Unterstützung für Fach- und Führungskräfte des Unternehmens im Vordergrund steht. Business Intelligence ergänzt somit die operativen Informationssysteme. Dieser Zusammenhang wird in Abbildung 18 dargestellt.

Business Intelligence basiert auf den operativen Informationssystemen. Operative Informationssysteme sind transaktionsorientiert und bilden die Leistungsprozesse eines Unternehmens ab.

Business Intelligence sammelt somit Daten aus operativen Informationssystemen und stellt diese Information für Fach- und Führungskräfte zu Analysezwecken zur Verfügung. Dabei kann die vom Business Intelligence-Systemen zur Verfügung gestellte Information nicht nur isoliert auf spezielle Unternehmensbereiche, sondern auch unternehmensweit oder unternehmensübergreifend betrachtet werden. (Chamoni & Gluchowski 2006, S. 10 f.)

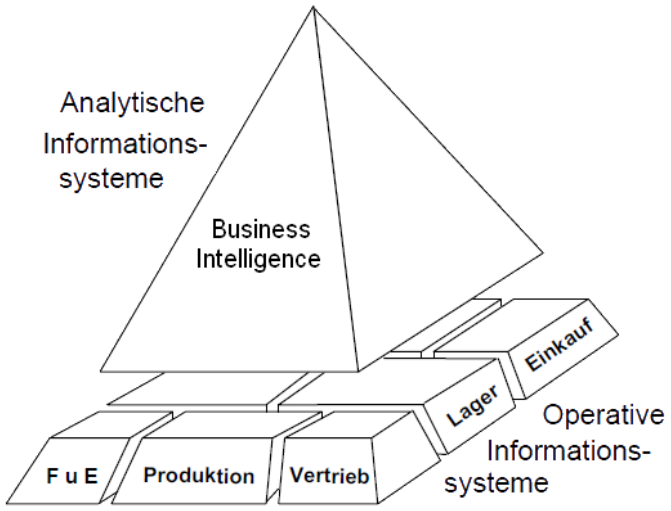


Abbildung 18: Business Intelligence als analytisches Informationssystem (nach Chamoni & Gluchowski 2006, S. 11)

2.4.3 Business Intelligence-Architektur

Der Aufbau eines Business Intelligence-Systems ist in drei Teile gegliedert. Den Grundstein der Business Intelligence-Architektur stellen die Datenquellen dar, die aus operativen Systemen gewonnen werden. Anschließend werden die für das BI-System relevanten Daten aus den Datenquellen extrahiert und in ein Data Warehouse-Schema transformiert. Im Folgenden werden die Schichten von BI-Systemen erläutert und in Abbildung 19 dargestellt.

1. **Datenbereitstellung**: Diese Schicht stellt die Grundlage für ein BI-System dar. Die Daten können aus vielen unterschiedlichen Systemen und Datenstrukturen stammen, die jeweils eine eigene Datenquelle darstellen. In Abbildung 19 sind hier unterschiedliche operative Informationssysteme als Datenquellen angeführt. Für die Datenbereitstellung wird schließlich im Regelfall ein Data Warehouse verwendet, da

dies die Anforderung von BI-Systemen in Bezug auf konsistente, aggregierte und dauerhafte Datenspeicherung erfüllt. (Kemper u. a. 2004, S. 10 f.)

2. **Informationsgenerierung**: Diese Systeme wandeln die in ein Data Warehouse zusammengefassten Daten in Information um (vgl. 2.2). Dadurch erfüllt ein BI-System die Anforderungen, die an ein Informationssystem gestellt werden. Die Informationsgewinnung kann etwa durch die Berechnung von im Unternehmen definierten Kennzahlen, wie etwa dem Zeitraum vom Eingang einer Bestellung bis zur Auslieferung der Ware, erfolgen.
3. **Portal:** Das Portal dient schließlich der Darstellung der in der vorgehenden Schicht generierten Information. Portale bieten die Möglichkeit, Information nicht nur in Tabellenform sondern auch in Diagrammform betrachten zu können. Zusätzlich bieten derartige Portale meist Möglichkeiten, die Informationsdarstellung an den jeweiligen Benutzer und dessen Fachbereich anzupassen. (Kemper u. a. 2004, S. 11)

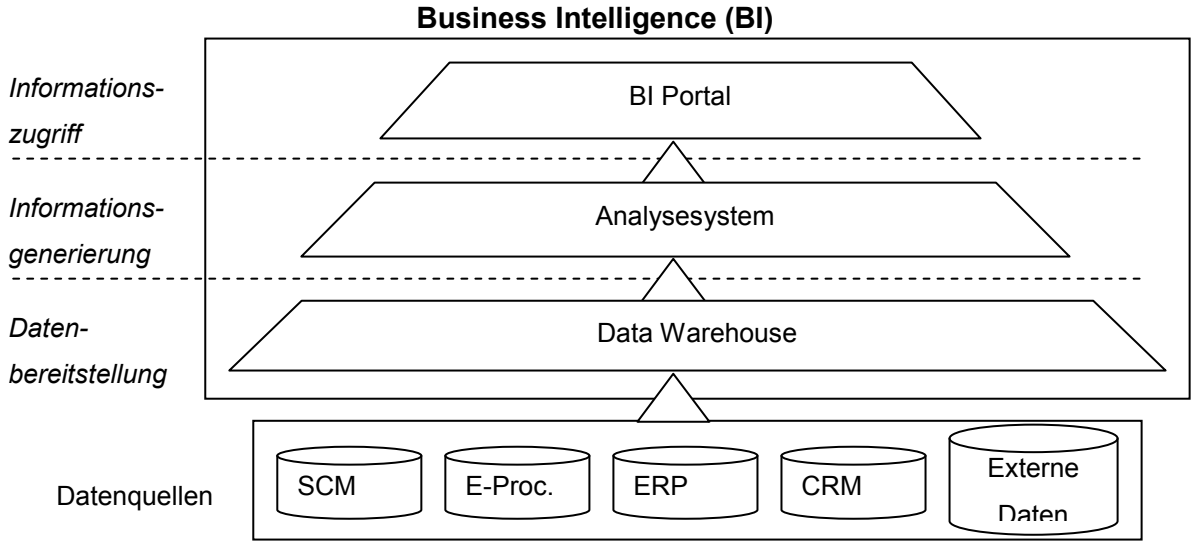


Abbildung 19: Business Intelligence Architektur (nach Kemper u. a. 2004, S. 10)
Die Datenquellen setzen sich aus SCM (Supply Chain-Management), E-Procurement, ERP (Enterprise Ressource Planning), CRM (Customer Relationship Management) und externen Systemen zusammen.

2.4.4 Business Intelligence-Datenquellen

Wie in Abbildung 19 zu erkennen, verwenden Business Intelligence-Systeme Daten aus unterschiedlichen Datenquellen. Da im vorliegenden Fall ein Business Intelligence-System für das SCM erstellt werden soll, ist besonders die Datenquelle Supply Chain-Management interessant. Wie in Abbildung 20 ersichtlich und in Abschnitt 2.1.1 erläutert, erstreckt sich Supply Chain-Management immer über mehrere Unternehmen. Aus diesem Grund ist mit

vielen unterschiedlichen Datenstrukturen und Datenquellen zu rechnen, die zunächst homogenisiert werden müssen. Daher kommt der Datenbereitstellung bei Business Intelligence-Systemen für das Supply Chain-Management ein besonderer Stellenwert zu.

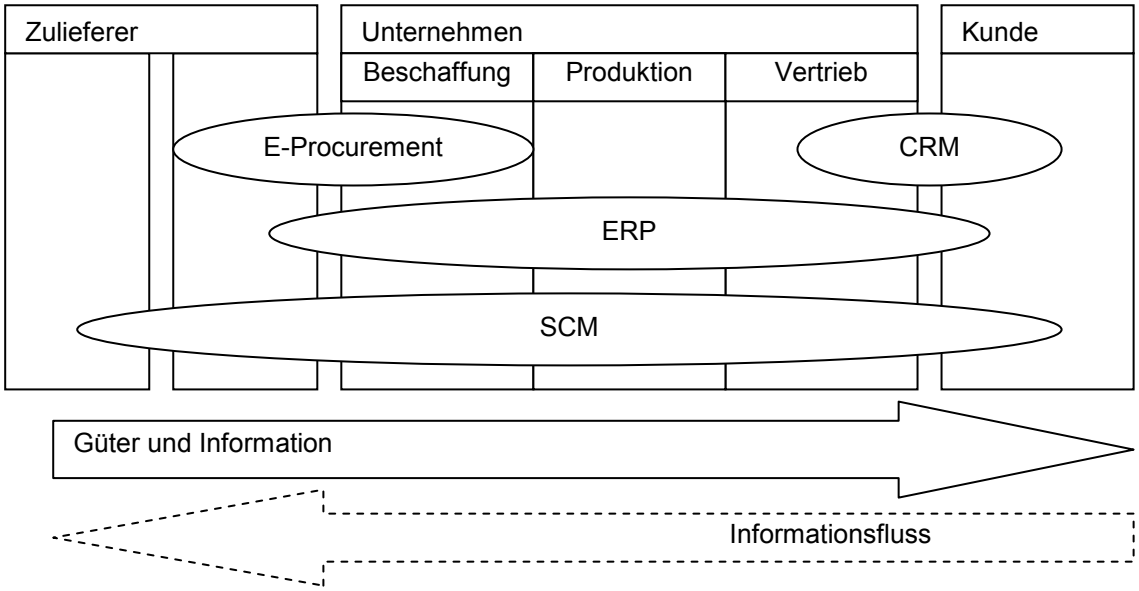


Abbildung 20: Informationssysteme in der Wertschöpfungskette (nach Kemper u. a. 2004, S. 7)
Man erkennt, dass SCM-Informationssysteme über die Unternehmensgrenzen hinweg im Einsatz sind und somit mit wesentlich mehr Akteuren interagieren als operative Informationssysteme wie ERP, E-Procurement und CRM.

2.5 Data Warehousing und Supply Chain-Management

Zur Umsetzung des in Abschnitt 2.4 beschriebenen BI-Systems ist gemäß der Aufgabenstellung ein Data Warehouse zur Datenbereitstellung zu erstellen. In diesem Abschnitt soll daher das Konzept des Data Warehousing anhand der Definition von Data Warehousing (vgl. 2.5.1), den Merkmalen (vgl. 2.5.2), den Anwendungsgebieten (vgl. 2.5.3) und den Zielen von Data Warehouse-Systemen (vgl. 2.5.4) beschrieben werden, um schließlich drei unterschiedliche Data Warehouse Architekturvarianten auf ihre Eignung für das Supply Chain-Management untersuchen (vgl. 2.5.5) und vergleichen (vgl. 2.5.6) zu können.

2.5.1 Definition Data Warehouse

Ein Data Warehouse in einem Unternehmen kann als integrierte Plattform jener Information gesehen werden, die für die Steuerung des Unternehmens von Bedeutung ist. Dabei wird Information aus unterschiedlichen Datenquellen so generiert, dass die gewünschten

Analyseergebnisse zumindest im Zeitraum weniger Minuten zur Verfügung stehen. Das Data Warehouse stellt also eine Grundlage für Analyse- und Auswertungstechniken für vom Management geforderte Information wie etwa Kennzahlen zu bestimmten Geschäftstätigkeiten dar. (Anahory & Murray 1997, S. 4)

Wesentlich für die Unterscheidung von herkömmlichen (transaktionsbasierten) und Data Warehouse-Systemen sind folgende Punkte, die auch der in Abschnitt 2.4.2 vorgenommenen Trennung von operativen und analyseorientierten Informationssystemen entsprechen (Bauer & Günzel 2004, S. 9 ff.):

- **Anfragen**: Im Gegensatz zu transaktionsbasierten Datenbanksystemen, die für den Zugriff auf einzelne Datenbestände ausgelegt sind, sei es zum Lesen, Aktualisieren oder Löschen, steht bei Data Warehouse-Systemen einzig der Lesezugriff über große Datenmengen im Vordergrund. Daneben wird das Data Warehouse in periodischen Abständen, im Normalfall durch das Hinzufügen neuer Datenbestände, auf den neuesten Stand gebracht.
- **Datenquellen**: Während transaktionsbasierte Systeme meist auf einer Datenquelle wie etwa einer Datenbank basieren, integrieren Data Warehouse-Systeme meist mehrere Datenquellen. Da diese Datenquellen sowohl physisch als auch logisch völlig unterschiedlich aufgebaut sein können ist es beim Beladen des Data Warehouse zunächst erforderlich, die extrahierten Daten der Datenquellen zu konsolidieren.
- **Anwender**: Die Anzahl der Anwender ist im Vergleich zu einem transaktionsbasierten Datenbanksystem gering. Dies liegt daran, dass die Information eines Data Warehouse zur Entscheidungsunterstützung für Manager, Controller und Analysten, nicht aber für einzelne Sachbearbeiter, dient. Die Antwortzeit für die Abfragen aus dem Data Warehouse sollte dementsprechend bei wenigen Sekunden bis wenigen Minuten liegen. Eine Antwortzeit im Millisekundenbereich, wie oft bei transaktionsbasierten Systemen, ist jedoch nicht erforderlich.

Tabelle 4 zeigt den Unterschied zwischen operativen und analytischen Informationssystemen anhand der zuvor geschilderten Unterscheidungsmerkmale:

System	**Operativ**	**Analytisch**
Eigenschaft	transaktionsbasiert	analyseorientiert
Anfragen	lesen & schreiben weniger Datensätze	lesen vieler Datensätze
Datenquellen	meist eine	mehrere heterogene Datenquellen
Anwender	viele, etwa einzelne Sachbearbeiter	wenige, wie Geschäfts- und Bereichsleiter

Tabelle 4: Gegenüberstellung von transaktionsbasierten und analyseorientierten Systemen

2.5.2 Merkmale eines Data Warehouse-Systems

Wie in Tabelle 4 erläutert ist ein Data Warehouse-System auf den lesenden Zugriff relativ weniger Anwender ausgelegt. Daraus ergeben sich für die im Data Warehouse zu speichernden Daten folgende Merkmale, die von operativen Systemen abweichen (Chamoni & Gluchowski 2006, S. 12 ff.):

1. **Themenorientierung:** Ein Data Warehouse bildet keine Geschäftsprozesse ab, wie dies bei operativen Systemen der Fall ist. Vielmehr werden inhaltliche Themenschwerpunkte, etwa auf Produkte oder Kunden gesetzt. Dabei werden nur jene Daten berücksichtigt, die der Entscheidungsfindung dienen. Somit findet bei der Themenorientierung eine Filterung aus dem operativen Datenbestand statt.
2. **Vereinheitlichung:** Da ein Data Warehouse aus vielen unterschiedlichen Datenquellen beladen wird, sind die Daten in diesen Quellen oft in unterschiedlichen Formen und Strukturen vorhanden. Ziel der Vereinheitlichung ist ein konsistenter Datenbestand, der zu einheitlichen Maßeinheiten und Kodierungen führt. Nur so ist die Vergleichbarkeit aller Daten im Data Warehouse sichergestellt.
3. **Zeitorientierung:** Da ein Data Warehouse kein operatives System ist, wird es nur in bestimmten Zeitabständen aktualisiert (vgl. 2.5.1). Das bedeutet, dass der im Data Warehouse vorliegende Datenbestand nie vollkommen aktuell ist sondern je nach Beladungsintervall einige Stunden, Tage oder Wochen zurückliegen kann. Aufgrund des analyseorientierten Fokus, der meist Information über längere Zeiträume bieten soll, reicht diese mäßige Aktualität aus.
4. **Beständigkeit:** Daten werden im Data Warehouse im Gegensatz zu operativen Systemen beständig hinterlegt, das bedeutet, die im Data Warehouse einmal hinterlegten Attributwerte werden später nicht mehr verändert. Dies führt zu einer Datenhaltung über große Zeiträume, die angepasste Datenhaltungstechniken

erfordert. Nur so kann sichergestellt werden, dass Abfragen und Auswertungen in für den Benutzer akzeptabler Zeit durchgeführt werden.

2.5.3 Anwendungsgebiete des Data Warehousing

Als Anwendungsgebiete für Data Warehousing-Systeme eignen sich all jene Tätigkeitsfelder, in denen große Datenmengen anfallen, eine Analyse dieser Daten aber trotzdem möglich sein soll. Dies sind vor allem wissenschaftliche, technische und betriebswirtschaftliche Tätigkeitsfelder (Bauer & Günzel 2004).

Im betriebswirtschaftlichen Einsatz ergeben sich für Data Warehousing vor allem die Einsatzbereiche Informationsbereitstellung und Informationsanalyse. Analyseorientierte Anwendungen verwenden die Information aus dem Data Warehouse vor allem für folgende Zwecke (Bauer & Günzel 2004, S. 13 ff.):

- **Erstellung von Kennzahlensystemen**: Bei Kennzahlensystemen handelt es sich um mehrere Kennzahlen, die zueinander über eine Hierarchie in Beziehung stehen. Die einzelnen Kennzahlen gipfeln somit in einer Spitzenkennzahl. Dies kann zum Beispiel die Berechnung des *Return on Investment* (ROI) oder die Kalkulation von Deckungsbeiträgen sein.
- **Kostenrechnung**: Data Warehousing kann für Kostenstellen- und Auftragskalkulationsrechnungen verwendet werden, wenn das Data Warehouse dementsprechend gestaltet wird. Für geschäftsfallbasierte Anwendungen eignet sich jedoch eher der Zugriff auf transaktionsbasierte Systeme (vgl. 2.5.1).
- **Marketing- und Vertriebscontrolling**: Um die Auswertung etwa von Marketingmaßnahmen auf bestimmte Regionen, Zielgruppen oder Produkte messen zu können, eignet sich die Struktur eines Data Warehouse aufgrund des multidimensionalen Modells gut. Die meisten Data Warehousing-Projekte werden entweder von der Geschäftsführung oder den Controllingabteilungen anstoßen (Totok 2000, S. 195 f.). Darüber hinaus können von dem neu geschaffenen Informationsangebot aber auch andere Unternehmensbereiche profitieren.

2.5.4 Ziele von Data Warehouse-Systemen

Oberstes Ziel von Data Warehouse-Systemen ist es, eines der wichtigsten Besitztümer eines Unternehmens, nämlich die Information über die Geschäftstätigkeit, als

Entscheidungsgrundlage für die zukünftige Geschäftstätigkeit zugänglich zu machen. Vereinfacht gesehen, dienen die in Abschnitt 2.1.5.1 angeführten operativen Systeme nur der Eingabe von Daten, während analytische Data Warehouse Systeme die übersichtliche und verständliche Ausgabe der großen Datenmengen ermöglichen. (Kimball u. a. 1998, S. 9)

Genauer formuliert sind die Ziele des Data Warehousing daher (Kimball u. a. 1998, S. 10 f.):

- **Die vom Unternehmen gesammelten Daten als Information zugänglich zu machen**: Dies bedeutet, auch große Informationsmengen werden im Data Warehouse verständlich und schnell verfügbar gemacht.
- **Die Integration der gesammelten Daten zu fachbereichsübergreifender konsistenter Information:** Dies bedeutet, die Daten müssen fachbereichs- und gegebenenfalls unternehmensübergreifend integriert werden, damit sie vergleichbar sind.
- **Eine Grundlage darzustellen, um Entscheidungen zu treffen:** Die im Data Warehouse dargestellte Information basiert auf Fakten und kann daher als Grundlage für fachbereichsspezifische und unternehmensübergreifende Entscheidungen dienen.

2.5.5 Data Warehouse-Architekturen im Supply Chain-Management

Die besondere Herausforderung für die Implementierung eines Data Warehouse für eine Supply Chain liegt in der Vielzahl der beteiligten Unternehmen. Je nach Kooperationsgrad der beteiligten Unternehmen ist daher eine angepasste Data Warehouse Architektur zu wählen. Für das Supply Chain-Management existieren nach Vahrenkamp (2007, S. 305 ff.) drei unterschiedliche Ansätze der Realisierung eines Data Warehouse. Diese sind *unabhängige Data Marts*, ein *virtuelles Data Warehouse* oder ein *zentrales Data Warehouse*.

	Datenhaltung	**Analyse**	**Kooperations-grad**	**Entwicklungs-aufwand**
Unabhängige Data Marts	verteilt	verteilt	gering	gering
Virtuelles Data Warehouse	verteilt	individuell	mittel	mittel
Zentrales Data Warehouse	zentral	zentral	hoch	hoch

Tabelle 5: Architekturansätze Data Warehousing (nach Vahrenkamp 2007, S. 308)

Wie in Tabelle 5 ersichtlich, unterscheiden sich die Ansätze hinsichtlich des Ortes der Datenhaltung, hinsichtlich der Art der Analysedurchführung, hinsichtlich des erforderlichen Kooperationsgrades sowie hinsichtlich des Entwicklungsaufwandes.

2.5.5.1 Unabhängige Data Marts

Unabhängige Data Marts sind für Supply Chains mit relativ geringem gegenseitigem Vertrauen der Akteure geeignet. Sind etwa nur die jeweils benachbarten Akteure der Supply Chain zum Datenaustausch gewillt, so können mehrere lokale voneinander unabhängige *Data Marts* erstellt werden. Ein *Data Mart* kann etwa ausschließlich auf die Daten der benachbarten Akteure der Supply Chain Zugriff bieten. Der Entwicklungsaufwand *unabhängiger Data Marts* ist gering, da nur eine geringe Anzahl an Akteuren berücksichtigt wird und somit auch die Anzahl der Datenquellen und Informationsarten geringer ist als bei anderen Architekturen. Der Nachteil *unabhängiger Data Marts* ist jedoch, dass eine spätere Integration der unabhängigen Data Marts in ein Data Warehouse aufgrund der unterschiedlichen Schemata erschwert wird (Bauer & Günzel 2004, S. 59 ff.).

2.5.5.2 Virtuelles Data Warehouse

Ein *virtuelles Data Warehouse* speichert die Daten der Supply Chain-Akteure dezentral bei den einzelnen Akteuren. Weitgehende organisatorische Anpassungen sind für die Implementierung eines virtuellen Data Warehouse nicht erforderlich, da keine neue physische Instanz zur Datenspeicherung geschaffen werden muss. Vielmehr ist eine Kooperation über die Standardisierung der Datenformate und Architektur notwendig, damit Abfragen über die gemeinsamen Datenbestände möglich werden. Eine geringere Performance im Vergleich zum zentralen Data Warehouse aufgrund der notwendigen Übertragungskanäle wird in Kauf genommen. Sollte bei den Akteuren der Supply Chain bereits ein Data Warehouse vorhanden sein, so kann ein *virtuelles Data Warehouse*, nach einer zuvor durchgeführten Konsolidierung der Data Warehouse-Schemata, eine gemeinsame Auswertung der Daten ermöglichen. Ein derartiges *virtuelles Data Warehouse* wird auch als heterogenes Data Warehouse bezeichnet (Du u. a. 2004, S. 7 f.). Der Entwicklungsaufwand ist höher als bei unabhängigen Data Marts, da beim virtuellen Data Warehouse die vorhandenen Datenbank- und Data Warehouse-Schemata integriert werden müssen.

2.5.5.3 Zentrales Data Warehouse

Die Einrichtung eines *zentralen Data Warehouse* ist vergleichbar mit den in Abschnitt 2.3 angeführten Fallstudien, in denen alle Akteure der Supply Chain auch Informationsbeteiligte sind. Ein *zentrales Data Warehouse* sieht allerdings nicht automatisch eine zentrale

Koordination der Supply Chain vor. Das heißt, die tatsächlichen Bestellmengen werden nicht vom *zentralen Data Warehouse* automatisch generiert. Der Entwicklungsaufwand eines *zentralen Data Warehouse* ist höher als der *eines virtuellen Data Warehouse*, da neben der logischen Zusammenführung der Daten auch eine zentrale physische Data Warehouse-Instanz geschaffen werden muss. Ein *zentrales Data Warehouse* erfordert somit einen hohen Kooperationsgrad der Supply Chain-Akteure.

2.5.6 Vergleich der Data Warehouse-Architekturen

Um die Eignung der in Abschnitt 2.5.5 vorgestellten Architekturen im Hinblick auf die Reduktion des Bullwhip-Effektes zu untersuchen wurde ähnlich wie in Abschnitt 2.3 ein Modell herangezogen. Das hier verwendete Modell basiert auf einer Supply Chain, die aus drei Akteuren besteht. Diese sind Hersteller, Großhändler und Lieferant. Die Nachfrage beträgt zunächst 1000 Einheiten pro Periode. Ab Periode fünf steigt die Nachfrage jedoch von 1000 auf konstant 1100 Einheiten pro Periode an. Beobachtet wird nun, ab wann sich die Produktionsmenge beim Hersteller auf die tatsächliche Nachfrage von 1100 Stück pro Woche einpendelt und wie hoch die in der Zwischenzeit eintretende maximale Produktions- und Lagermenge beim Hersteller ist. Das Data Warehouse tritt nie als zentrales Koordinationssystem auf, da Bestellzeitpunkt und -menge autonom festgelegt werden. (Vahrenkamp 2007, S. 311)

Abbildung 21 zeigt, dass der Einsatz eines Data Warehouse als Informationssystem in der Supply Chain in der Lage ist, den Bullwhip-Effekt zu reduzieren. Besonders die Implementierung eines virtuellen oder zentralen Data Warehouse ist in der Lage sowohl die maximale Produktions- als auch die maximale Lagermenge um etwa ein Drittel zu reduzieren. Der Gleichgewichtszeitpunkt kann mit einem virtuellen oder zentralen Data Warehouse sogar mehr als halbiert werden. Der Architekturtyp *unabhängige Data Marts* wurde im Modell *mit zwei unabhängigen Data Marts umgesetzt*, die jeweils zwischen Hersteller und Großhändler und Großhändler und Produzenten existieren. Diese Variante zeigt sich im Hinblick auf die Reduzierung des Bullwhip-Effektes als sinnvoll, sofern eine weitergehende Kooperation in der Supply Chain nicht möglich ist.

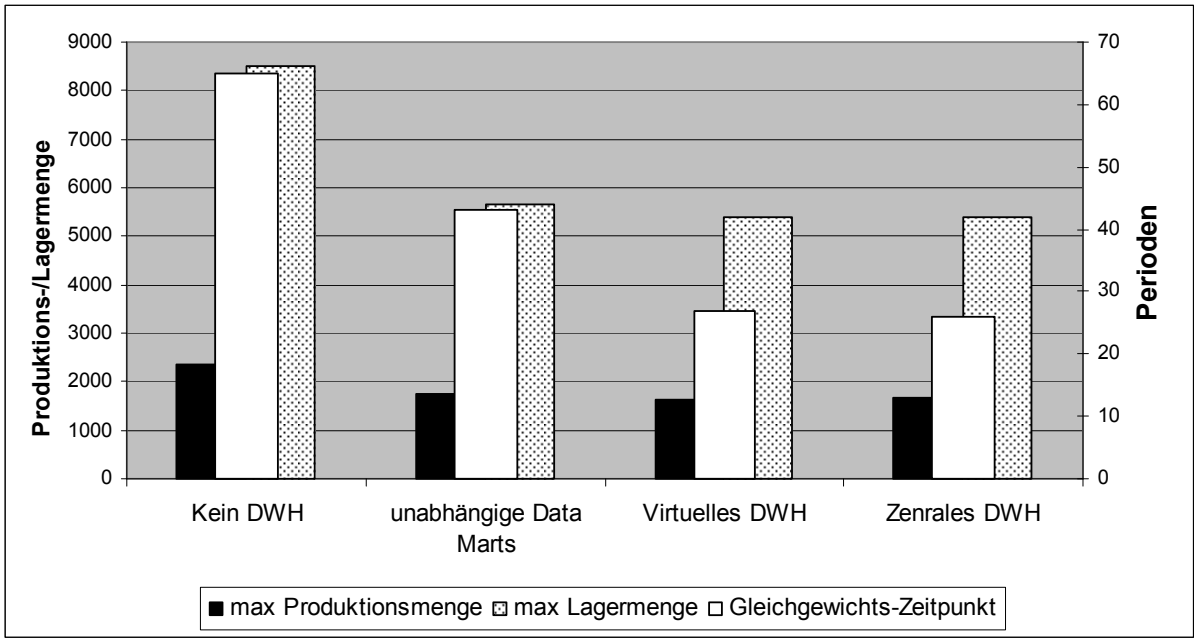


Abbildung 21: Auswirkung der unterschiedlichen Data Warehouse-Architekturen auf den Bullwhip-Effekt (nach Vahrenkamp 2007, S. 312)
Der schwarze Balken bezieht sich auf die maximale auftretende Produktionsmenge, der schwarz-weiße Balken auf die maximale Lagermenge und der weiße Balken auf die Anzahl der Perioden, bis sich die Produktionsmenge wieder dem tatsächlichen Bedarf annähert. Niedrigere Balken weisen auf eine größere Reduktion des Bullwhip-Effektes hin.

Zu beachten ist, dass aufgrund unterschiedlicher Modellannahmen ein direkter Vergleich zwischen den in Abschnitt 2.3 genannten Fallstudien und dem in diesem Abschnitt behandelten Modell nicht möglich ist. Eine Parallelität zur *Fallstudie von Milling und Größler* (vgl. 2.3.1) liegt jedoch im gleichen einmaligen Anstieg der Endkonsumentennachfrage um zehn Prozent innerhalb der ersten Periode. Auch der daraufhin eintretende Rückgang des maximalen Lagerbestandes liegt ähnlich wie in der *Fallstudie von Milling und Größler* bei etwa 30 Prozent. Dies lässt jedenfalls den Rückschluss zu, dass eine Business Intelligence-Lösung, die auf einem Data Warehouse basiert, ein geeignetes Informationssystem zur Reduzierung des Bullwhip-Effektes ist.

2.6 Datenquellen

Die Herausforderung bei der Entwicklung eines analytischen Informationssystems für eine Supply Chain liegt vor allem in der Integration der Daten in ein einheitliches Schema, da die Integration von Daten aus verschiedenen Unternehmen zunächst eine Vereinheitlichung der unterschiedlichen Datenmuster und Speicherformen erfordert.

Um geeignete Datenquellen für ein zu erstellendes Data Warehouse auszuwählen, müssen die Daten einer Datenquelle dem Zweck des Data Warehouse entsprechen, ein gewisses Qualitätsmaß erfüllen und eine gewisse Mindestverfügbarkeit aufweisen. Außerdem darf der Preis für die Quelldaten das Budget für das Data Warehouse-Projekt nicht übersteigen. Im Folgenden wird daher auf die Kriterien Zweck (vgl. 2.6.1), Qualität (vgl. 2.6.2), Verfügbarkeit (vgl. 2.6.3) und Preis der Quelldaten (vgl. 2.6.4) näher eingegangen.

2.6.1 Zweck des Data Warehouse

Die Daten einer Datenquelle müssen dem Zweck des Data Warehouse entsprechen. Das bedeutet, dass die Datenquelle jene Daten liefern muss, die für den Betrieb des Data Warehouse notwendig sind. Liefert eine Datenquelle etwa Daten über den Zustand eines Unternehmens zu einem bestimmten Zeitpunkt, die nicht fortlaufend aktualisiert werden, so entspricht diese Datenquelle nicht dem Zweck eines Data Warehouse, das wöchentlich neu beladen werden soll.

2.6.2 Qualität der Quelldaten

Um die Richtigkeit und Verwendbarkeit der Daten des Data Warehouse garantieren zu können, ist besonders auf die Qualität der Quelldaten zu achten. Datenqualitätsprobleme können, wie Abbildung 22 zeigt, in *Single-Source* und *Multi-Source* Probleme unterteilt werden.

Single-Source Daten-Qualitätsprobleme beziehen sich auf einzelne Datenquellen, deren Daten entweder unzureichend oder falsch strukturiert sind. So können fehlende Integritätsbedingungen, etwa dass eine bestimmte Kennzahl nicht negativ sein darf, dazu führen, dass fehlerhafte Datenbestände nicht erkannt werden. Andere Fehler, die trotz einer adäquaten Schemadefinition auftreten können, wie etwa Tipp- oder Rechtschreibfehler, werden dem *Instance-Level* zugeordnet.

Multi-Source Daten-Qualitätsprobleme am Schema-Level entstehen etwa bei unterschiedlicher Benennung von semantisch gleichartigen Feldern oder bei unterschiedlichen Kodierungen etwa durch unterschiedliche Notation von Adressen (vgl. 2.7.3 Konvertierung von Kodierungen). So können etwa zwei Namen in unterschiedlichen Datenquellen identisch sein, aber zwei verschiedene Personen repräsentieren (Homonym). Andererseits können zwei, in unterschiedlicher Schreibweise hinterlegte Namen, dieselbe Person repräsentieren

(Synonym). Daten-Qualitätsprobleme am *Instance-Level* sind durch unterschiedliche Repräsentationen gekennzeichnet. So kann zum Beispiel ein identisches Produkt in unterschiedlichen Datenquellen mit zwei unterschiedlichen Identifikationsnummern hinterlegt sein. In diesem Fall haben unterschiedliche Erfassungsmethoden zu unterschiedlichen Identifikationsnummern geführt. Der Beladungsprozess muss in diesem Fall erkennen, dass es sich trotz unterschiedlicher Identifikationsnummern um dasselbe Produkt handelt.

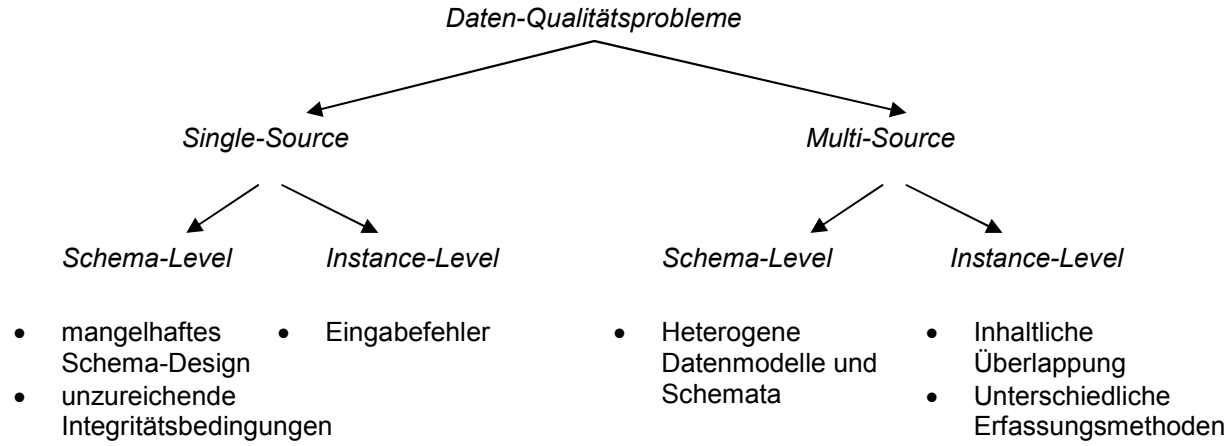


Abbildung 22: Klassifikation von Daten-Qualitätsproblemen (nach Rahm & Do 2000, S. 3)

In Abbildung 23 werden auftretende Qualitätsprobleme beim Abruf der Daten aus zwei unterschiedlichen Datenquellen illustriert. Bei *Multi-Source* Datenquellen kommt es zu folgenden Daten-Qualitätsproblemen:

- **Schema-Ebene**
  - *Namenskonflikt*: Zwei Felder, die semantisch die gleiche Bedeutung haben, sind unterschiedlich benannt.
  - *Struktureller Konflikt*: Tritt bei einer unterschiedlichen Art der Namensspeicherung auf. Bei Quelle-1 wird der Name in zwei Felder unterteilt; Quelle-2 repräsentiert den Namen in einem Feld und kürzt den Nachnamen ab.
- **Instanz-Ebene**
  - *Unterschiedliche Repräsentation*: Bei Quelle-1 wird das Geschlecht durch (f/m), bei Quelle-2 durch (1/2) dargestellt.
  - *Doppelte Einträge*: Es ist möglich, dass Einträge, die etwa die gleiche Person darstellen, mehrfach vorkommen.
  - *Quellspezifische Identifier* (Primärschlüssel): Die Primärschlüssel der Datenquellen sind für die Datenextraktion nicht von Bedeutung, da

unterschiedliche Primärschlüssel auf dieselbe Person verweisen können. Zufällig gleiche Primärschlüssel können hingegen für unterschiedliche Personen stehen.

- *Unterschiedliche Erfassungsmethoden*: Die Ausgaben der Personen werden in Quelle-1 in US-Dollar und in Quelle-2 in Euro hinterlegt.

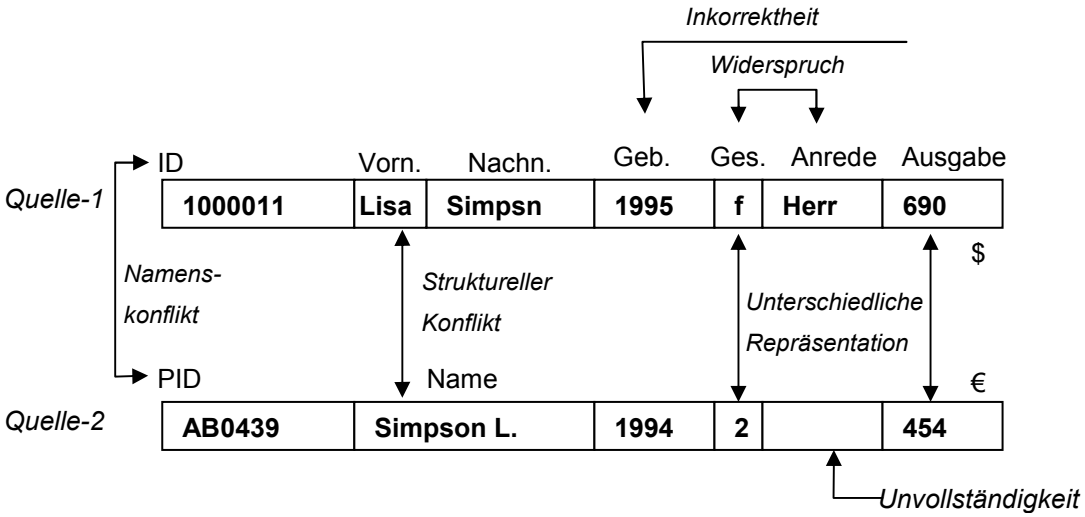


Abbildung 23: Beispiele für Daten-Qualitätsmängel (nach Bauer & Günzel 2004, S. 41)

Zusätzlich zu den *Multi-Source* Daten-Qualitätsproblemen kommt es zu folgenden *Single-Source* Daten-Qualitätsproblemen:

- **Schema-Ebene**
  - *Widerspruch:* Ein Widerspruch tritt beispielsweise auf, wenn im selben Datensatz das Geschlecht nicht mit der Anrede übereinstimmt.
  - *Unvollständigkeit*: Ist ein Feld, für das immer ein Wert zu erwarten ist, nicht befüllt, spricht man von unvollständigen Daten. Dies ist etwa beim Feld Anrede der Fall. Wäre hingegen ein Kommentarfeld nicht befüllt so würde dies nicht auf Unvollständigkeit hindeuten, da für ein Kommentarfeld nicht immer ein Wert zu erwarten ist.
- **Instanz-Ebene**
  - *Tippfehler:* Tippfehler können bei der manuellen Eingabe von Daten entstehen. Ein Tippfehler ist in Abbildung 23 beim Feld Nachname erkennbar.
  - *Inkorrektheit:* Werden Daten etwa von Kunden falsch angegeben, so kommt es zu inkorrekten Daten.

Fehler auf der Schema-Ebene können oft durch Zwangsbedingungen (Constraints) ausgeschlossen werden. Fehler auf der Instanz-Ebene hingegen sind nur durch domänenspezifisches Wissen, wie etwa einer Liste mit allen zulässigen Werten, erkennbar (vgl. 2.7.4).

2.6.3 Verfügbarkeit der Quelldaten

Dienen die Daten aus den Datenquellen dem Zweck des Data Warehouse und ist auch die Qualität der Daten zufriedenstellend, so ist als nächstes zu klären, ob die Daten, wie es der Zweck des Data Warehouse vorsieht, verwendet werden können. Dabei kann es zu organisatorischen und technischen Hindernissen kommen (Bauer & Günzel 2004, S. 41 f.):

2.6.3.1 Organisatorische Hindernisse

- **Rechtliche Einschränkungen**: Aufgrund von rechtlichen Vorschriften, die in Österreich im Datenschutzgesetz geregelt sind, ist etwa die Verwendung von personenbezogenen Daten auf Zulässigkeit zu überprüfen.
- **Betriebsinterne Einschränkungen**: Aufgrund von Vereinbarungen mit dem Betriebsrat oder unternehmensinternen Regelungen ist es möglich, dass Daten vertraulich zu behandeln sind. Diese Vertraulichkeit muss auch im erstellten Data Warehouse gegeben sein.
- **Zeitliche Verfügbarkeit**: Nur wenn die Daten rechtzeitig und in der benötigten Frequenz zur Verfügung stehen, können sie den Zweck des Data Warehouse erfüllen. Somit müssen etwa die Absatzzahlen am Endkonsumentenmarkt zumindest innerhalb weniger Tage zur Verfügung stehen, um sinnvoll verwendet werden zu können.

2.6.3.2 Technische Hindernisse

- **Zugriffsmöglichkeit**: Nur mit geeigneter Software ist ein Zugriff auf die Quelldaten möglich.
- **Sicherheit**: Bei der Datenübertragung muss zumindest
    - Vertraulichkeit (die übertragenen Daten sind nur einem berechtigten Personenkreis zugänglich) und
    - Integrität (die Daten werden bei der Übertragung nicht verändert)

  gewahrt bleiben (Heinrich 2001, S. 164).

- **Schnelligkeit**: Dazu muss sichergestellt sein, dass die erwarteten Datenmengen in ausreichender Geschwindigkeit aus den Quellsystemen in das Data Warehouse überführt werden können.

2.6.4 Preis für den Erwerb der Quelldaten

Werden externe Daten als Datenquelle genutzt, so ist zwischen Datenquellen zu unterscheiden, die frei etwa über das Internet verfügbar sind, und Datenquellen, die nur kostenpflichtig bezogen werden können. So bieten Marktforschungsinstitute Daten zu aktuellen Marktentwicklungen ausschließlich kostenpflichtig an. Hier ist unter Einbezug der drei anderen angeführten Kriterien zu entscheiden, ob und in welcher Höhe Kosten für den Zugriff auf externe Datenquellen getragen werden.

2.7 Phasen des Data Warehousing

Die in diesem Abschnitt beschriebenen Phasen sollen den Datenfluss ausgehend von den einzelnen Datenquellen bis zur Darstellung der Information beim Endbenutzer darstellen.

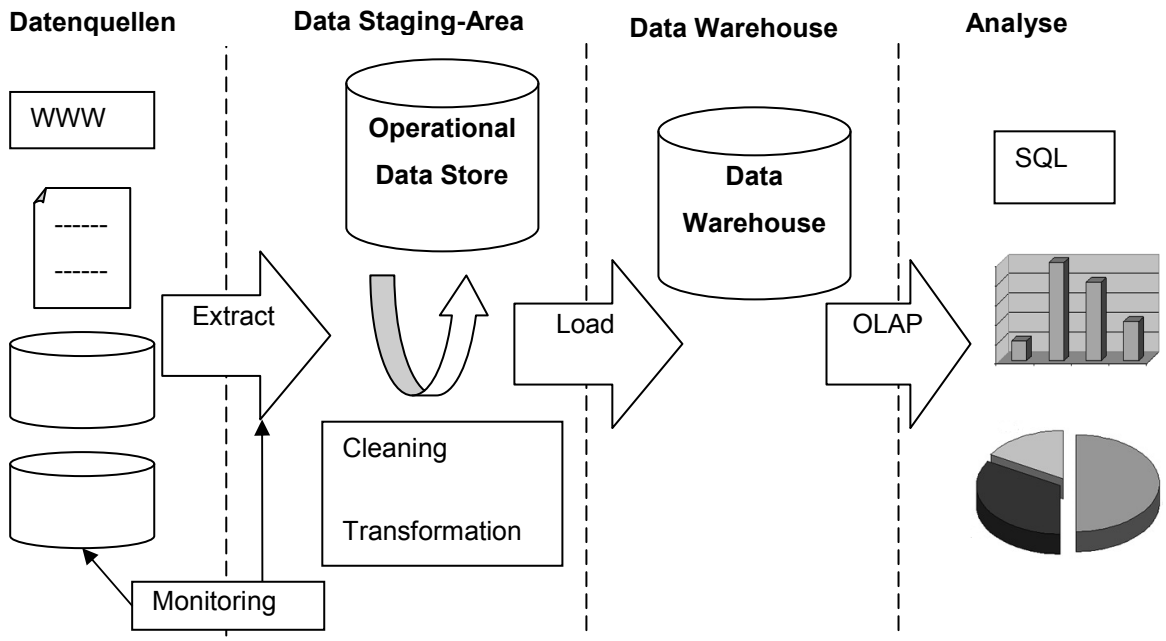


Abbildung 24: Phasen des Data Warehousing

Der Datenfluss wird, wie in Abbildung 24 veranschaulicht in folgende sechs Phasen unterteilt (Bauer & Günzel 2004, S. 75 ff.):

- **Monitoring** (vgl. 2.7.1): Beobachtung der Datenquellen
- **Extraktion** (vgl. 2.7.2): Übertragung der relevanten Daten
- **Transformation** (vgl. 2.7.3): Umwandeln der Daten in eine für das Data Warehouse geeignete Form
- **Bereinigung** (vgl. 2.7.4): Behebung von Datenqualitätsproblemen
- **Beladung** (vgl. 2.7.5): Befüllung des Data Warehouse-Schemas
- **Analyse** (vgl. 2.7.6): Aufbereitung der Daten für den Endbenutzer durch *Online Analytical Processing* (*OLAP*).

Im Folgenden wird das Monitoring, die Extraktions-, Transformations-, Bereinigungs- und Beladungsphase kurz als ETL-Prozess (*Extract, Transform and Load-Process*) bezeichnet.

2.7.1 Monitoring

Unter *Monitoring* wird die Überwachung von Datenquellen hinsichtlich Datenänderungen, die für das Data Warehouse relevant sind, verstanden. Die überwachten Datenquellen werden in kooperative und nicht kooperative Datenquellen unterteilt. Kooperative Datenquellen implementieren, im Gegensatz zu nicht kooperativen Datenquellen, Mechanismen, die es automatisch ermöglichen, jene Daten zu erkennen, die im Vergleich zum letzten Abruf geändert wurden. Zusätzlich bieten kooperative Datenquellen Benachrichtigungsfunktionen, sobald die Datenquelle über neue oder geänderte Daten verfügt.

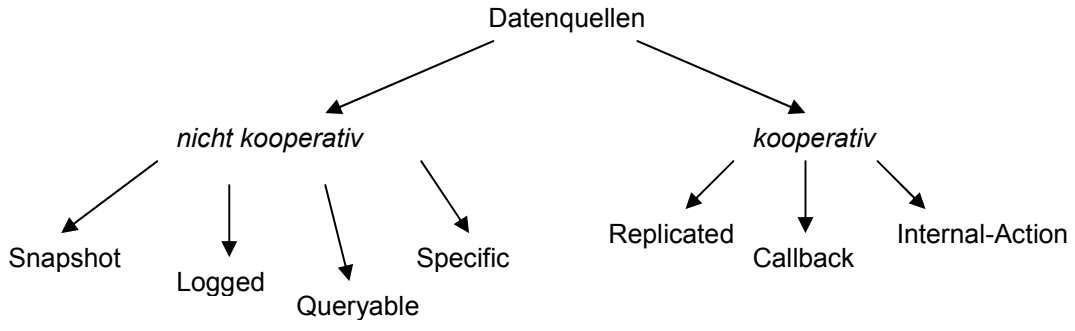


Abbildung 25: Einordnung der Datenquellen (nach Jarke u. a. 2003, S. 57)

2.7.1.1 Nicht kooperative Datenquellen

Wie in Abbildung 25 erkennbar, existieren vier unterschiedliche Arten von *nicht kooperativen Datenquellen* (Jarke u. a. 2003, S. 56 ff.). Diese werden im Folgenden näher erläutert:

Schnappschuss-Datenquellen (Snapshot Sources)

Derartige Datenquellen sind typischerweise einfache Dateien, die selektive Abfragen des Inhaltes nicht unterstützen. Das bedeutet, *Schnappschuss-Datenquellen* müssen immer zur Gänze übertragen und geöffnet werden. Um Änderungen festzustellen, ist es erforderlich, einen aktuell abgerufenen Schnappschuss mit dem zuvor abgerufenen Schnappschuss zu vergleichen. Besonders bei Schnappschuss-Datenquellen, die regelmäßig eine große Anzahl neuer Datensätze liefern, kann dies zu großen übertragenen Datenmengen und hohen Systembelastungen führen.

Protokollierte-Datenquellen (Logged Sources)

Datenquellen, die jegliche Änderungen im Datenbestand protokollieren, werden als *Protokollierte-Datenquellen* bezeichnet. Datenbanksysteme bieten diese Funktion standardmäßig an. Zu beachten ist jedoch zum einen, dass die Protokolle oftmals nur mit Administratorrechten abrufbar sind und meist in keinem standardisierten Format vorliegen. Zum anderen werden Protokolle häufig in einfachen Dateien hinterlegt, die zu ähnlichen Problemen wie bei Schnappschuss-Datenquellen führen.

Selektierbare-Datenquellen (Queryable Sources)

Selektierbare-Datenquellen bieten die Möglichkeit nur die für den Data Warehouse-Zweck relevanten Daten abzurufen. Modifizierte Daten werden bei Selektierbaren-Datenquellen, ähnlich wie bei Schnappschuss-Datenquellen, über einen Vergleich mit der bereits gespeicherten älteren Version der Daten erkannt. Dieser Vergleich basiert auf den jeweiligen Schlüsseln. Existieren neue Daten, so besitzen diese auch neue Schlüssel. Aktualisierte Daten werden am gleichen Schlüssel, aber unterschiedlichen Daten erkannt. Erfolgt eine Abfrage der Daten nur selten, so werden in der Zeit zwischen zwei Abfragen hinzugefügte, aber wieder gelöschte Daten vom Beladungsprozess nicht erkannt. Erfolgt die Abfrage zu häufig, kann dies zu einer verringerten Leistung der Datenquelle führen.

Spezifische-Datenquellen (Specific Sources)

Spezifische-Datenquellen stellen alle weiteren Datenquellen dar, die nicht kooperativ sind. Werden etwa ältere Systeme als Datenquelle benutzt, so muss das *Monitoring* an die spezifischen Eigenschaften dieser Datenquelle angepasst werden.

2.7.1.2 Kooperative Datenquellen

Kooperative Datenquellen ermöglichen es Veränderungen am Datenbestand anhand spezifischer Funktionen der Datenquelle zu erkennen. Es werden drei Typen von kooperativen Datenquellen unterschieden:

Replizierte-Datenquellen (Replicated Sources)

Replizierte-Datenquellen halten selbst eine Kopie des Datenbestandes vor. Aufgrund dieser Kopie sind Replizierte-Datenquellen in der Lage, die veränderten Daten selbstständig zu erkennen und an ein anderes System zu propagieren. Replizierte-Datenquellen sind damit in der Lage das Data Warehouse aktiv bei verändertem Datenbestand zu benachrichtigen.

Callback-Datenquellen (Callback Sources)

Callback-Datenquellen ermöglichen es direkt in der Datenquelle einen Trigger zu hinterlegen, der über einen I/O-Mechanismus (Input/Output Mechanismus) den Beladungsprozess starten kann. Es ist möglich, die Trigger so im Quellsystem zu hinterlegen, dass nur bei tatsächlich für das Data Warehouse relevanten Änderungen auch eine Benachrichtigung an den Beladungsprozess erfolgt. Es ist zu beachten, dass eine derartige Datenquelle das Erstellen von Triggern erlauben muss.

Internal-Action-Datenquellen (Internal-Action Sources)

Im Gegensatz zu Callback-Datenquellen ist es *Internal-Action-Datenquellen* nicht möglich, Benachrichtigungen an externe Systeme zu versenden. In der Datenquelle werden daher eigene Tabellen angelegt, die alle Änderungen an den relevanten Dateien aufzeichnen. Der Beladungsprozess greift somit nur mehr auf jene Daten zu, die in den Tabellen stehen, die Änderungen aufzeichnen. Da die Datenquelle nicht erkennt, wann der letzte Zugriff erfolgte muss vom Beladungsprozess zusätzlich die Nummer des erstellten Änderungseintrages protokolliert werden, damit nur neue Änderungseinträge abgerufen werden können. Zu

beachten ist, dass die Änderungstabellen mindestens alle Daten beinhalten müssen, die verändert wurden.

2.7.1.3 Merkmale von Datenquellen

Weiters werden Datenquellen in die Merkmale *Entdeckung aller Änderungen*, *Internes Monitoring* und *Push-Prinzip* unterteilt (Bauer & Günzel 2004, S. 76 f.):

- Das Merkmal ***Entdeckung aller Änderungen*** bedeutet, dass alle Änderungen seit dem letzten Zugriff auf eine Datenquelle erfasst werden. Das heißt, selbst wenn ein bestimmter Datensatz hinzugefügt und wieder gelöscht wird, so erkennt das Monitoring diesen Änderungsvorgang obwohl sich der Datenbestand letztlich nicht geändert hat.
- Das Merkmal ***Internes Monitoring*** beschreibt, ob eine Datenquelle durch *internes Monitoring* in der Lage ist, die für ein Zielsystem relevanten Änderungen zu entdecken.
- Eine Datenquelle die das ***Push-Prinzip*** implementiert ist in der Lage ein Zielsystem zu informieren, sobald Änderungen stattgefunden haben. In diesem Fall muss der Beladungsprozess nicht in periodischen Abständen prüfen, ob sich Änderungen ergeben haben, sondern das Quellsystem macht den Beladungsprozess aktiv auf Änderungen aufmerksam.

Tabelle 6 gibt einen Überblick über die Merkmale der in Abschnitt 2.7.1.1 und 2.7.1.2 genannten Arten von Datenquellen:

Art der Datenquelle	**Entdeckung aller Änderungen**	**Internes Monitoring**	**Push-Prinzip**
Snapshot			
Logged	✓	✓	
Queryable			
Replicated	✓		✓
Callback	✓	✓	✓
Internal Action	✓	✓	

Tabelle 6: Vergleich der Eigenschaften von Datenquellen.

2.7.2 Extraktionsphase

Die *Extraktionsphase* (*Extraction*) der Daten baut auf der gewählten Datenquelle und Art des *Monitorings* auf. In der *Extraktionsphase* wird die Entscheidung getroffen, welche Daten selektiert werden, sofern diese Entscheidung nicht bereits durch internes *Monitoring* in den Quelldaten getroffen wurde. Zusätzlich muss die Frequenz der Datenextraktion festgelegt werden. Diese Entscheidung ist abhängig vom Zweck des Data Warehouse und von der Art der Datenquelle. Die Extraktionsphase kann in folgenden Ereignissen gestartet werden (Bauer & Günzel 2004, S. 82):

- **Periodisch:** Der Extraktionsprozess wird nach einem bestimmten Zeitintervall gestartet. Dieses Zeitintervall kann je nach Zweck des Data Warehouse und Art der Quelldaten variieren. So müssen etwa Börsenkurse häufiger aktualisiert werden als produktspezifische Daten.
- **Anfragegesteuert:** In diesem Fall wird entsprechend dem *Pull-Prinzip* in periodischen Abständen eine Anfrage an das Quellsystem gestellt. Wird eine relevante Änderung der Quelldaten erkannt, so wird der Extraktionsprozess erneut angestoßen. Dieses Verfahren kann etwa bei Selektierbaren-Datenquellen angewandt werden, um Veränderungen am Datenbestand zu erkennen.
- **Ereignisgesteuert:** Die ereignisgesteuerte Extraktion wird im Gegensatz zur anfragegesteuerten Extraktion direkt vom Quellsystem angestoßen. Das bedeutet, der Extraktionsprozess verläuft nach dem *Push-Prinzip*. Für die ereignisgesteuerte Extraktion eigenen sich daher *Replicated-* oder *Callback Sources*.
- **Sofort:** Bei besonders hohen Anforderungen an die Aktualität des Data Warehouse, kann es erforderlich sein, dass etwa Daten zu Börsenkursen sofort nach jeder Änderung der Quelldaten auch im Data Warehouse aktualisiert werden. Dies entspricht einer ereignisgesteuerten Extraktion, die auf jede Veränderung hin den Extraktionsprozess startet.

2.7.3 Transformationsphase

Ziel der *Transformationsphase* (*Transform*) ist es, die in der *Extraktionsphase* geladenen Daten an die Anwenderanforderungen anzupassen. Das bedeutet, dass die Daten sowohl integriert, als auch bereinigt werden müssen. Die Transformation der Daten geschieht in einem Bereich, der auch als Staging-Area bezeichnet wird. In diesem Bereich werden die

Daten so aufbereitet, dass sie im Beladungsprozess in das Data Warehouse überführt werden können. Die in den folgenden Abschnitten 2.7.3.1 bis 2.7.3.7 vorgestellten Maßnahmen erläutern wesentliche Aufgaben der Transformationsphase.

2.7.3.1 Schlüsselbehandlung

Aufgrund der bereits in Abschnitt 2.6 erwähnten heterogenen Datenstruktur der unterschiedlichen Datenquellen können die Schlüssel der Datenquellen im Normalfall nicht in das Data Warehouse übernommen werden, da die Schlüssel nur für eine spezifische Datenquelle gelten. Darüber hinaus ist es möglich, dass Schlüssel eine implizite semantische Bedeutung besitzen. So kann ein Schlüssel etwa der Personalnummer entsprechen. In diesem Fall muss die Personalnummer zusätzlich übernommen werden.

Im Data Warehouse selbst wird zur Identifikation eines Datensatzes immer ein künstlich erzeugter Schlüssel (surrogate key) verwendet. Aus diesem Grund muss im Transformationsprozess ein derartiger Schlüssel, der im Normalfall eine natürliche Zahl ist, die sequenziell erhöht wird, erzeugt werden.

2.7.3.2 Anpassung von Datentypen

Falls der Datentyp einer Datenquelle nicht dem Datentyp im Data Warehouse entspricht, muss eine Umwandlung vorgenommen werden. Wird etwa ein Datumswert in einer Datenquelle als Zeichenkette hinterlegt so kann eine Umwandlung zum Datentyp Date notwendig sein.

Eine derartige Vereinheitlichung von Datumsangaben zum Datentyp Date hat darüber hinaus den Vorteil, dass die Daten an der Benutzerschnittstelle an die jeweilige regionale Darstellungsform angepasst werden kann, während die Datumsangaben im Data Warehouse eindeutig hinterlegt sind.

2.7.3.3 Konvertierung von Kodierungen

Unterschiedliche Kodierungen müssen beim Transformationsprozess vereinheitlicht werden. Ist etwa das Geschlecht einer Person in zwei Datenquellen unterschiedlich kodiert, so ist eine einheitliche Kodierung einzuführen (vgl.

Abbildung 26).

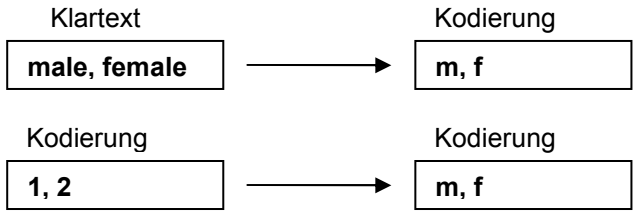


Abbildung 26: Konvertierung von Kodierungen

2.7.3.4 Vereinheitlichung von Zeichenketten

Aufgrund regionaler sprachlicher Besonderheiten, wie etwa Umlaute oder Sonderzeichen, ist es notwendig, eine einheitliche Schreibweise festzulegen. Darüber hinaus ist festzulegen, ob die Daten einheitlich in Klein- oder Großbuchstaben festgehalten werden. Auch Trennzeichen wie Leerzeichen, Punkte oder Bindestriche müssen vereinheitlicht werden.

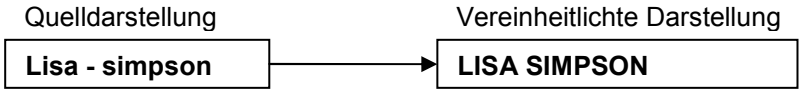


Abbildung 27: Vereinheitlichung von Zeichenketten

Die Vereinheitlichung einer Zeichenkette ist in Abbildung 27 exemplarisch gezeigt. Diese Vereinheitlichung dient der Integration der aus unterschiedlichen Datenquellen geladenen heterogenen Daten.

2.7.3.5 Kombination/Separierung von Attributwerten

Um die Quelldaten an das Zielschema anzupassen, kann es notwendig sein, Attributwerte zu kombinieren oder zu separieren (vgl.

Abbildung 28). Das bedeutet, dass zwei Datensätze, die zum Beispiel aus unterschiedlichen Datenquellen extrahiert wurden, zu einem Feld zusammengeführt, also kombiniert werden. Aufgrund des Data Warehouse-Schemas kann auch eine Separierung eines Quellfeldes in mehrere einzelne Felder vorgenommen werden.

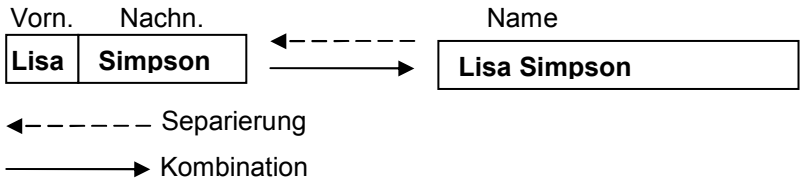


Abbildung 28: Kombination/Separierung von Attributwerten.

2.7.3.6 Berechnung abgeleiteter Werte

Wenn die Quelldaten in ihrer ursprünglichen Form im Zielschema nicht benötigt werden oder abgeleitete Werte zusätzlich im Data Warehouse benötigt werden um eine neuerliche Berechnung in der Analysephase aus Effizienzgründen zu vermeiden, ist es notwendig, im Transformationsprozess Berechnungen durchzuführen. Dies kann etwa zur Berechnung von Kennzahlen dienen, wie in Abbildung 29 exemplarisch dargestellt.

Abbildung 29: Berechnung des abgeleiteten Wertes „aktueller Lagerstand“

2.7.3.7 Aggregation

Zusätzlich kann die Berechnung abgeleiteter Werte für die Aggregation von Daten benötigt werden. Ziel einer Aggregation ist es die Geschwindigkeit der Benutzerabfragen zu erhöhen, indem voraussichtlich vom Endbenutzer häufig benötigte Aggregationsebenen bereits vorberechnet werden. Eine Aggregation wird vorgenommen, wenn die Quelldaten entweder in feinerer Granularität vorliegen als für den Data Warehouse-Zweck benötigt, oder wenn von den Endbenutzern Daten häufig auch in gröberen Aggregationsebenen abgefragt werden. Eine so vorgenommene Aggregation im Transformationsprozess steigert die Performance, da nun die Berechnung der aggregierten Werte nicht für jede Endbenutzeranfrage neu vorgenommen werden muss (vgl.

Abbildung 30).

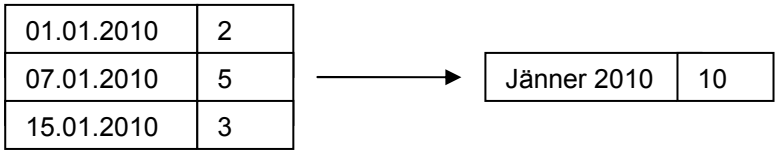


Abbildung 30: Aggregation von Tages- auf Monatsebene

Um zu entscheiden, welche Aggregationen vorgenommen werden, ist zum einen zu klären, wie viele Datenreihen aufgrund der Aggregation eingespart werden können. Existiert für jeden Tag ein Eintrag in einer Tabelle, so würde eine Aggregationstabelle auf Monatsebene die Anzahl der Einträge bereits auf etwa 1/30 reduzieren, wie in Abbildung 31 ersichtlich. Eine zusätzliche Reduktion auf Jahresebene würde hingegen die Anzahl der Datenreihen nur mehr auf 1/12 reduzieren. (Kimball u. a. 1998, S. 543 ff.)

Level	Rows
Day	1826
Month	60
Quarter	20
Year	5
Total	1

Abbildung 31: Anzahl der Datenreichen (Rows) bei unterschiedlichen Aggregationsebenen (Levels) innerhalb von etwa fünf Jahren (nach Kimball u. a. 1998, S. 547).

Bei der Ausführung von Abfragen muss vom Analyseprogramm erkannt werden, dass eine Aggregationstabelle vorhanden ist. Dies ist mittels eines Aggregations-Navigators möglich, der Abfragen zur effizienteren Bearbeitung automatisch und transparent für den Anwender gegebenenfalls auf eine Aggregationstabelle umleitet. (Kimball u. a. 1998, S. 544)

2.7.4 Bereinigungsphase

Die Bereinigung der Quelldaten ist häufig aufgrund mangelnder Qualität (vgl. Abschnitt 2.6.2) der Quelldaten erforderlich. Zunächst ist es zur Bereinigung erforderlich, einen Fehler in den Quelldaten zu erkennen. Dies geschieht anhand der Kriterien Korrektheit (vgl. 2.7.4.1), Konsistenz (vgl. 2.7.4.2), Vollständigkeit (vgl. 2.7.4.3) und Redundanzfreiheit (vgl. 2.7.4.4), die im Folgenden näher erläutert werden (Bauer & Günzel 2004, S. 89 ff.):

2.7.4.1 Korrektheit

Inkorrekte Daten in den Quelldaten zu erkennen ist nicht trivial. Untersucht man etwa Adressen, so ist ein Abgleich mit einer zentralen Datenbank nötig, um zu erkennen, ob eine angegebene Adresse auch tatsächlich existiert. Aufgrund großer Datenmengen ist ein derartiger Vergleich aber oft nur stichprobenartig möglich. Ist ein Datensatz aufgrund einer Stichprobe als inkorrekt identifiziert, so müssen auch andere Datensätze derselben Datenquelle genauer untersucht werden, um etwaige häufiger auftretende Fehler erkennen zu können. Darüber hinaus können statistische Verfahren zur Erkennung von inkorrekten Daten genutzt werden. Weicht etwa ein neuer Datensatz stark von der Varianz der Daten der letzten Jahre ab, so ist dieser möglicherweise inkorrekt. Zur Erkennung von inkorrekten Daten ist immer domänenspezifisches Wissen, etwa über vorhandene Adressen, notwendig.

2.7.4.2 Konsistenz

Dateninkonsistenzen treten dann auf, wenn sich zwei oder mehrere Werte widersprechen. So kann eine Inkonsistenz auftreten, wenn das Lieferdatum eines Produktes vor dem Bestelldatum liegt. Inkonsistenzen können meist mithilfe von Regeln erkannt werden (vgl. Abbildung 32).

WENN lieferdatum < bestelldatum *DANN* Mangel mit Schweregrad 0.9

Abbildung 32: Regel um Inkonsistenzen zu erkennen

Die Behebung von Konsistenzfehlern ist allerdings meist schwierig, da es oft nicht möglich ist, den korrekten Wert anhand der übrigen Quelldaten zu ermitteln. Daher ist es bei derartigen Fehlern oft nötig, Rücksprache mit den Verantwortlichen der betroffenen Datenquelle zu halten, damit die Konsistenzfehler behoben werden können.

2.7.4.3 Vollständigkeit

Fehlen bei den gelieferten Daten die Werte für einzelne oder mehrere Felder, so kann dies folgende Ursachen haben:

1. Es gibt in der realen Welt keinen Wert für dieses Attribut. Dies kann etwa die Haltbarkeitsdauer für ein nicht verderbliches Produkt sein.

2. Es existiert ein Wert für das Attribut; er wurde jedoch im Quellsystem nicht erfasst. Dies kann etwa an der niedrigen Gewichtung liegen, die das Quellsystem einem Attribut aufgrund unterschiedlicher Analysezwecke zukommen lässt. Jedenfalls handelt es sich in diesem Fall um einen Qualitätsmangel beim Quellsystem.
3. Es ist unklar, ob in der realen Welt ein Wert für das Attribut existiert. Dies kann etwa bei Telefonnummern oder E-Mail Adressen der Fall sein.

Jedenfalls sollten fehlende Werte im Data Warehouse einheitlich und, wenn möglich, durch einen NULL-Wert repräsentiert werden. Aufgrund unterschiedlicher Ausprägungen in den Quellsystemen, in denen etwa der NULL-Datentyp nicht existiert ist, ist es notwendig, zu erkennen, ob etwa eine leere Zeichenkette tatsächlich einen NULL-Wert repräsentiert. Dies erfolgt anhand festgelegter Konvertierungsregeln (vgl. Abschnitt 2.7.3).

2.7.4.4 Redundanzfreiheit

Im Gegensatz zur Korrektheit wird, um redundante Datenbestände zu erkennen, nicht direkt auf Vergleiche zur Realwelt zurückgegriffen. Vielmehr wird ein Abgleich der Datenquellen anhand jener Attributwerte versucht, die einen Vergleich der Datensätze ermöglichen. Ein Abgleich der Schlüsselwerte über mehrere Datenquellen hinweg ist aufgrund der unterschiedlichen Schlüsselzuordnung jedoch meist nicht durchführbar. Daher kann die Redundanzfreiheit nur aufgrund von Ähnlichkeitsvergleichen, etwa von Namen und Adressen festgestellt werden. Dazu existieren unter anderem folgende Ansätze:

Probabilistic Record Linkage Verfahren: Bei diesem Verfahren werden in Abhängigkeit der Ähnlichkeit zweier Attributwerte Gewichtungen vergeben. Die Summe der Gewichtungen aller Attributwerte eines Datensatzes ergibt die Wahrscheinlichkeit, dass beide Datensätze dieselbe Entität beschreiben und somit redundant vorhanden sind.

Sorted Neighbourhood Verfahren: Dieses Verfahren sortiert zunächst die Werte nach zu vergleichenden Attributen. Anschließend werden die jeweils nebeneinander liegenden Attribute auf Ähnlichkeit verglichen, um redundante Einträge zu ermitteln. Der Vorteil dieses Verfahrens liegt vor allem in der reduzierten Anzahl der notwendigen Vergleiche, da nur mehr die jeweils benachbarten Einträge verglichen werden. Um sicherzustellen, dass ähnliche Daten, je nachdem, ob der Vor- oder Nachname vorangestellt ist, richtig sortiert werden, sollten die Felder vor der Sortierung vereinheitlicht, und Attributwerte so weit wie möglich separiert werden (Lee u. a. 1999).

2.7.5 Ladephase

Nach dem Abschluss der Transformationsphase, in der die Daten angepasst und bereinigt wurden, folgt die Ladephase. Aufgabe der Ladephase ist es, die Daten in das Data Warehouse-Schema zu übertragen. Es wird zwischen dem ersten Beladungsprozess (initiale Beladung) und der regelmäßigen Aktualisierung der Daten im Data Warehouse unterschieden. Der initiale Beladungsprozess ist sehr ressourcenintensiv, da oft die Daten der vergangenen Jahre in einem Beladungsprozess in das Data Warehouse integriert werden. Die darauf folgenden inkrementellen Ladevorgänge sind weniger ressourcenintensiv, da nur jene Daten in das Data Warehouse geladen werden, die sich seit dem letzten Beladungsprozess verändert haben.

Der Beladungsprozess kann in weiterer Folge durch erworbenes Erfahrungswissen und eine verbesserte Kenntnis der Datenquellen effizienter gestaltet werden. Dies ist etwa durch optimierte Abfragen auf selektive Datenquellen, ein effizienteres *Monitoring* oder mit verbesserten Methoden zur Bereinigung der Quelldaten möglich.

Der Beladungsprozess sollte möglichst zu einem Zeitpunkt durchgeführt werden, an dem die Datenbank einer geringen Belastung ausgesetzt ist. Dies ist typischerweise nachts oder an Wochenenden der Fall. Aufgrund der zunehmenden Globalisierung und den dadurch beteiligten unterschiedlichen Zeitzonen ist ein derartiges Zeitfenster aber meist klein. Der Ladevorgang wird typischerweise online vorgenommen, das bedeutet, das Data Warehouse steht auch während des Ladevorgangs zur Verfügung. Sollte die Zeit für den Beladungsprozess zu knapp bemessen sein, ist es etwa durch Parallelisierung der Ladevorgänge möglich, den Beladungsprozess zu beschleunigen.

2.7.6 Analysephase

Die Analysephase dient der Auswertung der im Data Warehouse erfassten Daten. Diese Auswertung kann für die Benutzer der einzelnen Fachbereiche unterschiedlich sein. Zur Analyse werden zwei unterschiedliche Zugriffsmöglichkeiten auf ein Data Warehouse unterschieden, die im Folgenden erläutert werden.

2.7.6.1 Data Access

Ziel des Data Access ist es, Antworten auf spezifische Fragen zu geben, die sich aus dem Geschäftsbetrieb ergeben. So könnte eine Fragestellung etwa lauten, wie viele Stück von Produkt X in Woche fünf in Endverkaufsstelle Y verkauft wurden. Zur Ermittlung der Ergebnisse bietet sich entweder SQL (Structured Query Language) oder MDX (Multidimensional Expressions) an. Die Abfragesprache MDX bieten den Vorteil, dass sie auf mehrdimensionale Datenstrukturen, wie sie in Data Warehouses genutzt werden, spezialisiert ist (Whitehorn u. a. 2006, S. 31). Aufgrund des großen Verbreitungs- und Bekanntheitsgrades wird trotzdem häufig die zweidimensional orientierte Abfragesprache SQL verwendet. (Bauer & Günzel 2004, S. 97)

2.7.6.2 Online Analytical Processing

Unter Online Analytical Processing (OLAP) wird die Abfrage und Präsentation von Abfrageergebnissen in einem dynamischen Kontext verstanden. Das bedeutet, im Gegensatz zum Data Access stehen hier Fragestellungen im Vordergrund, die dynamische Auswertungen ermöglichen. Etwa „In welchem Bezirk macht Produkt X den größten Umsatz?“ oder „Wie hat sich der Umsatz im Vergleich zu den letzten Monaten entwickelt?“.

Damit ein System als OLAP-System gilt, muss es folgende fünf Kriterien erfüllen, die kurz als Fast Analysis of Shared Multidimensional Information (FASMI) bezeichnet werden (Pendse & Creeth 2008):

- **Geschwindigkeit (Fast)**: OLAP-Systeme sollten die Anfragen von Anwendern in weniger als fünf Sekunden beantworten können. Der Durchschnitt der Abfragezeit sollte aber deutlich darunter liegen. Eine Antwortzeit von über 30 Sekunden gilt darüber hinaus als nicht mehr akzeptabel, da bei einer Bearbeitungszeit von über 30 Sekunden viele Benutzer dazu neigen das System als abgestürzt einzustufen und dementsprechend nicht mehr auf die Anfragebeantwortung warten. Umfragen haben gezeigt, dass hohe Antwortzeiten oft zu Akzeptanzproblemen von OLAP-Systemen führen.
- **Analysemöglichkeit**: Das OLAP-System muss eine anwenderfreundliche und intuitive Analyse der Daten ermöglichen. Dies bedeutet, dass das OLAP-System auch Endbenutzern ohne Programmierkenntnisse die Anfertigung von Berichten nach

benutzerdefinierten Wünschen ermöglichen muss. Daneben ist auch die Einbindung externer Werkzeuge wie Excel zur Analyse und Auswertung zulässig.

- **Sicherheit**: OLAP-Systeme müssen einen sicheren Mehrbenutzerbetrieb ermöglichen. Das bedeutet, dass Zugriffsrechte auf feinster Granularitätsstufe vergeben werden können müssen, um sicherzustellen, dass die Nutzer nur Daten einsehen können, für die sie eine Berechtigung besitzen. Bieten die OLAP-Systeme auch Schreibmöglichkeiten auf die Datenbank, so muss eine Mehrbenutzerkontrolle implementiert sein.
- **Multidimensionalität**: Die Multidimensionalität bei OLAP ist der Hauptunterschied im Vergleich zu anderen Datenzugriffsmethoden und somit die Kerneigenschaft von OLAP. Damit diese Eigenschaft erfüllt ist, muss es möglich sein, beliebige Dimensionen zu kombinieren und deren Hierarchie für Auswertungen zu nutzen. Die zugrundeliegende Datenbanktechnologie ist hingegen für dieses Kriterium nicht relevant.
- **Kapazität (Information)**: OLAP-Systeme werden hier nach der maximalen Datenkapazität, die sie verwalten können, kategorisiert. Nicht die Größe des Dateisystems, das verwaltet werden kann, sondern die Anzahl der Datensätze ist für die Kapazität wesentlich. Darüber hinaus ist es erforderlich, dass derartige Systeme auch bei stetig steigenden Datenmengen stabile Antwortzeiten liefern.

3 Anforderungen und Rahmenbedingungen

Die Anforderungen für die erstellte BI-Lösung haben sich aus der Aufgabenstellung, der im Grundlagenteil herangezogenen Literatur, sowie aus den Erkenntnissen des evolutionären Prototyping (vgl. 1.2) ergeben. Die Anforderungen werden im Folgenden in funktionale (vgl. 3.1) und nicht funktionale Anforderungen (vgl. 3.2) unterteilt. Abschnitt 3.3 erläutert schließlich die Rahmenbedingungen, unter denen die Umsetzung der Anforderungen erfolgte.

3.1 Funktionale Anforderungen

Unter funktionalen Anforderungen eines Systems versteht man die erwartete Reaktion des Systems auf Eingaben. Das bedeutet, eine Funktionalität beschreibt, zu welchen Ausgabedaten eine Eingabe in das System führen soll. Die funktionalen Anforderungen beschreiben ausdrücklich nicht, wie der gewünschte Zustand erreicht wird, sondern nur, welche Ausgabedaten vom System auf bestimmte Eingaben erwartet werden. (Versteegen 2002, S. 19 f.)

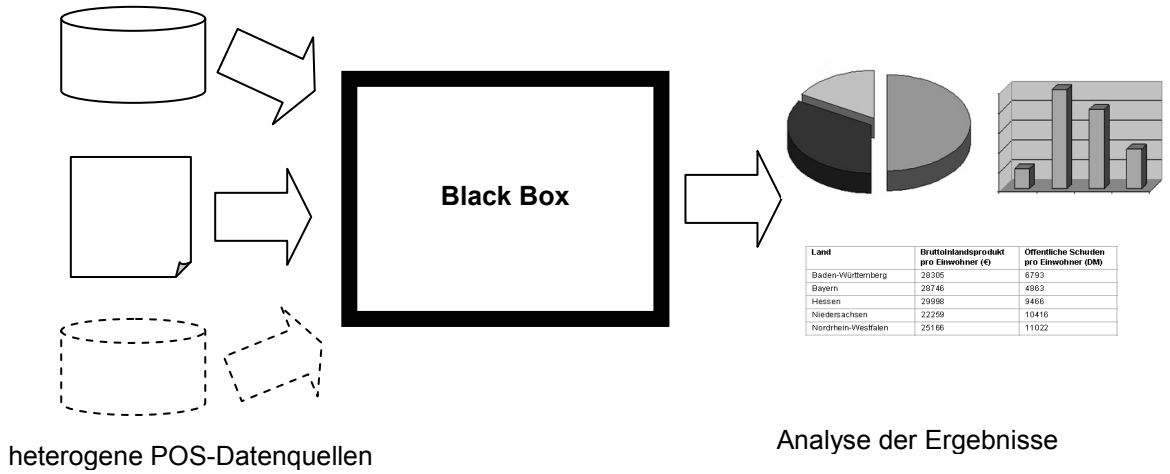


Abbildung 33: Blackbox Darstellung der BI-Lösung
Die gestrichelte Darstellung steht stellvertretend für eine beliebige Anzahl an Datenquellen des BI-Systems.

In Abbildung 33 sind die funktionalen Anforderungen vereinfacht dargestellt. Eingangsdaten der BI-Lösung, die hier als Black Box dargestellt ist, sind die Datenquellen, die an die BI-Lösung angebunden werden müssen. Die Black Box übernimmt die weitere Bearbeitung der Daten, die letztlich zur Bereitstellung der Daten für den Endbenutzer führt.

Wesentlich ist, dass, wie in der Grafik durch ein gestricheltes Quellsystem angedeutet, die Anzahl der Quellsysteme variabel ist. Bieten weitere Partner in der Supply Chain Daten zur Endkonsumentennachfrage an, so können diese Daten in die POS BI-Lösung integriert werden.

Um das Ergebnis des Beladungsprozesses zu veranschaulichen bietet sich eine grafische Darstellung der Analyseergebnisse an. Diese Darstellung kann vom Endbenutzer angepasst und benutzerspezifisch erweitert werden. Die Architektur des Beladungsprozesses bleibt dabei für den Endbenutzer transparent.

Somit unterteilen sich die funktionalen Anforderungen in zwei Teile. Dies sind zum einen die in den Abschnitten 3.1.1 bis 3.1.3 angeführten Anforderungen an den Beladungsprozess des POS-Data Warehouse und zum anderen die in den Abschnitten 3.1.4 bis 3.1.6 angeführten Anforderungen an die Analyse der Ergebnisse.

3.1.1 Hinzufügen und Entfernen von Datenquellen

- **Funktionalität:** Es ist für den Benutzer mit geringem Anpassungsaufwand möglich, neue Datenquellen zum Beladungsprozess hinzuzufügen.
- **Eingabedaten:** Das Hinzufügen neuer Daten erfolgt durch das Anlegen eines entsprechenden Mappings im ETL-Prozess zur Beladung der Staging-Area (vgl. 5.4.4). Im Mapping wird die Datenstruktur der Datenquellen an die Datenstruktur der Zieltabellen angepasst.
- **Ausgabedaten:** Die Daten der jeweiligen Datenquelle werden in den Beladungsprozess integriert.
- **Fehlerbehandlung:** Wird eine ungültige Datenquelle angegeben oder ein ungültiges Mapping hinzugefügt, wird die betroffene Datenquelle bei der Integration der POS-Daten umgangen und es ergeht eine dementsprechende Fehlermeldung.

3.1.2 Integration der Datenquellen

- **Funktionalität:** Die aus unterschiedlichen Datenquellen abgeleiteten Daten werden vereinheitlicht und in ein zentrales Schema übertragen, das als Staging-Area bezeichnet wird.
- **Eingabedaten:** Heterogene Daten aus Datenquellen.

- **Ausgabedaten:** Integrierter Datenbestand der Staging-Area
- **Fehlerbehandlung:** Sollten eine oder mehrere Datenquellen ungültige Daten liefern, die eine Integration der Daten nicht ermöglicht, so erfolgt eine Warnung.

3.1.3 Beladung des Data Warehouse

- **Funktionalität:** Befüllung des Data Warehouse-Schemas mit den Daten der Staging-Area. Nur die im Data Warehouse integrierten Daten stehen dem Endbenutzer zur Analyse zur Verfügung (vgl. Abbildung 36).
- **Eingabedaten:** Keine Endbenutzereingabe erforderlich, da der Beladungsvorgang die Daten der Staging-Area verwendet.
- **Ausgabedaten:** Neben den aktualisierten Daten im Data Warehouse liefert der Beladungsvorgang Metadaten, die festhalten zu welchem Zeitpunkt welche Dimension mit wie vielen Datensätzen beladen wurde. Zusätzlich wird der aktuelle Status des Beladungsprozesses auf „RUNNING“ gesetzt. Ist der Beladungsprozess erfolgreich abgeschlossen, wird der Status auf „COMPLETED“ gesetzt.
- **Fehlerbehandlung:** Tritt während des Beladungsvorganges ein Fehler auf, so wird der Beladungsvorgang abgebrochen und der Status des Beladungsprozesses auf „FAILED“ gesetzt.

3.1.4 Bereitstellung der POS-Daten

- **Funktionalität**: Zusammengefasste Anzeige der POS-Daten
- **Eingabedaten**: Es sind keine Eingaben erforderlich, es können jedoch Filter auf Produkt- oder Handelskettenebene gesetzt werden.
- **Ausgabedaten**: Information über die Endkonsumentennachfrage aggregiert auf Monats- und Tagesebene.
- **Fehlerbehandlung:** Sollte die Berechnung der Aggregate nicht möglich sein, wird statt den Analyseergebnissen eine Fehlermeldung angezeigt.

3.1.5 Multidimensionale Analyse der POS-Daten

Um eine multidimensionale Analyse der POS-Daten zu ermöglichen, ist es zunächst notwendig, die dafür notwendigen Dimensionen festzulegen. Diese Entscheidung ist primär von den Endbenutzern der BI-Lösung zu treffen. Dazu ist es zunächst sinnvoll, dass die

Endanwender explizit jene Fragen auflisten, die später von der BI-Lösung beantwortet werden sollen. Es ist darauf zu achten, dass die gestellten Fragen auch mit den Ergebnissen der Literatur im Bezug auf die Reduzierung des Bullwhip-Effektes übereinstimmen. Bei der Anforderungsanalyse haben sich zunächst folgende Fragen ergeben:

1. Wie haben sich die Verkaufszahlen aller Produkte in den letzten Monaten entwickelt?
2. Wie viele Stück eines Produktes wurden vor zwei Tagen an allen Verkaufsstellen abgesetzt?
3. Wie viele Stück eines Produktes wurden vor zwei Tagen in einer bestimmten Region abgesetzt?
4. Welche Artikel haben sich in der letzten Kalenderwoche besonders gut verkauft?
5. Wie entwickelt sich der Absatz eines bestimmten Produkttyps im Vergleich zu einem anderen (etwa DVD im Vergleich zu BluRay)?

Zu beachten ist, dass für eine abschließende Analyse eine wesentlich größere Anzahl an Fragen zu erwarten ist. Aber auch diese geringe Anzahl an Fragen hilft dem Data Warehouse-Designer, die nötigen Dimensionen und Hierarchien festzulegen. Dieses Vorgehen entspricht in vereinfachter Form dem als *Business Exploration* bezeichneten Vorgehensmodell zur Identifikation der im Data Warehouse benötigten Dimensionen und Hierarchien. (Westerman 2001, S. 59 ff.)

Wie sich zeigt, ist zur Beantwortung der Fragen 1 bis 4 eine Zeitdimension notwendig, da sich diese Fragen auf einen bestimmten Zeitraum beziehen. Zur Beantwortung der Fragen zwei bis vier ist darüber hinaus eine Produktdimension notwendig, die es ermöglicht, eine Anfrage zusätzlich auf bestimmte Produkte einzuschränken. Zur Beantwortung von Frage 3 ist weiters eine Dimension erforderlich, die eine Unterscheidung nach den Verkaufsstellen ermöglicht. Frage 5 schließlich verlangt nach einer Gliederung der Produkttypen. Dazu ist keine neue Dimension erforderlich, da die Produkttypen in die Hierarchie der Produkte eingefügt werden können.

- **Funktionalität**: Die BI-Lösung ermöglicht es, die POS-Daten nach unterschiedlichen Kriterien auszuwerten. Diese wurden aus der Beantwortung der fünf charakteristischen Fragen abgeleitet. Die Funktionalität umfasst daher Auswertungen nach Zeit, Produkt und Verkaufsstelle. Darüber hinaus soll ein *Drill-down* auf einzelne Title, Endverkaufsstellen oder Verkaufsvorgänge pro Tag möglich sein. Das

bedeutet, der Endbenutzer kann etwa von auf Monatsebene aggregierten Verkaufszahlen einen *Drill-down* auf Wochen- oder Tagesebene durchführen.

- **Eingabedaten**: Gewünschte Dimensionsattribute und gewünschter Datenbereich
- **Ausgabedaten**: Information über die Endkonsumentennachfrage gemäß den angegebenen Kriterien.
- **Fehlerbehandlung**: Führt die Auswahl der Filter zu einer leeren Ergebnismenge, so wird eine dementsprechende Warnung angezeigt. Ebenso, wenn die Verbindung zum Data Warehouse nicht möglich ist.

3.1.6 Benutzerdefinierte Ansicht der POS-Daten

- **Funktionalität**: Der Benutzer ist in der Lage, eigenständig Abfragen an das System zu stellen oder eigenständig Berichte zu erstellen
- **Eingabedaten**: Abfrage in entsprechender Syntax oder Auswahl der entsprechenden Parameter zur Anfertigung eines benutzerdefinierten Berichts.
- **Ausgabedaten**: Die jeweils vom Benutzer angeforderten Daten.
- **Fehlerbehandlung**: Wählt der Benutzer ungültige Kombinationen, etwa von Filterkriterien, so erhält der Benutzer eine dementsprechende Fehlermeldung. Eine derartige Fehlermeldung ergeht auch, wenn im zugrundeliegenden BI-System ein Fehler aufgetreten ist, der eine weitere Auswertung nicht mehr möglich macht.

3.2 Nichtfunktionale Anforderungen

Neben den funktionalen Anforderungen sind auch die nicht-funktionalen Anforderungen vor der Entwicklung von Software zu präzisieren. Unter nicht-funktionalen Anforderungen versteht man vor allem qualitative Anforderungen an das zu erstellende System. Die nichtfunktionalen Anforderungen tragen erheblich zur Benutzerzufriedenheit bei. (Versteegen 2002, S. 19 f.)

Aktualität der Daten: Die von der BI-Lösung zur Verfügung gestellten Daten sollten tagesaktuell sein. Das bedeutet, der Beladungsprozess sollte alle 24 Stunden ausgeführt werden. Aufgrund des Fokus der POS-Daten auf Europa würden sich, um die Auslastung des Systems tagsüber nicht zu stören, die Nachtstunden anbieten.

Leistung: Gemäß den FASMI Kriterien (vgl. 2.7.6) darf das System mit zunehmender Datenmenge nicht signifikant an Leistung verlieren und muss zum anderen in der Lage sein,

Abfragen von Benutzern in maximal 30 Sekunden zu beantworten, wobei die durchschnittliche Antwortzeit unter fünf Sekunden liegen soll.

Portierbarkeit: Das entwickelte System muss portierbar sein; das bedeutet, es sollte auf unterschiedlichen Plattformen einsetzbar sein und nicht an eine spezielle Umgebung gebunden sein.

3.3 Rahmenbedingungen bei der Entwicklung der Business Intelligence-Lösung

Die in Abschnitt 3.1 und 3.2 angeführten Anforderungen wurden im Unternehmensumfeld umgesetzt. Aus diesem Grund ergaben sich Rahmenbedingungen, die bei der Entwicklung der BI-Lösung berücksichtigt werden mussten. Diese Rahmenbedingungen erstreckten sich zum einen auf Datenquellen, die zur Beladung des Data Warehouse zu verwenden waren (vgl. 3.3.1) und zum anderen auf Werkzeuge, die für die Umsetzung heranzuziehen waren (vgl. 3.3.2).

3.3.1 Datenquellen

Für den ETL-Prozess wurden von *Sony DADC* zwei Datenquellen vorgegeben. Beide Datenquellen bieten Daten zu Verkaufsvorgängen, die dem Unternehmen im Rahmen der Zusammenarbeit der Supply Chain-Akteure zur Verfügung gestellt wurden. Da diese Daten alleine für den Zweck des Data Warehouse nicht ausreichen würden, wurden zusätzlich noch Daten einbezogen, die nicht unternehmensintern verfügbar waren. Diese externen Datenquellen sind somit nicht Teil der Rahmenbedingungen. Sie werden jedoch aufgrund der Übersichtlichkeit auch in diesem Abschnitt angeführt.

Im Folgenden werden die Eigenschaften der zur Verfügung stehenden Datenquellen näher erläutert und hinsichtlich ihrer Eignung gemäß der in Abschnitt 2.7.1 genannten Anforderungen geprüft.

3.3.1.1 Interne Datenquellen

Supply Chain intern werden POS-Daten derzeit von drei Einzelhandelsketten zur Verfügung gestellt, die von zwei unterschiedlichen Datenquellen geliefert werden.

Datenquelle 1 (*Source 1*)

Die erste Datenquelle wird von einer großen Filmvertriebsgesellschaft zur Verfügung gestellt. Die *Datenquelle 1* umfasst somit alle Umsätze, die mit Discs erzielt werden, die bei *Sony DADC* produziert werden und deren Rechteinhaber die Filmvertriebsgesellschaft ist. Derzeit werden die Endverkaufsdaten von folgenden Einzelhandelsketten zur Verfügung gestellt:

- **Dirk Rossmann GmbH:** Die *Dirk Rossmann GmbH* (kurz: *Rossmann*) ist mit etwa 1500 Verkaufsstellen die drittgrößte deutsche Drogeriemarktkette. Da das Sortiment der *Dirk Rossmann GmbH* breit gefächert ist, finden sich neben den klassischen Drogeriemarktprodukten in der Sparte Technik und Multimedia auch zahlreiche DVDs. (Rossmann 2010)

- **Netto Marken-Discount AG & Co. KG:** Die *Netto Marken-Discount AG & Co. KG* (kurz: *Netto*) betreibt derzeit deutschlandweit etwa 4000 Verkaufsstellen. *Netto* ist damit die drittgrößte deutsche Handelskette im Discount-Segment. Das Sortiment von Netto legt den Schwerpunkt auf Lebensmittel, daneben sind aber auch DVDs im Angebot enthalten. (Netto 2010)

Die Endverkaufsdaten dieser beiden Handelsketten werden in einer zentralen Tabelle einer relationalen Datenbank gesammelt. Diese Tabelle wird im Folgenden als „ORDER_LINE“ bezeichnet. Um die tatsächlichen Verkaufszahlen feststellen zu können, ist es nötig, die Daten zu filtern, da nicht alle Daten der Datenquelle einen für den Data Warehouse-Zweck relevanten Verkaufsvorgang darstellen. Zusätzlich werden in der Tabelle auch Produktrückgaben (Retouren) festgehalten. Diese dienen gleichzeitig auch der Fehlerkorrektur, sofern Verkaufsvorgänge fehlerhaft in der Datenbank gelistet wurden. Somit ist es nur möglich, die Verkaufsvorgänge eines Kalendertages korrekt zu extrahieren, wenn jeweils alle Kauf- und Umtausch- oder Korrektureinträge eines Tages aggregiert werden.

Datenquelle 1 kann als eine selektierbare, aber nicht kooperative, Datenquelle (vgl. Abschnitt 2.7.1) angesehen werden. Die *Datenquelle 1* ist nicht kooperativ, da auf ihr keine Schreibrechte bestehen, die es etwa ermöglichen würden, einen Trigger zu hinterlegen. Anhand eines fortlaufenden Schlüssels (*Sequence-ID*) ist es möglich, neue Einträge zu identifizieren. Dazu ist es notwendig, die höchste *Sequence-ID* des letzen Beladungsprozesses vorzuhalten, um beim nächsten Beladungsprozess nur Einträge mit einer höheren *Sequence-ID* selektieren zu können.

Datenquelle 2 (*Source 2*)

Datenquelle 2 liefert jene Umsätze, die mit Discs erzielt werden, die bei *Sony DADC* produziert werden und deren Rechteinhaber ein großer Entwickler und Herausgeber von Unterhaltungssoftware ist. Derzeit werden von *Datenquelle 2* Endverkaufsdaten der folgenden Einzelhandelskette zur Verfügung gestellt:

- **El Corte Inglés, S.A.:** Die spanische Einzelhandelskette *El Corte Inglés, S.A.* ist gemessen am Umsatz die größte Handelskette in Spanien. *El Corte Inglés, S.A.* verfügt über zahlreiche Marken die nach außen als eigene Handelsketten auftreten. Den größten Teil des Umsatzes erwirtschaften allerdings die 67 Kaufhäuser unter der Marke *Grandes almacenes El Corte Inglés.* Da das Produktsortiment dieser Kaufhäuser sehr weit gefächert ist, finden sich auch zahlreiche Discs darunter. (El Corte Inglés 2009)

Datenquelle 2 ist gemäß der Kategorisierung in Abschnitt 2.7.1 als Schnappschuss-Datenquelle anzusehen, da die Endverkaufsdaten von *Datenquelle 2* als EDIFACT-Dateien (vgl. 6.1.4) zur Verfügung gestellt werden. Um zu erkennen, ob neue Daten vorliegen, ist es daher erforderlich, die Datei zur Gänze zu übertragen und zu öffnen. Anhand des eindeutigen Schlüssels jeder EDIFACT-Datei, der im Beladungsprozess vorgehalten wird, kann letztlich erkannt werden, ob die in der Datei aufgezeichneten Verkaufsvorgänge tatsächlich noch nicht geladen wurden.

3.3.1.2 Externe Datenquellen

Da in den Supply Chain-internen Datenquellen keine oder nur lückenhafte Information zu den Verkaufsstellen angeboten wird, obwohl diese Daten aber für den Zweck des Data Warehouse wesentlich sind, wurde dazu eine Supply Chain externe Datenquelle herangezogen:

Global Electronic Party Information Register (GEPIR)

Das *Global Electronic Party Information Register* (*GEPIR*) stellt eine weltweit verteilte Datenbank zu Unternehmen und Produkten in über 100 Ländern dar. Diese Datenbank wird vom Unternehmen GS1 (Global Standards 1) betrieben. Die Aufgabe von GS1 ist es, Standards zu entwickeln und zu verwalten, die unternehmensübergreifend gültig sind. Derartige Standards haben besonders im Supply Chain-Management Bedeutung, da hier die

Zusammenarbeit mehrerer Unternehmen in den Vordergrund tritt. Für den vorliegenden Zweck sind besonders Daten zu den Endverkaufsstellen, wie etwa Standort und Unternehmenszugehörigkeit, sowie Daten zu verkauften Artikeln interessant. Das GEPIR kann hier zu folgenden Schlüsseln Daten liefern:

- *GLN (Global Location Number):* Eine *GLN* stellt einen weltweit eindeutigen Schlüssel zu einem bestimmten Unternehmen oder einem geografischen Standort dar. Zu beachten ist, dass auch unternehmensintern mehrere *GLN* vergeben werden können, wenn ein Unternehmen auf mehrere Standorte verteilt ist. (GS1 2010, S. 12) Der Aufbau der *GLN* ist an die *GTIN* (vgl. Abbildung 34) angelehnt.
- *GTIN (Global Trade Item Number):* Die *GTIN* stellt einen weltweit eindeutigen Schlüssel zu einem bestimmten Produkt dar. Weichen die Produkteigenschaften geringfügig ab, so muss in der Regel eine neue *GTIN* vergeben werden. Der Aufbau der *GTIN* ist in Abbildung 34 illustriert.

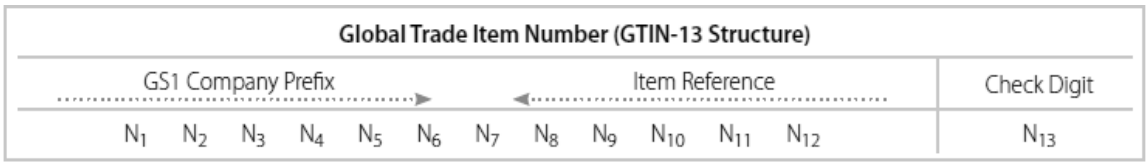


Abbildung 34: Aufbau GTIN (GS1 2010, S. 10)
Die ersten sieben Stellen dienen zur Identifikation des Unternehmens, Stelle sieben bis zwölf zur Identifikation des Produktes, Stelle 13 ist die Prüfziffer. Bei der GLN tritt anstelle der Produktidentifikationsnummer die Standortidentifikationsnummer.

Beide Schlüssel bestehen aus 13 Ziffern, wobei der erste Teil jeweils unternehmensspezifisch und die weiteren Stellen standort- oder produktspezifisch sind. Zusätzlich besteht jeder Schlüssel an der letzten Stelle aus einer Prüfziffer. Die Prüfziffer wirkt sich positiv auf die Datenqualität aus, da so Fehler auf Instanz-Ebene (vgl. 2.6.2), wie etwa Tippfehler, einfach erkannt werden können.

Abbildung 35 verdeutlicht den Einsatz der *GLN* und *GTIN* entlang der Supply Chain. Deutlich zu erkennen ist, dass der Produzent, das Distributionszentrum und die Verkaufsstelle über eine eigene GLN verfügen. Die in der Supply Chain versendeten Produkte verfügen über eine GTIN, anhand der das Produkt in allen Stufen der Supply Chain eindeutig identifiziert werden kann.

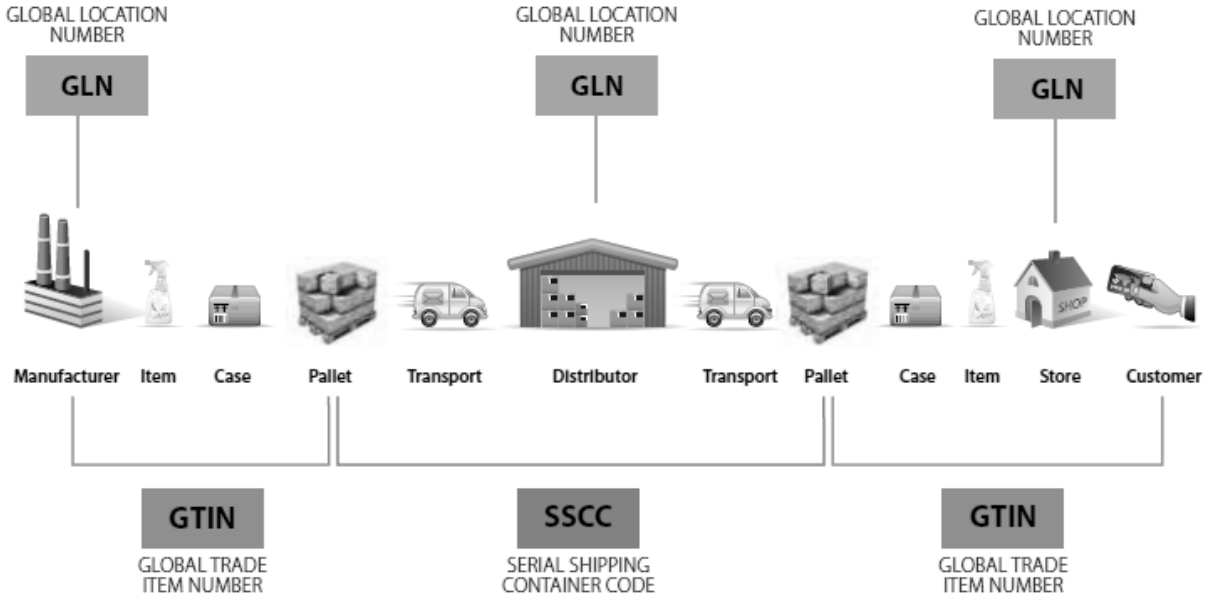


Abbildung 35: Einsatz von GLN und GTIN in der Supply Chain (GS1 2010, S. 7)[4]

Die Datenquelle *GEPIR* hat Nachrang gegenüber den Supply Chain-internen Datenquellen, da die internen Datenquellen gewöhnlich einen höheren Detaillierungsgrad aufweisen. Das bedeutet, nur wenn die Supply Chain-internen Datenquellen keine Daten liefern, wird die Datenquelle *GEPIR* benutzt, um etwa die Standortdaten zu einer *GLN* abrufen zu können.

Die Datenquelle *GEPIR* wird als nicht kooperative Schnappschuss-Datenquelle eingestuft, da die GEPIR-Daten in einfachen Dateien im XML (Extensible Markup Language)-Format übertragen werden (vgl. 6.1.3). Pro XML-Datei wird weiters nur ein Datensatz übertragen. Um zu erkennen, ob sich seit dem letzten Beladungsprozess Daten geändert haben, ist es somit erforderlich jeden Datensatz einzeln zu überprüfen. Da pro Beladungsvorgang nur wenige Daten von *GEPIR*, wie etwa Daten über neu registrierte Endverkaufsstellen, abgerufen werden und eine Änderung der geladenen Datensätze nur selten zu erwarten ist, erfüllen die GEPIR-Daten den Data Warehouse-Zweck. Für den bisher benötigten Umfang stehen die GEPIR-Daten darüber hinaus kostenlos zur Verfügung, sodass die Kriterien von Abschnitt 2.6 als erfüllt gelten.

[4] Der in der Abbildung 35 zusätzlich angeführte SSCC (Serial Shipping Container Code) dient der Zusammenfassung von Produkten zu Verpackungseinheiten. Aufgrund der Ausrichtung des Informationssystems auf den Absatz beim Einzelhändler ist der SSCC für das Informationssystem nicht relevant.

3.3.1.3 Eigenschaften der verwendeten Datenquellen

In Tabelle 7 wird eine Übersicht über die Eigenschaften der verwendeten Datenquellen gemäß der in Abschnitt 2.6 angeführten *Eigenschaften von Datenquellen* gegeben. Es zeigt sich, dass alle Datenquellen für die Verwendung geeignet sind, da die Datenquellen den Zweck des Data Warehouse erfüllen. Darüber hinaus ist die Datenaktualität mit einer Verzögerung von etwa 24 bis 72 Stunden ausreichend. Da die externe Datenquelle im vorliegenden Fall kostenlos zur Verfügung steht, war keine Abwägung zwischen Kosten und Nutzen notwendig.

Datenquelle	Zweck	Art der Datenquelle	Datenaktualität	Preis
Source 1	Endverkaufsdaten	Selektierbar, nicht kooperativ	24 bis 36 Stunden Verzögerung	Supply Chain-intern
Source 2	Endverkaufsdaten	Schnappschuss, nicht kooperativ	24 bis 36 Stunden Verzögerung	Supply Chain-intern
GEPIR	Daten zu Verkaufs-stellen und Produkten	Schnappschuss, nicht kooperativ	unbekannt	kostenlos

Tabelle 7: Übersicht über die Eigenschaften der Datenquellen

3.3.2 Werkzeuge

Zur Implementierung der BI-Lösung wurden von *Sony DADC* ein Datenbanksystem sowie eine OLAP-Anwendung bereitgestellt (vgl. 1.1.2):

Relationale Datenbank Oracle 10g

Als Plattform sowohl für die Staging-Area als auch für das Data Warehouse wurde von *Sony DADC* ein *relationales Data Base Management System (RDBMS)* der Firma Oracle in Version 10g zur Verfügung gestellt.

Oracle Business Intelligence Suite - Enterprise Edition Plus

Als Werkzeug zur Anzeige und Auswertung der im Data Warehouse vorgehaltenen Daten wurde von *Sony DADC* das Tool *Oracle Business Intelligence Suite - Enterprise Edition Plus (OBIEE)* in der Version 10.1.3.4.1 zur Verfügung gestellt (vgl. Oracle 2010). Dieses Werkzeug dient in der vorliegenden Architektur als OLAP-System (vgl. 2.7.6).

4 Architektur

Grundlage für die *Architektur* der BI-Lösung sind die Erkenntnisse von Kapitel 2 im Zusammenhang mit den in Kapitel 3 erläuterten Anforderungen und Rahmenbedingungen von *Sony DADC*. Im Besonderen beruht die Umsetzung auf den in Abschnitt 2.5 diskutierten Architekturvarianten und den in Abschnitt 2.7 vorgestellten Techniken zur Beladung eines Data Warehouse.

Im Gegensatz zur Anforderungsdefinition in Kapitel 3 wird in diesem Abschnitt auf die gewählte Architektur zur Umsetzung der in Abschnitt 3.1.1 bis 3.1.3 genannten Anforderungen eingegangen. Die Architektur lehnt sich an die in Abschnitt 2.4 bis 2.7 erzielten Erkenntnisse zu BI-Lösungen im Supply Chain-Management und an den bisher bei *Sony DADC* für andere Data Warehouse Projekte verwendeten Beladungsprozess an. Aufgrund der Ergebnisse von Abschnitt 2.5.6 wurde eine zentrale Data Warehouse-Architektur gewählt, da diese Architekturvariante am besten zur Reduzierung des Bullwhip-Effektes geeignet ist.

Die Anforderungen 3.1.4 bis 3.1.6 werden vom Analysewerkzeug OBIEE, basierend auf dem Data Warehouse-Schema, implementiert. Die Architektur für deren Umsetzung ist somit spezifisch für OBIEE und wird in diesem Kapitel nicht näher ausgeführt.

Zunächst wird in Abschnitt 4.1 ein Überblick über die Architektur des Beladungsprozesses gegeben um in Abschnitt 4.2 den Ablauf des Beladungsprozesses anhand von UML-Diagrammen zu verdeutlichen.

4.1 Überblick über den Beladungsprozess

In Abbildung 36 ist die Architektur des Beladungsprozesses zusammenfassend dargestellt. Die Architektur ist neben den Datenquellen in *Front-Room* und *Back-Room* unterteilt. Der *Back-Room* dient der Aufbereitung der Daten, ist aber für keine weiteren Anwendungen zugänglich. Der *Front-Room* wird in der vorliegenden Architektur durch das Data Warehouse realisiert, das für den Zugriff auf Analysewerkzeuge optimiert ist. (Kimball u. a. 1998, S. 609 ff.)

Die Architektur des gewählten Beladungsprozesses entspricht den in Abschnitt 2.7 angeführten Phasen des Data Warehousing. Die im Beladungsprozess verwendete Staging-Area ist vergleichbar mit der bei Kimball u. a. (1998, S. 15 f.) beschriebenen Data Staging-Area und dem bei Bauer und Günzel (2004, S. 48 f.) beschriebenen Arbeitsbereich.

Die Staging-Area dient somit „vornehmlich der Integration von Daten aus heterogenen Quellen“ (Bauer & Günzel 2004, S. 48 f.). Das bedeutet, die Staging-Area ist ein Hilfsmittel für die Bereinigungs-, Transformations- und Ladephase.

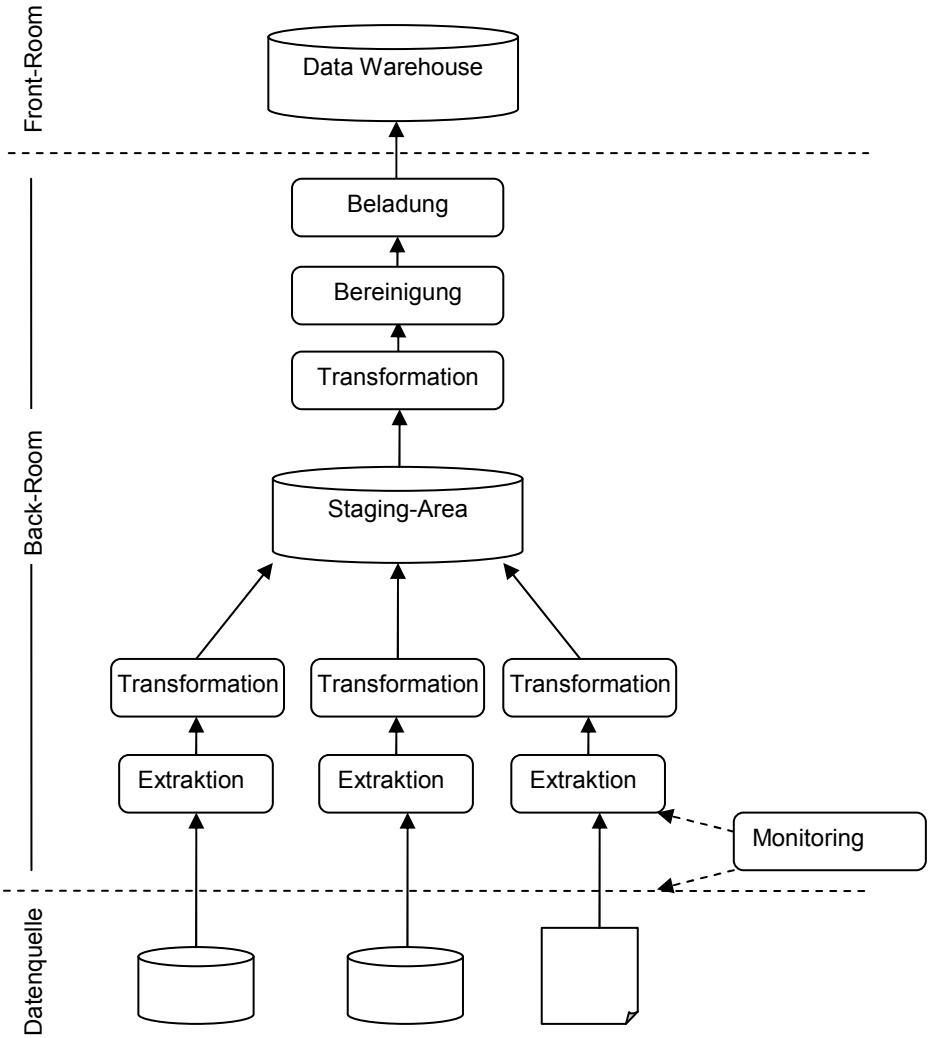


Abbildung 36: Architektur des Beladungsprozesses.
Die Grafik ist unterteilt in die *Datenquellen*, den *Back-Room*, der als Arbeitsbereich des Beladungsprozesses dient, und den *Front-Room*, der für die Datenanalyse zur Verfügung steht.

4.1.1 Monitoring und Extraktion

Der Aufbau des Beladungsprozesses beginnt mit dem *Monitoring*. Das *Monitoring* dient dazu, jene Daten zu erkennen, die vom folgenden Extraktionsprozess erfasst werden sollen. Dabei muss, je nach Eigenschaften der Datenquelle, ein passendes *Monitoringverfahren* angewendet werden, um die jeweils geänderten Daten zuverlässig erkennen zu können (vgl. 2.7.1).

In der anschließend folgenden Extraktionsphase werden jene Daten aus den Quelldatenbeständen extrahiert, die vom Monitoring als neu oder verändert angezeigt wurden. Darüber hinaus verwendet das Extraktionsverfahren Filter, die nur jene Attribute der Daten extrahieren, die für den Data Warehouse-Zweck relevant sind.

4.1.2 Beladungsprozess der Staging-Area

Im Anschluss an das *Monitoring* werden die extrahierten Daten transformiert und in die Staging-Area geladen. Die Integration der Daten in die Staging-Area erfolgt, damit die Anzahl und Art der Datenquellen für den nachfolgenden Teil des Beladungsprozesses transparent sind. Das bedeutet, nur jene Prozesse, die vor der Beladung der Staging-Area ausgeführt werden, müssen an die Anzahl und Art der Datenquellen angepasst werden. Alle der Staging-Area nachgelagerten Prozesse bauen auf das einheitliche Schema der Staging-Area auf und sind somit von der Anzahl und Art der Quellsysteme unabhängig.

4.1.3 Beladungsprozess des Data Warehouse

Ist der ETL-Prozess zur Beladung der Staging-Area abgeschlossen, kann der Transformations-, Bereinigungs- und Beladungsprozess des POS-Data Warehouse beginnen. Dazu werden die Daten in das Data Warehouse-Schema transformiert und, falls notwendig, bereinigt. Anschließend wird das Data Warehouse beladen.

4.2 Ablauf des Beladungsprozesses

Neben der Darstellung der Architektur in Abbildung 36 wurde zur Veranschaulichung des Ablaufes des Beladungsprozesses in Abbildung 37 ein Aktivitätsdiagramm erstellt. Für ein Aktivitätsdiagramm ist charakteristisch, dass alle Schritte mit einer Verarbeitung verbunden sind. Eine Aktivität wird erst verlassen, sobald die mit diesem Zustand verbundene Bearbeitung beendet ist. Mithilfe eines Splittings ist es in einem Aktivitätsdiagramm möglich,

Verarbeitungsschritte als parallel zu kennzeichnen. In diesem Fall kann bei der Implementierung entschieden werden, ob die Verarbeitung tatsächlich parallel oder sequenziell erfolgt. Wesentlich ist, dass nach Ende eines parallelen Prozesses alle parallelen Aktivitäten beendet sein müssen, damit die nächste Aktivität beginnt. (Balzert 1999, S. 85)

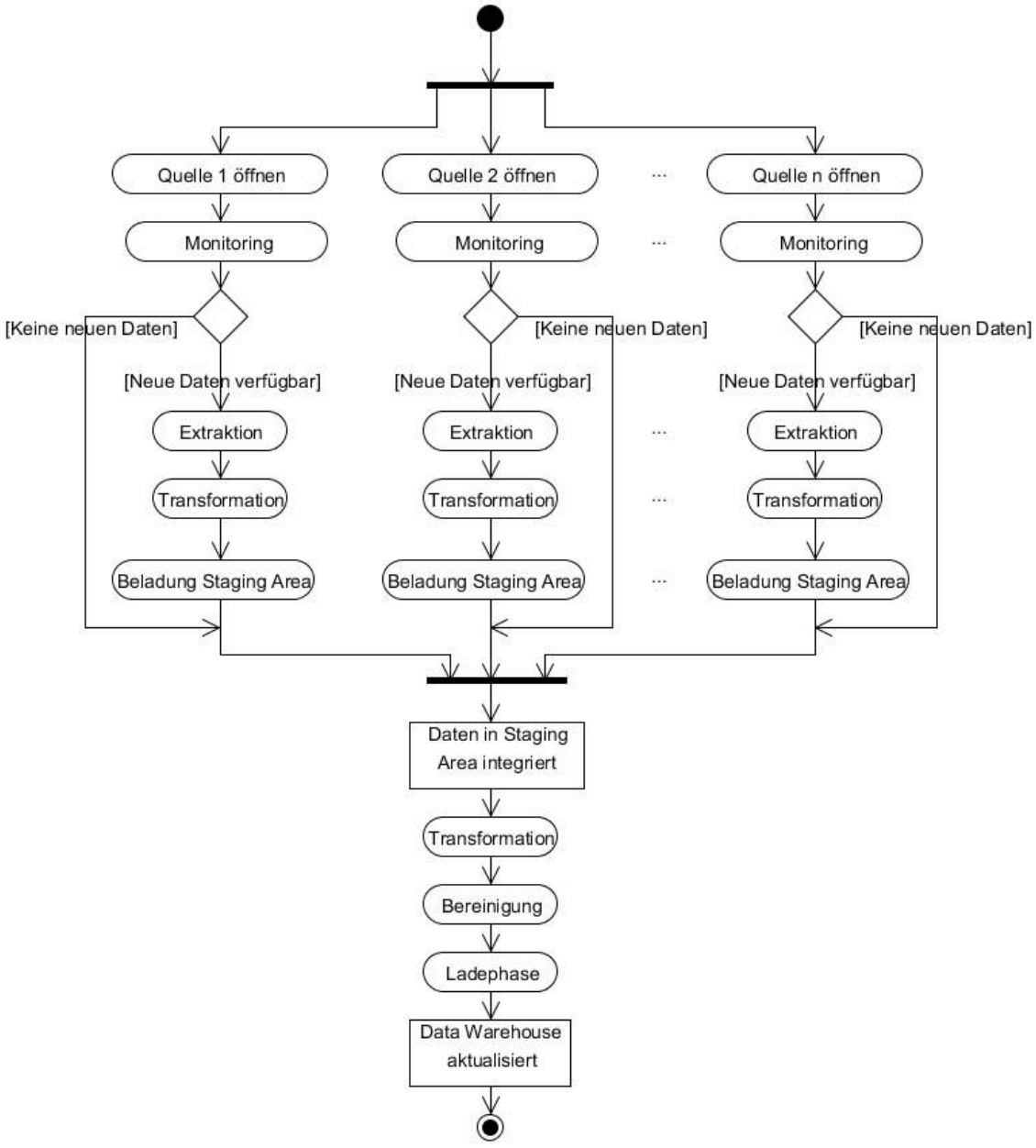


Abbildung 37: Beladungsprozess (UML- Aktivitätsdiagramm)

Die Integration mehrerer Datenquellen kann, wie in Abbildung 37 ersichtlich, parallel erfolgen. Die Anzahl der Datenquellen ist beliebig und im Aktivitätsdiagramm mit *1,2,...,n* spezifiziert, wobei *n* für die Zahl der genutzten Datenquellen steht. Der an die Staging-Area angeschlossene Beladungsprozess erfolgt sequenziell.

5 Design

In diesem Kapitel wird, aufbauend auf der in Kapitel 4 erläuterten Architektur der BI-Lösung, auf das Design der entworfenen BI-Lösung eingegangen. Das Design wird unterteilt in das konzeptuelle, logische und physische Design. Wie in Abbildung 38 erkennbar, ist das konzeptuelle Design unabhängig vom später für die Umsetzung herangezogenen System, während das logische Design bereits vom verwendeten System beeinflusst ist.

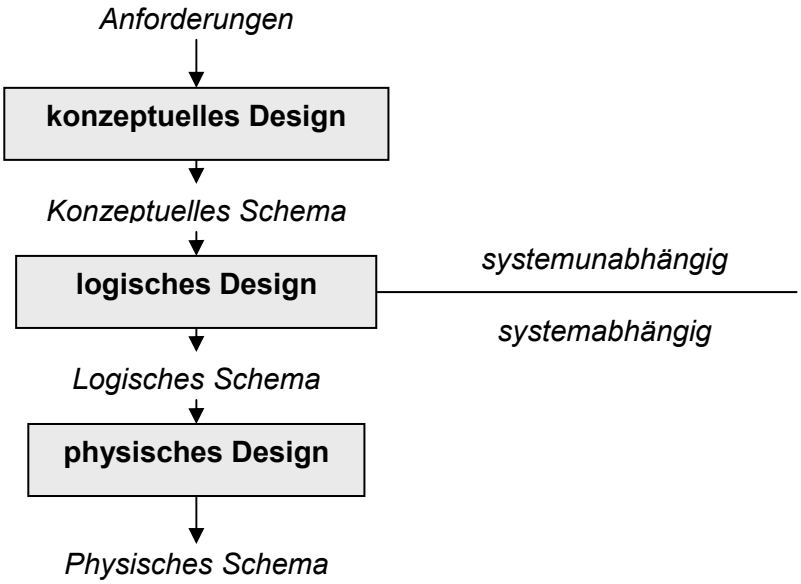


Abbildung 38: Datenbankentwurfsprozess nach (Schrefl 2006, S. 11)

Dieser Entwurfsprozess erfolgt sowohl für das Data Warehouse-Schema als auch für das Schema der Staging-Area. Dazu wird zunächst in Abschnitt 5.1 das konzeptuelle Design des Data Warehouse und der Staging-Area vorgestellt. Basierend auf dem konzeptuellen Design wird daraufhin der logische Entwurf in Abschnitt 5.2 vorgestellt, der die Umsetzung systemabhängig präzisiert. Anschließend wird auf physische Designaspekte in Abschnitt 5.3 eingegangen.

In den Abschnitten 5.4 und 5.5 wird, aufbauend auf dem Design der Staging-Area und des Data Warehouse, auf das Design des Beladungsprozesses eingegangen. Da vom Beladungsprozess die Staging-Area und das Data Warehouse zu befüllen sind, wird der Beladungsprozess in den ETL-Prozess zur Beladung der Staging-Area (vgl. 5.4) und den ETL-Prozess zur Beladung des Data Warehouse (vgl. 5.5) unterteilt.

Abschließend wird in Abschnitt 5.6 auf die Metadatenverwaltung im Data Warehouse eingegangen.

5.1 Konzeptuelles Design

Das konzeptuelle Design sieht den Entwurf eines konzeptuellen Schemas vor, das die relevanten Objekte mit ihren Eigenschaften und Beziehungen zu anderen Objekten erfasst (Schrefl 2006, S. 10). Aufgrund der Architektur erfolgt das konzeptuelle Design im Folgenden sowohl für die Data Staging-Area (vgl. 5.1.1) als auch für das Data Warehouse (vgl. 5.1.2).

5.1.1 Staging-Area

Zur Darstellung der Verkaufsvorgänge an den einzelnen Verkaufsstellen wurde ein UML-Klassendiagramm nach der bei (Schrefl 2006, S.24 ff.) angegebenen Notation für das konzeptuelle Datenbankdesign erstellt.

Zur schematischen Darstellung werden die in (Booch u. a. 2006, S. 137 ff.) und (Schrefl 2006, S. 24 ff.) angeführten Notationselemente verwendet. Neben den Standardnotationsobjekten Klasse und Assoziation sind folgende erweiterte Notationselemente für das konzeptuelle Design der Staging-Area von Bedeutung (Booch u. a. 2006, S. 180 ff.):

- **Komposition:** Eine Komposition ist eine Aggregationsbeziehung, also eine Beziehung zwischen zwei Objekten, die ein Ganzes von seinen Teilen unterscheidet. Zusätzlich zeichnet sich eine Komposition durch die Bedingung aus, dass ein Teil immer nur mit einem Ganzen verbunden ist. Das Ganze wird dabei als Kompositum (engl. composite) bezeichnet. So ist etwa ein Rahmen ein Teil des Kompositums Fenster. Ein Rahmen kann immer nur Teil eines Fensters sein. Im Gegensatz zu anderen Aggregationsbeziehungen ist es bei der Komposition nicht möglich, dass ein Rahmen zu mehreren Fenstern gehört. Eine Kompositionsbeziehung wird, wie in Abbildung 39 ersichtlich, mit einer schwarzen Raute gekennzeichnet.

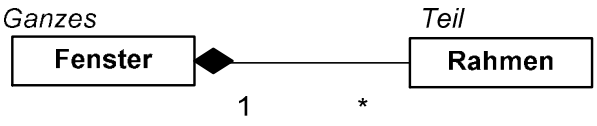


Abbildung 39: Kompositum

- **Assoziationsklasse:** Besitzt die Assoziation zweier Klassen selbst Eigenschaften, so ist es sinnvoll, diese Eigenschaften mithilfe einer Assoziationsklasse zu modellieren. Eine Assoziationsklasse beschreibt also eine Assoziation näher. Eine Assoziationsklasse kann beispielsweise eingesetzt werden, um zwischen den Klassen Unternehmen und Person ein Arbeitsverhältnis zu modellieren. Eine Assoziationsklasse kann etwa die Attribute Beginn- und Enddatum eines Arbeitsverhältnisses umfassen, wie in Abbildung 40 erkennbar. Zu beachten ist, dass in dieser Modellierungsvariante eine Person nur ein Arbeitsverhältnis mit einem Unternehmen eingehen kann (Oestereich 2005, S. 85). Kann es zu mehreren Arbeitsverhältnissen einer Person mit demselben Unternehmen kommen, so ist das Arbeitsverhältnis als eigene Klasse zu modellieren (vgl. Abbildung 40).

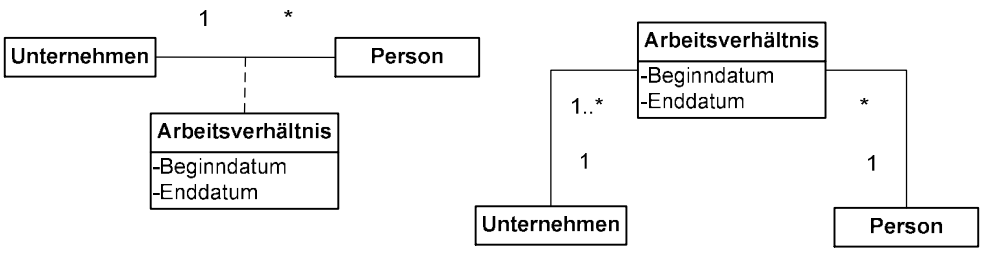


Abbildung 40: Modellierung eines Arbeitsverhältnisses mit Assoziationsklasse (links) oder mit einem zusätzlichen Objekt (rechts)
Die Assoziationsklasse erlaubt für eine Person nur ein Arbeitsverhältnis pro Unternehmen, bei Modellierung mit einem zusätzlichen Objekt sind mehrere Arbeitsverhältnisse mit demselben Unternehmen möglich.

Die Modellierung der Staging-Area erfolgte schließlich unter Einbeziehung der folgenden Klassen, die im Hinblick auf die spätere Transformation der Daten in das Data Warehouse-Schema erstellt wurden. Das konzeptuelle Design der Staging-Area wird zunächst in Abbildung 41 grafisch dargestellt und anschließend näher erläutert.

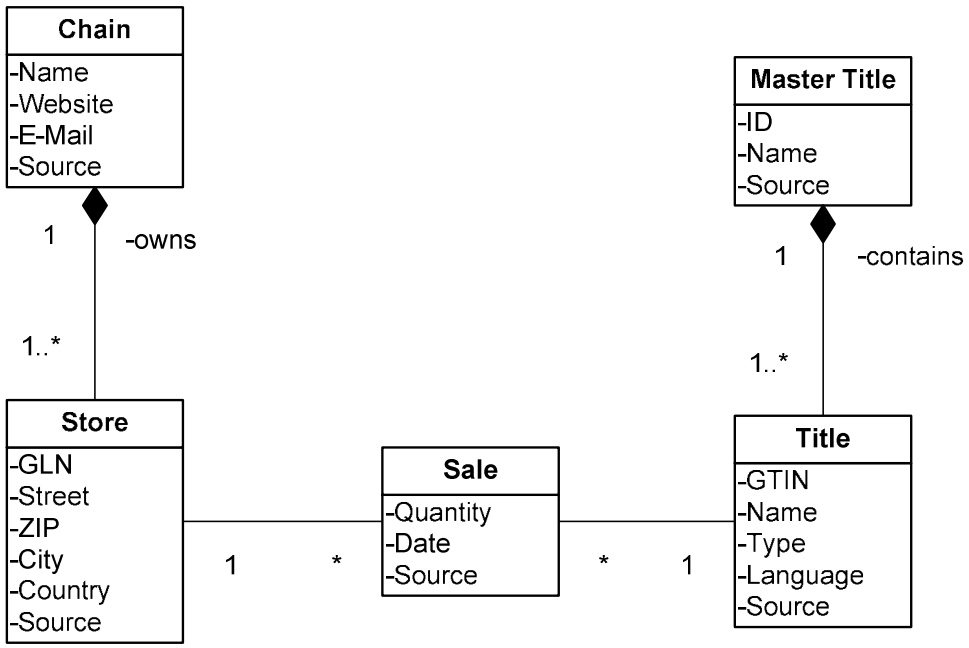


Abbildung 41: Konzeptionelles Schema (UML-Klassendiagramm) der Staging-Area

Handelskette (Chain)

Diese Klasse stellt eine Handelskette dar, die am Endkonsumentenmarkt auftritt. Eine Handelskette besitzt üblicherweise eine Vielzahl an Verkaufsstellen, mindestens jedoch eine. Die Handelskette tritt als Kompositum auf, da eine Verkaufsstelle nur genau Teil einer Handelskette sein kann. Eine Handelskette besitzt darüber hinaus folgende Attribute:

- *Name*: Der Name des Unternehmens, das die Handelskette betreibt.
- *Website*: URL der Homepage der Handelskette.
- *E-Mail*: E-Mail Adresse der Handelskette.
- *Source*: Bezeichnet die Datenquelle, die die Daten zum Unternehmen zur Verfügung gestellt hat.

Verkaufsstelle (Store)

Ein *Store* stellt eine physische Verkaufsstelle dar. Eine Verkaufsstelle ist immer Teil genau einer *Chain*. Eine Verkaufsstelle besitzt darüber hinaus folgende Attribute:

- *GLN*: Jede Verkaufsstelle besitzt eine *Global Location Number* (vgl. Abschnitt 3.3.1.2).
- *Street*: Dieses Attribut umfasst Straße und Hausnummer des Stores.
- *ZIP*: Postleitzahl der Verkaufsstelle.

- *City*: Stadt oder Gemeinde in der die Verkaufsstelle liegt.
- *Country*: Land in der die Verkaufsstelle liegt.
- *Source*: Bezeichnung der Datenquelle.

Verkäufe (Sale)

Diese Klasse stellt Verkäufe an den Endkonsumenten dar. Ein Verkauf findet immer in genau einer Verkaufsstelle statt und umfasst im vorliegenden Modell immer genau einen Titel. Umfasst ein Verkaufsvorgang mehrere Titel so wird ein Verkaufsvorgang in mehrere Verkäufe derselben Verkaufsstelle aufgeteilt. Somit wird die Anzahl der verkauften Titel korrekt erfasst, eine Zuordnung zu einem Verkaufsvorgang ist aber nicht möglich. Dies ist aber für den Data Warehouse-Zweck nicht erforderlich (vgl. 3.1). Darüber hinaus liefern die bisher vorhandenen Datenquellen keine Daten zu erfolgten Verkaufsvorgängen. Ein Verkauf kann zugleich auch den Umtausch eines Titels modellieren. Ein Umtauschvorgang ist am negativen Wert des Attributes *Quantity* zu erkennen. Zur näheren Beschreibung eines Verkaufes sind folgende Attribute vorhanden:

- *Quantity*: Gibt die Stückzahl der verkauften Titel an. Wird ein Titel etwa zweimal am selben Tag in derselben Verkaufsstelle erworben, so beinhaltet dieses Attribut den Wert „2“. Bei einem Umtausch von zwei Artikeln beinhaltet das Attribut den Wert „-2“.
- *Datum*: Gibt den Kalendertag des Verkaufsvorganges an. Auf eine feinere zeitliche Unterteilung wird aufgrund des Data Warehouse-Zwecks und der Datenqualität der Quelldaten verzichtet.
- *Source:* Bezeichnung der Datenquelle des Verkaufsvorganges.

Titel (Title)

Diese Klasse stellt ein Produkt dar, das zum Verkauf steht. Ein derartiges Produkt wird als *Titel* bezeichnet. Ein *Titel* besteht aus folgenden Attributen:

- *GTIN*: Die *Global Trade Item Number* stellt einen eindeutigen Schlüssel für jeden Titel dar (vgl. Abschnitt 3.3.1.2).
- *Name*: Dieses Attribut stellt die Bezeichnung des verkauften Produktes dar.
- *Type*: Der Typ des Produktes gibt an, um welche Art von Disc es sich handelt. Bei den vorhandenen Datenquellen ist der Typ DVD am häufigsten.

- *Language*: Gibt die Sprache des verkauften Produktes an. Das Attribut *Language* gibt die Sprache der Artikelverpackung und nicht, gegebenenfalls mehrfach vorhandene, Sprachversionen des Inhaltes der Disc an.
- *Source*: Bezeichnung der Datenquelle des *Titels*.

Master Title

Ein *Master Title* stellt eine Aggregationsbeziehung zu einem konkreten *Titel* dar. Das bedeutet ein *Master Title* umfasst zumindest einen Titel. Vor allem bei populären *Master Title* ist es möglich, dass ein *Master Title* aus mehreren Titel besteht. So kann es vorkommen, dass etwa der *Master Title* „Spider Man" aus den Titeln „Spider Man Gold Edition" und „Spider Man Limited Edition" besteht. Da sich beide Titel nur geringfügig unterscheiden, aber denselben Film beinhalten, ist es sinnvoll, hier die Titel in der Komposition mit *Master Title* zu aggregieren. Ein Titel kann somit nur Teil genau eines *Master Title* sein.

- *Name***:** Der Name des *Master Title*. Dieser sollte keine Zusatzbezeichnungen wie etwa „Gold Edition" beinhalten.
- *Source***:** Bezeichnung der Datenquelle des *Master Title***.**

5.1.2 Data Warehouse

Für die Modellierung des konzeptuellen Designs des POS-Data Warehouse wird das *Dimensional Fact Model (DFM)* herangezogen (vgl. Golfarelli u. a. 1998). Daneben existieren zahlreiche weitere Designnotationen wie das multidimensionale Entity-Relationship-Modell (vgl. Sapia u. a. 1998) oder multidimensionales UML (vgl. Harren & Herden 1999). Diese Designnotationen sind jedoch nur Erweiterungen bereits vorhandener Standards und daher nicht optimal zur Darstellung multidimensionaler Datenstrukturen geeignet. Aus diesem Grund wurde für diese Arbeit das Dimensional Fact Model gewählt, das ausschließlich für die Darstellung von multidimensionalen Datenmodellen entwickelt wurde. (Bauer & Günzel 2004, S.165 ff.)

Ein DFM besteht aus folgenden Teilen (Golfarelli u. a. 1998, S. 4):

- **Fakt (*Fact*).** Ein Fakt ist der wesentlichste Bestandteil eines DFM. Er stellt jene für Geschäftsprozesse typische Handlung dar, die mit Hilfe des Data Warehouse analysiert werden soll. Dies sind beispielsweise Verkäufe oder Bestellungen.

- **Kennzahlen (*Measures*)** werden zur Beschreibung eines Fakts verwendet. Ein Fakt besteht aus mindestens einer Kennzahl, oft werden aber mehrere Kennzahlen zur Beschreibung eines Facts verwendet. Eine typische Kennzahl für die Verkäufe wäre etwa der Umsatz.
- **Dimensionen (*Dimensions*)** sind konkrete Eigenschaften eines Fakts auf niedrigster Detailebene. So besitzt ein Verkauf etwa die Eigenschaft, dass er in einer Verkaufsstelle stattfindet. Dies ist gleichzeitig die niedrigste Detailebene zur Bestimmung des Ortes eines Verkaufes.
- **Dimensions-Attribute (*dimension attributes*)** beschreiben eine Dimension näher. Die feinstgranulare Ebene eines Dimensions-Attributes entspricht zugleich der Bezeichnung der *Dimension*. Alle weiteren *Dimensions-Attribute* einer Dimension stellen eine zunehmend gröbere Granularitätsstufe dar. Das bedeutet etwa, wenn ein *Store* die feinste Hierarchieebene darstellt, dann ist die Stadt, in der sich ein *Store* befindet, die nächst höhere Hierarchieebene. Als höchste Hierarchieebene für die Dimension *Store* sind je nach Zweck des Data Warehouse Länder oder Kontinente denkbar. Die gröbste Granularitätsstufe jeder Dimension wird im DFM nicht angeführt und entspricht der alles (all)-Ebene. Diese Ebene ist sinnvoll, um etwa die Summe aller Verkäufe ohne Einschränkungen zu bilden. Zu beachten ist, dass die Beziehung immer ausgehend von der feineren zur gröberen Granularitätsstufe eine „viele-zu-eins“ Beziehung darstellt. Das bedeutet, in einer Stadt kann es gemäß der Dimensionshierarchie mehrere *Stores* geben, ein *Store* kann aber immer nur in einer Stadt liegen.
- **Nicht-dimensionale Attribute (*Non-Dimension Attributes*):** Eine Dimensionshierarchie kann aus mehreren nicht-dimensionalen Attributen bestehen. Nicht-dimensionale Attribute beschreiben eine Dimensionshierarchie näher. Ein nicht-dimensionales Attribut steht immer in einer „eins-zu-viele“ Beziehung im Bezug auf die Dimensionshierarchie. Das bedeutet ein nicht-dimensionales Attribut, wie etwa ein Wochentag, kann viele unterschiedliche Tage bezeichnen. Zu beachten ist, dass nicht-dimensionale Attribute nur beschreibende Eigenschaften haben, aber nicht für Hierarchien oder Aggregationen herangezogen werden können.

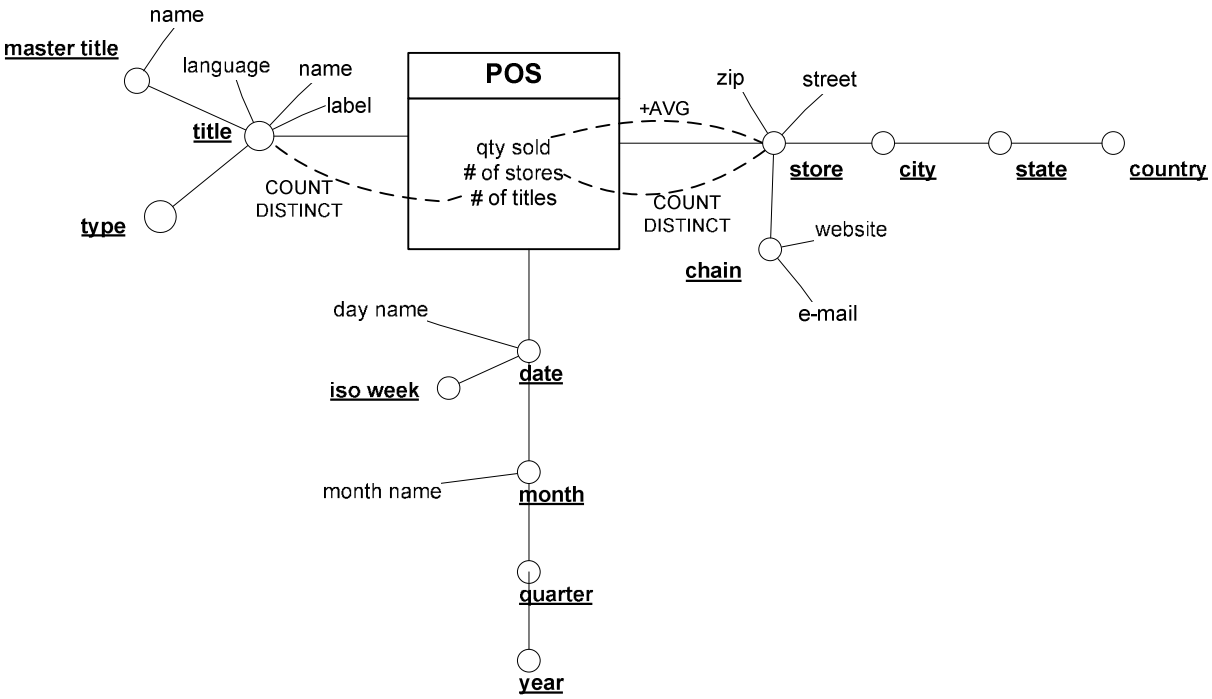


Abbildung 42: Dimensional Fact Model des POS-Data Warehouse

Der Aufbau des DFM, das in Abbildung 42 grafisch veranschaulicht ist, basiert zum einen direkt auf den funktionalen Anforderungen. Zum anderen ist das Design direkt aus dem Aufbau der Data Staging-Area abgeleitet, die hier als relationales Ausgangsschema dient (Golfarelli u. a. 1998, S. 22 ff.). Ausgangspunkt für den Fakt im DFM ist immer eine Entität, die häufigen Änderungen unterworfen ist. Diese Entität ist im zugrundeliegenden relationalen Modell ein Verkauf (*Sale*), der wesentlich öfter stattfindet als etwa das Hinzufügen eines neuen Produkts zum Sortiment oder die Eröffnung einer neuen Endverkaufsstelle (vgl. Abbildung 41).

Gemäß den funktionalen Anforderungen (vgl. 3.1.5) und den in der Staging-Area vorhandenen Daten besteht der konzeptuelle Entwurf aus folgenden Dimensionen zur Darstellung der POS-Verkaufsdaten. Diese sind mit dem Fakt *POS* verbunden:

- *title:* In dieser Dimension sind alle Artikel, die von *Sony DADC* produziert und bisher in den erfassten Verkaufsstellen verkauft wurden, hinterlegt. Ein *title* stellt immer ein Produkt von *Sony DADC* dar, das mit seinem Namen, dem vertreibenden Label sowie der Sprache, in der der Film verkauft wird, gespeichert ist.
    - *master title:* Die Dimensionshierarchie *Master Title* wurde angelegt, um unterschiedliche Titel, die den gleichen Film beinhalten, zu erkennen. Ist also

ein Film etwa in einer „Platin“ und in einer „Gold“ Edition erhältlich, so handelt es sich um zwei verschiedene *Titel*, aber um denselben *Master Title.*

- *type:* Die Dimensionshierarchie *type* ist vorgesehen, um zu erkennen, auf welchem Medium ein *Titel* hinterlegt ist. Zu einem *type* sind mehrere *Titel* hinterlegt. *Type* kann jedoch nicht als gröbere Granularitätsstufe im Vergleich zu *Master Title* auftreten, da ein *Master Title* aus mehreren *types* und ein *type* aus mehreren *Master Title* bestehen kann.

- *store*: In dieser Dimension werden alle Verkaufsstellen erfasst. Eine Verkaufsstelle verfügt über die nicht-dimensionalen Attribute *street*, das die Adresse angibt, und *zip*, das für die Postleitzahl steht.
  - *chain*: Ein *store* ist immer Teil einer Handelskette, das heißt, zu einer Handelskette zählen meist mehrere *stores.* Da es möglich sein soll, die Verkäufe über die einzelnen Handelsketten auszuwerten, ist *chain* hier als dimensionales Attribut hinterlegt.
  - Zur Abgrenzung der Lage eines *store* sind folgende geografischen Ortsangaben hinterlegt:
    - *city*: Ein *store* muss immer in einer Stadt liegen, wobei in einer Stadt mehrere *stores* liegen können. *City* ist daher die nächst gröbere Granularitätsstufe zu *store.*
    - *state:* Dieses Dimensions-Attribut stellt die Bundesländer als nächst gröbere Granularitätsstufe zu den Städten dar.
    - *country:* Dieses Dimensions-Attribut stellt im aktuellen DFM die gröbste Granularitätsstufe dar und ist daher als letztes Dimensions-Attribut angeführt.

- *date*: Ein Kalendertag, hier als *date* bezeichnet, stellt die feinste Granularitätsstufe im Bezug auf die zeitliche Einschränkung der POS-Daten dar. Das Dimensions-Attribut wird durch das nicht-dimensionale Attribut *day name* näher beschrieben.
  - *Iso week*: Um auch Auswertungen, die etwa auf den letzten Wochen basieren, zu ermöglichen, wurde die *iso week* als Dimensions-Attribut aufgenommen. Die *iso week* definiert den Montag als den ersten Tag der Woche. Als erste Woche im Jahr wird jene Woche angesehen, die den ersten Donnerstag im Jahr beinhaltet. (ISO 8601 2010)

- *Month:* Dieses Dimensions-Attribut steht für ein Kalendermonat. Zur Beschreibung wird das nicht dimensionale Attribut *month name* verwendet.
  - *Quarter:* Dieses Dimensions-Attribut stellt ein Quartal dar.
  - *Year:* Dieses Dimensions-Attribut stellt die gröbste Granularitätsstufe dar, die bei der Analyse der POS-Daten notwendig ist.

Darüber hinaus wurden folgende Kennzahlen (*measures)* festgelegt:

- *qty sold*: Diese Kennzahl stellt die Summe der Anzahl aller Verkäufe für die jeweils gewählten Dimensionshierarchien und Filter dar. Da die Anzahl der Verkaufsstellen, die von den Datenquellen berücksichtigt werden, stetig steigt, ist es sinnvoll, im Bezug auf die Dimension *store* neben der Summe auch den Durchschnitt der Verkaufszahlen pro Store zu berechnen. Steigt also die Kennzahl *qty sold* nur aufgrund einer größeren Anzahl an Verkaufsstellen an, so bleibt der Durchschnitt konstant.
- # *of stores*: Diese Kennzahl zählt die Anzahl der unterschiedlichen Verkaufsstellen, in denen Verkaufsvorgänge registriert wurden, im Bezug auf die jeweils gewählten Dimensionen und Filter.
- # *of title*: Diese Kennzahl ermöglicht es, die Anzahl der unterschiedlichen verkauften Titel im Bezug auf die jeweils gewählten Dimensionen und Filter anzugeben.

5.2 Logisches Design

In diesem Abschnitt wird das vom konzeptuellen Design abgeleitete logische Design der Staging-Area und des Data Warehouse erläutert. Dabei wurde das logische Design gemäß den Rahmenbedingungen (vgl. 3.3.2) mit einem relationalen Datenmodell umgesetzt.

Zunächst wird in Abschnitt 5.2.1 auf das logische Design der Staging-Area eingegangen. Anschließend wird in Abschnitt 5.2.2 das multidimensionale Design des Data Warehouse näher betrachtet, das aufgrund der Rahmenbedingungen (vgl. 3.3.2) als ROLAP (relationales OLAP) modelliert wurde. Alternativ möglich wäre, das logische Design als MOLAP (multidimensionales OLAP) umzusetzen. Dazu wäre jedoch ein multidimensionales Database Management System erforderlich. Zu beachten ist, dass MOLAP und ROLAP nicht impliziert, dass ein OLAP-Werkzeug zur Auswertung verwendet werden muss (Bauer & Günzel 2004, S. 201).

5.2.1 Staging-Area

Das logische Design der Staging-Area erfolgte in Anlehnung an den in Abschnitt 5.1.1 vorgestellten konzeptuellen Entwurf der Staging-Area. Da zu erwarten ist, dass viele Datenquellen, wie derzeit die *Datenquelle 1,* bereits Daten aus relationalen Datenbanken zur Verfügung stellen, scheint die Umsetzung des logischen Designs der Staging-Area in dritter Normalform als sinnvoll. Da die Staging-Area Teil des *Back-Rooms* (vgl. 4) ist, steht sie nicht für Abfragen von Endbenutzern zur Verfügung. Somit spielt die Antwortzeit auf Abfragen, im Gegensatz zum *Front-Room*, keine übergeordnete Rolle. Darüber hinaus ist die Bereinigung und Transformation der Daten in einer relationalen Datenbank in dritter Normalform einfacher möglich, da es bei der Änderung von einzelnen Attributwerten nicht zu Änderungsanomalien kommt (Kimball u. a. 1998, S. 15 f.).

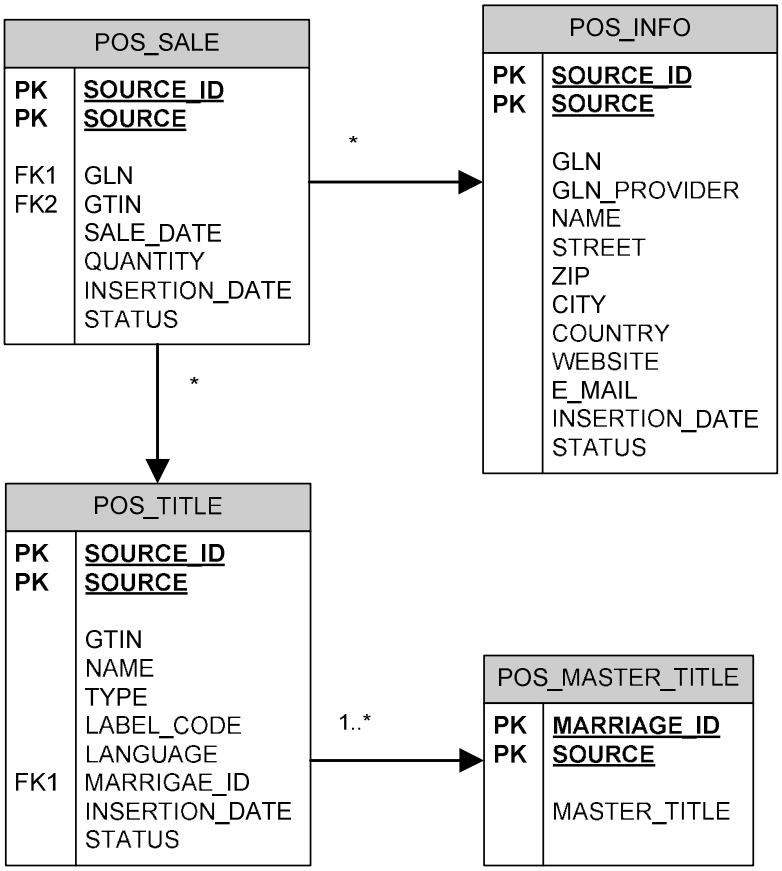


Abbildung 43: Logisches Design der Staging-Area (UML-Diagramm)
Die Primärschlüssel (PK) und die Fremdschlüssel (FK) sind entsprechend gekennzeichnet.

Die in Abbildung 43 abgebildeten Entitäten entsprechen dem konzeptuellen Modell der Staging-Area (vgl. 5.1.1). So bildet die Tabelle POS_SALE die konzeptuelle Klasse *Sale*, die Tabelle POS_TITLE die konzeptuelle Klasse *Ttitle* und die Tabelle POS_MASTER_TITLE

die konzeptuelle Klasse *Master Title* ab. Die Kompositionsbeziehung zwischen *Chain* und *Store* wurde in der Tabelle POS_INFO abgebildet. Diese Entscheidung wurde getroffen, da die Datenquellen kein eindeutiges Merkmal zur Zuordnung einer Verkaufsstelle zu einer Handelskette bieten. Die Zuordnung, etwa aufgrund des Namen der Handelskette (Attribut NAME in POS_INFO), ist somit Aufgabe des Transformationsprozesses (vgl. 2.7.3).

Der Primärschlüssel der Tabellen POS_SALE, POS_INFO, POS_TITLE setzt sich aus der SOURCE_ID und der SOURCE, der Primärschlüssel der Tabelle POS_MASTER_TITLE aus MARRIAGE_ID und SOURCE zusammen.

Zu beachten ist, dass die Beziehungen der Tabellen in der Implementierung (vgl. 6.2) nicht über Fremdschlüssel-Zuweisungen realisiert wurden, da es aufgrund des Designs des Beladungsprozesses möglich ist, dass Datensätze mit gleichen natürlichen Schlüsseln mehrfach in die Staging-Area geladen werden (vgl. 5.4.4). In diesem Fall ist eine Abfrage auf den jeweils zuletzt geladenen Datensatz eines natürlichen Schlüssels einzuschränken um die in Abbildung 43 dargestellten Beziehungen nutzen zu können.

Die SOURCE_ID ist ein Wert, der vom Quellsystem als Primärschlüssel für einen Verkaufsvorgang verwendet wird. Liegt ein derartiger Primärschlüssel in der Datenquelle nicht vor wird ein fortlaufender Schlüssel vom ETL-Prozess zur Beladung der Staging-Area erstellt. In Kombination mit der eindeutigen Bezeichnung der Datenquelle (SOURCE) ergibt sich somit der Primärschlüssel eines Eintrages. Dieser Primärschlüssel wird verwendet, um den Ursprung jedes Datensatzes in der Datenquelle rückverfolgen zu können.

Das Attribut STATUS gibt an, ob ein Datensatz bereits in das Data Warehouse-Schema eingefügt wurde und somit vom nächsten ETL-Prozess zur Beladung des Data Warehouse nicht mehr zu erfassen ist.

5.2.2 Data Warehouse

In diesem Abschnitt wird das logische Design des Data Warehouse erläutert. Dazu wird zunächst im Abschnitt 5.2.2.1 auf die möglichen logischen Abbildungsmöglichkeiten des konzeptuellen Data Warehouse-Entwurfes eingegangen bevor in Abschnitt 5.2.2.2 das gewählte Umsetzungsmodell erläutert wird.

5.2.2.1 Abbildungsmöglichkeiten des konzeptuellen Data Warehouse-Entwurfes

Zur Abbildung des DFM aus Abschnitt 5.1.2 auf ein logisches Design gibt es zwei Möglichkeiten, nämlich das *Snowflake-* und das *Starflake-Schema*. Weiters existieren von diesen Schemata zahlreiche Misch- oder abgeleitete Formen. Zentrales Element aller logischen Darstellungsformen ist die Faktentabelle. Diese Tabelle beinhaltet alle *Kennzahlen (measures)* des Data Warehouse auf granularster Ebene. Dies könnte etwa ein Eintrag für jeden Verkaufsvorgang sein. Darüber hinaus beinhaltet die Faktentabelle einen Fremdschlüssel zur jeweils granularsten Ebene der verbundenen Dimensionen. Alle Fremdschlüssel bilden automatisch auch einen zusammengesetzten Schlüssel der Faktentabelle. (Bauer & Günzel 2004, S. 202 ff.)

Die Faktentabelle unterhält immer viele-zu-eins Beziehungen zu allen Dimensionen. Das bedeutet, dass ein Produkt etwa in mehreren Verkaufsstellen verkauft werden kann, ein Verkaufsvorgang aber immer nur in einer Verkaufsstelle stattfinden kann. Aufgrund verschiedener Speicherkonzepte für Dimensionen wird zwischen dem Star- und Snowflake-Schema wie folgt unterschieden (Bauer & Günzel 2004, S. 203 ff.):

1. **Snowflake-Schema**

   Im Snowflake-Schema erfolgt die Speicherung der Dimensionen in dritter Normalform. Das bedeutet, dass für jede Klassifikationsstufe auch eine eigene Tabelle angelegt wird. So würde in einem Snowflake-Schema etwa für jede Produktgruppe und jede Produktkategorie eine eigene Tabelle angelegt. Dieser Tabellenaufbau ist daher vergleichbar mit anderen relationalen Datenstrukturen in dritter Normalform. Der Name Snowflake (Schneeflocke) ergibt sich aus der grafischen Darstellung, die aufgrund der meist großen Zahl an Tabellen zur Klassifikation der Dimensionen an die Form einer Schneeflocke erinnert.

   Der Vorteil des *Snowflake-Schemas* liegt in der normalisierten Struktur, die Änderungsanomalien vermeidet und die Daten zu den Klassifikationen nicht mehrfach vorhält. Nachteilig auf die Geschwindigkeit der Anfragebeantwortung wirken sich die relativ große Anzahl an Tabellen und die damit benötigten Verknüpfungsoperationen (Joins) aus.

2. **Star-Schema**

   Im *Star-Schema* werden die Dimensionen im Gegensatz zum *Snowflake-Schema* denormalisiert hinterlegt. Damit ist für jede Dimension nur eine Tabelle erforderlich.

   Der Vorteil des Star-Schemas ist, dass alle Daten zu einer Dimension zentral in einer Tabelle hinterlegt sind. Dadurch ist eine kurze Antwortzeit auf Anfragen möglich, da bei Abfragen weniger Verknüpfungsoperationen notwendig sind. Negativ wirkt sich die Denormalisierung aus, die spätere Änderungen erschwert. Diese können im *Star-Schema* zu inkonsistenten Datenbeständen führen. Die mehrfache Speicherung von Klassifikationen, wie etwa der Name und die Eigenschaft einer Produktkategorie, führt zudem zu höherem Speicherplatzbedarf.

Darüber hinaus existieren Mischformen, in denen einige Dimensionen wie bei einem *Snowflake-Schema* und die restlichen wie bei einem *Star-Schema* modelliert werden.

Fact-Constellation-Schema

Aufgrund der enormen Datenmenge, die in einem Data Warehouse zu erwarten ist, kann es, um die Antwortzeit auf Abfragen zu senken, zusätzlich erforderlich sein, die Kennzahlen der Faktentabelle zu aggregieren. Das bedeutet, neben den auf feinster Ebene eingetragenen Werten in der Faktentabelle existieren weitere Aggregationstabellen, die etwa die Umsatzzahlen nicht auf Ebene jeder Endverkaufsstelle sondern bereits auf Ebene eines Landes speichern. Die Speicherung der Aggregate in einer eigenen Tabelle hat folgende Vorteile (Kimball u. a. 1998, S. 555 ff.):

- Die ursprüngliche Faktentabelle bleibt unberührt.
- Aufgrund der unterschiedlichen Tabellenbezeichnung ist es nicht möglich, dass aggregierte Werte unabsichtlich in Berechnung auf feinst-granularer Ebene einbezogen, und somit mehrfach gezählt werden.
- Aufgrund angepasster Dimensionstabellen können Attribute, die auf der jeweiligen Aggregationsebene keinen Sinn ergeben, einfach weggelassen werden und müssen nicht durch Platzhalter ersetzt werden.

5.2.2.2 Umsetzung als Fact-Constellation-Schema

Aufgrund der oben genannten Vorteile des Star-Schemas und der Rahmenbedingung, die besagt, dass OBIEE als OLAP-Werkzeug zu verwenden ist, wurde das *Fact-Constellation-Schema* für die Umsetzung von Aggregationstabellen gewählt. Dies ist vorteilhaft, da OBIEE vorhandene Aggregationstabellen erkennen kann und somit kürzere Antwortzeiten erzielt werden können.

Das Data Warehouse-Design basiert auf einem *Star-Schema*, da im vorliegenden Fall die geringere Antwortzeit im Hinblick auf die FASMI Kriterien (vgl. 2.7.6) wichtiger erschien als eine Speicherplatzersparnis, die ein *Snowflake-Schema* mit sich bringen würde. Eventuelle Erschwernisse beim Aktualisieren von Dimensionsdaten werden in Kauf genommen, da etwa Änderungen von Filialadressen oder Produktspezifika nur selten zu erwarten sind.

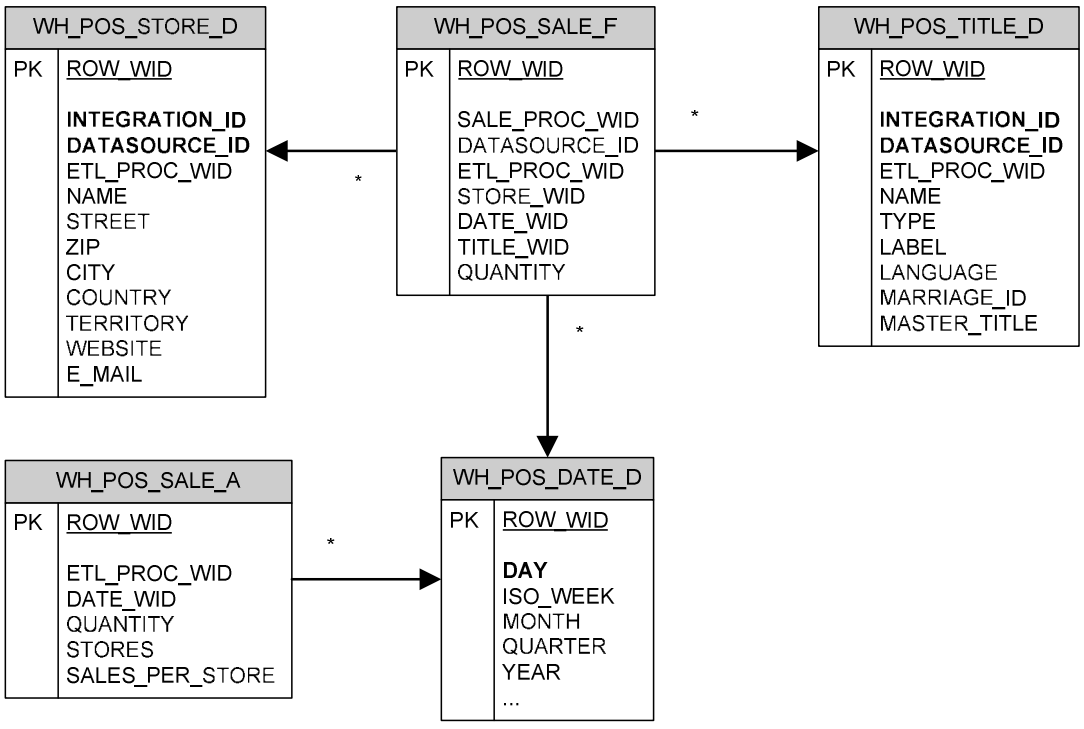


Abbildung 44: Fact-Constellation Star-Schema
Die künstlichen Primärschlüssel sind unterstrichen, die natürlichen Schlüssel der Dimensionstabellen sind fett dargestellt.

Abbildung 44 zeigt die logische Modellierung des Data Warehouse *Star-Schema*. Die Tabelle WH_POS_SALE_F stellt die zentrale Faktentabelle dar. Sie ist mit den drei

Dimensionstabellen WH_POS_STORE_D, WH_POS_DATE_D und WH_POS_TITLE_D über die drei Fremdschlüssel STORE_WID, DATE_WID und TITLE_WID verbunden. Die Tabelle WH_POS_SALE_A stellt eine Aggregationstabelle im Sinne des *Fact-Constellation-Schema* dar. Da diese Tabelle alle Verkäufe auf Monatsebene ohne Beachtung von Verkaufsstellen oder Produkten aggregiert, besteht nur eine Verbindung zur Zeitdimension (WH_POS_DATE_D).

Die Primärschlüssel der einzelnen Dimensionen sind künstlich erzeugte Schlüssel (surrogate keys). Die fett hervorgehobenen Attribute stellen den natürlichen Schlüssel einer Dimension dar. Dieser setzt sich im vorliegenden Design aus INTEGRATION_ID und SOURCE_ID zusammen. Falls der natürliche Schlüssel aus mehreren Attributen besteht, so sind auch mehrere INTEGRATION_IDs erforderlich. Der natürliche Schlüssel wurde um eine SOURCE_ID ergänzt, um sicherzustellen, dass sich die natürlichen Schlüssel unterschiedlicher Datenquellen nicht überschneiden.

5.3 Physisches Design

Das physische Design von Datenbanken legt fest, wie die Tupel von Relationen in Dateien abgebildet werden (Schrefl 2006, S. 67).

Beim Data Warehouse Design spielt vor allem die Optimierung des Datenzugriffes im Hinblick auf eine möglichst hohe Performance eine Rolle (vgl. 2.7.6). Aufgrund der zufriedenstellenden Performance im Hinblick auf die FASMI Kriterien, die bereits durch Aggregationstabellen erreicht werden konnte (vgl. 7.1), wurden im vorliegenden Kontext mögliche weitere Optimierungsmaßnahmen nicht eingehend untersucht.

Es steht jedoch außer Zweifel, dass Bitmap-Indizes einen Performancegewinn für Punktabfragen ermöglichen, die etwa nach dem Produkttyp unterscheiden, der nur eine relativ geringe Anzahl an Ausprägungen bietet. (Bauer & Günzel 2004, S. 269 ff.)

Weiters würde sich für die Faktentabelle WH_POS_SALE_F eine horizontale Partitionierung etwa nach Verkaufsjahren oder Handelsketten anbieten. Die Partitionierung ist im Besonderen vorteilhaft, wenn sich ein Großteil der Anfragen nur auf die Verkäufe innerhalb eines bestimmten Zeitraumes oder auf eine bestimmte Handelskette beschränkt. Die Partitionierung würde dazu führen, dass die Daten, je nach Teilungskriterium, in getrennten

Partitionsbereichen gespeichert werden und somit bei einer Anfrage nur die jeweils relevante Partition durchlaufen werden muss. (Bauer & Günzel 2004, S. 277 ff.)

5.4 ETL-Prozess zur Beladung der Staging-Area

Neben dem Design der Staging-Area und des Data Warehouse war es erforderlich, eine Anwendung zu erstellen, die in der Lage ist, die Anforderungen in Abschnitt 3.1.1 und 3.1.2 zu erfüllen. Das bedeutet, die Anwendung muss die Datenquellen einem *Monitoring* unterziehen und anschließend die Daten in die Staging-Area integrieren. Darüber hinaus muss die Anwendung, die im Folgenden kurz als *PosIntegrator* bezeichnet wird, auch das Hinzufügen und Entfernen von unterschiedlichen Datenquellen unterstützen.

Im Folgenden wird das gewählte Design zur Umsetzung der Integrationsfunktion sowie das Design zur Unterstützung unterschiedlicher Datenquellen erläutert und begründet. Das Design dieser Anforderung deckt damit auch die in Abschnitt 4 angeführten Prozesse zur Befüllung der Staging-Area ab.

Dazu werden zunächst in Abschnitt 5.4.1 die für das Design herangezogenen Entwurfsmuster erläutert um in den Abschnitten 5.4.2 bis 5.4.5 die Umsetzung des *Monitoring*, die *Extraktion*, die *Transformation* und die Beladung der Staging-Area, deren Design in Abschnitt 5.1.1 geschildert ist, zu erläutern. In Abschnitt 5.4.6 schließlich wird der Beladungsprozess der Staging-Area zusammenfassend dargestellt.

5.4.1 Entwurfsmuster

Zur besseren Verständlichkeit werden im Folgenden die im Design verwendeten Entwurfsmuster *Fabrikmethode*, *abstrakte Fabrik* und *Singelton* kurz vorgestellt.

Fabrikmethode

Sinn der Fabrikmethode ist es, ein Objekt vom System anzufordern, ohne die tatsächliche Implementierung der Klassendeklaration zu kennen, da diese Kenntnis nur die Fabrikmethode besitzt.

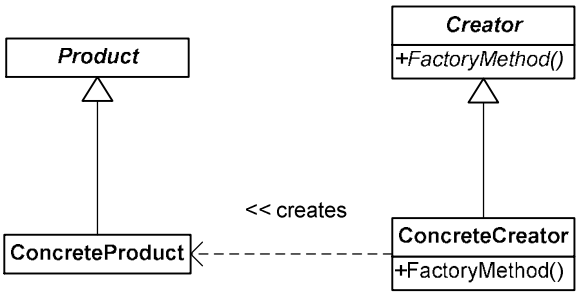


Abbildung 45: Fabrikmethode aus (Gamma u. a. 1995, S. 108)

Wie in Abbildung 45 erkennbar, tritt die Klasse *ConcreteCreator* auf, um Objekte vom Typ *Product* zu erzeugen. Die Klasse *Product* dient als Schnittstelle für alle vom Produkt vererbten Typen. Wird ein *ConcreteProduct* erzeugt, so ist die Operation *FactoryMethod* der Klasse *ConcreteCreator* aufzurufen. Existieren unterschiedliche Produkte, so sind auch mehrere dementsprechende Creator-Klassen zu erstellen.

Alternativ dazu ist es möglich, dass die Creator-Klasse konkret deklariert wird und die *FactoryMethod* anhand eines übergebenen Parameters entscheidet, welches *Product* zu erstellen ist. Wird ein Produkt gegen ein anderes ausgetauscht oder aktualisiert, so muss diese Information nur an die *FactoryMethod* weitergegeben werden; gegenüber dem Klienten der *FactoryMethod* ist diese Änderung transparent. (Gamma u. a. 1995, S.107 ff.)

Abstrakte Fabrik

Eine abstrakte Fabrik ist im Gegensatz zur Fabrikmethode sinnvoll, wenn ein System, unabhängig von der Art der Erzeugung der Objekte, mehrere Produktfamilien anbieten soll. Im Gegensatz zur Fabrikmethode verläuft die Hierarchie der Fabrikmethoden parallel zur Hierarchie der zu erzeugenden Produkte. Um in Abbildung 46 etwa das *ProductA2* zu erstellen, muss der Client zunächst ein Objekt der *ConcreteFactory2* erstellen und anschließend die Operation *CreateProductA* aufrufen. Somit ist sichergestellt, dass ein Objekt nur erstellt wird, wenn beide Eigenschaften des Produktes angegeben werden. Der Produktionsprozess ist, wie bei der Fabrikmethode, für den *Client* transparent.

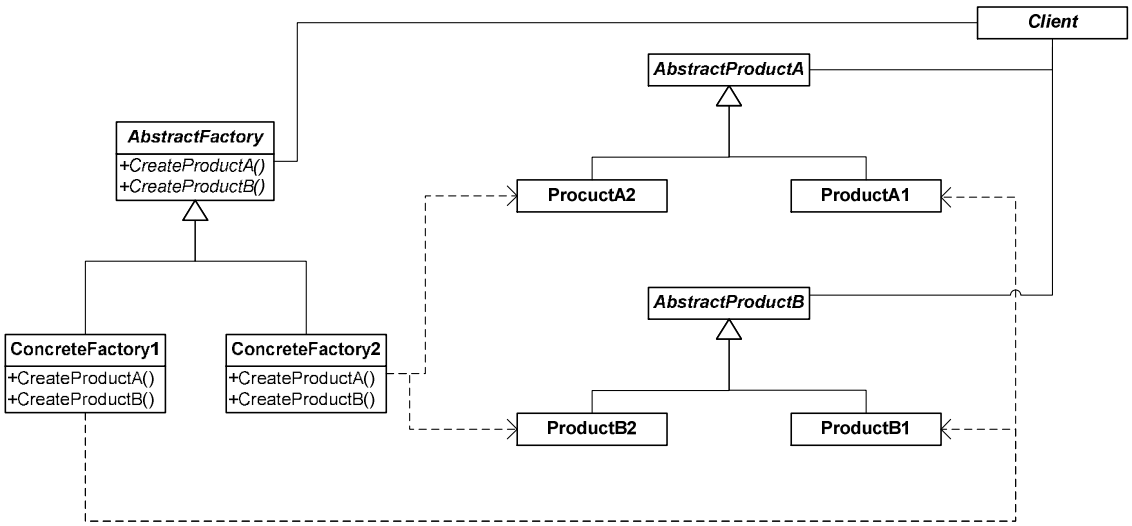


Abbildung 46: Abstrakte Fabrik nach (Gamma u. a. 1995, S. 88).
Der *Client* kommt nur mit den jeweils abstrakten Typen der *Factory* und der *Product*-Klassen in Kontakt.

Wird ein neuer Produkttyp hinzugefügt, so ist diese Anpassung auch in den *Factory*-Klassen vorzunehmen, da eine neue Operation oder eine neue *Factory* zu erstellen ist. Dies könnte umgangen werden, indem die *AbstractFactory* lediglich über eine Operation verfügen würde, die aufgrund unterschiedlicher Parameter die konkreten Produkte erzeugt. Diese Art der Implementierung ist jedoch aufgrund der variablen Parameterwerte als unsicher einzustufen. Aus diesem Grund sind die Änderungen, die für ein neues Produkt notwendig sind, in den Fabriksklassen nachzuziehen.

Darüber hinaus wird von abstrakten Fabriken das Singleton Muster verwendet (vgl. 5.4.2), um zu vermeiden, dass mehrere Instanzen der Fabriksklassen erstellt werden. (Gamma u. a. 1995, S. 87 ff.)

Singleton

Dieses Entwurfsmuster stellt sicher, dass von einer Klasse nur ein Objekt erstellt werden kann. Dabei sollte diese Bedingung, wenn möglich, direkt in der Klasse abgelegt werden. Vorteilhaft ist ein Singleton überall dort, wo nur ein einziger Zugriffspunkt auf die Funktionalität einer Klasse vorhanden sein soll. (Gamma u. a. 1995, S. 127)

5.4.2 Monitoring

Der Monitoring-Prozess ist für jede Datenquelle und jede Informationsart in geeigneter Form durchzuführen. Das Monitoring selbst wird aber nicht in den konkreten Klassen wie

Sphe2posSale durchgeführt, da für das Monitoring ein Abgleich der Daten mit der Staging-Area notwendig ist und somit ein direkter Zugriff auf die Staging-Area notwendig ist. Ein direkter Zugriff auf die Staging-Area ist für die Transformationsklassen (vgl. Abbildung 48) nicht vorgesehen, da diese nur auf die Datenquellen zugreifen.

Das Monitoring wurde daher mittels zusätzlicher Klassen umgesetzt, die mithilfe einer Fabrikmethode instanziert werden. Die Fabrikmethode wurde gewählt, damit der Zugriff auf die Monitoring-Daten unabhängig von deren tatsächlicher Implementierung der Staging-Area erfolgt. Um festzustellen, ob die zu ladenden Daten bereits in die Staging-Area integriert wurden, können die zu ladenden Daten direkt mit den Daten der Staging-Area abgeglichen werden oder der Abgleich erfolgt mit einer eigenen Datenstruktur, die für das Monitoring erstellt wird. Im vorliegenden Fall basiert der Vergleich, der für das Monitoring notwendig wurde, direkt auf den Daten der Staging-Area.

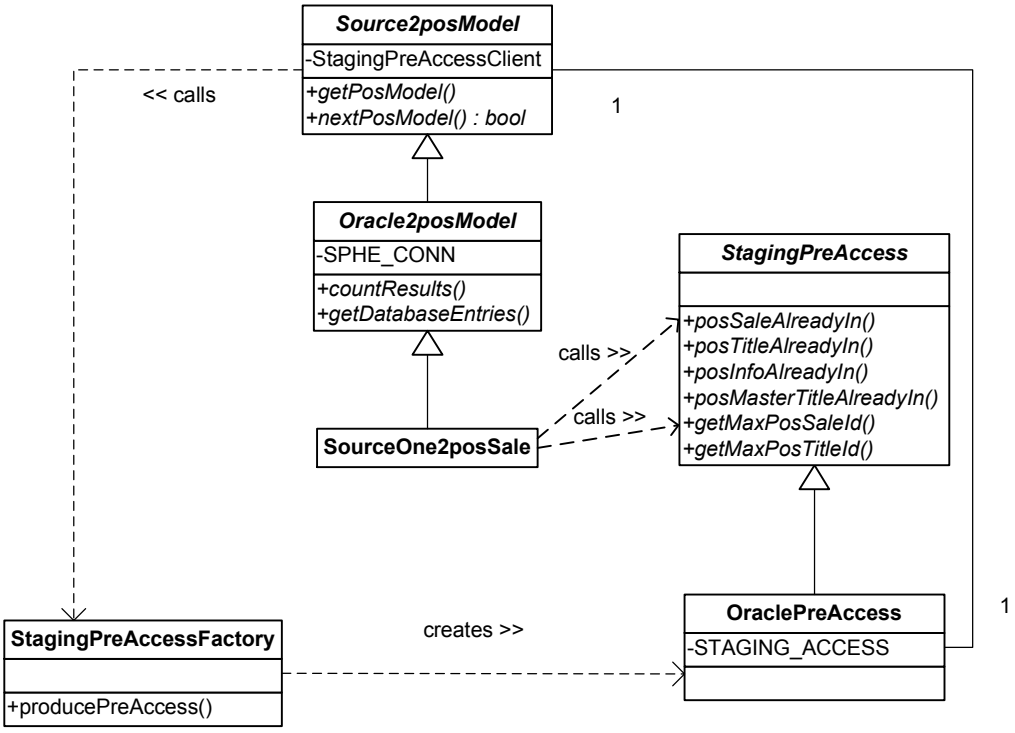


Abbildung 47: Monitoring Fabrik *StagingPreAccessFactory* am Beispiel von SourceOne2posSale (UML-Klassendiagramm)

Die Klasse *StagingPreAccessFactory* dient als Fabrik und liefert im vorliegenden Fall eine Instanz von *OraclePreAccess*, die die in der abstrakten Klasse *StagingPreAccess* vorhandenen Monitoring-Operationen implementiert. Wie in Abbildung 47 ersichtlich kann die konkrete Klasse *SourceOne2posSale* nun etwa die Operation *posSaleAlreadyIn* aufrufen, um zu

erfahren, ob ein Verkaufsvorgang bereits in die Staging-Area integriert wurde. Zusätzlich kann *SourceOne2posSale* die Operation *getMaxPosSaleId* aufrufen (vgl. Abbildung 43 SOURCE_ID in POS_SALE), anhand der erkannt werden kann bis zu welcher fortlaufenden *Sequence-ID* (vgl. 3.3.1.1) die Daten bereits in die Staging-Area integriert wurden. Alle Daten mit einer höheren *Sequence-ID* sind somit in den Extraktionsprozess aufzunehmen

5.4.3 Extraktion

Das Design des *PosIntegrator* ist so ausgelegt, dass zur Integration immer sowohl die Art der Datenquelle als auch die Art der gelieferten Daten berücksichtigt werden. Dies berücksichtigt den Umstand, dass eine Datenquelle mehrere Informationsarten, wie Produkte und Verkäufe liefern kann. In diesem Fall wäre es unzweckmäßig, die Verbindung zu einer Datenquelle mehrfach herzustellen.

Zur Realisierung der Unterteilung in Datenquellen wurde eine Vererbungshierarchie erstellt, die aus Abbildung 48 hervorgeht. Jede Datenquelle erbt vom abstrakten Typ *Source2PosModel*. *Source2PosModel* wurde nicht als Interface modelliert, da *Source2PosModel* zugleich Prozesse implementiert, die datenquellenunabhängig sind. Von *Source2PosModel* erbt pro Datenquelle eine abstrakte Klasse. Die Kernaufgabe der abstrakten Klassen *Oracle2posModel*, *AxisWs2PosModel* und *Edi2PosModel* ist, die Verbindung zu den jeweiligen Datenquellen, unabhängig von der Art der gelieferten Information, herzustellen. Dadurch ist es möglich, die Verbindung für unterschiedliche Informationsarten zu nutzen. Dies wird in Abbildung 48 etwa beim Typ *Oracle2posModel* genutzt, indem die Verbindung *Oracle2posSale* für *Sphe2posSale* (*Sale*), *Sphe2posTitle* (*Title*) und *Sphe2posMasterTitle* (*Master Title*) verwendet wird.

Zur Umsetzung der Anforderung, die vorsieht, dass Datenquellen hinzugefügt oder entfernt werden können, wurde das Entwurfsmuster der abstrakten Fabrik herangezogen. Die abstrakte Fabrik übernimmt die Aufgabe Objekte zu erzeugen, ohne dass diese direkt aufgerufen werden müssen. Das bedeutet, die Datenquellen können einfach ausgetauscht werden, da bei einem Austausch etwa einer Datenquelle nur ein Eingriff in die Fabrik, nicht aber direkt in die Programmlogik erforderlich ist. (Gamma u. a. 1995, S. 24 f.)

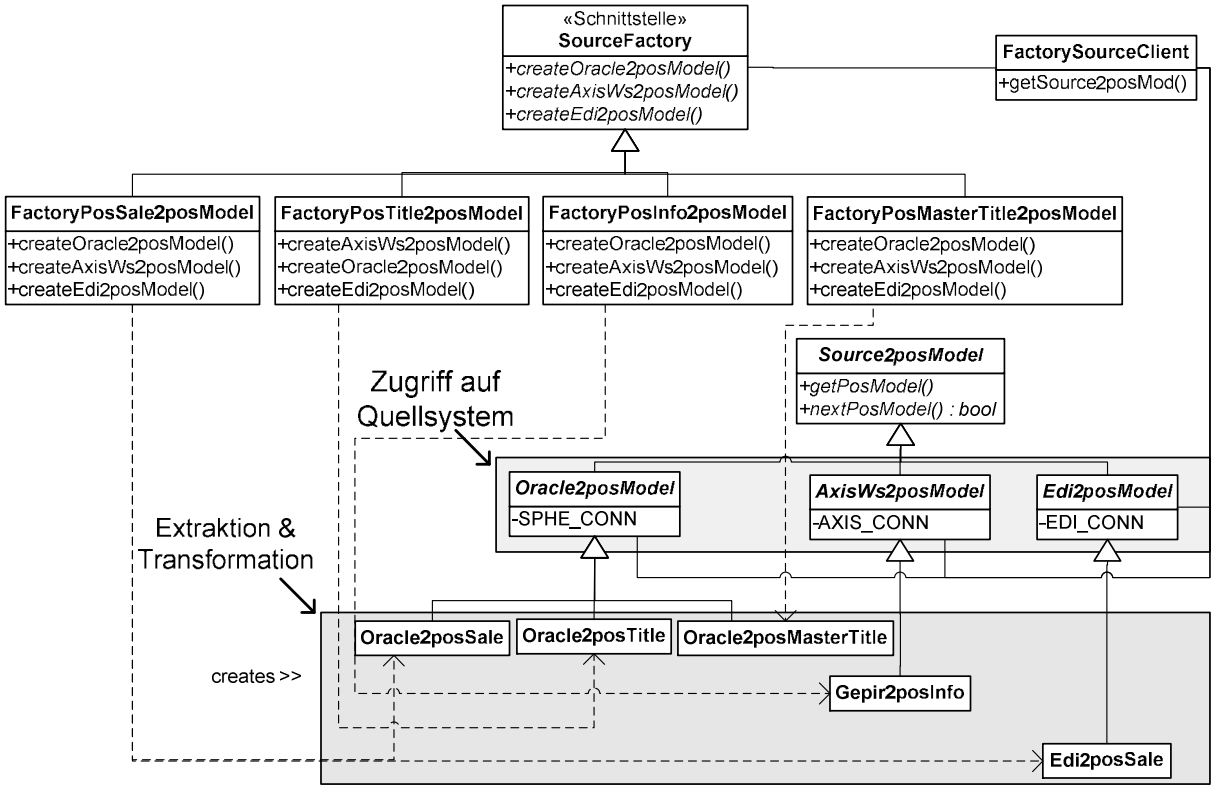


Abbildung 48: Abstrakte Fabrik zum Zugriff auf die Datenquellen und Informationsarten

Das Entwurfsmuster *abstrakte Fabrik* wurde hier realisiert, um zum einen eine klare Trennung zwischen den Datenquellen und ihren Charakteristika zu erreichen, und um zum anderen eine zusätzliche Abgrenzung der Informationsarten zu ermöglichen. Dies ist in Abbildung 48 durch die visuelle Trennung von Datenzugriff und Datenselektion veranschaulicht. Der Zugriff auf eine Datenquelle wird aufgrund der abstrakten Fabrik nur im Zusammenhang mit einer Informationsart möglich. Wird etwa der Zugriff auf Verkaufsdaten (*posSale*) der Datenquelle *Oracle2posModel* gewünscht, so muss der *FactorySourceClient* sowohl darüber in Kenntnis gesetzt werden, welche Operation der *SourceFactory* (und somit welche Datenquelle) auszuwählen ist, als auch welche konkrete *FactoryKlasse* verwendet werden soll. Nur mit diesen beiden Parametern ist der Zugriff auf eine Datenquelle möglich. Im konkreten Fall wäre dies die Operation *createOracle2posModel* der Klasse *FactoryPosSale2posModel*.

Die in den konkreten Klassen *Oracle2posSale*, *Oralce2posTitle*, *Oralce2posMasterTitle*, *Gepir2posInfo* und *Edi2posSale* extrahierten Datensätze basieren auf den Ergebnissen des Monitorings (vgl. 5.4.2).

5.4.4 Transformation

Ist die Extraktion der Daten abgeschlossen, folgt die Transformationsphase. Dazu wird jeder Datensatz der Datenquellen in ein Objekt transformiert, das von der abstrakten Klasse *PosModel* erbt. Diese Objekte werden von den in Abbildung 49 dargestellten Klassen *POS_MASTER_TITLE*, *POS_TITLE*, *POS_SALE* und *POS_INFO* instanziert. Die Attribute dieser Klassen entsprechen den Attributen der gleichnamigen Tabellen des logischen Designs der Staging-Area (vgl. 5.2.1).

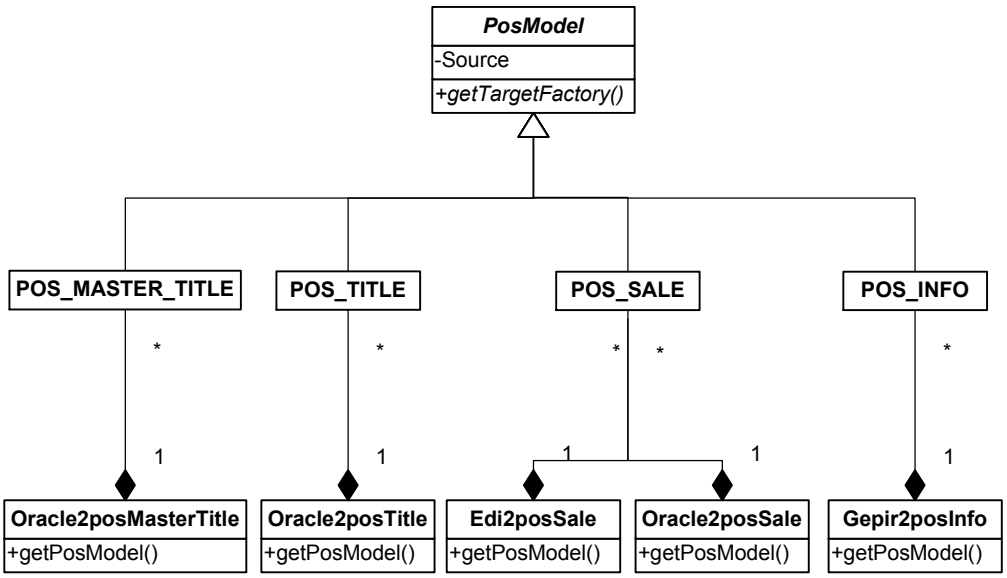


Abbildung 49: Transformation der Quelldaten in das Design der Staging-Area (UML-Klassendiagramm)
Die Kompositionsbeziehungen zwischen den Transformationsklassen und den Klassen, die von PosModel erben, zeigen, dass die konkreten *PosModel*-Objekte nur existieren, wenn sie von den Transformationsklassen mit der Operation *getPosModel* erzeugt werden.

Abbildung 49 zeigt die bei einem Transformationsvorgang benötigten Klassen. Die Transformationsklassen *Oracle2posMasterTitle*, *Oracle2posTitle*, *Edi2posSale*, *Oracle2posSale* und *Gepir2posInfo* fanden bereits bei der Extraktion der Quelldaten Anwendung (vgl. Abbildung 48). Die Bezeichnung der Extraktions- und Transformationsklassen orientiert sich im vorderen Teil an der Datenquelle und im hinteren Teil an der zu erstellenden *PosModel*-Klasse, die zugleich die jeweilige Informationsart bezeichnet, die zuvor aus der Datenquelle extrahiert wurde. Das bedeutet, dass beispielsweise die Klasse *Oracle2posSale* immer *Sales*-Daten der Datenquelle *Oracle* in *POS_SALE*-Objekte transformiert.

Wie anhand der POS_SALE-Klasse in Abbildung 49 erkennbar ist, können die konkreten POS_MODEL-Klassen auch von mehreren Transformationsklassen, wie *Edi2posSale* und *Oracle2posSale* erstellt werden, wenn, wie etwa für POS_SALE, zu einer Informationsart mehrere Datenquellen vorhanden sind.

Wird die Operation *getPosModel* aufgerufen, so transformiert diese Operation eine extrahierte Datenreiche in ein konkretes *PosModel*-Objekt, das aus den gleichen Attributen wie die gleichnamige Tabelle der Staging-Area besteht (vgl. Abbildung 44). Somit ist die Transformationslogik, die auch als Mapping bezeichnet wird, in den konkreten Klassen hinterlegt, die von *Source2posModel* erben (vgl. Abbildung 48).

Das erwartete Ergebnis der Transformation ist immer ein *PosModel*-Objekt. Sobald die aus den Datenquellen extrahierten Daten als Objekt vorliegen, ist der Transformationsvorgang aus logischer Perspektive abgeschlossen, da die *PosModel*-Objekte bereits der Datenstruktur der Staging-Area entsprechen.

5.4.5 Beladung der Staging-Area

Der letzte Schritt des *PosIntegrators* ist die Beladung der Staging-Area. Die Beladung basiert immer auf einem *PosModel*-Objekt, das bei der Transformation erzeugt wurde. Das bedeutet, dass der ETL-Prozess zur Beladung der Staging-Area nach der Transformation in ein *PosModel*-Objekt logisch unabhängig von den darunterliegenden Datenquellen erfolgt. Einzig das in allen *PosModel*-Objekten vorhandene Attribut *Source* (vgl. Abbildung 49) ermöglicht einen Rückschluss auf die Datenquelle. Wie in Abbildung 49 erkennbar, verfügt jedes *PosModel* auch über die Operation *getTargetFactory*. Diese Operation ermöglicht es, dass das *PosModel*-Objekt bei der Beladung der Staging-Area selbst entscheiden kann, in welche Zieltabelle es einzufügen ist. Dadurch ist ausgeschlossen, dass beim ETL-Prozess ein Fehler aufgrund einer falschen Zuweisung zu einer Zieldatenstruktur entsteht. Der Zugriff auf die Staging-Area wurde mithilfe der Entwurfsmuster *abstrakte Fabrik* und *Singleton* gestaltet.

Aufgabe der *abstrakten Fabrik* ist es, die jeweils richtigen Objekte zum Beladen der Staging-Area zu erstellen. Jede Tabelle in der Staging-Area wird von einem entsprechenden Objekt beladen. Die *abstrakte Fabrik* übernimmt die Funktion das jeweils richtige Objekt zur Beladung der Staging-Area zur Verfügung zu stellen. Neben dem Parameter, der angibt um welche Informationsart, also um welchen konkreten Typ von *PosModel*, es sich handelt, muss der Fabrik der Typ der Staging-Area mitgeteilt werden. Das vorliegende Design ermöglicht

somit, die Staging-Area auch auf anderen Plattformen als der gewählten Plattform zu realisieren. Da im vorliegenden Fall immer die gleiche Staging-Area verwendet wurde, wurde keine weitere Staging-Area als Ziel des ETL-Prozesses (*Target*) modelliert.

Um zu verhindern, dass bei jedem Ladevorgang die *Model2Staging*-Klassen und somit auch die Verbindung zur Staging-Area neu aufgebaut werden muss, wurden alle konkreten Produkte und Fabriken mithilfe des Entwurfsmusters *Singleton* implementiert (vgl. 6.4.1). Das bedeutet, zu jeder konkreten Fabriks- und Produktklasse existiert maximal eine Instanz. Die Fabrik wird also in diesem Fall nur bei der ersten Erzeugung tatsächlich als Erzeuger des Produktes verwendet. Später dient die Fabrik dazu, einen Zeiger auf das richtige Produkt zu geben.

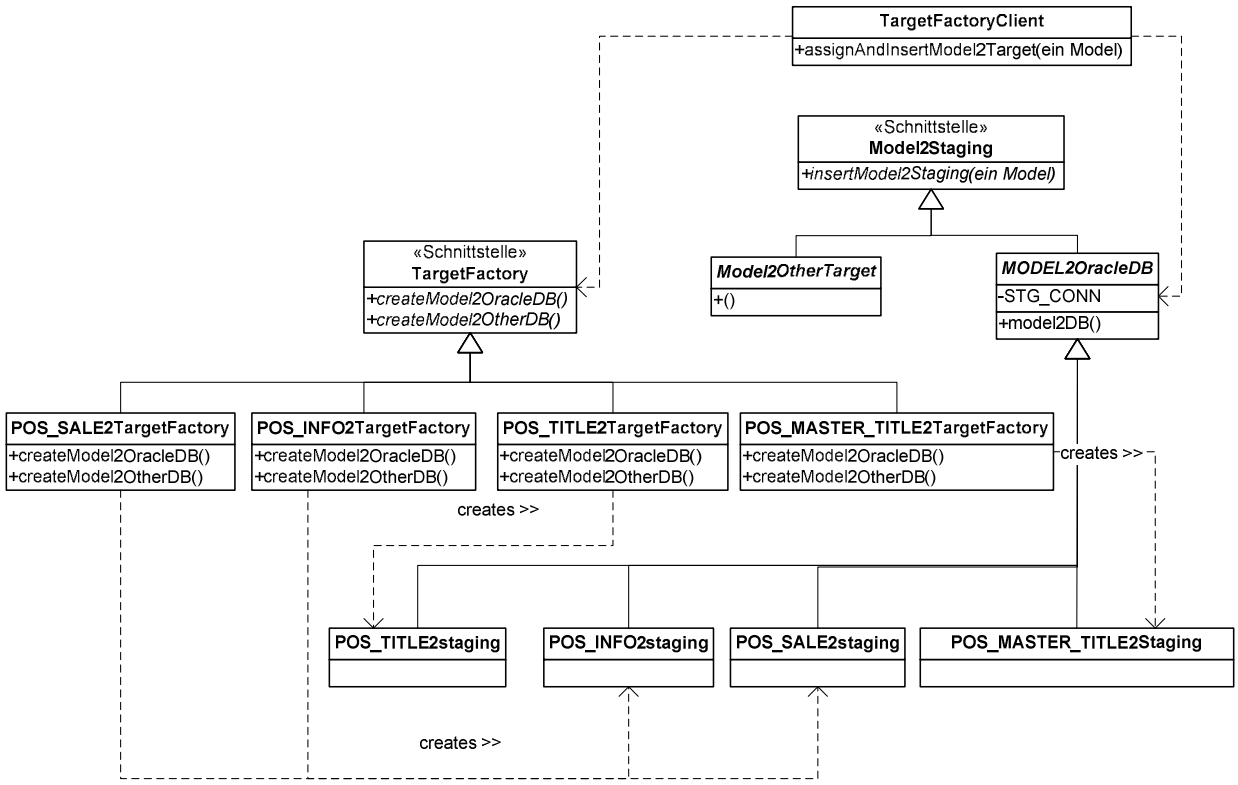


Abbildung 50: Abstrakte Fabrik zur Beladung der Staging-Area (UML-Klassendiagramm)
Die Objekte, die von *MODEL2OracleDB* erben, stellen die Produkte dar.

Die Objekte, die von *MODEL2OracleDB* erben, können nur erstellt werden, wenn die *TargetFactory* mit der gerade verwendeten Implementierung der Staging-Area aufgerufen wird. Die Entscheidung, welche konkrete *Factory* verwendet wird, wird vom übergebenen *PosModel*-Objekt getroffen. Nicht dargestellt ist in Abbildung 50 der Zugriff auf eine zweite Staging-Area, die ein weiteres Produkt darstellen würde.

5.4.6 Ablauf des ETL-Prozesses zur Beladung der Staging-Area

Um neben den statischen Klassendiagrammen auch den zeitlichen Ablauf der Programmausführung in Zusammenhang mit den eingeführten Fabriksklassen besser verständlich zu machen, stellt Abbildung 51 ein Sequenzdiagramm dar, das den Ablauf der Programmausführung in zeitlicher Abfolge darstellt. Im Sequenzdiagramm ist vertikal der Zeitablauf von oben nach unten dargestellt; horizontal sind die beteiligten Objekte angeführt, zwischen denen Nachrichten versendet werden (Balzert 1999, S. 189).

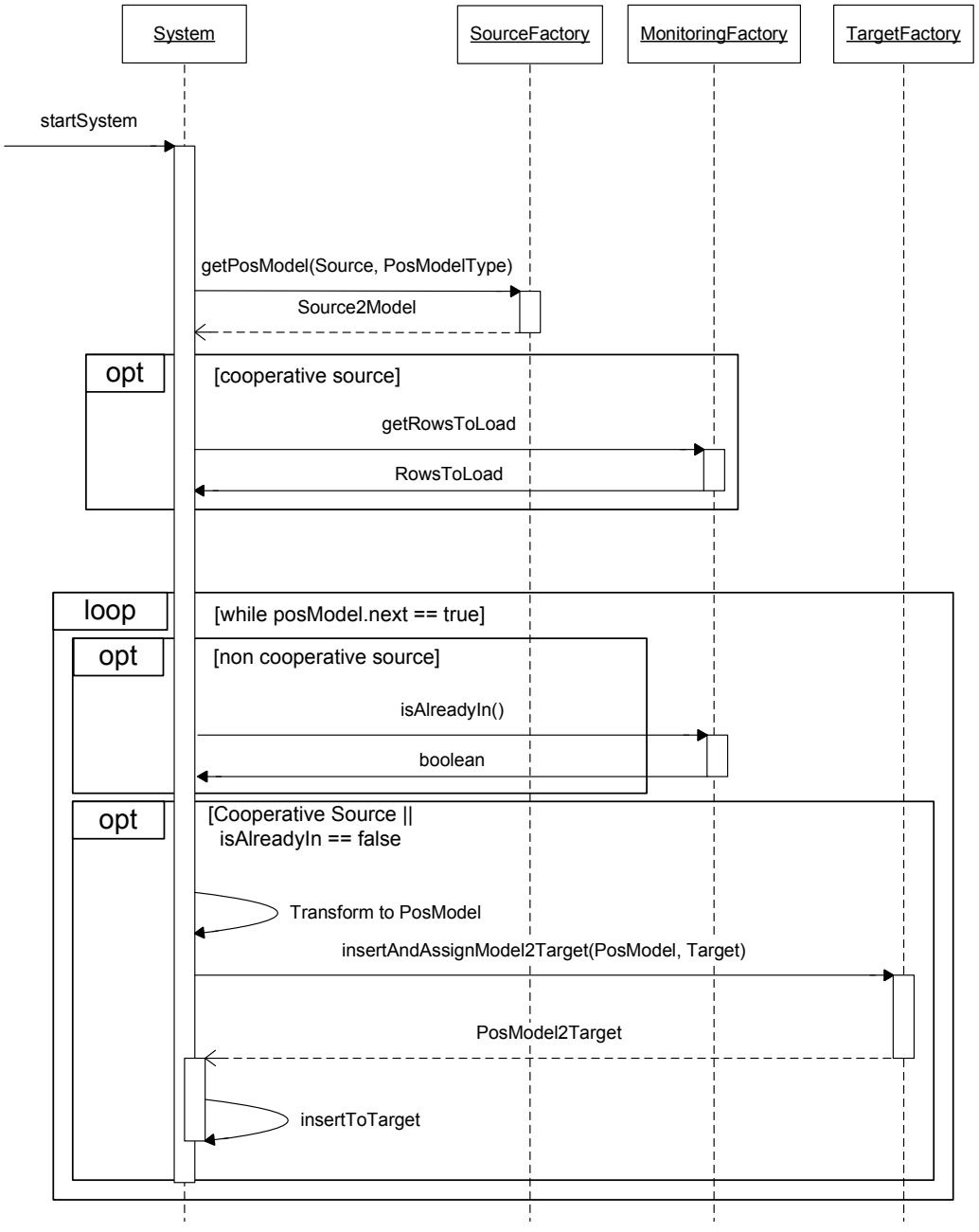


Abbildung 51: Beladung der Staging-Area mit einer unbestimmten Datenquelle (UML-Sequenzdiagramm)

Das Sequenzdiagramm in Abbildung 51 wurde so gestaltet, dass besonders die Interaktion mit den vorhandenen Fabriken besser verständlich wird. Beim Programmaufruf wird zunächst Kontakt mit der *SourceFactory* aufgenommen, deren Aufgabe es ist, das den Parametern entsprechende *Source2Model*-Objekt zurückzugeben. Als Parameter dient zum einen die Variable *Source,* die das zu verwendende Quellsystem spezifiziert und die Variable *PosModel*, die die für den ETL-Prozess zu verwendende Informationsart vorgibt. Anschließend wird das entsprechende *Source2posModel*-Objekt zurückgegeben (vgl. Abbildung 49). Handelt es sich um eine kooperative Datenquelle (vgl. 2.7.1), kann das System sofort Kontakt mit der *MonitoringFactory* aufnehmen, die mitteilt, welche Daten von der Datenquelle zu laden sind.

Anschließend wird eine Schleife betreten, deren Laufbedingung so lange gültig ist, wie ein neuer Datensatz in der Datenquelle vorhanden ist. Handelt es sich um eine *nicht-kooperative Datenquelle*, so erfolgt nun bei jedem Datensatz ein Abgleich, ob dieser Datensatz bereits in der Staging-Area vorhanden ist. Bei *nicht-kooperativen Datenquellen* muss für jeden Datensatz explizit geprüft werden, ob die zu ladenden Daten bereits in der Staging-Area vorhanden sind. Dieser Abgleich erfolgt anhand der Attribute SOURCE_ID und SOURCE, die auch in der Staging-Area hinterlegt sind. Da dieser Abgleich für jeden Datensatz durchzuführen ist, ist der ETL-Prozess bei *nicht-kooperativen Datenquellen* langsamer als der ETL-Prozess bei *kooperativen Datenquellen*, da bei *kooperativen Datenquellen* sichergestellt werden kann, dass nur Daten extrahiert werden, die noch nicht in die Staging-Area integriert wurden.

Ist ein Datensatz noch nicht in der Staging-Area vorhanden, wird der aktuell vorliegende Datensatz in ein *PosModel*-Objekt transformiert. Anschließend wird von der *TargetFactory* das für das aktuelle *PosModel*-Objekt passende *posModel2Target*-Objekt zurückgegeben, um den Datensatz anschließend in die Staging-Area einzufügen. Dieser Vorgang wird wiederholt bis die Datenquelle keine neuen Datensätze mehr zur Verfügung stellt (*posModel.next == false*).

Die in Abbildung 51 dargestellte Sequenz wird beim Programmablauf für jede angegebene Kombination aus Datenquelle und Informationsart wiederholt. So würde die in Abbildung 48 dargestellte Datenstruktur aufgrund fünf verschiedener konkreter Extraktionsklassen zu ebenso vielen Durchläufen der in Abbildung 51 dargestellten Sequenz führen.

5.5 ETL-Prozess zur Beladung des Data Warehouse

Nachdem im vorhergehenden Kapitel das Design der Befüllung der Staging-Area beschrieben wurde, ist es Ziel dieses Abschnittes, aufzuzeigen, wie der Prozess der Befüllung der Data Warehouse-Tabellen modelliert wurde. Dieser Prozess ist an bereits bei *Sony DADC* implementierte Data Warehouse Projekte angelehnt. Dazu wird zunächst ein Überblick über den Ablauf des Prozesses gegeben, bevor auf das die Umsetzung des ETL-Prozesses eingegangen wird.

5.5.1 Ablauf des ETL-Prozesses

Die Aufgabe ETL-Prozesses ist es, die Data Warehouse Tabellen (vgl. 5.2.2.2) mit den Daten der Staging-Area zu befüllen. Die Beladung des Data Warehouse unterteilt sich in folgende Hauptaktivitäten, die je nach Anzahl der Dimensions-, Fakten- und Aggregationstabellen mehrfach auszuführen sind (vgl. Abbildung 52):

1. Prüfung der *Handshake*-Tabelle (Check Handshake Table)
2. Erzeugung der Prozessmetadaten (Load Process Metadata)
3. Beladung der Dimensionstabelle (Load Dimension)
4. Beladung der Faktentabelle (Load Fact)
5. Beladung der Aggregationstabelle (Load Aggragate)
6. Aktualisierung der Prozessmetadaten (Update Process Metadata)

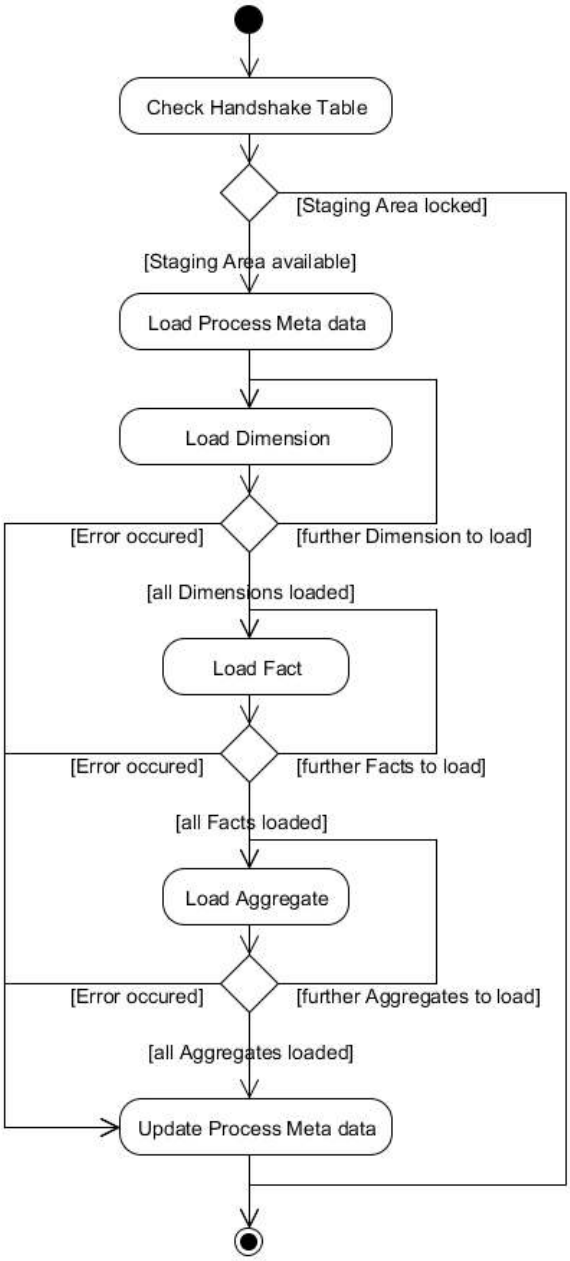


Abbildung 52: ETL-Prozess zu Beladung des Data Warehouse (UML-Aktivitätsdiagramm)

Vor Beginn des ETL-Prozesses zur Beladung des Data Warehouse muss zunächst geprüft werden ob die Staging-Area von einem anderen Prozess gesperrt ist. Diese Prüfung erfolgt anhand der *Handshake-Tabelle* (vgl. 5.6.2.2). Sollte die Staging-Area gesperrt sein, so wird der ETL-Prozess abgebrochen. In diesem Fall muss der ETL-Prozess in einem bestimmten Zeitintervall so lange neu gestartet werden bis die Staging-Area verfügbar ist.

Die Aufgabe des ETL-Prozesses ist es zunächst, Meta-Daten zum Prozess zu hinterlegen (vgl. 5.6). Anschließend werden in drei Schritten die Data Warehouse-Tabellen beladen:

1. Beladung aller Dimensionstabellen.
2. Beladung aller Faktentabellen. Die Beladung der Faktentabellen kann nur erfolgen, wenn die Beladung der Dimensionstabellen abgeschlossen ist.
3. Beladung aller Aggregationstabellen. Die Beladung der Aggregationstabellen basiert auf den Faktentabellen und kann somit erst erfolgen, nachdem die Beladung der Dimensionstabellen abgeschlossen ist.

Ist die Beladung der Dimensions-, Fakt- und Aggregationstabellen abgeschlossen, werden die Metadaten zum ETL-Prozess aktualisiert. Sollte ein Fehler auftreten, so wird der ETL-Prozess beendet und in den Meta-Daten hinterlegt, dass der Prozess abgebrochen wurde. Wurde der Prozess ohne Fehler beendet, so wird in den Meta-Daten hinterlegt, dass der ETL-Prozess erfolgreich abgeschlossen wurde.

5.5.2 Monitoring der Staging-Area

Aufgrund des uneingeschränkten Zugriffsrechts auf die Staging-Area war es möglich, die Quelldaten mit einer Spalte Status zu ergänzen, die signalisiert, ob die Daten bereits in das Data Warehouse geladen wurden oder noch zu laden sind (vgl. 5.2.1). Bei der Beladung des Data Warehouse werden somit nur jene Daten der Staging-Area berücksichtigt, deren Status signalisiert, dass sie noch nicht in das Data Warehouse integriert wurden.

5.5.3 Extraktion, Transformation und Beladung

Der Ablauf des Beladungsprozesses für Dimensionen, Fakten und Aggregate ist mit wenigen Ausnahmen für alle Data Warehouse-Tabellen gleich. Aus diesem Grund wird der Beladungsprozess nicht für Dimensions-, Fakt- und Aggregationstabellen einzeln beschrieben, sondern zusammengefasst in Abbildung 53. Der Unterschied zwischen Dimensions-, Fakt- und Aggregationstabellen wird mit der ersten Verzweigung im Aktivitätsdiagramm in Abbildung 53 berücksichtigt.

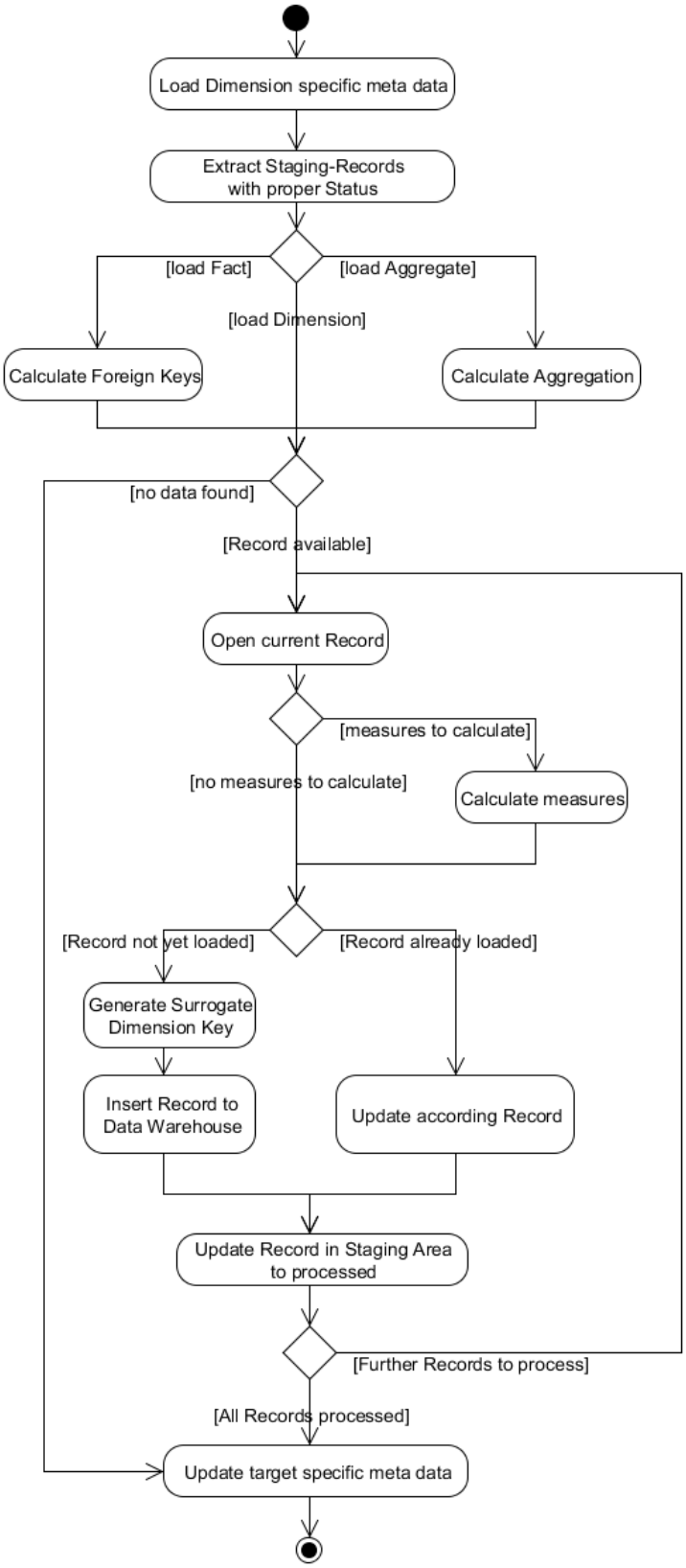


Abbildung 53: ETL-Prozess zur Beladung einer Data Warehouse-Tabelle (UML-Aktivitätsdiagramm)

Wie aus Abbildung 53 hervorgeht, werden zunächst die zu jedem Beladungsprozess spezifischen Metadaten erzeugt (vgl. 5.6). Anschließend werden alle Daten, die für die

Beladung der aktuellen Entität relevant sind, in den ETL-Prozess aufgenommen. Die Selektion basiert auf dem Status, der signalisiert, welche Datensätze noch nicht in das Data Warehouse integriert wurden, und somit noch zu laden sind. Ist eine Aggregationstabelle das Ziel des aktuellen Beladungsvorganges, so erfolgt anschließend eine Gruppierung der Zieldaten, um die gewünschte Aggregationsebene herzustellen. Ist eine Faktentabelle das Ziel, so ist es notwendig, Fremdschlüssel zum Verweis auf die einbezogenen Dimensionen zu erzeugen (vgl. 6.5).

Anschließend wird der erste zu ladende Datensatz gelesen. Falls nötig, werden nun zusätzliche Kennzahlen berechnet (vgl. Berechnung abgeleiteter Werte in 2.7.3) oder Bereinigungsmaßnahmen durchgeführt (vgl. 2.7.4).

Danach erfolgt der Abgleich mit den bereits im Data Warehouse vorhandenen Daten, um festzustellen, ob der zu ladende Datensatz bereits im Data Warehouse hinterlegt wurde. Dieser Abgleich erfolgt anhand des natürlichen Schlüssels, der in der Staging-Area und im Data Warehouse hinterlegt ist. Wird ein Datensatz mit gleichem natürlichen Schlüssel gefunden, so wird der vorhandene Datensatz mit dem zu ladenden Datensatz überschrieben, ansonsten wird der Datensatz neu in das Data Warehouse eingefügt.

Beim Einfügen eines neuen Datensatzes wird ein neuer künstlicher Schlüssel generiert, um zusammengesetzte Schlüssel zu vermeiden (vgl. Schlüsselbehandlung 2.7.3). Diese Vorgehensweise bietet zwei Vorteile:

1. Im Data Warehouse vorhandene Datensätze können einfach aktualisiert werden, falls sich deren Werte geändert haben. Der ETL-Prozess erkennt automatisch, dass der zu ladende Datensatz bereits im Data Warehouse vorhanden ist und überschreibt diesen.
2. Werden aufgrund eins Fehlers in den Datenquellen oder des ETL-Prozesses zur Beladung der Staging-Area gleiche Datensätze mehrfach in die Staging-Area eingefügt, so wirkt sich dies nicht auf das Data Warehouse aus. Im Data Warehouse wird der betroffene Datensatz anhand des gleichen natürlichen Schlüssels identifiziert und mit, in diesem Fall, gleichen Werten überschrieben. Da keine Einfügeoperation vorgenommen wird, ändert sich der Bestand der Data Warehouse-Tabellen somit nicht.

Sobald das Einfügen oder Aktualisieren des neuen Datensatzes abgeschlossen ist, erfolgt die Aktualisierung des Status in der Staging-Area, da der Datensatz nun als verarbeitet markiert

werden kann. Sind weitere Datensätze vorhanden, so wird mit diesen nun auf selbe Weise verfahren, ansonsten werden die Metadaten zum ETL-Prozess aktualisiert und die Beladung der Entität ist abgeschlossen.

5.6 Metadatenverwaltung des Data Warehouse

Unter Data Warehouse-Metadaten versteht man jene Daten des Data Warehouse, die nicht direkt dem Data Warehouse-Zweck dienen (Kimball u. a 1998, S. 22 f.). Metadaten können klassifiziert werden in

- *Metadaten über Primärdaten* (vgl. 5.6.1), die die Strukturdefinitionen sowohl der Quelldaten als auch der Staging-Area und des Data Warehouse umfassen und in
- *Prozessmetadaten* (vgl. 5.6.2), die Metadaten über den ETL-Prozess etwa in Form von Protokolldateien oder speziellen Datenbanktabellen sammeln (Bauer & Günzel 2004, S. 336 f.).

5.6.1 Metadaten über Primärdaten

In der vorliegenden Architektur des POS-Data Warehouse existieren *Metadaten über Primärdaten* für das Schema der Quelldaten, für das Schema der Staging-Area und für das Schema des Data Warehouse.

Für das Schema der Quelldaten sind die Metadaten über Primärdaten in den konkreten Klassen, die von der abstrakten Klasse *Source2posModel* erben (vgl. Abbildung 48) hinterlegt. Diese Klassen stellen Operationen zur Transformation des Ausgangschemas in das einheitliche Schema der Staging-Area zur Verfügung und enthalten somit Metadaten über die Datenstrukturen der Datenquellen.

Die Metadaten über die Primärdaten der Staging-Area, die zur Beladung der Staging-Area notwendig sind, sind in den konkreten Klassen, die von der abstrakten Klasse *Model2Staging* erben (vgl. Abbildung 50), vorhanden, da die Metadaten über Primärdaten benötigt werden, um Einfügeoperationen auf der Staging-Area ausführen zu können. Darüber hinaus befinden sich die Metadaten über die Primärdaten der Staging-Area in der Definition des Schemas der Staging-Area im RDBM der Staging-Area. Außerdem sind Metadaten über die Primärdaten der Staging-Area im ETL-Prozess zur Beladung des Data Warehouse vorhanden, da der ETL-Prozess lesend und schreibend auf die Staging-Area zugreifen muss.

Die Metadaten über die Primärdaten des Data Warehouse sind schließlich im ETL-Prozess zur Beladung des Data Warehouse, im RDBMS des Data Warehouse und in den darauf aufsetzenden OLAP Anwendungen, wie im vorliegenden Fall OBIEE, hinterlegt.

5.6.2 Prozessmetadaten

Prozessmetadaten können in folgende Bestandteile gegliedert werden (Vassiliadis 2009, S. 7):

1. **Ablauf des ETL-Prozesses**: Unter Prozessmetadaten zum Ablauf des ETL-Prozesses ist der Algorithmus zu verstehen, dem der ETL-Prozess folgt.
2. **Ausnahmebehandlung**: Prozessmetadaten zur Ausnahmebehandlung definieren wie auf Ausnahmen im ETL-Prozess reagiert wird. Sie sind somit Teil des Ablaufes des ETL-Prozesses.
3. **Ausführungsplan des ETL-Prozesses**: Darunter versteht man die Metadaten, die notwendig sind, um festzulegen, bei welchem Ereignis der ETL-Prozess angestoßen wird.
4. **Prozessstatistik**: In der Prozessstatistik werden Prozessmetadaten zum Beladungsprozess hinterlegt. Die Prozessstatistik erlaubt somit die Anzahl der Ladeprozesse sowie deren Status nachverfolgen zu können.
5. **Datenherkunftsinformation**: Prozessmetadaten zur Datenherkunft geben an, aus welcher Datenquelle die Daten im Data Warehouse stammen. Dies dient der besseren Nachvollziehbarkeit, kann aber auch für Auswertungen beim Endbenutzer verwendet werden.

In den folgenden Abschnitten 5.6.2.1 bis 5.6.2.4 wird auf die Verwendung der einzelnen Teile der Prozessmetadaten im ETL-Prozess zur Beladung des Data Warehouse näher eingegangen. Abschnitt 5.6.2.5 bietet schließlich eine Übersicht über Produzenten und Konsumenten von Prozessmetadaten.

5.6.2.1 Ablauf des ETL-Prozesses und Ausnahmebehandlung

Der *Ablauf des ETL-Prozesses* und die *Ausnahmebehandlung* wurde bereits in den Abschnitten 5.4 und 5.5 näher erläutert und wird daher hier nicht mehr näher ausgeführt.

5.6.2.2 Ausführungsplan des ETL-Prozesses

Der *Ausführungsplan des ETL-Prozesses* basiert auf den nicht funktionalen Anforderungen, die vorsehen, dass der ETL-Prozess in einem 24-Stunden-Intervall auszuführen ist (vgl. 3.2). Dazu ist zunächst der ETL-Prozess zur Beladung der Staging-Area, etwa von dem Betriebssystem, auf dem der *PosIntegrator* ausgeführt wird, anzustoßen.

Da der ETL-Prozess aus der Beladung der Staging-Area und der anschließenden Beladung der Data Warehouse-Tabellen besteht, muss der Ausführungsplan so gestaltet sein, dass sichergestellt ist, dass die Beladung der Data Warehouse Tabellen erst beginnt, wenn die Beladung der Staging-Area abgeschlossen ist. Umgekehrt soll der ETL-Prozess zur Beladung der Staging-Area nicht beginnen, bevor der ETL-Prozess zur Beladung des Data Warehouse beendet ist. Dazu wird in der Staging-Area eine *Handshake-Tabelle* eingeführt, die zur Kommunikation zwischen dem ETL-Prozess zur Beladung der Staging-Area und dem ETL-Prozess zur Beladung des Data Warehouse dient. Sobald ein ETL-Prozess angestoßen wird, wird in die *Handshake-Tabelle* ein Eintrag eingefügt, der den aktuellen Beladungsprozess beschreibt und den Startzeitpunkt des aktuellen Beladungsprozesses erfasst. Dieser Eintrag darf nur vom jeweils aktuellen laufenden ETL-Prozess entfernt werden, sobald dieser beendet ist. In Abbildung 54 ist der Zustand der *Handshake-Tabelle,* während der ETL-Prozess zur Beladung der Staging-Area ausgeführt wird, ersichtlich.

HANDSHAKE	
DESCRIPTION	**START_DT**
LOADING STAGING	01.01.2010 12:33

Abbildung 54: Handshake-Tabelle während Ausführung des ETL-Prozesses zur Beladung der Staging-Area

Die *Handshake-Tabelle* wurde eingerichtet, um zu verhindern, dass der nachgelagerte ETL-Prozess zur Beladung der Data Warehouse-Tabellen gestartet wird, während die Beladung der Staging-Area noch im Gang oder fehlgeschlagen ist. Das bedeutet, vor der Beladung der Staging-Area oder des Data Warehouse muss zunächst geprüft werden, ob in der *Handshake-Tabelle* ein Eintrag vorhanden ist, der anzeigt, dass der jeweils andere ETL-Prozess aktuell ausgeführt wird.

Der ETL-Prozess zur Beladung der Data Warehouse-Tabellen sollte bei der aktuell erwarteten Datenmenge in einem Zeitabstand von etwa einer Stunde nach dem Start des ETL-Prozesses

zur Beladung der Staging-Area angestoßen werden, da die Beladung der Staging-Area innerhalb einer Stunde abgeschlossen sein sollte. Schlägt der Ladevorgang der Staging-Area fehl oder er ist nach einer Stunde noch nicht beendet, so beginnt der ETL-Prozess zur Beladung des Data Warehouse aufgrund des Eintrages in der *Handshake-Tabelle* nicht.

5.6.2.3 Ausnahmebehandlung

Ein Teil der Ausnahmebehandlung wurde bereits beim Ausführungsplan des ETL-Prozesses beschrieben. Sollte der ETL-Prozess zur Beladung der Staging-Area fehlschlagen, so wird automatisch verhindert, dass der ETL-Prozess zur Beladung des Data Warehouse startet.

Bricht der ETL-Prozess zur Beladung der Staging-Area ab, so kann dieser nach Lokalisierung und Behebung der Ursache wieder ausgeführt werden, ohne zu inkonsistenten Daten zu führen. Dies wird durch das Monitoring sichergestellt, das bei jedem Ladevorgang prüft, welche Daten im Data Warehouse-Schema bereits festgeschrieben wurden, sodass nur jene Daten, die noch nicht festgeschrieben wurden, bei der neuerlichen Ausführung des *PosIntegrators* erfasst werden. Somit ist auch im Abbruchfall ausgeschlossen, dass Datensätze mehrfach geladen oder übersprungen werden.

Der ETL-Prozess zur Beladung des Data Warehouse kann ebenso nach einem Fehler wieder ausgeführt werden, ohne zu einem inkonsistenten Zustand zu führen. Dies ist im Design des Beladungsprozesses begründet, das verhindert, dass Datensätze mit gleichem natürlichem Schlüssel mehrfach geladen werden (vgl. Abbildung 53). Sollten Daten in der Staging-Area aufgrund eines Fehlerfalls noch nicht als bereits geladen markiert sein, obwohl sie bereits im Data Warehouse festgeschrieben wurden, so führt dies dazu, dass der betroffene Datensatz mit gleichen Daten überschrieben wird. Es kann somit im Fehlerfall nur zu einer – in diesem Fall unnötigen – Aktualisierung des Datensatzes mit gleichen Werten, nicht aber zu Dateninkonsistenzen kommen.

5.6.2.4 Prozessstatistik

Der *PosIntegrator* erstellt bei jedem ETL-Prozess zur Beladung der Staging-Area ein Protokoll, in das neben Start- und Endzeit des *PosIntegrators* Daten zum derzeitigen Status des Beladungsvorganges geschrieben werden. Weiters werden Daten zum Beladungsvorgang für jede Datenquelle und Informationsart festgehalten. Dabei werden jeweils zum Start- und

Endzeitpunkt des Beladungsvorganges für die aktuelle Kombination aus Datenquelle und Informationsart folgende Daten festgehalten:

- Ein *Zeitstempel*, der den Beginn und das Ende des ETL-Prozess aufzeichnet.
- Der *Status* des ETL-Prozesses, der den Zustand des aktuellen Einfügevorganges anzeigt.
- Die *Anzahl* der vom Monitoring-Prozess ermittelten, zu ladenden Datensätze.
- Die *Anzahl* der tatsächlich eingefügten Datensätze, sofern der ETL-Prozess erfolgreich abgeschlossen wurde.

Diese Daten sind für die Ausführung des ETL-Prozesses nicht unbedingt notwendig; sie ermöglichen jedoch im Fehlerfall schnell zu erkennen, bei welcher Datenquelle und Informationsart ein Fehler aufgetreten ist.

Vom ETL-Prozess zur Beladung des Data Warehouse werden die Tabellen WH_LOAD_HISTORY_F und WH_POS_PARAM_G befüllt (vgl. Abbildung 55). Die Tabelle WH_LOAD_HISTORY_F tritt als Faktentabelle der Prozessmetadatenverwaltung auf. Die Kennzahlen dieser Faktentabelle sind die Attribute NO_INSERT und NO_UPDATE, die die Anzahl der Einfüge- und Aktualisierungsoperationen pro Beladungsprozess für jede Data Warehouse-Tabelle speichern. In Kombination mit der Zeitdimension (WH_POS_DATE_D) sind Auswertungen über die Prozessstatistik nach der ETL_PROC_WID und nach Zeiteinheiten wie Tag, Woche, Monat, Quartal und Jahr möglich.

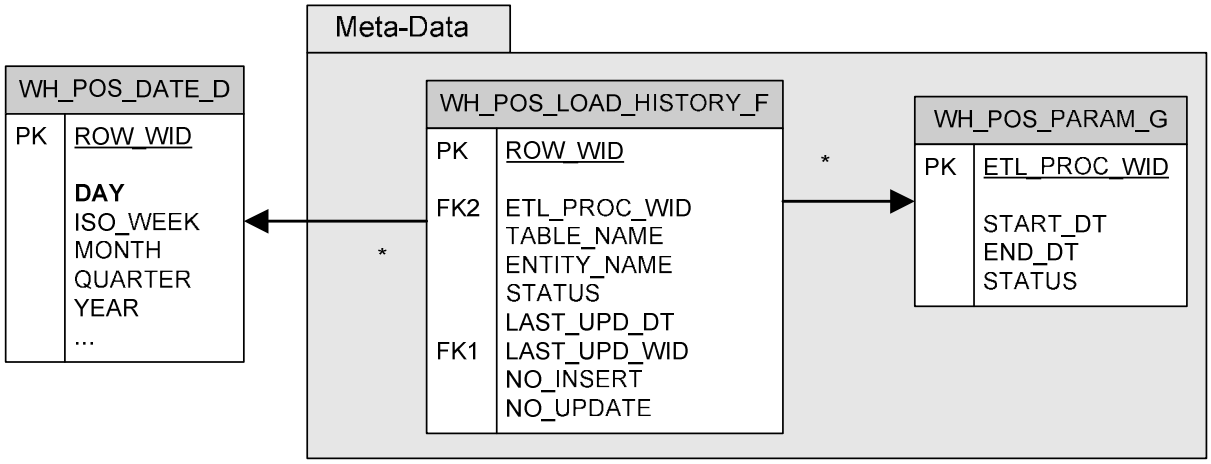


Abbildung 55: Schema der Prozessstatistik-Metadatenverwaltung des Data Warehouse

Die Tabelle WH_POS_LOAD_HISTORY_F ist mit der Tabelle WH_POS_DATE über den Fremdschlüssel LAST_UPD_WID und mit der Tabelle WH_POS_PARAM_G über den Fremdschlüssel ETL_PROC_WID verbunden.

Im Folgenden wird nun näher auf den Zweck der beiden Tabellen WH_POS_LOAD_HISTORY_F und WH_POS_PARAM_G und deren Attribute eingegangen.

Parametertabelle (WH_POS_PARAM_G)

Diese Tabelle zeigt den Gesamtzustand aller bisher gestarteten ETL-Prozesse, sowie die für jeden Ladevorgang eindeutige ETL_PROC_WID an. Die ETL_PROC_WID wird vom ETL-Prozess sequenziell erzeugt und bei jedem neuen Ladevorgang um eins erhöht, sofern der letzte Ladevorgang erfolgreich abgeschlossen wurde. Abbildung 56 zeigt den Algorithmus, nach dem die Parametertabelle befüllt wird. Jeder ETL-Prozess erhält eine eindeutige ETL_PROC_WID; sollte der ETL-Prozess fehlschlagen, so wird die ETL_PROC_WID des vorigen Prozesses wiederverwendet, bis der ETL-Prozess erfolgreich abgeschlossen ist. Das Feld STATUS kann die folgenden drei Werte annehmen:

- RUNNING: Dieser Status zeigt an, dass der ETL-Prozess gerade ausgeführt wird.
- FAILED: Dieser Status zeigt an, dass der ETL-Prozess aufgrund eines Fehlers abgebrochen wurde. Wurde der Fehler behoben, kann der ETL-Prozess mit RUNNING neu gestartet werden
- COMPLETED: Dieser Status zeigt an, dass der ETL-Prozess erfolgreich abgeschlossen wurde.

Die Felder START_DT und END_DT zeigen an wann der ETL-Prozess gestartet wurde. Das bedeutet auf den Status RUNNING gesetzt wurde und zu welchem Zeitpunkt der ETL-Prozess mit dem Status COMPLETED abgeschlossen wurde.

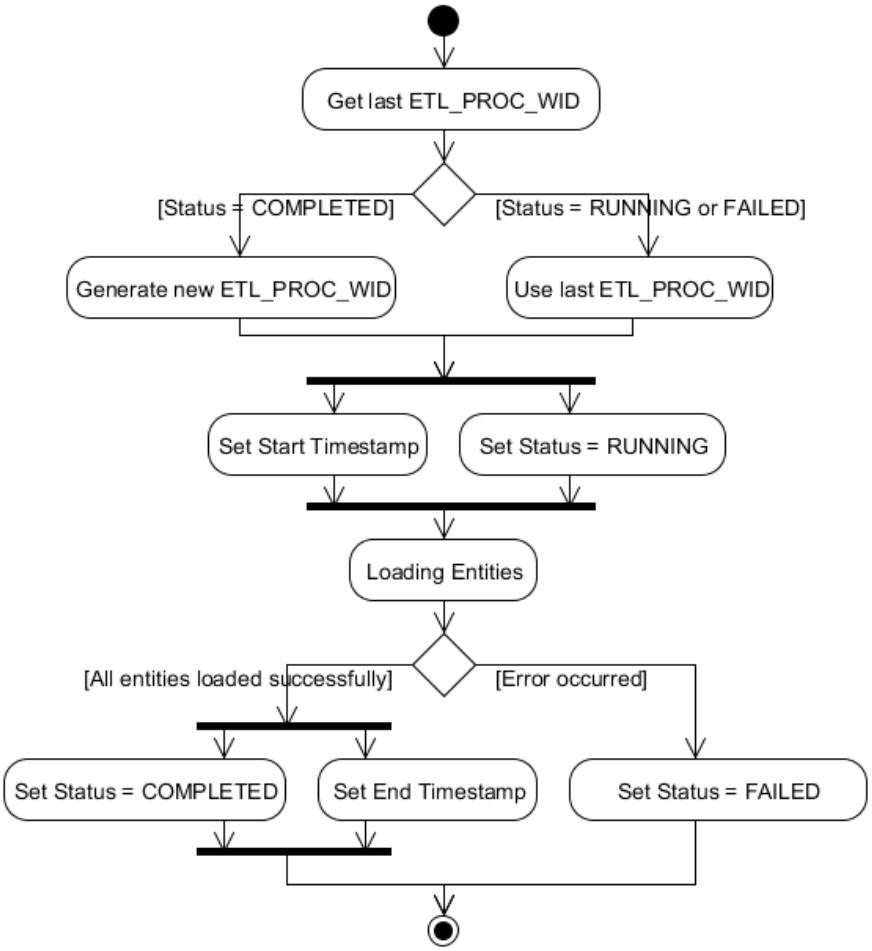


Abbildung 56: Generierung der Prozessmetadaten der Parametertabelle (UML-Aktivitätsdiagramm)

Ladeverlauftabelle (WH_LOAD_HISTORY_F)

Diese Tabelle erzeugt vergleichbar mit der Statistiktabelle der Staging-Area einen Eintrag für jede beladene Entität. Die Beladungslogik folgt dem in Abbildung 56 aufgezeigten Verlauf, jedoch auf Ebene der einzelnen Data Warehouse-Tabellen (TABLE_NAME), die mit einer logischen Beschreibung (ENTITY_NAME) ergänzt werden. Das Feld STATUS nimmt die gleichen Werte wie bei der Parametertabelle auf Ebene der einzelnen Data Warehouse-Tabellen ein. Mithilfe der Ladeverlauftabelle ist es somit möglich, ein Problem bei der Beladung auf eine Entität einzugrenzen. LAST_UPD_DT speichert jenen Zeitpunkt, an dem der Status des Beladungsvorganges einer Entität auf COMPLETED gesetzt wurde. Weiters wird für jede Entität die Anzahl der erfolgten Einfüge- und Aktualisierungsvorgänge der Data Warehouse-Tabellen in den Feldern NO_INSERT und NO_UPDATE hinterlegt.

Datenherkunftsinformation

Die Datenherkunftsinformation wird im POS-Data Warehouse mit dem Attribut DATASOURCE_ID zu jedem Datensatz hinterlegt (vgl. Abbildung 44). Mithilfe dieses Attributs ist es somit möglich, die Datenquelle des Datensatzes zu bestimmen.

5.6.2.5 Produzenten und Konsumenten der Prozessmetadaten

In diesem Abschnitt wird auf Konsumenten und Produzenten der Prozessmetadaten eingegangen. Da der *Ablauf des ETL-Prozesses*, der *Ausführungsplanes des ETL-Prozesses* und die *Ausnahmebehandlung,* vom Design des ETL-Prozesses festgelegt ist, tritt das Design auch als Produzent dieser Prozessmetadaten auf. Tabelle 8 bietet daher eine Übersicht über Produzenten und Konsumenten der übrigen Prozessmetadaten, die vom ETL-Prozess erzeugt werden

	Produzent	Konsument
Prozessstatistik	ETL-Prozess	ETL-Prozess, Administrator & Endbenutzer
Datenherkunftsinformation	ETL-Prozess	Administrator & Endnutzer

Tabelle 8: Produzenten und Konsumenten der Prozessmetadaten.

Produzent der *Prozessstatistik* und der *Datenherkunftsinformation* ist der ETL-Prozess. Die *Prozessstatistik* wird zugleich vom ETL-Prozess verwendet, um zu erkennen, ob der zuvor durchgeführte ETL-Prozess erfolgreich abgeschlossen wurde.

Weiters treten Administratoren und Endbenutzer als Konsumenten der *Prozessstatistik* und der *Datenherkunftsinformation* auf. Administratoren des Data Warehouse können die Prozessstatistik zur Fehlerbehebung oder Optimierung des ETL-Prozesses nutzen. Darüber hinaus können auch Endbenutzer die *Prozessstatistik* und die *Datenherkunftsinformation* abfragen.

6 Implementierung

In diesem Kapitel wird auf die Art der Umsetzung des in Kapitel 5 angeführten Designs der BI-Lösung eingegangen. Dafür werden zunächst die verwendeten Technologien in Abschnitt 6.1 kurz erläutert, um anschließend, in den Abschnitten 6.2 und 6.3, auf die Implementierung der Staging-Area und des Data Warehouse einzugehen. Die weiteren Abschnitte 6.4 und 6.5 erläutern die Implementierung des ETL-Prozesses zur Beladung der Staging-Area und des Data Warehouse. Abschnitt 6.6 schließlich zeigt auf wie die Analyseumgebung konfiguriert wurde.

6.1 Technologien

In diesem Abschnitt wird auf die Eignung der verwendeten Technologien Java (vgl. 6.1.1), JDBC (vgl. 6.1.2), Axis (vgl. 6.1.3), Smooks (vgl. 6.1.4) und PL/SQL (vgl. 6.1.5) zur Implementierung des in Kapitel 5 vorgestellten Designs eingegangen.

6.1.1 Java

Die entwickelte Anwendung zur Integration der Daten aus den unterschiedlichen Datenquellen, die den ETL-Prozess zur Beladung der Staging-Area implementiert, wird als *PosIntegrator* bezeichnet. Das Design des *PosIntegrators* wurde mithilfe der Programmiersprache Java in der aktuellen Version 1.6 realisiert. Java wurde herangezogen, da es zum einen die Umsetzung des objektorientierten Designs unterstützt und zum anderen aufgrund der Java Virtual Machine (JVM) einen plattformübergreifenden Einsatz ermöglicht. Darüber hinaus existiert eine Vielzahl an Programmbibliotheken, die die Anbindung an unterschiedliche Datenquellen erleichtern. Somit ermöglicht Java, der Forderung nach einer generischen Umsetzung sowohl im Hinblick auf unterschiedliche Datenquellen als auch auf im Hinblick auf einen plattformunabhängigen Einsatz nachzukommen. (Ullenboom 2009, S. 57 ff.)

6.1.2 JDBC

Eine der Programmbibliotheken, die für Java zur Verfügung gestellt werden, ist JDBC (Java Database Connectivity). JDBC ermöglicht es, eine Verbindung zu einer Datenbank oder zu anderen tabellenförmig angelegten Datenstrukturen zu erstellen. Aufbauend auf einer

Verbindung kann die Datenselektion mittels SQL vorgenommen werden. Die Daten werden schließlich an ein Objekt der JDBC-Programmbibliothek übergeben, sodass die Daten im Programm weiterverarbeitet werden können. Je nach Datenquelle ist dafür ein spezifischer JDBC-Treiber erforderlich, der, falls sich die zugrundeliegende Datenbank, nicht aber das Datenmodell ändert, transparent für den Code der erstellten Anwendung ist. (Sun 2010)

6.1.3 Axis

Axis ist ein Java-Framework, das den SOAP (Simple Object Access Protocol)-Standard für Web-Services implementiert. Wie in Abbildung 57 erkennbar, basiert die Kommunikation bei Web-Services auf WSDL (Web-Service Description Language) und SOAP XML-Dokumenten.

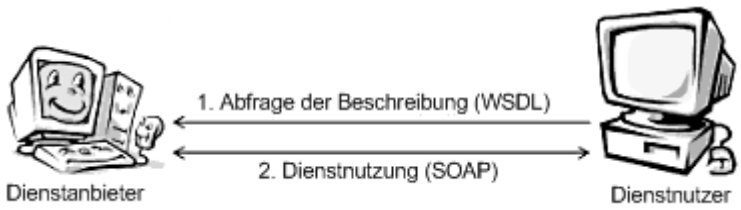


Abbildung 57: Web-Service Kommunikationsstruktur (nach Melzer 2010, S. 14)

Will ein Dienstnutzer einen Web-Service konsumieren, so benötigt dieser zunächst das zugehörige WSDL-Dokument. In einem WSDL-Dokument werden die Funktionen, die der Web-Service zur Verfügung stellt, näher beschrieben. Ist der Dienstnutzer im Besitz des WSDL-Dokumentes, so kann es in einer beliebigen Java-Anwendung von der Axis-Bibliothek verwendet werden, um die für den Zugriff notwendigen Funktionen auf der Seite des Dienstnutzers zu implementieren. Das bedeutet, Axis stellt mithilfe eines WSDL-Dokumentes Methoden in Java zur Verfügung, die eine Anfrage an den Dienstanbieter in eine SOAP-Nachricht übersetzen. Da eine SOAP-Nachricht plattformunabhängig ist, ist es für den Dienstnutzer unerheblich, welches System vom Dienstanbieter verwendet wird, sofern er die Spezifikationen des WSDL-Dokumentes erfüllt. Die Antwort des Dienstanbieters erfolgt ebenso in einem SOAP-Dokument, das mithilfe der von Axis erstellten Methoden in einer Java-Anwendung verwendet werden kann.

In der vorliegenden Implementierung wurde der Axis Version 1.4 gegenüber Axis2 der Vorzug gegeben, da Axis 1.4 alle notwendigen Funktionen bietet und als ausgereift gilt (Axis 2010).

6.1.4 Smooks und EDIFACT

Smooks ist eine Programmbibliothek für Java, die es ermöglicht, Dateien, wie etwa XML-Dokumente an Java-Objekte zu binden. Neben XML werden auch zahlreiche weitere Dateiformate wie CSV und EDIFACT unterstützt. Der Transformationsprozess von *Smooks* basiert auf XML. Das bedeutet eine EDIFACT- oder CSV-Datei wird zunächst in ein XML-Dokument konvertiert bevor die Umwandlung in ein Java-Objekt erfolgt. Das Mapping der XML-Dokumente zu Java-Klassen wird ebenfalls in einem XML-Dokument vorgenommen.

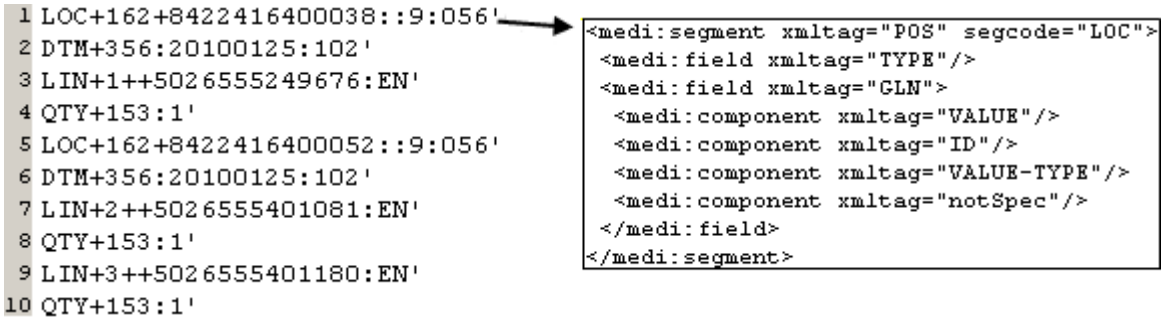
Bei der Implementierung wurde die aktuelle Version *Smooks 1.2.4* verwendet. (Smooks 2010).

EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport) ist, wie XML, ein Dateiaustauschformat (Bauer & Günzel 2004, S.146 f.). Es existieren zahlreiche unterschiedliche EDIFACT-Nachrichtentypen, die zur Beschreibung unterschiedlicher Geschäftsvorgänge verwendet werden. Zur Übermittlung von Endverkaufsdaten wird der Nachrichtentyp SLSRPT verwendet, der von der UNECE (*United Nations Economic Commision for Europe*) standardisiert ist (UNECE 2010).

Ein EDIFACT-Dokument gliedert sich immer in folgende Abschnitte:

- Segmente (Trennzeichen „**'**“), die aus mehreren
    - Datenfeldern (Trennzeichen „+“) bestehen. Datenfelder können weiters in mehrere
        - Komponenten (Trennzeichen „**:**“) unterteilt sein.

Zur besseren Lesbarkeit erfolgt nach dem Trennzeichen eines Segmentes meist ein Zeilenumbruch.


1 LOC+162+8422416400038::9:056'
2 DTM+356:20100125:102'
3 LIN+1++5026555249676:EN'
4 QTY+153:1'
5 LOC+162+8422416400052::9:056'
6 DTM+356:20100125:102'
7 LIN+2++5026555401081:EN'
8 QTY+153:1'
9 LIN+3++5026555401180:EN'
10 QTY+153:1'

<medi:field xmltag="TYPE"/>
<medi:field xmltag="GLN">
<medi:component xmltag="VALUE"/>
<medi:component xmltag="ID"/>
<medi:component xmltag="VALUE-TYPE"/>
<medi:component xmltag="notSpec"/>
</medi:field>

Abbildung 58: Ausschnitt EDIFACT-SLSRPT (links) und Smooks-Mapping für Zeile 1 (rechts)

Abbildung 58 zeigt einen Ausschnitt eines EDIFACT-SLSRPT. In Zeile eins wird mit LOC ein Segment eingeleitet, das einen Standort näher bestimmt. Im nächsten Datenelement wird mit der Zahl 162 festgelegt, dass eine Verkaufsstelle, also ein POS, näher bestimmt wird. Im dritten Datenelement wird eine Nummer angegeben, die anhand der VALUE-TYPE Komponente dieses Datenelementes, als GLN identifiziert werden kann (vgl. Abbildung 58). In den weiteren Segmenten wird der Verkaufszeitpunkt bestimmt sowie der verkaufte Artikel anhand der GTIN spezifiziert. Mit dem Segment QTY wird schließlich die Verkaufsmenge festgelegt. Ab Zeile fünf wird weiterer Verkaufsvorgang mit dem Segment LOC eingeleitet.

Mithilfe eines Mappings, wie in rechts in Abbildung 58 dargestellt, wird ein EDIFACT-Dokument in ein XML-Dokument konvertiert. Das nun vorliegende XML-Dokument kann mithilfe eines weiteren Mappings mit vergleichbarer Syntax, an die *PosModel*-Objekte (vgl. Abbildung 59) gebunden werden.

6.1.5 PL/SQL

Um die Daten von der Staging-Area in das Data Warehouse zu übertragen, wurde die Programmiersprache PL/SQL verwendet. Die Entscheidung fiel auf PL/SQL, da sowohl die Staging-Area als auch das Data Warehouse in einem Oracle RDBMS (vgl. 3.2) realisiert wurden. Da PL/SQL Prozeduren direkt im RDBMS abgelegt werden können ergeben sich im Vergleich zu Programmiersprachen, wie Java, folgende Vorteile (Pribyl & Feuerstein 2002, S. 9 ff.):

- **Fehler werden zur Übersetzungszeit erkannt**: Da PL/SQL direkt im RDBMS abgelegt ist, kann PL/SQL bereits beim Kompilieren erkennen, wenn Anfragen an die Datenbank nicht dem aktuellen Schema entsprechen. Derartige Fehler werden bei externem Zugriff erst zur Laufzeit erkannt.
- **Weniger Codezeilen**: Da PL/SQL direkt in der Datenbank ausgeführt wird, ist es nicht notwendig, zunächst eine Datenbankverbindung herzustellen oder diese im Nachhinein zu trennen.
- **Keine Konvertierung von Datentypen**: Würde eine externe Programmiersprache verwendet, müssten die RDBMS internen Datentypen in die Datentypen der jeweiligen Programmiersprache konvertiert werden.
- **Bessere Leistung**: Da die Prozeduren direkt im RDBMS abgespeichert sind, ist keine Netzwerkkommunikation oder eine zusätzliche Übersetzung der SQL Anweisungen

zur Laufzeit erforderlich, da die erforderlichen Daten bereits auf dem RDBMS abgelegt sind.

Aufgrund der großen zu erwartenden Datenmengen waren vor allem die Performancevorteile von PL/SQL ausschlaggebend bei der Entscheidung PL/SQL für die Beladung einzusetzen. Bei der Beladung der Staging-Area ist PL/SQL aber aufgrund der proprietären Strukturen, die lediglich auf die Oracle RDBMS-Umgebung angepasst sind, ungeeignet. Daneben würden sich zahlreiche Werkzeuge für die Beladung des Data Warehouse anbieten, diese wurden aufgrund des bisher bei *Sony DADC* verwendeten Rahmenprozesses hier nicht näher untersucht. Dies könnte allerdings bei in Zukunft erheblich steigenden Datenmengen notwendig werden (vgl. 8.3).

6.2 Staging-Area

Die Implementierung der Staging-Area beruht auf dem in Abschnitt 5.2.1 vorgestellten logischen Design der Staging-Area und wurde, wie in den Rahmenbedingungen angeführt, auf einem RDBMS ausgeführt.

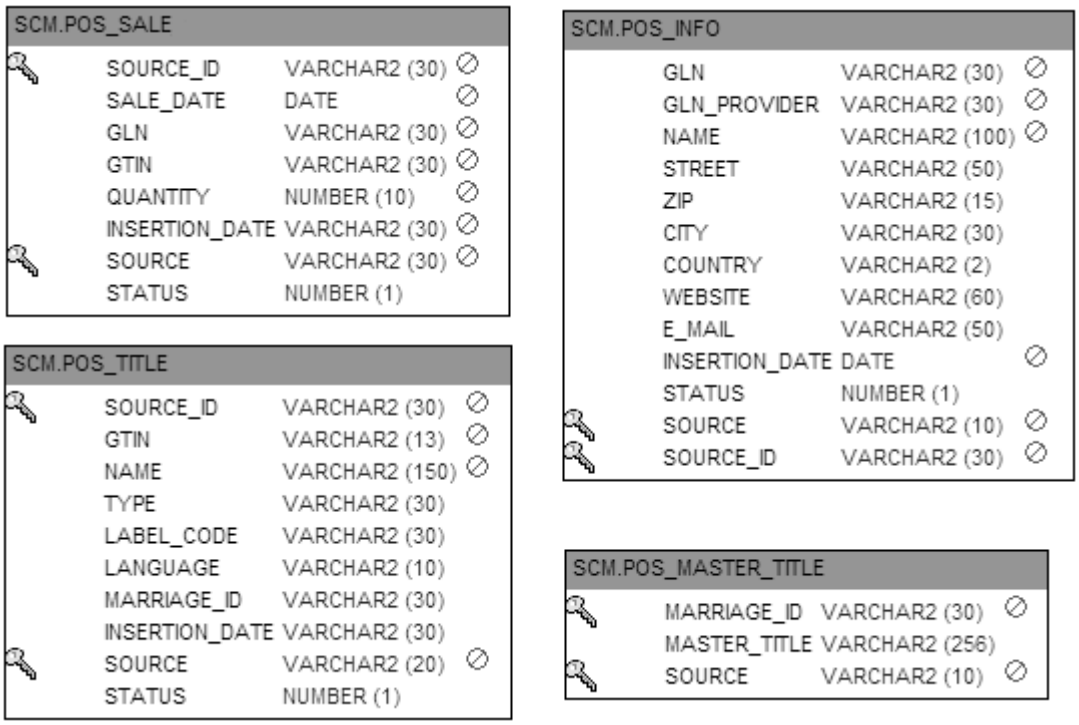


Abbildung 59: Physisches Schema der Staging-Area

Die Schlüsselsymbole signalisieren jeweils die zusammengesetzten Primärschlüssel einer Tabelle. Rechts davon werden Name und der Datentyp der Attribute angeführt. Der Kreis mit Strich symbolisiert, dass dieses Attribut immer befüllt werden muss.

Wie in Abbildung 59 ersichtlich wurde jeweils die Kombination der Attribute SOURCE und SOURCE_ID als Primärschlüssel der Tabellen festgelegt. Der Wert der Spalte SOURCE_ID wird, falls die Datenquelle über eine sequenzielle ID verfügt direkt von dieser übernommen und ansonsten vom ETL-Prozess zur Beladung der Staging-Area erzeugt. Damit ist sichergestellt, dass die Kombination aus SOURCE und SOURCE_ID einzigartig ist.

Die natürlichen Schlüsselattribute, wie die GTIN in der Tabelle POS_TITLE und die GLN in der Tabelle POS_INFO, wurden in der Implementierung nicht als Schlüssel oder Fremdschlüssel gekennzeichnet, da es möglich ist, dass etwa ein POS_TITLE mit der gleichen GLN mehrfach in die Staging-Area geladen wird um die Attribute eines *Title* zu aktualisieren. Der ETL-Prozess zur Beladung des Data Warehouse erkennt mehrfach vorhandene natürliche Schlüssel und aktualisiert die jeweils betroffenen Datensätze (vgl. 5.4).

6.3 Data Warehouse

Die Implementierung des Data Warehouse beruht auf dem in Abschnitt 5.2.2 vorgestellten logischen Design des Data Warehouse und wurde, wie in den Rahmenbedingungen angeführt, auf einem RDBMS ausgeführt.

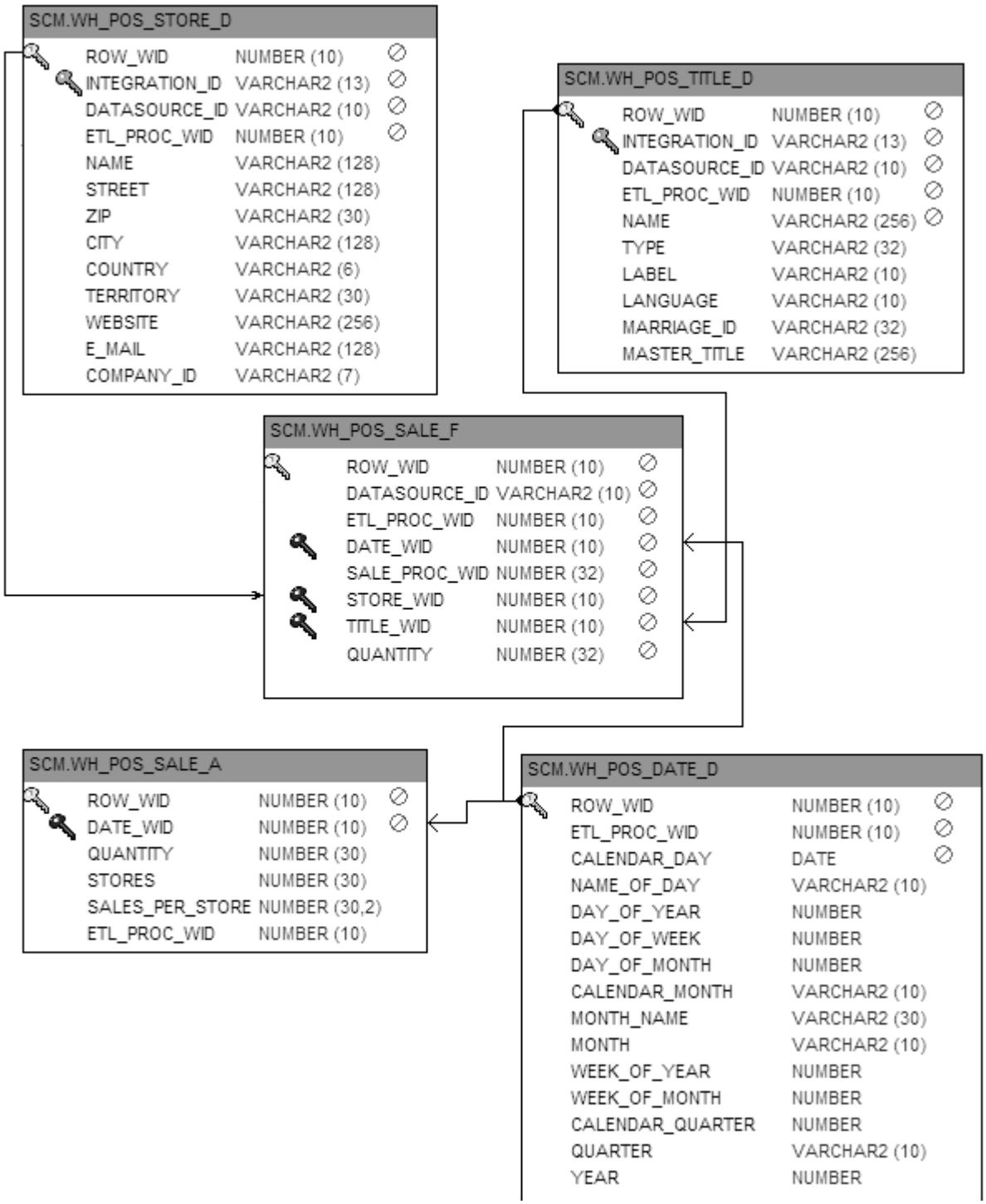


Abbildung 60: Physisches Schema des Data Warehouse

Die hellen Schlüsselsymbole stehen für den Primärschlüssel, die dunklen Schlüsselsymbole für Fremdschlüssel. Ein Kreis mit Strich symbolisiert, dass diese Attribut immer befüllt sein muss.

Abbildung 60 zeigt die Implementierung des logischen Data Warehouse-Schemas, das gemäß den Rahmenbedingungen in einem RDBMS durchgeführt wurde. Die Faktentabelle WH_POS_SALE_F ist mit den Dimensionstabellen über Fremdschlüssel verbunden. Zusätzlich wurde für das INTEGRATION_ID-Attribut der Dimensionstabellen, das dem

natürlichen Schlüssel der feinsten Granularitätsstufe der Dimensionshierarchien entspricht, ein UNIQUE-Key Constraint angelegt, da im Gegensatz zur Staging-Area auf Ebene des Data Warehouse nur ein Eintrag pro INTEGRATION_ID vorhanden sein darf.

6.4 ETL-Prozess zur Beladung der Staging-Area

Ziel dieses Abschnittes ist es, aufzuzeigen, wie die im vorhergehenden Abschnitt angeführten Technologien verwendet werden, um das in Abschnitt 5.4 erläuterte Design des ETL-Prozesses zur Beladung der Staging-Area umzusetzen. Dazu wird zunächst der Einsatz des Entwurfsmuster Singleton (vgl. 6.4.1) und Abstrakte Fabrik (vgl. 6.4.2) demonstriert. Abschnitt 6.4.3 zeigt schließlich die Implementierung des Programmablaufes.

6.4.1 Einsatz des Entwurfsmuster Singleton

Um zu verhindern, dass Klassen, wie etwa die Fabriksklasse *FactoryPosSale2PosModel,* mehrfach instanziert werden, wurde das Entwurfsmuster Singleton auf folgende Weise umgesetzt:

```
1  public class FactoryPosSale2posModel extends FactorySource {
2    //FactoryPosSale2posModel Object is created on first call
3    private static final FactoryPosSale2posModel INSTANCE
4                                = new FactoryPosSale2posModel();
5
6    /**
7     * Constructor - is only called once
8     */
9    private FactoryPosSale2posModel() {
10   }
11
12   /**
13    * Static method for singleton pattern. An object of
14    * FactoryPosSale2posModel can only be referenced
15     by using this method
16    * @return this
17    */
18   public static FactoryPosSale2posModel getInstance() {
19     return INSTANCE;
20   }
```

Abbildung 61: Singleton Entwurfsmuster in Java (Quellcodeausschnitt)

Da der Konstruktor in Zeile 9 und die statische Variable INSTANCE in Zeile 3 *private* sind, kann zur Initialisierung der Klasse, wie in Abbildung 61 erkennbar, nur die öffentliche Methode *getInstance* verwendet werden. Diese gibt einen statischen Verweis auf die Klasse

zurück. Diese Implementierung wurde gewählt, da der Aufruf des Konstruktors direkt bei der Deklaration des Attributes INSTANCE in Zeile zwei automatisch zu einer *thread*-sicheren Implementierung des Singleton Entwurfsmusters führt. (Rias A. Sherzad 2006)

6.4.2 Einsatz des Entwurfsmuster Abstrakte Fabrik

Das in Abschnitt 5.4 behandelte Design der abstrakten Fabrik lässt bezüglich der Implementierung der Fabriksklassen nur wenige Wahlmöglichkeiten, da hier die zu verwendenden Schnittstellen klar vom Design vorgegeben sind. Die Implementierung des Klienten wird jedoch vom Entwurfsmuster nicht vorgegeben. In Abbildung 62 wird daher beispielhaft der Klient der abstrakten Fabrik für den Zugriff auf die jeweils benötigte Datenquelle und Informationsart aufgezeigt.

```
1 public class FactorySourceClient {
2
3   public enum PosModelType {PosSale
4                            ,PosTitle
5                            ,PosInfo
6                            ,PosMasterTitle
7   }
8
9   public Source2posModel getPosModel(SourceType sourceType
10                                     ,PosModelType posModelType) {
11    FactorySource factorySource = null;
12
13    switch (posModelType) {
14      case PosSale:
15         factorySource = FactoryPosSale2posModel.getInstance();
16         break;
17      case PosTitle:
18        factorySource = FactoryPosTitle2posModel.getInstance();
19        break;
20      case PosInfo:
21        factorySource = FactoryPosInfo2posModel.getInstance();
22        break;
23      case PosMasterTitle:
24        factorySource = FactoryPosMasterTitle2posModel.getInstance();
25        break;
26    }
27
28    switch(sourceType) {
29      case Source1:
30        return factorySource.createOracle2posModel();
31      case Gepir:
32        return factorySource.createAxisWs2posModel();
33      case Source2:
34        return factorySource.createEdi2posModel();
35    }
36    return null;
37  }
38 }
```

Abbildung 62: Klient der abstrakten Fabrik (Quellcodeausschnitt)

Die Methode getPosModel gibt anhand der Parameter einen Verweis auf das angeforderte *Source2posModel*-Objekt.

Um zu verhindern, dass an die öffentliche Methode *getPosModel* ein ungültiger Parameter übergeben wird, wird in Zeile 3 der Aufzählungstyp *PosModelType* definiert. An anderer Stelle wurde bereits der Aufzählungstype *SourceType* definiert, der die konkrete Datenquelle bezeichnet. Die Methode *getPosModel* benötigt als Parameter sowohl den *SourceType* als auch den *PosModelType,* also die Informationsart, die von der Datenquelle übermittelt werden soll.

Mithilfe dieser Parameter instanziert der Klient zunächst ab Zeile 13 die für die Informationsart passende Fabrik. Anschließend wird mithilfe des Parameters *SourceType* ab Zeile 28 die für die jeweilige Datenquelle passende Klasse zurückgegeben. Sollte die spezifizierte Kombination aus Informationsart und Datenquelle nicht existieren, so wird anstatt einer konkreten Klasse *null* zurückgegeben.

6.4.3 Programmablauf

Neben der Schilderung der Umsetzung der Entwurfsmuster gibt Abbildung 63 Einblick in die Testklasse des *PosIntegrators*. Abbildung 63 soll aufzeigen, wie die *abstrakten Fabriken* verwendet werden, um die angeforderten Daten von den Datenquellen in die Staging-Area zu integrieren.

```
1  public static void main (String[] args) {
2
3      startUpLoggerInformation(); //Create Log file and log time
4      FactorySourceClient sourceClient = new FactorySourceClient();
5      TargetFactoryClient targetClient = new TargetFactoryClient();
6      targetClient.setTargetType(TargetType.ORACLE_TEST);
7
8      logger.info("*****Begin with pos_sale from source1 ***********");
9      Source2posModel sm1 = sourceClient.getPosModel(SourceType.Source1
10                                                    ,PosModelType.PosSale);
11     //run while new models are available
12     while(sm1.nextPosModel()) {
13       targetClient.assignAndInsertModel2Target(sm1.getPosModel());
14     }
15
16     logger.info("*****Begin with pos_sale from source2 ***********");
17     Source2posModel sm2 = sourceClient.getPosModel(SourceType.Source2
18                                                    ,PosModelType.PosSale);
19     while(sm4.nextPosModel()) {
20       targetClient.assignAndInsertModel2Target(sm2.getPosModel());
21     }
```

Abbildung 63: Testklasse, die zur Ausführung des *PosIntegrators* dient (Quellcodeausschnitt)

In Zeile 3 wird die Erstellung des Protokolls angestoßen. Anschließend werden die beiden Klienten der Fabriken seitens der Datenquelle und seitens der Staging-Area erzeugt. In Zeile 6 erfolgt die Anweisung, an die Fabrik nur jene Klassen zu instanzieren, die die Einfügeoperationen auf die Staging-Area ORACLE_TEST implementieren. Diese Zuweisung erfolgt bereits an dieser Stelle, da für einen Programmablauf nur ein Staging-Area Typ verwendet wird.

In Zeile 9 wird das erste *Source2posModel*-Objekt angefordert, das in diesem Fall eine Instanz der Zugriffsklasse für die Informationsart *PosSale* der Datenquelle *Source1* vorsieht.

In Zeile 12 tritt eine Besonderheit dieser Implementierung hervor, die sich anhand einer Schleife zeigt, deren Laufbedingung wahr ist, solange eine weitere Zeile in der Datenquelle, und somit ein weiteres, transformiertes, *PosModel*, vorhanden ist. Diese Schleife wurde auf Höhe der Testklasse gezogen, um zu verhindern, dass eine sehr hohe Anzahl an gelieferten Datensätzen, etwa bei initialen Beladungsvorgängen, zu Speicherplatzproblemen mit dem für Java verfügbaren Arbeitsspeicher führt. Stattdessen wird in Zeile 13 mithilfe der Methode *assignAndInsertModel2Target* das von der Methode *getPosModel* zurückgegebene Objekt sofort in die richtige Zieltabelle der Data Staging-Area eingefügt.

Die *targetFactory* erkennt den *PosModelTyp* anhand der Methode *getTargetFactory* (vgl. Abbildung 49). Somit ist sichergestellt, dass immer nur der aktuelle Datensatz im Arbeitsspeicher liegt und nicht alle verfügbaren Datensätze in einer Liste gespeichert werden. Sobald die Methode *nextPosModel* anzeigt, dass keine weiteren *PosModel*-Typen vorhanden sind wird die Schleife verlassen. In Abbildung 63 wurde beispielhaft noch eine weite Datenquelle angeführt, die nach dem gleichen Muster in die Staging-Area integriert wird.

6.5 ETL-Prozess zur Beladung des Data Warehouse

Ziel dieses Abschnittes ist es, aufzuzeigen, wie PL/SQL verwendet wurde, um das in Abschnitt 5.5 erläuterte Design umzusetzen.

Um die, für neue Datensätze notwendigen künstlichen Schlüssel zu erzeugen, wurde für jede Entität eine Sequenz erzeugt, die bei 1 startet und bei jedem Aufruf den Wert der Sequenz um 1 erhöht. Eine Sequenz wird auch für die ETL_PROC_ID verwendet, die jedem ETL-Prozess eine eindeutige ID zuweist.

Um zu prüfen, ob ein Datensatz bereits in der jeweiligen Zieltabelle vorhanden ist, wird ein Cursor verwendet, der den erste zu ladende Datensatz öffnet und anschließend anhand des in der Staging-Area vorhandenen natürlichen Schlüssels überprüft, ob der entsprechende Datensatz bereits in der Zieltabelle vorhanden ist. Ist dies nicht der Fall, wird der neue Datensatz mittels INSERT-Anweisung in die Zieltabelle eingefügt, ansonsten wird der vorhandene Datensatz mittels UPDATE-Anweisung aktualisiert (vgl. 5.5.3).

Für die Faktentabellen müssen vom ETL-Prozess zusätzlich die Fremdschlüssel für die jeweilige Dimensionstabelle erzeugt werden. Diese Fremdschlüssel verweisen direkt auf den künstlich erzeugten Dimensionsschlüssel. Um auf den jeweils richtigen Dimensionsschlüssel zu verweisen, wird ein *LEFT OUTER JOIN* verwendet, um sicherzustellen, dass auch Datensätze, für die kein Eintrag in der Dimensionstabelle existiert, in das Data Warehouse übertragen werden.

```
1 SELECT NVL(wh_dimension_d.generated_key,0) AS dimension_fk
2       ,staging_table.*
3     FROM staging_table
4     LEFT OUTER JOIN wh_dimension_d ON
5       dimension_key = wh_dimension_d.integration_id
```

Abbildung 64: *Left outer join* zur Befüllung der Faktentabelle

Wie in Abbildung 64 angeführt, wird, falls kein passender Eintrag in der Dimensionstabelle existiert, das Attribut mit dem *NVL* Befehl auf 0 gesetzt. Eine Reihe mit der ID 0 ist in allen Data Warehouse Dimensionstabellen vorhanden und verweist auf den künstlich eingefügten Datensatz *unspecified*. Der *LEFT OUTER JOIN* wird über den Schlüssel der Staging-Area gebildet. Selektiert wird aber der künstlich erstellte Schlüssel.

Um Aggregationstabellen zu befüllen, wird ähnlich vorgegangen wie in Abbildung 64 zu erkennen. Als Datenquelle wird die bereits vorhandene Faktentabelle verwendet und mittels GROUP BY Anweisungen aggregiert. Wird auf Monatsebene aggregiert, und ist für den fraglichen Monat ein neuer Datensatz vorhanden, so muss die Berechnung für den gesamten Monat neu ausgeführt werden.

6.6 OLAP-Benutzerschnittstelle

In diesem Abschnitt wird auf die Umsetzung der OLAP-Benutzerschnittstelle in OBIEE eingegangen.

Dazu wird zunächst in den Abschnitten 6.6.1 und 6.6.2 auf das physische und logische Mapping des Data-Warehouse Designs im Administration-Tool von OBIEE eingegangen. Abschnitt 6.6.3 erläutert die darauf Aufbauende Konfiguration der Präsentations-Ebene. Abschnitt 6.6.4 geht schließlich auf das grafische Design der Benutzerschnittstelle ein.

6.6.1 Physische Ebene

Auf physischer Ebene ist das Design der Data Warehouse-Tabellen zu modellieren. Dazu sind die Data Warehouse-Tabellen und deren Attribute zu konfigurieren. Darüber hinaus sind die Beziehung der einzelnen Tabellen wie in Abbildung 65 gezeigt zu modellieren. Diese Modellierung lehnt sich an das logische Design des Data Warehouse in Abbildung 44 an.

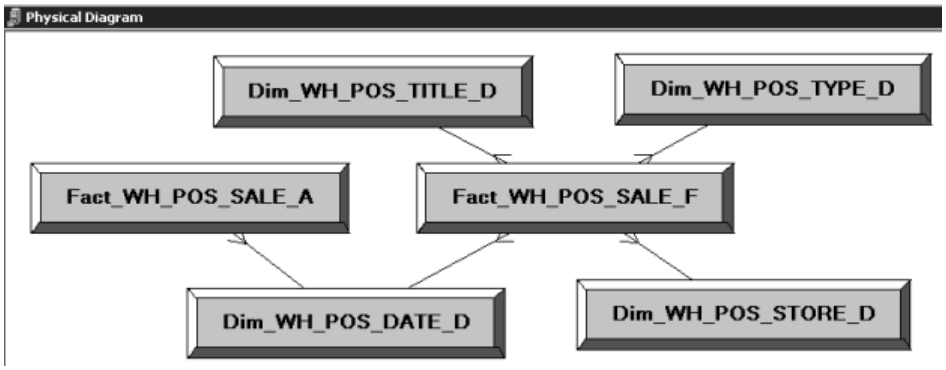


Abbildung 65: Physische Konfiguration Administration-Tool

6.6.2 Logische Ebene

Auf der logischen Ebene sind die Kennzahlen der Faktentabellen und die Dimensionshierarchien zu definieren. Die Definition dieser Hierarchien erfolgt in Anlehnung an die Dimensionshierarchien des konzeptuellen Designs in Abbildung 42.

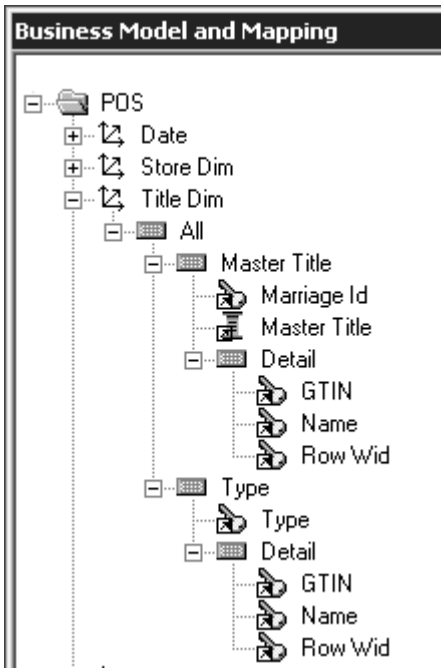


Abbildung 66: Konfiguration der Hierarchie der Title-Dimension im Administration-Tool (Screenshot)

Abbildung 66 zeigt exemplarisch die Konfiguration der *Title*-Dimension (Title-Dim). Die Konfiguration erfolgt von der gröbsten zur feinsten Granularitätsstufe (vgl. DFM in Abbildung 42). Das bedeutet Ausgangspunkt jeder Dimensionshierarchie ist die *All*-Ebene. Anschließend folgen die beiden hierarchischen Dimensionsattribute *Master Title* und *Type*. Die *Title*-Ebene wird in der Modellierung als *Detail*-Ebene angegeben.

Abbildung 67: Faktentabelle mit Kennzahlen im Adminstration-Tool

Abbildung 67 zeigt die Kennzahlen der Faktentabelle. Diese entsprechen dem DFM in Abbildung 42. Um die Aggregationsbeziehung zur Dimension Store auszudrücken, musste neben der Kennzahl *Sum-Sales*, die die Anzahl der Verkäufe in Endverkaufsstellen angibt, auch die Kennzahl *Sum-Sales-per-Store* definiert werden, die die durchschnittlichen Verkaufszahlen pro Endverkaufsstelle angibt. Die Berechnung der Kennzahl *Sum Sales per Store* ist als Division der Kennzahlen *Sum Sales* und *# of Stores* zu modellieren.

6.6.3 Präsentations-Ebene

In der Präsentations-Ebene kann konfiguriert werden, welche Dimensionen, Kennzahlen und Attribute letztlich für den Endbenutzer zur Auswahl zur Verfügung stehen. Darüber hinaus kann die Bezeichnung der Attribute und Kennzahlen angepasst werden. Bei der Implementierung wurden in der Präsentations-Ebene alle Namen und Dimensionen der logischen Ebene übernommen. Lediglich die nur auf logischer Ebene erforderliche ROW_WID der Data Warehouse-Tabellen wurde nicht auf die Präsentations-Ebenen übernommen.

6.6.4 Benutzerschnittstelle

Auf Ebene der Benutzerschnittstelle wird in OBIEE zwischen dem *Dashboard* und *Answers* unterschieden. Während Answers dem Benutzer die Möglichkeit bietet selbst Berichte zu erstellen, so stellt das *Dashboard* bereits in *Answers* gefertigte Berichte für den Endbenutzer dar. In der prototypischen Implementierung wurden vier Dashboard Seiten eingerichtet:

Die *Overview*-Dashboard-Seite soll einen Überblick über die Verkaufszahlen aller einbezogenen Verkaufsstellen bieten. Ein Ausschnitt der Overview-Dashboard-Seite ist in Abbildung 68 dargestellt. Links oben stellt eine Verkaufsanzeige das Verhältnis der Verkaufsmenge mit dem Vormonat dar. Das Balkendiagramm rechts oben visualisiert die Verkaufszahlen der letzten Monate. Links unten wird ein Kreisdiagramm angezeigt, dass die Verteilung der Verkaufszahlen auf die teilnehmenden Handelsketten visualisiert. Weiters, werden rechts unten zwei Balkendiagramme zur Entwicklung der Verkaufszahlen angezeigt.

Für alle Diagramme besteht eine *Drill-down* Möglichkeit, sodass etwa von Wochen- und Monatsverkaufszahlen einfach auf Tagesverkaufszahlen eines Monats oder einer Woche navigiert werden kann. Des Weiteren besteht die Möglichkeit von Tagesumsätzen zu den Umsätzen einzelner Verkaufsstellen oder zu den Verkaufszahlen einzelner Titel an diesem Tag zu navigieren. Darüber hinaus ist es möglich, Filter für alle dargestellten Diagramme zu setzen, um etwa den Absatzverlauf eines bestimmten Titels beobachten zu können.

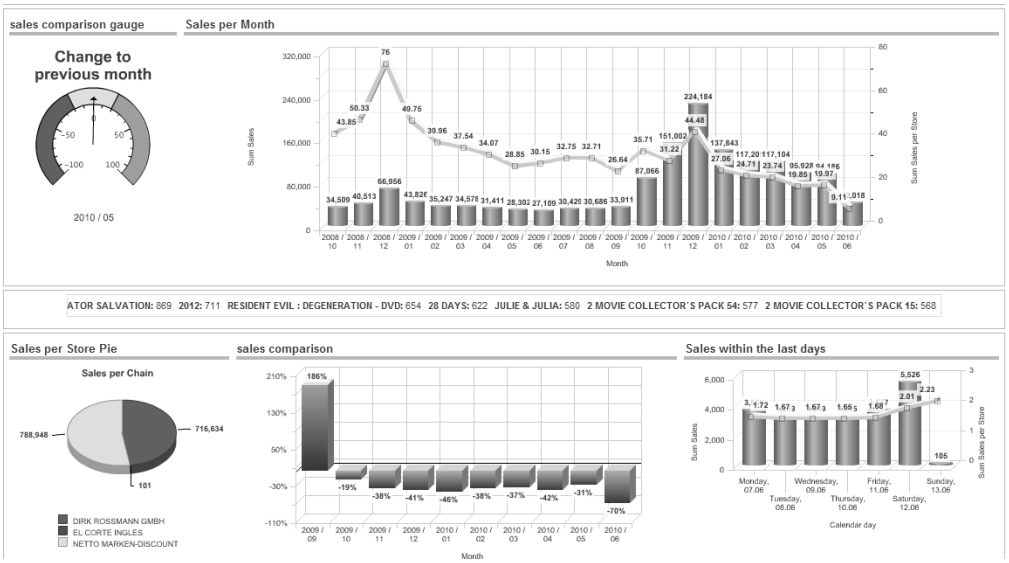


Abbildung 68: Screenshot der Overview-Dashboard-Seite in OBIEE mit einer Anzeige (links oben), einem Kreisdiagramm (links unten) und drei Balkendiagrammen

Das *Title*-Dashboard bietet die Möglichkeit eine Liste aller *Title* einzusehen, für die bisher Verkäufe registriert wurden. Darüber hinaus ist es möglich, nach bestimmten *Title* zu suchen, um deren Verkaufszahlen analysieren zu können.

Die *Store*-Dashboard-Seite bietet die Möglichkeit, eine Liste aller *Stores* einzusehen, an denen bisher Verkäufe registriert wurden. Darüber hinaus ist es möglich, nach bestimmten *Stores* regional, anhand einer Handelskette oder anhand der GLN zu suchen.

Die *Sales*-Dashboard-Seite bietet schließlich die Möglichkeit, Verkaufszahlen anhand eines benutzerdefinierten Zeitrahmens zu beobachten. Darüber hinaus werden die bestverkauften *Master Title* dieses Zeitraums, nach Verkaufszahlen absteigend sortiert, angezeigt.

Zu beachten ist, dass der Benutzer auch selber Berichte erstellen kann. Somit ist die Berichterstattung nicht auf die vorgefertigten Dashboard-Berichte beschränkt. Dies wird unter anderem in Kapitel 7 exemplarisch aufgezeigt.

7 Test

In diesem Kapitel wird die entwickelte BI-Lösung exemplarischen Tests unterzogen. Dazu wird in Abschnitt 7.1 auf die Erfüllung der funktionalen Anforderungen eingegangen. Abschnitt 7.2 untersucht darüber hinaus die erwarteten Auswirkungen der entwickelten BI-Lösung auf den Bullwhip-Effekt.

7.1 Test der Funktionalität

In diesem Abschnitt wird die Funktionalität der BI-Lösung getestet. Dazu wurde zunächst in Abschnitt 7.1.1 der Beladungsprozess untersucht, um in Abschnitt 7.1.2 die Analyseergebnisse der BI-Lösung und in Abschnitt 7.1.3 die erreichten Antwortzeiten untersuchen zu können.

7.1.1 Beladungsprozess

Da es nicht ausreicht den Beladungsprozess lediglich einmal auf seine Funktionsfähigkeit zu testen, weil es bei den Datenquellen zu unterschiedlicher Datenqualität und Datenmengen kommen kann, wurde der Beladungsprozess einige Monate getestet, wobei der Beladungsprozess in unterschiedlichen Intervallen zwischen täglich und einmal wöchentlich ausgeführt wurde. Dabei kam es bei allen durchgeführten Beladungsprozessen zu keinen Fehlern, sodass der Beladungsprozess bisher nicht abgebrochen werden musste.

	initiale Beladung	**wöchentliche Änderungen**
Verkaufsvorgänge	900 000	10 000 - 70 000
Neue Titel	5 000	0 - 20
Neue Verkaufsstellen	5 000	0 - 5

Tabelle 9: Übersicht über die Anzahl der geladenen Datenreihen je Beladungsprozess

Wie in Tabelle 9 ersichtlich, haben sich vor allem bei der Anzahl der wöchentlichen Verkaufsvorgänge große Schwankungen ergeben, da die Anzahl der Verkäufe auch saisonal abhängig ist. Die Anzahl neuer Titel oder Verkaufsstellen ist hingegen, unabhängig vom Beladungsintervall, sehr gering. Bemerkenswert ist jedoch, dass der initiale Beladungsprozess im vorliegenden Szenario ein Vielfaches an Daten lieferte, da eine Datenquelle die Verkaufsdaten bereits seit Oktober 2008 vorhielt und somit die seit über einem Jahr anfallenden Daten in nur einem Beladungsprozess erfasst wurden.

Der ETL-Prozess zur Beladung der Staging-Area nimmt aktuell für die Sammlung und Auswertung der Monitoring-Daten und die Integration der Daten bei wöchentlicher Ausführung etwa zwei bis drei Minuten in Anspruch, die Transformation der Daten in das Data Warehouse eine weitere Minute. Bei täglicher Ausführung des ETL-Prozesses zur Beladung der Staging-Area ist die Beladungsdauer um etwa 30 Prozent schneller, da nur der Extraktionsprozess, nicht aber die Auswertung der Monitoring-Daten durch eine geringere Datenmenge beschleunigt wird. Der ETL-Prozess zur Beladung des Data Warehouse ist bei täglicher Beladung etwa um das Vierfache schneller, als bei wöchentlicher Beladung des Data Warehouse. Diese Aussagen gelten jedoch nur als Richtwert und können bei anderem Datenaufkommen oder anderen Datenquellen variieren.

7.1.2 Analyseergebnisse

Um die von OBIEE erzeugten Analyseergebnisse prüfen zu können, wurde eine SQL-Abfrage direkt an das Data Warehouse gestellt. Das Ergebnis dieser Abfrage wird anschließend mit den Ergebnissen vergleichen, die von OBIEE erzielt werden.

```
1 SELECT wh_pos_store_d.CITY
2       ,SUM(wh_pos_sale_f.QUANTITY)
3 FROM wh_pos_store_d, wh_pos_sale_f
4 WHERE wh_pos_sale_f.STORE_WID = wh_pos_store_d.ROW_WID
5 GROUP BY wh_pos_store_d.CITY
```

Abbildung 69: Testabfrage an das Data Warehouse, um die Summe aller Verkäufe pro Stadt zu ermitteln

Die in Abbildung 69 gestellte Testanfrage listet zu allen Städten die dort erzielten Verkaufszahlen auf. Die Abfrage kam bei direkter Anfrage an das Data Warehouse ohne Zuhilfenahme einer Aggregationstabelle nach fünf Sekunden zu einem Ergebnis.

Zum Vergleich wurde mit OBIEE-Answers dieselbe Anforderung realisiert:

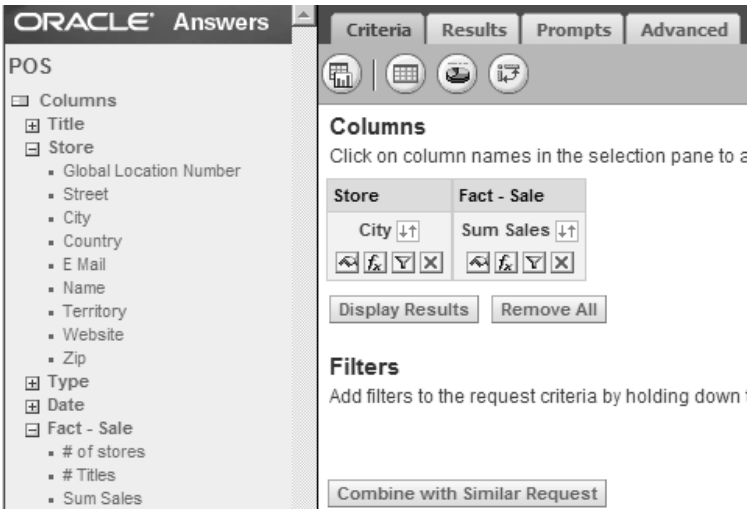


Abbildung 70: Screenshot OBIEE-Answers
Auswahl des Fakts *Sum Sales* und des Attributes *City* der Dimension *Store* aus dem Drop Down Menü links. Rechts sind die ausgewählten Spalten ersichtlich.

Dazu wurde wie in Abbildung 70 ersichtlich das Attribut *City* und der Fakt *Sum Sales* ausgewählt. OBIEE generiert anschließend auf Basis des vorgenommenen Mappings (vgl. 6.6) automatisch die in Abbildung 71 gezeigte SQL-Abfrage:

```
1 select T1460.CITY as c1,
2      sum(T1450.QUANTITY) as c2
3 from
4      WH_POS_STORE_D T1460 /* Dim_WH_POS_STORE_D */ ,
5      WH_POS_SALE_F T1450 /* Fact_WH_POS_SALE_F */
6 where  ( T1450.STORE_WID = T1460.ROW_WID )
7 group by T1460.CITY
8 order by c1
```

Abbildung 71: Von OBIEE generierte SQL-Abfrage (vgl. Abbildung 69)

Das erzielte Analyseergebnis der beiden Abfragen ist in Abbildung 72 dargestellt. Es zeigt sich, dass die beiden Abfrageergebnisse trotz unterschiedlicher Darstellung übereinstimmen. Darüber hinaus stimmt die Abfrage in Abbildung 71 logisch mit der in Abbildung 69 gestellten Abfrage überein.

CITY	SUM(WH_POS_SALE_F.QUANTITY)
BURGWEDEL	706165
MADRID	101
unspec	13
WORMS	57241
HODENHAGEN	45305
MAXHÜTTE-HAIDHOF	540568
GUTEBORN	48383
DETTINGEN	58361

City	Sum Sales
BURGWEDEL	706,165
DETTINGEN	58,361
GUTEBORN	48,383
HODENHAGEN	45,305
MADRID	101
MAXHÜTTE-HAIDHOF	540,568
WORMS	57,241
unspec	13

Abbildung 72: Analyseergebnis mit direkter SQL-Abfrage (links) und mit OBIEE (rechts)

Da das Abfrageergebnis in OBIEE mit wenigen Mausklicks erzielbar war zeigt sich, dass die Anwendung des OLAP-Werkzeuges OBIEE wesentlich benutzerfreundlicher ist, als direkte SQL-Abfragen und somit auch von Anwendern ohne SQL-Kenntnisse verwendet werden kann.

Dieser Test untermauert, dass der Endnutzer nicht direkt Zugriff auf das Data Warehouse bekommen soll, da das Werkzeug OBIEE den erforderlichen Funktionsumfang bei gleichzeitig wesentlich höherem Komfort für den Endnutzer bietet. Der Endnutzer kann je nach Rechtekonfiguration auch SQL-Anfragen direkt an das Data Warehouse stellen, diese orientieren sich jedoch an eine OBIEE spezifische Syntax, da etwaige Aggregationstabellen für den Anwender einer OLAP-Anwendung transparent sind.

7.1.3 Antwortzeit

Um alle FASMI-Kriterien, die an eine OLAP-Anwendung gestellt werden erfüllen zu können ist es nötig, dass die Antwortzeit auf gestellte Abfragen nicht über 30 Sekunden steigt. Daher wurde im Rahmen dieser Arbeit die Antwortzeit der Abfragen der *Overview*-Dashboard-Seite (vgl. 6.6.4) herangezogen. Die Berichte auf dieser Seite basieren auf der Faktentabelle, die aufgrund der größten Zahl an Einträgen auch die höchste Antwortzeit vermuten lassen.

Beim Aufruf der *Overview*-Dashboard-Seite werden derzeit neun Abfragen gleichzeitig auf die Datenbank abgesetzt

Wie in Tabelle 10 ersichtlich, liefern mehr als die Hälfte aller Abfragen nach weniger als drei Sekunden ein Ergebnis. Bei diesen Abfragen ist die Verzögerung im Zuge der Darstellung der Ergebnisse für den Benutzer kaum erkennbar. Weitere drei Abfragen liefern innerhalb von zehn Sekunden ein Ergebnis. Bei diesen Abfragen ist ebenfalls kaum eine Verzögerung

wahrnehmbar. Eine Abfrage, die die durchschnittlichen Umsatzzahlen basierend auf den Kalendermonaten berechnet, liefert nach 25 Sekunden ein Ergebnis. Diese Antwortzeit liegt noch innerhalb der FASMI-Kriterien. Bei steigendem Datenaufkommen sollte allerdings die Erstellung einer Aggregationstabelle für diese Abfrage in Erwägung gezogen werden. Die in Tabelle 10 erzielten Antwortzeiten wurden direkt nach einem neuen Beladungsvorgang erzielt.

Antwortzeit	**Anzahl der Abfragen**
< 3 Sekunden	5
3-10 Sekunden	3
10-20 Sekunden	0
20-30 Sekunden	1
> 30 Sekunden	0

Tabelle 10: Antwortzeit der Abfragen der *Overview*-Dashboard-Seite

Da die in OBIEE erzielten Abfrageergebnisse bis zum nächsten Beladungsvorgang zwischengespeichert werden, liegt die durchschnittliche Antwortzeit wesentlich unter den Antwortzeiten in Tabelle 10.

7.2 Auswirkung der Analyseergebnisse auf den Bullwhip-Effekt

Da die entwickelte BI-Lösung bisher nur zu Testzwecken verwendet wird, ist noch keine praktische Auswirkung auf Produktions- oder Lagerstände nachzuweisen. Um das Potenzial der entwickelten BI-Lösung im Hinblick auf eine Reduktion des Bullwhip-Effektes aufzuzeigen, wurden daher die Bestellmengen der Einzelhändler mit den Endverkaufszahlen der Einzelhändler, die die entwickelte BI-Lösung zur Verfügung stellt, verglichen. Die Supply Chain besteht in allen untersuchten Fällen aus den drei Gliedern Produzent, Distributionszentrum und Einzelhändler.

Der Vergleich der Bestellmenge der Einzelhandelsketten mit den Endverkaufszahlen wurde gewählt, da nur die Bestellmengen im Distributionszentrum direkt auf die bisher beteiligten Einzelhandelsketten zurechenbar sind. Ein Vergleich des gesamten Lagerbestandes eines Produktes, etwa im Distributionszentrum oder im Werk, mit den Endverkaufszahlen ist nicht sinnvoll, da aktuell nur ein Bruchteil aller Endverkaufsstellen von der entwickelten BI-Lösung erfasst wird. Somit ist ausgeschlossen, dass Lagerbestände, die für mehrere hundert Einzelhändler vorgesehen sind, mit den Absatzzahlen von lediglich zwei Einzelhändlern verglichen werden.

7.2.1 Bullwhip-Effekt in der Supply Chain von Sony DADC bei der Einzelhandelskette Netto

In diesem Abschnitt werden die in der BI-Lösung verfügbaren Endverkaufszahlen mit den Bestellmengen der Einzelhandelskette *Netto* (vgl. 3.3.1.1) beim Distributionszentrum von *Sony DADC* verglichen. Für den Vergleich wurde ein populärer *Title* gewählt, der Seit März 2010 bei der Einzelhandelskette Netto verfügbar ist.

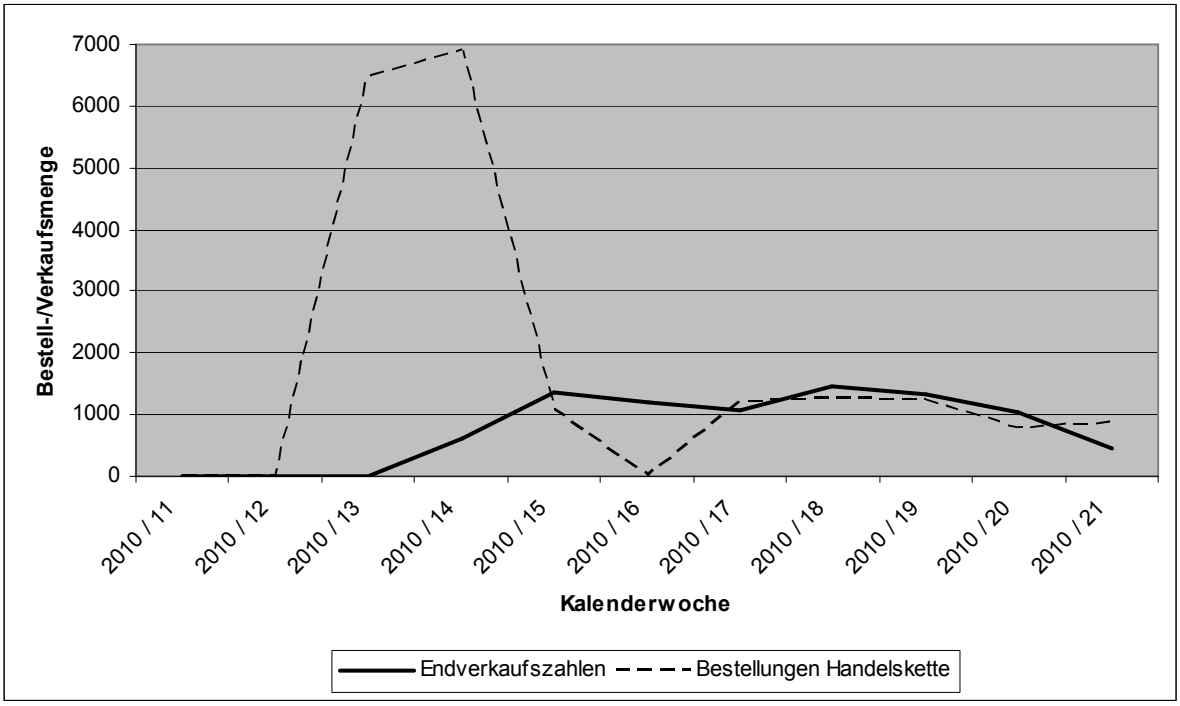


Abbildung 73: Verlauf der Verkaufszahlen und Bestellmengen der Einzelhandelskette Netto

Abbildung 73 zeigt den Verlauf der Bestellmengen der Handelskette im Vergleich zu den Verkaufszahlen beim Einzelhändler Netto. Es fällt auf, dass bei der Markteinführung des *Title* in Kalenderwoche 13 bis zu 7000 Stück pro Woche bestellt wurden. Die Bestellmengen fallen anschließend ab und sinken bis Kalenderwoche 16 auf 0. In den folgenden Perioden nähert sich die Bestellmenge den Endverkaufszahlen schrittweise an.

Die Bestellmengen weichen zum Zeitpunkt der Markteinführung stark von den Endverkaufszahlen ab. Die hohen Bestellmengen in Kalenderwoche 12 und 13 lassen sich zum mit der initialen Auffüllung der Lagerbestände in den einzelnen Filialen erklären. So dienen die Bestellmengen in Kalenderwoche 12 und 13 dem initialen Auffüllen der Verkaufsregale in den Verkaufsstellen der Handelsketten.

Der Einbruch der Bestellmengen in Kalenderwoche 16 zeigt jedoch, dass die Bestellmengen in Kalenderwoche 12 und 13 zu hoch angesetzt waren, da der Einzelhändler in den Kalenderwochen 15 und 16 seine Bestellmenge stark reduziert. Ab Kalenderwoche 17 entspricht die Bestellmenge etwa den Endverkaufszahlen. Es zeigt sich, dass die Endverkaufszahlen gegen Ende des Beobachtungszeitraumes stetig sinken. Dies lässt auf den natürlichen Produktlebenszyklus von Disc schließen, der mit zunehmendem zeitlichen Abstand zum Erscheinungsdatum stetig fallende Verkaufszahlen erwarten lässt.

Vor allem in den Kalenderwochen 13 bis 17 zeigt sich, dass die Bestellmengen stark von der Endkonsumentennachfrage abweichen. Dies ist vermutlich in der *Struktur der Bestellmenge* begründet, die vor allem bei der Markteinführung eines neuen Produktes einen hohen Sicherheitsbestand in den Lagern des Einzelhändlers vorsehen könnte. Dieser sehr hohe Sicherheitsbestand lässt darauf schließen, dass der Einzelhändler die Endkonsumentennachfrage bei Einführung des Produktes nicht abschätzen kann.

Ein weiterer Entstehungsgrund des in Kalenderwoche 13 bis 17 auftretenden Bullwhip-Effektes ist die *Entkoppelung der Nachfrage von der Endkonsumentennachfrage*. Diese ist, ähnlich wie bei Rabattaktionen, bei Einführung eines neuen Title, etwa aufgrund von Marketingaktionen, höher. Weiters dürften die eingesetzten *Prognoseverfahren* nicht in der Lage sein die Endkonsumentennachfrage bei Einführung eines Produktes einzuschätzen.

Der Einsatz der BI-Lösung in Kombination mit einem Prognosesystem, dass etwa aufgrund vorhergegangener Kinobesuchszahlen und ausgeführter Marketingaktionen Aussagen über die Endkonsumentennachfrage bereits bei Produkteinführung bieten kann, sollte in der Lage sein, die Bestellmenge auch bei Produkteinführung an die Endkonsumentennachfrage anzupassen.

Darüber hinaus können die Akteure der Supply Chain, anhand der Endverkaufszahlen der BI-Lösung, bereits in Kalenderwoche 14 erkennen, dass die Bestellmengen nicht mit den Endverkaufszahlen übereinstimmen. Somit können das Distributionszentrum und das Werk bereits abschätzen, dass die Bestellmengen in den folgenden Perioden, aufgrund der nicht bedarfsgerechten Bestellmengen in den ersten Perioden, stark absinken werden. Würde die Endkonsumentennachfrage jedoch den hohen Bestellmengen in Kalenderwoche 13 bis 14 der Endkonsumentennachfrage entsprechen, so erkennt das Distributionszentrum und das Werk, das die Bestellmengen tatsächlich mit der Endkonsumentennachfrage übereinstimmen. Bisher konnten das Distributionszentrum und das Werk, aufgrund der ausschließlichen *lokalen*

Information, nur mutmaßen ob die aktuelle Bestellmenge auch tatsächlich der Endkonsumentennachfrage entspricht.

7.2.2 Bullwhip-Effekt in der Supply Chain von Sony DADC unter Einbeziehung aller bisher registrierten Endverkaufszahlen.

In diesem Abschnitt werden, wie im vorhergehenden Abschnitt, die Endverkaufszahlen mit den Bestellmengen des Einzelhändlers verglichen. Nun werden jedoch die Bestellmengen beider Einzelhändler zusammengefasst. Weiters wurde ein *Title* gewählt, der bereits zu Beginn des vierten Quartals 2009 im Einzelhandel erschienen ist, um den Verlauf der Bestell- und Verkaufszahlen über einen längeren Zeitraum als bei 7.2.1 verfolgen zu können.

St

Abbildung 74: Verlauf der Verkaufszahlen und Bestellmengen der Handelsketten Netto und Rossmann

Abbildung 74 zeigt, dass die Bestellungen der Handelsketten bei Markteinführung ähnlich wie in 7.2.1 stark von den Endverkaufszahlen abweichen. Der vorliegende Verlauf der Bestellmengen ab Kalenderwoche 2010 / 01 legt zudem nahe, dass es zusätzlich zu den in Abschnitt 7.2.1 genannten Entstehungsgründen des Bullwhip-Effektes auch zur *Bündelung der Bestellmengen* kommt.

Die *Bündelung der Bestellmengen* führen dazu, dass Bestellungen möglichst selten, aber in großen Mengen getätigt werden um Bearbeitungs- und Transportkosten, die mit einer Bestellung verbunden sind, einzusparen. Die BI-Lösung ist in der Lage diese Bündelungsbestellungen zu erkennen.

Trifft bei *Sony DADC* eine Bestellung ein, so kann anhand der aktuellen Endverkaufszahlen erkannt werden, ob es sich um eine gebündelte Bestellung handelt, wenn die Bestellmenge bereits die erwartete Endkonsumentennachfrage der nächsten Perioden umfasst. Somit ist die vorhandene BI-Lösung in der Lage den Entstehungsgrund *Lokale Information* zu verhindern.

Deutlich zu erkennen ist in Kalenderwoche 50 und Kalenderwoche 13 der saisonale Einfluss von Weihnachten und Ostern auf die Endkonsumentennachfrage. Die Nachfrage nach Discs ist somit hauptsächlich vom Produktlebenszyklus und von saisonalen Einflüssen betroffen. Da die Nachfrage aufgrund dieser Einflüsse Schwankungen unterworfen ist, ist die Information

über die Endkonsumentennachfrage, wie in der Fallstudie von Lee, So und Tang (vgl. 2.3.2) geschildert, als wertvoll in Bezug auf die Reduktion des Bullwhip-Effektes einzuschätzen.

8 Fazit und Ausblick

In diesem Kapitel wird in Abschnitt 8.1 zunächst ein Resümee über die Entwicklung der BI-Lösung gezogen. Abschnitt 8.2 geht näher auf die Erkenntnisse ein, die aus der Anfertigung dieser Arbeit gezogen wurden. Abschnitt 8.3 bietet schließlich einen Ausblick auf Erweiterungsmöglichkeiten der entwickelten Business Intelligence-Lösung.

8.1 Fazit

Zusammenfassend lässt sich feststellen, dass sowohl auf dem Gebiet des Data Warehousing als auch am Gebiet des Supply Chain-Management bereits umfangreiche wissenschaftliche Grundlagenarbeit zum Bullwhip-Effekt in Supply Chains und zur Entwicklung von BI-Lösungen geleistet wurde. Ein Großteil der bisher publizierten wissenschaftlichen Arbeiten beschränkt sich jedoch auf die bloße Betrachtung der betriebswirtschaftlichen oder technischen Sichtweise. Das bedeutet, von betriebswirtschaftlichen Veröffentlichungen, wie sie in Kapitel 2 vorgestellt wurden, wird die Einrichtung eines Informationssystems gefordert, ohne die technischen Implikationen weiter zu beachten. Umgekehrt gehen Veröffentlichungen auf dem Gebiet der Business Intelligence zwar häufig auf organisatorische Probleme ein, die bei der Kommunikation des Entwicklungsteams der BI-Lösung mit Fachabteilungen auftreten können, nicht aber auf den wissenschaftlichen Zweck, auf dessen Grundlage ein BI-System überhaupt erst erstellt werden sollte. Das bedeutet, die Anforderungen, die an eine BI-Lösung gestellt werden, sollten auf wissenschaftlichen Erkenntnissen beruhen und nicht einzig auf dem Wunsch der Mitarbeiter von Fachabteilungen.

Um diese Verbindung der betriebswirtschaftlichen Anforderungen mit den informationstechnischen Aspekten zu ermöglichen, sollte daher für Business Intelligence-Projekte die Wirtschaftsinformatik als Wissenschaft herangezogen werden, da sie betriebswirtschaftliche Anforderungen mit dem nötigen technischen Hintergrundwissen verknüpft (vgl. Heinrich 2001).

Am aktuellen Stand kann von der BI-Lösung nur ein geringer Einfluss auf den Bullwhip-Effekt erwartet werden, da noch von zu wenigen Verkaufsstellen Daten an *Sony DADC* übermittelt werden. So wurden im Jahr 2009 Daten von etwa 750 000 Verkäufen übermittelt, wobei im gleichen Zeitraum etwa 700 Millionen Discs produziert wurden (vgl. Abschnitt 1.1.1). Das bedeutet, dass momentan deutlich weniger als ein Prozent aller produzierten

Verkaufseinheiten von der BI-Lösung erfasst werden, auch wenn man berücksichtigt, dass Verkaufseinheiten aus mehreren Discs bestehen können. Diese Implikationen waren jedoch bereits bei Beginn der Entwicklung der BI-Lösung bekannt, die Umsetzung unterstützt jedoch das Hinzufügen neuer Datenquellen, sodass der aktuelle Stand der Umsetzung der BI-Lösung als Ausgangssituation für die Integration weiterer Funktionen und Datenquellen zu sehen ist.

Die Testergebnisse in Abschnitt 7.2 zeigen, dass die BI-Lösung großes Potenzial bietet, da anzunehmen ist, dass die Bestellverläufe auch bei weiteren Einzelhändlern ähnlich wie bei den bereits in BI-Lösung integrierten Einzelhändlern verlaufen werden. Somit bietet die vorliegende BI-Lösung, sofern die Anzahl der Verfügbaren Datenquellen ausgeweitet wird, das Potenzial die Kosten für Lagerung und Transport um etwa 13 Prozent senken (vgl. 2.3.4.1).

8.2 Erkenntnisse

Die Erkenntnisse, die aus der Erstellung dieser Arbeit gezogen wurden, sind vielschichtig und können sowohl auf die informationstechnische als auch auf die betriebswirtschaftliche Komponente unterteilt werden.

8.2.1 Informationstechnische Ebene

Auf informationstechnischer Ebene hat sich gezeigt, dass bei der Entwicklung einer BI-Lösung aufgrund der unterschiedlichen Datenquellen zahlreiche Hürden zu überwinden sind, um die Integration der Daten zu ermöglichen. Bei vielen Datenquellen war vor allem eine mangelhafte Dokumentation anzukreiden, sowohl was die Zugriffsmöglichkeiten auf die Daten als auch was das Schema der Datenspeicherung betrifft. Letztlich waren jedoch die notwendigen Metadaten durch „Trial and Error“ oder bei externen Datenquellen durch ausführliche Recherche im Internet aufzufinden. Es zeigt sich also, dass bei BI-Projekten damit zu rechnen ist, dass ein nicht unerheblicher Teil der Entwicklungszeit in die Recherche zu den jeweiligen Datenquellen investiert werden muss, um die ungenügende Dokumentation auszugleichen.

Mit der Datenqualität der Quelldaten wurden unterschiedliche Erfahrungen gemacht. So waren die Daten von externen Datenquellen oft unvollständig oder ungenau, allerdings nur sehr selten offensichtlich unrichtig. Bei Daten zu den Verkaufsvorgängen konnte in einigen Fällen ebenso eine fehlerhafte Übermittlung nachgewiesen werden. Das bedeutet, dass BI-

Systeme mit falschen Quelldaten richtig umgehen müssen. Das bedeutet, es sind Mechanismen vorzusehen, die es ermöglichen unrichtige Quelldaten, wenn immer möglich, zu erkennen um zu verhindern, dass diese Daten in das Data Warehouse geladen werden. Besonders Daten, die trotz eines offensichtlich fehlerhaften Zeitstempels in das Data Warehouse gelangen, werden bei den Endnutzern Zweifel an der Richtigkeit der Daten im Data Warehouse aufkommen lassen. Daher sollte versucht werden, solche Daten im Beladungsprozess zu erkennen um zu verhindern, dass unrichtige Daten dem Endbenutzer zur Verfügung gestellt werden.

Bezüglich der Performance der eingerichteten BI-Lösung ist anzumerken, dass besonders Aggregationstabellen beim vorliegenden Data Warehouse eine erhebliche Steigerung der Antwortzeit ermöglichen. Dies gilt vor allem im Bezug auf die eingesetzte OLAP-Anwendung OBIEE, die beim Aufruf vorgefertigte Grafiken anzeigt, die großteils auf aggregierten Werten basieren. Aus diesem Grund wird, zumindest im aktuellen Stadium der Entwicklung, besonders der Einsatz von Aggregationstabellen zur Verringerung der Antwortzeit empfohlen.

Sollten die Datenmengen für den Beladungsprozess erheblich zunehmen, so kann in Zukunft auch der Einsatz externer ETL-Werkzeuge, die eventuell eine performantere Integration der Daten in das Data Warehouse ermöglichen, evaluiert werden. Dies wird jedoch zunächst nur den ETL-Prozess zur Beladung des Data Warehouse betreffen, da ein ETL-Werkzeug kaum die Mächtigkeit des *PosIntegrators* im Bezug auf die Integration von Daten aus Web-Services oder kompliziert aufgebauten Dokumenten (vgl. 6.1.4) erreichen wird.

8.2.2 Betriebswirtschaftliche Ebene

Auf betriebswirtschaftlicher Ebene wurde die Erkenntnis erzielt, dass die Umsetzung einer BI-Lösung vor allem auf dem Willen und dem Vertrauen der Supply Chain-Partner zur Zusammenarbeit beruht. Das bedeutet, vor der Entwicklung einer BI-Lösung sind zunächst Aufklärungsarbeit und vertrauensbildende Maßnahmen zu setzen, um eine dauerhafte Teilung der Endverkaufsdaten sicherzustellen. Dies ist vor allem im Hinblick darauf zu sehen, dass die beste BI-Lösung wertlos ist, wenn sie nicht mit ausreichend Daten versorgt wird. Somit ist festzuhalten, dass besonders die Kooperation auf betriebswirtschaftlicher Ebene für eine erfolgreiche Implementierung einer unternehmensübergreifenden BI-Lösung notwendig ist.

Alle Probleme, die bisher auf informationstechnischer Ebene aufgetreten sind, waren innerhalb kurzer Zeit lösbar. Treten jedoch Schwierigkeiten bei der Zusammenarbeit der Supply Chain-Akteure auf so kann eine unternehmensübergreifende Business Intelligence-Lösung substanziell gefährdet werden, da eine derartige BI-Lösung nur dann zu einem nutzbaren Ergebnis führen kann wenn möglichst viele Supply Chain-Akteure Daten zur Verfügung stellen und bereit sind die Informationen der Business Intelligence-Lösung zu nutzen. Somit ist der Erfolg einer Business Intelligence-Lösung zur Reduzierung des Bullwhip-Effektes vor allem vom Willen zur Zusammenarbeit der Supply Chain-Akteure abhängig.

Der in Abschnitt 7.2 durchgeführte Vergleich zwischen Bestellzahlen des Einzelhändlers und der von der BI-Lösung dargestellten tatsächlichen Endkonsumentennachfrage bietet weiters die Erkenntnis, dass bereits die Bestellmenge des Einzelhändlers, vor allem bei Produkteinführungen, erheblich von der tatsächlichen Endkonsumentennachfrage abweicht. Auch der im Grundlagenteil vorgestellte Verzögerungs- und Aufschaukelungseffekt ließ sich nachvollziehen. Dies lässt eine weitere Aufschaukelung beim den Bestellmengen des Distributionszentrum beim Werk und somit auch bei der Produktionsmenge vermuten. Die Information der BI-Lösung über die Endkonsumentennachfrage kann somit eingesetzt werden um zu erkennen wie sich die Bestellmengen in den folgenden Perioden entwickeln werden.

8.3 Ausblick

Die aktuell wesentlichste Einschränkung der vorliegenden BI-Lösung im Hinblick auf die Reduzierung des Bullwhip-Effektes stellt die relativ geringe Anzahl an beteiligten Verkaufsstellen dar. Eine Ausweitung der Anzahl an beteiligten Endverkaufsstellen ist daher unumgänglich. Dies ist aber nur durch eine weiter verbesserte Kommunikation innerhalb der Supply Chain möglich. Das bedeutet, es müssen die in Abschnitt 2.1.4 vorgeschlagenen Maßnahmen, wie etwa eine Reduktion der Wiederbeschaffungszeit für alle teilnehmenden Händler ergriffen werden, um die Einzelhandelsketten zu einer Teilnahme an der Supply Chain-Kooperation zu überzeugen.

Darüber hinaus kann die aktuell vorliegende BI Lösung, wie in den Modellen in Abschnitt 2.3 erläutert, als zentrale Planstelle fungieren. Dazu ist ein Ausbau der aktuellen Lösung aus technischer Sicht notwendig, um Daten etwa zum Lagerstand oder zu Umsatz in die BI-Lösung zu integrieren. Ein derartiges zentrales Planungssystem könnte schließlich auch für

die Implementierung eines VMI-Konzeptes genutzt werden. Dies würde zu einer Supply Chain vom *Typ 3* führen, bei denen ein VMI auf die im zentralen Planungssystem vorhandene Lagerstandsinformation aufbauen kann. Die Umsetzung als zentrales Planungssystem könnte gemäß den Erkenntnissen aus Abschnitt 2.3 noch zu einer wesentlich weiter reichenden Reduktion des Bullwhip-Effektes führen, als dies die derzeitige Implementierung vermag.

Für die im zentralen Planungssystem vorgesehene Prognose wird im zentralen Planungssystem auch die Einbeziehung externer Faktoren vorgeschlagen. Besonders zu erwähnen ist in diesem Fall der Zusammenhang, der zwischen Einspielergebnissen eines Kinofilmes und den späteren Verkaufszahlen des Filmes auf DVD oder BluRay erwartet wird. Dazu wäre es möglich, zunächst mit einer einfachen linearen Regression einen Zusammenhang zwischen den Verkaufszahlen und den Einspielergebnissen herzustellen. Diese Daten könnten etwa mit Data Mining-Werkzeugen erhoben werden, um in die zentrale Prognose einzufließen. Damit eine derartige Regressionsanalyse ermöglicht wird, ist es allerdings erforderlich, die Umsatzzahlen von Kinofilmen sowohl national als auch international als externe Datenquelle für das Data Warehouse unter Beachtung der in Abschnitt 2.6 festgelegten Kriterien anzukaufen.

Zusammenfassend lässt sich feststellen, dass die vorliegende BI-Lösung noch in vielen Punkten ausgebaut werden kann. Zunächst ist jedoch der Gewinn weiterer Partner des Einzelhandels für die Übermittlung von Endverkaufszahlen anzustreben. Das bedeutet beim aktuellen Stand des Projektes ist neben einem technischen Ausbau der BI-Lösung vor allem eine kontinuierliche Verbesserung der Kooperation der Supply Chain-Partner erforderlich.

9 Abkürzungsverzeichnis

BI	Business Intelligence
CRM	Customer Relationship Management
CSV	Comma Separated Values
DFM	Dimensional Fact Model
EDIFACT	Electronic Data Interchange For Administration, Commerce and Transport
ERP	Enterprise Resource Planning
ETL	Extract, Transform, Load
FASMI	Fast Analysis of Shared Multidimensional Information
GE	Geldeinheiten
GLN	Global Location Number
GTIN	Global Trade Item Number
ID	Identifier
IO	Input/Output
MDX	Multidimensional Expressions
MOLAP	Multidimensional OLAP
OBIEE	Oracle Business Intelligence Enterprise Edition
OLAP	Online Analytical Processing
POS	Point of sale
ROI	Return on Invenstment
RDBMS	Relational Database Management System
ROLAP	Relational OLAP
SCM	Supply Chain-Management
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
u.a.	und andere
URL	Uniform Resource Locator
vgl.	vergleiche
WSDL	Web Service Description Language
XML	Extensible Markup Language

10 Abbildungsverzeichnis

11 Tabellenverzeichnis

12 Literaturverzeichnis

Alicke, K., 2005. *Planung und Betrieb von Logistiknetzwerken: Unternehmensübergreifendes Supply Chain Management* 2. Aufl., Berlin: Springer.

Anahory, S. & Murray, D., 1997. *Data Warehousing in the Real World: A Step-by-step Guide for Building Decision Support Data Warehouses*, Amsterdam: Addison-Wesley.

Arentzen, U., 2006. *Gabler Kompakt-Lexikon Wirtschaft,* 9. Aufl., Wiesbaden: Gabler.

Axis, 2010. *WebServices - Axis*. Available at: http://ws.apache.org/axis/ [Zugegriffen Mai 22, 2010].

Balzert, H., 1999. *Lehrbuch der Objektmodellierung*, Heidelberg; Berlin: Spektrum, Akad. Verlag.

Bauer, A. & Günzel H., 2004. *Data-Warehouse-Systeme*, Heidelberg: dpunkt-Verlag.

Beckmann, H., 2004. *Supply Chain Management*, Berlin; Heidelberg: Springer-Verlag.

Booch, G., Rumbaugh, J. & Jacobson, I., 2006. *Das UML-Benutzerhandbuch*, München: Addison-Wesley Verlag.

Chamoni, P. & Gluchowski, P., 2006. *Analytische Informationssysteme*, S. 3-22, Berlin; Heidelberg: Springer.

Corsten, H. & Gössinger, R., 2001. *Einführung in das Supply Chain Management*, Oldenbourg Wissenschaftsverlag.

Davis, M. & Heineke, J., 2004. *Operations Management: Integrating Manufacturing and Services* 5. Aufl., McGraw-Hill/Irwin.

Disney, S., 2003. The effect of vendor managed inventory (VMI) dynamics on the Bullwhip Effect in supply chains. *International Journal of Production Economics*, 85(2), S. 199-215.

Du, T.C., Wong, J. & Lee, M., 2004. Designing Data Warehouses for supply chain management. In *Proceedings of the IEEE International Conference on E-Commerce Technology*. S. 170-177.

El Corte Inglés, 2009. *El Corte Inglés Informe Anual 2008*. Available at: http://sgfm.elcorteingles.es/SGFM/ECI/recursos/doc/Datos_Economicos/Memorias/2007/1913266246_2882009112433.pdf.

Floyd, C., 1984. A systematic look at Prototyping. *Approaches to prototyping.* S. 1-18.

Forrester, J.W., 1961. *Industrial Dynamics,* Student Edition., Productivity Press.

Gamma, E., Helm, R. & Johnson, R.E., 1995. *Design Patterns. Elements of Reusable Object-Oriented Software.* 1. Aufl., Addison-Wesley.

Golfarelli, M., Maio, D. & Rizzi, S., 1998. The Dimensional Fact Model: A conceptual model for Data Warehouses. *International Journal of Cooperative Information Systems,* vol. 7, n. 2&3, S 215-247.

GS1, 2010. *The Value and Benefits of the GS1 System of Standards*. Available at: http://www.gs1.org/sites/default/files/docs/GS1_System_of_Standards.pdf [Zugegriffen Mai 8, 2010].

Harren, A. & Herden, O., 1999. MML und mUML - Sprache und Werkzeug zur Unterstützung des konzeptionellen Data Warehouse Designs. *Proceedings 2.GI-Workshop "Data Mining und Data Warehousing als Grundlage moderner entscheidungsunterstützender Systeme (DMDW99)"*, S. 57-68.

Heinrich, L.J., 2001. *Wirtschaftsinformatik. Einführung und Grundlegung*, München: Oldenbourg Wissenschaftsverlag.

Heusler, K.F., 2004. *Implementierung von Supply Chain Management*, Wiesbaden: Deutscher Universitäts-Verlag.

Holweg, M. u. a., 2005. Supply chain collaboration: making sense of the strategy continuum. *European Management Journal*, 23(2), S. 170-181.

Hosoda, T. u. a., 2008. Is there a benefit to sharing market sales information? Linking theory and practice. *Computers & industrial engineering*, 54(2), S. 315-326.

ISO 8601, 2010. ISO Numeric representation of Dates and Time. Available at: http://www.iso.org/iso/support/faqs/faqs_widely_used_standards/widely_used_standards_other/date_and_time_format.htm [Zugegriffen Mai 9, 2010].

Jarke, M., 2003. *Fundamentals of Data Warehouses*, Berling; Heidelberg: Springer.

Keller, S., 2004. *Die Reduzierung des Bullwhip-Effektes*, Wiesbaden: Deutscher Universitäts-Verlag.

Kemper, H., Mehanna, W. & Unger, C., 2004. *Business Intelligence - Grundlagen und praktische Anwendungen*, Wiesbaden: Vieweg+Teubner Verlag.

Kimball, R. u. a., 2008. *The Data Warehouse lifecycle toolkit*, Wiley.

Kuhn, A. & Hellingrath, B., 2002. *Supply Chain Management*, Springer.

Lee, H.L., Padmanabhan, V. & Whang, S., 1997. Information Distortion in a Supply Chain: The Bullwhip Effect. *Management Science*, 43(4), S. 546-558.

Lee, H.L., So, K.C. & Tang, C.S., 2000. The value of information sharing in a two-level supply chain. *Management science*, 46(5), S. 626-643.

Lee, M.L. u. a., 1999. Cleansing Data for Mining and Warehousing. *Proceedings of the 10th International Conference on Database and Expert Systems Applications,* S. 751-760.

Melzer, I., 2010. *Service-orientierte Architekturen mit Web Services*, Heidelberg: Spektrum Akademischer Verlag.

Mertens, P., 2002. Business Intelligence - ein Überblick. *Arbeitspapier an der Universität Erlangen-Nürnberg 2/2002*, 2002(2).

Milling, Gößler, 2001. Management von Material und Informationsflüssen. *Supply Chains: System-Dynamics-basierte Analysen,* Industrieseminar Mannheim.

Netto, 2010. *Netto Marken-Discount*. Available at: http://www.netto-online.de/presse/index.php?168 [Zugegriffen Mai 8, 2010].

Oestereich, B., 2005. *Die UML 2.0 Kurzreferenz für die Praxis*, München: Oldenbourg Wissenschaftsverlag.

Oracle, 2010. OBIEE. Available at: http://www.oracle.com/appserver/business-intelligence/enterprise-edition.html [Zugegriffen Mai 3, 2010].

Pendse, N. & Creeth, R., 2008. What is OLAP? Available at: http://www.bi-verdict.com/fileadmin/FreeAnalyses/fasmi.htm [Zugegriffen Mai 2, 2010].

Pribyl, B. & Feuerstein, S., 2002. *Learning Oracle PL/SQL*, O'Reilly Media, Inc.

Rahm, E. & Do, H.H., 2000. Data cleaning: Problems and current approaches. *Bulletin of the Technical Committee on Data Engineering*, S. 3-14.

Rias A. Sherzad, 2006. Singleton Pattern in Java. Available at: http://www.theserverside.de/singleton-pattern-in-java/ [Zugegriffen Januar 2, 2010].

Rossmann, 2010. Rossmann - Der Drogerie-Markt. Available at: http://www3.rossmann.de/frm_index.htm [Zugegriffen Mai 8, 2010].

Sapia, C. u. a., 1998. Extending the E/R Model for the Multidimensional Paradigm. *Advances in Database Technologies Lecture Notes in Computer Science, Springer Berlin*, (1552), 105-116.

Schönsleben, P., 2007. *Integrales Logistikmanagement: Operations and Supply Chain Management in umfassenden Wertschöpfungsnetzwerken*, Berlin: Springer.

Schrefl, M., 2006. Datenmodellierung - Grundlagen und Richtlinien für den relationalen Datenbankentwurf. *Datenmodellierung Vorlesungsunterlagen Wintersemester 2006/2007*.

Smooks, 2010. Main Page - Smooks. Available at: http://www.smooks.org/mediawiki/index.php?title=Main_Page [Zugegriffen Januar 2, 2010].

Sun, 2010. JDBC Overview. Available at: http://java.sun.com/products/jdbc/overview.html#2 [Zugegriffen Mai 22, 2010].

Totok, A., 2000. *Modellierung von OLAP- und Data-Warehouse-Systemen*, Wiesbaden: Deutscher Universitäts-Verlag.

Ullenboom, C., 2009. *Java ist auch eine Insel*, Bonn: Galileo Press. Available at: http://openbook.galileocomputing.de/javainsel8/index.htm.

UNECE, 2010. UN/EDIFACT Message SLSRPT Release: 00A. Available at: http://www.unece.org/trade/untdid/d00a/trmd/slsrpt_c.htm [Zugegriffen Januar 6, 2010].

Vahrenkamp, R., 2007. *Risikomanagement in Supply Chains*, Berlin: Erich Schmidt Verlag.

Vassiliadis, P., 2009. Data Warehouse Metadata. *Encyclopedia of Database Systems, Springer, 2009*. Available at: http://www.cs.uoi.gr/~pvassil/publications/2009_DB_encyclopedia/DW_metadata.pdf .

Versteegen, G., 2002. *Software-management*, Berlin; Heidelberg: Springer.

Westerman, P., 2001. *Data warehousing: using the Wal-Mart model*, Academic Press.

Whitehorn, M., Zare, R. & Pasumansky, M., 2006. *Fast track to MDX*, Springer.

Zäpfel, G. & Wasner, M., 1999. Der Peitschenschlageffekt in der Logistikkette und Möglichkeiten der Überwindung chaotischen Verhaltens. *Logistikmanagement*, 1(4), S. 297- 309.