technische universität
dortmund

Bachelorarbeit

Modellierung von Betriebsdaten als Planungsgrundlage in der Logistik

Tim Lutz

September 2014

Gutachter:
Prof. Dr. Teubner
Prof. Dr. ten Hompel

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Datenbanken und
Informationssysteme (6)
http://dbis.cs.tu-dortmund.de/

In Kooperation mit:
Fakultät für Maschinenbau
Lehrstuhl für Förder- und Lagerwesen

Inhaltsverzeichnis

1. Einleitung **1**
1.1. Motivation . . . 1
1.2. Zielsetzung . . . 1
1.3. Aufbau der Arbeit . . . 2

2. Grundlagen **3**
2.1. Data-Warehouse . . . 3
2.1.1. Ziele eines Data-Warehouse . . . 3
2.1.2. Dimensionale Modellierung . . . 4
2.2. ETL-Prozess . . . 9
2.3. Prozesse in einem Lagersystem . . . 10
2.3.1. Wareneingang . . . 11
2.3.2. Einlagerung . . . 11
2.3.3. Auslagerung . . . 12
2.3.4. Kommissionierung . . . 12
2.3.5. Verpackung und Versand . . . 13
2.4. Warehouse Managementsystem . . . 13
2.4.1. Lagerverwaltung . . . 13
2.4.2. Reorganisation . . . 14
2.4.3. Fördermittelverwaltung . . . 14
2.4.4. Datenerfassung . . . 14
2.4.5. Inventur . . . 15

3. Analyse **16**
3.1. Basisdaten & Kennzahlen . . . 16
3.1.1. Basisdaten . . . 16
3.1.2. Kennzahlen . . . 17
3.2. Quelldaten . . . 18
3.2.1. Stammdaten . . . 19
3.2.2. Bewegungsdaten . . . 22

4. Konzept **27**
4.1. Dimensionen . . . 27
4.1.1. Datumsdimension . . . 27
4.1.2. Tageszeitdimension . . . 29
4.1.3. Artikeldimension . . . 30
4.1.4. Lagerplatzdimension . . . 32
4.1.5. Ladehilfsmitteltypendimension . . . 33
4.1.6. Mitarbeiterdimension . . . 34
4.2. Fakten . . . 35
4.2.1. Wareneingangsfakt . . . 35
4.2.2. Einlagerungsfakt . . . 36
4.2.3. Auslagerungsfakt . . . 37
4.2.4. Kommissionierfakt . . . 37
4.2.5. Warenverladungsfakt . . . 39
4.3. Mögliche Erweiterungen . . . 39
4.3.1. Dimensionserweiterungen . . . 40
4.3.2. Faktenerweiterung . . . 41

5. Implementierung **43**
5.1. Architektur . . . 43
5.2. Projekt Struktur . . . 44
5.3. ETL-Prozess . . . 44
5.3.1. Extraktion . . . 44
5.3.2. Transformation . . . 46
5.3.3. Laden . . . 50
5.3.4. Fehlerbehandlung . . . 51
5.4. Auswertung . . . 52
5.4.1. Datenquelle . . . 52
5.4.2. Schemadatei . . . 53

6. Zusammenfassung und Ausblick **57**
6.1. Zusammenfassung . . . 57
6.2. Ausblick . . . 57
6.2.1. Quelldaten . . . 57
6.2.2. Anwendung . . . 58
6.2.3. Kennzahlenbildung . . . 58

A. Weitere Informationen **59**

Abbildungsverzeichnis **60**

Literaturverzeichnis **62**

Erklärung **62**

1. Einleitung

1.1. Motivation

Jegliche Form der logistischen Planung benötigt Daten, auf deren Grundlage Entscheidungen getroffen werden. Häufig sind diese Daten jedoch nicht strukturiert und mit Fehlern behaftet. Dies liegt insbesondere an der heterogenen Struktur der verfügbaren Software-Systeme, welche zur Datenhaltung eingesetzt werden. In Unternehmen werden diverse Anwendungen eingesetzt, um den Materialfluss zu steuern (Materialflussrechner), zu koordinieren (Warehouse Managementsystem) und zu verwalten (ERP-System). Diese Anwendungen unterscheiden sich in der eingesetzten Datenbank und speichern die anfallenden Daten in ihren eigenen proprietären Formaten. Aufgrund der verschiedenen Anwendungsbereiche, in denen diese Applikationen eingesetzt werden, sind die generierten Daten sehr inhomogen. Ein Warehouse Managementsystem generiert Daten zu Warenbewegungen eines Lagersystems. Ein ERP-System erzeugt hingegen betriebswirtschaftlich relevante Daten. Ein weiteres Problem stellt die wachsende Datenmenge dar, die durch eine automatische Generierung entsteht.

1.2. Zielsetzung

Im Rahmen dieser Arbeit wird ein Datenmodell entwickelt, welches die Daten eines Warehouse Managementsystems einheitlich abbildet und miteinander in Relation setzt. Das Modell ist flexibel aufgebaut, sodass nachträgliche Änderungen und Erweiterungen möglich sind. Eine Überführung der Rohdaten in das neu zu entwickelnde Modell ist deshalb wichtig, weil nur so eine robuste, homogene und valide Datenbasis geschaffen werden kann, welche eine effiziente und übersichtliche Auswertung der Daten ermöglicht. Es wird eine Software-Lösung entwickelt, die den Import von logistischen Rohdaten in das erstellte Datenmodell ermöglicht. Die Importfunktion beziehungsweise die Anpassung der Rohdaten an das neue Datenmodell ist so konzipiert, dass die Anwendung für ungeschulte Anwender mit geringem Einarbeitungsaufwand genutzt werden kann.

1.3. Aufbau der Arbeit

Zunächst werden im einführenden Kapitel 2 die für diese Arbeit grundlegenden Begriffe und Techniken erläutert, sodass ein einheitliches Begriffsverständnis als Grundlage vorausgesetzt werden kann. Die Quelldaten werden in Kapitel 3 analysiert und erklärt. In Kapitel 4 wird das entwickelte Datenmodell vorgestellt und es werden mögliche Erweiterungen diskutiert. Anschließend wird in Kapitel 5 die Architektur und Funktionsweise der prototypisch entwickelten Anwendung genauer betrachtet. Abschließend folgt in Kapitel 6 eine Zusammenfassung über die gewonnenen Erkenntnisse der Arbeit und es wird ein Ausblick über mögliche Erweiterungen und Verbesserungen des Datenmodells und der Anwendung gegeben.

2. Grundlagen

In diesem Kapitel werden die für den weiteren Verlauf der Arbeit notwendigen Grundlagen eingeführt. In Teil 2.1 wird der Begriff der dimensionalen Modellierung erläutert. Nachfolgend werden die Kernkonzepte für den Umgang mit leeren Feldern, die degenerierten- und Role-Playing-Dimensionen, das Prinzip der Stellvertreterschlüssel und das Konzept der Additivität in Faktentabellen vorgestellt. Diese werden für die Modellierung, des in Kapitel 4 vorgestellten Datenmodells, benötigt.

Anschließend folgt in 2.2 eine kurze Einführung in den ETL-Prozess.

In Abschnitt 2.3 des Grundlagenkapitels werden die grundlegenden Prozesse eines Lagersystems vorgestellt. Abschließend folgen in 2.4 Erläuterungen zu den Grundfunktionen von Warehouse Managementsystemen.

2.1. Data-Warehouse

In einem operationalen Datenbanksystem werden Daten erfasst und abgefragt. Typischerweise werden viele kurze Schreib- und Lesetransaktionen durchgeführt. Die Struktur der Anfragen ist dabei simpel und das Datenvolumen einer Anfrage ist klein.

Ein Data-Warehouse ist hingegen eine in sich abgeschlossene Datenbank, welche nur zu Analysezwecken genutzt wird. In einem Data-Warehouse werden Daten nur einmal geladen. Die Daten können dabei aus verschiedenen operationalen Systemen stammen.

Im Folgenden werden in Abschnitt 2.1.1 die Ziele eines Data-Warehouse genauer erläutert. Anschließend wird in Abschnitt 2.1.2 das Konzept der dimensionalen Modellierung vorgestellt.

2.1.1. Ziele eines Data-Warehouse

Ein Data-Warehouse muss Informationen zugänglich machen. Das heißt, es muss verständlich und schnell sein. Die Bezeichner der Daten sind umgangssprachlich und nicht kryptisch. Beispielsweise sollte ein männlich / weiblich Indikator nicht '1' und '0', sondern 'männlich' und 'weiblich' lauten.

Die Konsistenz der Daten in einem Data-Warehouse muss zu jeder Zeit gewährleistet sein. Zum einen ist mit der Konsistenz die Bezeichnung der Daten gemeint. Wenn mehrere Inhalte die gleiche Bezeichnung haben, so muss ihr Inhalt identisch sein. Zum anderen

muss beim Import darauf geachtet werden, dass die Ursprungsdaten konsistent sind. Wenn nicht, müssen sie korrigiert oder gelöscht werden.

Das System muss auf Änderungen reagieren können. Anforderungen eines Nutzers, Technologien und die Daten des operativen Systems ändern sich im Laufe der Zeit. Ein Data-Warehouse System muss sich diesen Änderungen anpassen können, ohne bereits enthaltene Daten zu invalidieren und somit unbenutzbar zu machen.

Die Informationen, welche ein Data-Warehouse System darstellt, müssen zuverlässig und verbindlich sein, da auf Basis dieser Informationen teilweise kritische Entscheidungen gefällt werden. [KR13, vgl S. 3-5]

2.1.2. Dimensionale Modellierung

Mehrdimensionale Datenmodelle sind konzeptioniert, um eine effiziente Datenauswertung zu ermöglichen. Ein mehrdimensionales Datenmodell ermöglicht eine hohe Abfragegeschwindigkeit und ist - im Gegensatz zu einem stark normalisierten Datenmodell - verständlicher. [KR13, S. 7] Dadurch sind zwei Ziele eines Data-Warehouse erfüllt. Obwohl dimensionale Modelle in einem relationalen Datenbanksystem umgesetzt werden, unterscheiden sie sich von einem normalisierten Datenbankmodell.

Normalisierte Modelle teilen Daten in Tabellen ein, um Datenredundanzen und damit Anomalien oder Inkonsistenzen zu vermeiden. Dies ist besonders im operationalen Betrieb einer Datenbank notwendig, da eine 'update'-oder 'insert'-Transaktion die Datenbank an einer Stelle verändert. Ein dimensionales Modell enthält die gleichen Informationen wie ein normalisiertes Modell, diese sind lediglich in einer anderen Art und Weise abgespeichert.[KR13, S. 8]

Sternschema

In einem mehrdimensionalen Modell wird zwischen zwei verschiedenen Arten von Tabellen unterschieden. Zum einen existieren Faktentabellen und zum anderen Dimensionstabellen. Die Bezeichnung des Schemas leitet sich aus der sternförmigen Anordnung der Tabellen ab. Ein Sternschema ist in Abbildung 2.1 dargestellt.

Eine generelle Regel bei der dimensionalen Modellierung ist es, pro Prozess genau eine Faktentabelle anzulegen. So kann jeder Prozess einzeln analysiert werden. [Ada10, S. 61]

Adamson schlägt für eine gegebene Menge an Fakten zwei Fragen vor, anhand derer entschieden werden kann, ob es sich um separate Prozesse handelt. Jeder Fakt hat einen bestimmen Detaillierungsgrad. Dieser wird im nachfolgenden als Granularität bezeichnet.

1. Passieren die Fakten zu gleicher Zeit?
2. Besitzen die Fakten die gleiche Granularität?

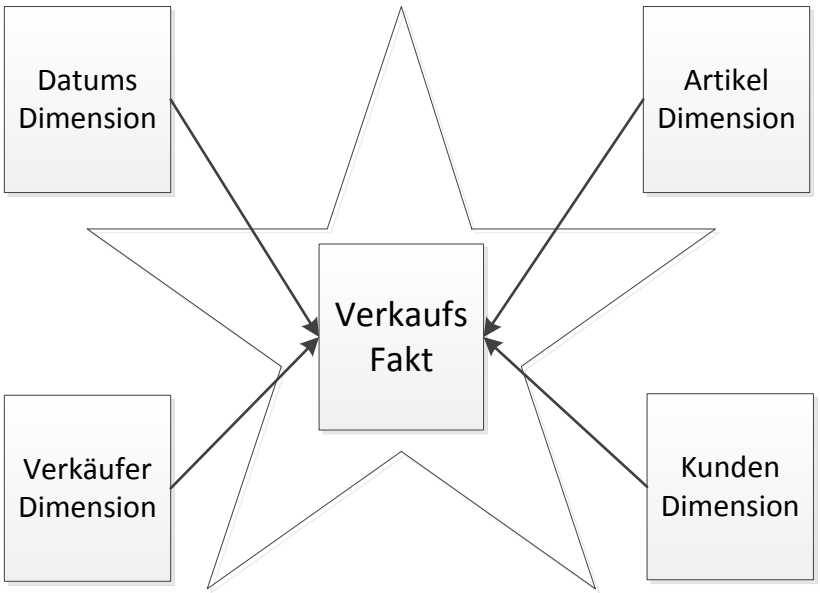

Abbildung 2.1.: Beispiel für ein Sternschema

Wenn eine der beiden Fragen mit "nein" beantwortet werden kann, so handelt es sich um zwei verschiedene Prozesse. [Ada10, S. 62]

Prozesse in einem Unternehmen lassen sich in mehrere Unterprozesse teilen. Adamson veranschaulicht dies am Beispiel von Verkäufen. Der Verkaufsprozess kann in mehrere Subprozesse aufgeteilt werden: Bestelleingang, Versand, Rechnungserstellung und Retoure. Wenn eine Bestellung eingeht, müssen Informationen zum Versand erstellt werden. Bestellmengen und Versandmenge sind außerdem in ihrer Granularität verschieden. Die Versandmenge ist durch verschiedene Versandbedingungen begrenzt, hingegen eine Bestellmenge nicht.

In diesem Fall muss auf beide oben genannten Fragen mit "nein" geantwortet werden. Die Bestellung und der Versand sind unterschiedliche Prozesse. [Ada10, S. 62]

Faktentabellen

Eine Faktentabelle ist eine Datenbanktabelle. In Faktentabellen werden die Werte abgespeichert, welche das Geschäftsfeld oder den Prozess repräsentieren. Typischerweise bilden Fakten Ereignisse ab, wie sie in der realen Welt vorkommen. Eine einfache Faktentabelle ist der Prozess an einer Kasse im Supermarkt. Für jeden Artikel werden die Menge, der Wert und eine Transaktionsnummer festgehalten. Dabei entspricht eine Zeile in der Faktentabelle einem an der Kasse gescannten Artikel. Hierbei ist wichtig, welche Granularität gewählt wird. Um größtmögliche Flexibilität bei der Analyse zu gewährleisten, wird eine möglichst kleine Granularität gewählt. Datensätze können problemlos in einer Abfrage zu-

sammengefasst werden. Bereits zusammengefasste Datensätze können unmöglich in ihre ursprüngliche Granularität zurückgeführt werden. Jede Zeile eines Fakts muss die gleiche Granularität haben. Am Beispiel der Kasse heißt dies, dass immer pro Kunde und pro Artikel eine Zeile eingefügt wird. [KR13, S. 75]

Faktentabellen haben typischerweise eine große Anzahl an Zeilen, aber eine geringe Anzahl an Spalten. Kimball spricht von mehreren Milliarden Zeilen pro Faktentabelle. [KR13, S. 31] [KR13, S. 13]

Dimensionstabellen

Dimensionstabellen enthalten den Kontext für die Faktentabellen. Sie beschreiben das „wer, was, wo, wann, wie und wieso“ eines Ereignisses in der Faktentabelle. Im Gegensatz zu den Faktentabellen haben Dimensionen weniger Zeilen, dafür werden die Tabellen durch beschreibende Werte in Textform ”breiter”. Kimball spricht von 50 bis 100 Spalten pro Dimensionstabelle. [KR13, S. 13] Jede Dimension wird durch ihren einzigartigen Primärschlüssel identifiziert, welcher die referentielle Integrität mit einer Faktentabelle sichert. [KR13, vgl S. 13]

Dimensionen sind in Abfragen die primäre Quelle von Einschränkungen oder Gruppierungen. [KR13, vgl S. 13] Wenn ein Benutzer abfragen will, wie viele Paletten eines bestimmten Artikels eingelagert sind, so benutzt er die Dimension Artikel und den Fakt der Einlagerung. Die Attribute in den Dimensionstabellen tragen zur Verständlichkeit von Abfragen bei. Eine Beispielabfrage ist in Listing 2.1 dargestellt.

Listing 2.1: Zwei Beispiele für eine Abfrage in SQL

```
SELECT *
FROM einlagerungs_fakt, artikel
WHERE einlagerungs_fakt.key = artikel.key
AND artikel.name = "PrSW14"

SELECT *
FROM einlagerungs_fakt, artikel
WHERE einlagerungs_fakt.key = artikel.key
AND artikel.name = "Printer S/W Model 2014"
```

In der zweiten Abfrage (siehe Listing 2.1) wird keine Abkürzung benutzt. Aufgrund der Verwendung des vollen Produktnamens ist die zweite Abfrage verständlicher als die erste.

Abkürzungen werden in Dimensionen separat abgespeichert, da es vorkommen kann, dass ein Nutzer nur die Abkürzung aus dem operationalen Betrieb eines Systems kennt.

Hierarchien von mehreren Tabellen, wie die Artikel - Artikelgruppe Hierarchie, werden in einer Dimensionstabelle abgespeichert. [KR13, vgl S. 84] In einem normalisierten Modell

werden die Artikelgruppen und Artikel in separaten Tabellen abgespeichert. Aufgrund der Einfachheit und der Abfragegeschwindigkeit wird in einem dimensionalen Modell darauf verzichtet.

Kernkonzepte

Leere Felder Leere Felder, sind Felder, denen ein Wert fehlt. Diese werden als *null*-Felder bezeichnet. In einem dimensionalen Modell werden *null*-Felder vermieden. Sie verursachen unzählige Komplikationen und Verwirrungen. Diese Komplikationen werden in den nachfolgenden Abschnitten genauer dargestellt.

Obwohl *null* kein Teil des Konzept des relationalen Modells ist, unterstützt so gut wie jede relationale Datenbank *null*-Felder. [Ada10, vgl S. 132] In den nächsten beiden Abschnitten werden die Auswirkungen von *null*-Felder in Dimensionen und Faktentabellen erläutert.

Null in Dimensionen Adamson erläutert das Problem an einem Beispiel [Ada10, vgl. S. 133 - 134]:

Angenommen in der Kunden-Tabelle eines Unternehmens existiert eine Spalte, welche angibt, ob ein Kunde von Steuern befreit ist oder nicht. Wenn die oben genannte Regel umgesetzt wird, dass Indikatoren ausgeschrieben werden, lautet der Inhalt der Spalte "steuerpflichtig" oder "steuerfrei". Weiter wird angenommen, dass aus irgendeinem nicht näher spezifizierbaren Grund die Spalte für einen Kunden nicht ausgefüllt ist. Also enthält sie einen *null*-Wert. Wenn ein Bericht erstellt werden soll, der alle Kunden aufzählt, die keine Steuerbefreiung haben, wird der "not equal"-Operator benutzt (siehe Listing 2.2).

Listing 2.2: Kunden ohne Steuerbefreiung "not equal"-Operator

```
WHERE steuer_status <> "steuerfrei"
```

Der Kunde, welcher einen *null*-Wert in der steuer_status Spalte hat, wird nicht mit ausgegeben. Sein Status ist *null*. Dies ist nicht mit einer Steuerbefreiung und nicht mit einer Steuerpflicht gleichzusetzen. Der "korrekte" Weg in einer SQL Abfrage *null*-Werte abzufragen ist mit "IS NULL" oder "IS NOT NULL". Eine Abfrage, um alle Kunden zu finden, welche nicht steuerbefreit sind, ist in Listing 2.3 dargestellt.

Listing 2.3: Kunden ohne Steuerbefreiung "not equal"-Operator und "IS NULL"

```
WHERE steuer_status <> "steuerfrei"
OR steuer_status IS NULL
```

Obwohl die Abfrage in diesem Fall das richtige Ergebnis liefert, basiert sie auf einer Annahme. Um diese Unsicherheiten zu vermeiden, muss ein Platzhaltewert anstatt eines *null*-Werts verwendet werden. [Ada10, vgl. S. 133 - 134]

Null in Faktentabellen In einer Faktentabelle muss zwischen zwei verschiedenen Arten von *null*-Werten unterschieden werden. Als Fremdschlüssel in einer Faktentabelle verletzt ein *null*-Wert die referentielle Integrität. Da eine "join"-Operation nicht auf einem *null*-Wert ausgeführt werden kann, ist ein Arbeiten mit diesem Datenschema nicht möglich. Jede Dimension muss einen Platzhaltedatensatz anbieten, der im Fall eines *null*-Werts referenziert werden kann. [KR13, S. 92]

Wenn metrische Fakten *null*-Werte enthalten, können diese im Normalfall unbehandelt bleiben, da alle Aggregationsfunktionen, wie SUM, MIN, MAX, COUNT und AVG mit *null*-Werten korrekt umgehen. Wenn stattdessen ein Platzhaltewert wie 0 verwendet wird, berechnen diese Funktionen den falschen Wert. [KR13, vgl. S. 92]

Stellvertreterschlüssel Der Primärschlüssel einer Dimensionstabelle darf nicht der ursprüngliche Primärschlüssel des operationalen Systems sein. Stattdessen wird ein Stellvertreterschlüssel benutzt.

Stellvertreterschlüssel sind einfache, inhaltslose Zahlen-Werte. Die erste Zeile einer Dimension bekommt den Schlüssel mit dem Wert "1", in der zweiten Zeile hat der Schlüssel den Wert "2". Ein Vorteil ist, dass Änderungen am operationalen System nicht das Data-Warehouse betreffen. Weiter ist es so möglich, mehrere operationale Systeme in ein einzelnes Data-Warehouse zu integrieren. [KR13, vgl. S. 98f.]

Degenerierte Dimension Es ist nicht sinnvoll, alle Dimensionen eines Geschäftsprozesses in Dimensionstabellen auszulagern. Eine oder mehrere Dimensionen können direkt in der Faktentabelle abgespeichert werden. Spalten, in denen Dimensionen abgespeichert sind, heißen degenerierte Dimensionen. Beispielsweise wird bei einer Bestellung pro Bestellposition eine Zeile in der Faktentabelle angelegt. Um den Bezug zur Bestellung nicht zu verlieren, wird die Bestellung nicht als eigene Dimension abgespeichert, sondern der Bestellschlüssel als degenerierte Dimension in der Faktentabelle. [Ada10, vgl. S.43]

Role-Playing-Dimension Eine einzelne physische Dimension kann in einer Faktentabelle mehrmals referenziert werden. Die Faktentabelle einer aufgegebenen Bestellung kann beispielsweise das Bestelldatum und ein geplantes Versanddatum enthalten. Jedes dieser Daten ist ein Fremdschlüssel in der Faktentabelle. Eine "join"-Operation in SQL setzt voraus, dass beide Daten identisch sind, was selten der Fall ist. Um dieses Problem zu umgehen, werden in SQL aus der Dimension, die zwei Mal referenziert werden soll, zwei View-Tabellen erstellt, welche einen "join"-Operation problemlos zulassen. [KR13, vgl. S. 170-171]

Additivität in Faktentabellen Fakten wie Menge oder Preis sind die nützlichsten Fakten für Analysen, da sie numerisch und additiv sind. Additivität ist die Eigenschaft eines

Fakts. Analysen werden nicht auf einzelnen Zeilen einer Faktentabelle ausgeführt, sondern vielmehr auf mehreren hundert, wenn nicht tausenden. Daher ist es von Vorteil, wenn sich Fakten einer Zeile addieren lassen. Additive Fakten lassen sich über alle Dimensionen addieren. [KR13, vgl. S. 11]

Fakten wie die Gesamtmenge eines Artikels im Lager oder ein Kontostand, heißen semi-additiv, da sie nicht über alle Dimensionen additiv sind. Es macht keinen Sinn, Momentaufnahmen, wie die Bestandsmenge, über eine zeitliche Dimension zu addieren. Der einzig sinnvolle Weg diese Fakten miteinander zu vergleichen, ist die Bildung eines Durchschnitts über mehrere Zeitperioden. Semiadditive Fakten lassen sich nicht über alle Dimensionen addieren. [KR13, vgl. S. 114]

Nicht-additive Fakten lassen sich über keine Dimension addieren. Ein Beispiel für einen solchen Fakt ist der Preis eines Artikels. Eine Addition ergibt in keiner Dimension einen Sinn. Es ist sinnvoller, nicht-additive Fakten in einer Dimension, anstatt einer Faktentabelle zu speichern, da sie kaum Analysen zulassen. [KR13, vgl. S. 78]

2.2. ETL-Prozess

Der ETL-Prozess übernimmt alle Aufgaben, die zwischen dem operationalen System und dem Data-Warehouse ablaufen. ETL steht für *Extract*, *Transform* und *Load*. In Abbildung 2.2 ist der Datenfluss eines ETL-Systems schematisch dargestellt.

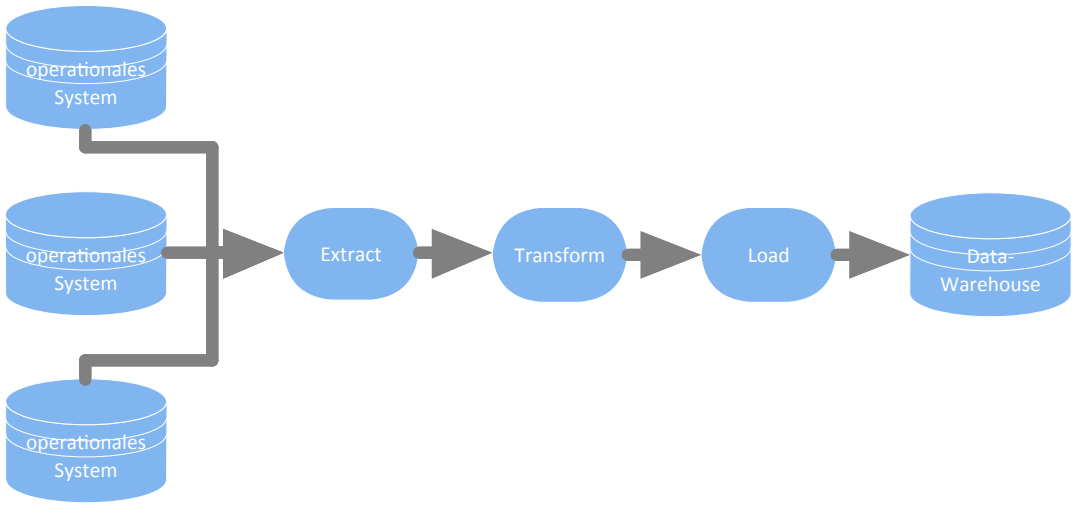

Abbildung 2.2.: Schematischer Datenfluß eines ETL-Systems

Der erste Schritt ist die Extraktion der Daten aus dem operativen System. Extraktion bedeutet auch ein Verständnis für die Quelldaten zu entwickeln. Sobald die Daten aus dem operationalen System gelesen und kopiert sind, werden sie nur noch vom ETL-Prozess verwaltet und sind nicht mehr an das operationale System gekoppelt. Dann können die Daten ohne Einschränkungen manipuliert und verändert werden. [KR13, vgl. S. 20]

Nachdem die Daten extrahiert sind, können sie im Transformationsschritt verändert werden. Zu diesen Veränderungen gehört eine Bereinigung der Daten. Die Bereinigung beinhaltet die Korrektur von Schreibfehlern, den Umgang mit Duplikaten, die Auflösung von Konflikten, den Umgang mit fehlenden und leeren Feldern und das Parsen in Standardformate. Parsen bedeutet in diesem Fall, dass zum Beispiel verschiedene Datumsformate in ein Standardformat gebracht werden. Zu den weiteren Aufgaben in diesem Schritt gehört das Kombinieren von Daten, wenn diese aus unterschiedlichen Quellsystemen stammen. [JPT10, S. 68]

Nachdem die Daten transformiert sind, folgt als letzter Schritt das physische Laden in das dimensionale Modell des Data-Warehouse Systems. Beim Laden werden die Daten auf das dimensionale Modell abgebildet und die in Abschnitt 2.1.2 genannten Konzepte umgesetzt. [JPT10, S. 69]

2.3. Prozesse in einem Lagersystem

Im Folgenden werden die grundlegenden Abläufe eines Lagersystems dargestellt. Diese sind in Abbildung 2.3 schematisch dargestellt. Jeder dieser fünf Prozesse wird in einem der folgenden Abschnitte getrennt betrachtet.

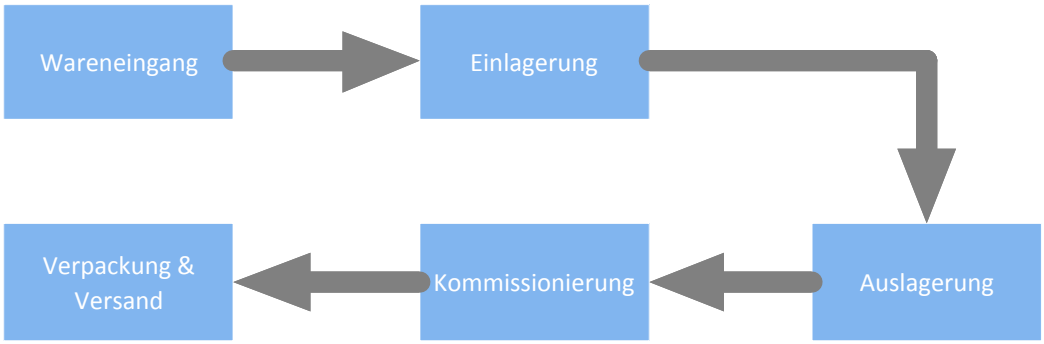

Abbildung 2.3.: Prozesse in einem Lagersystem

2.3.1. Wareneingang

Der Materialfluss innerhalb eines Unternehmens wird als interner Materialfluss bezeichnet und beginnt mit der Warenannahme. Dabei wird im ersten Schritt ein eventuell vorhandenes Lieferavis mit dem Frachtbrief des Spediteurs abgeglichen. Anschließend wird die Lieferung unter Vorbehalt in das System eingebucht. Wenn eine Lieferung avisiert wird, können gleichzeitig Pufferstellen im Wareneingangsbereich reserviert, eine Annahmestelle ausgewählt und eventuelle Etiketten für den internen Gebrauch gedruckt werden. Nach dem Transport des Guts in den Wareneingangsbereich, findet eine Wareneingangsprüfung statt. Die Lieferung wird in diesem Schritt auf Art und Menge kontrolliert. Meist findet eine Qualitätssicherung statt und die Lieferung wird auf Beschaffenheit überprüft. Eventuelle Fehler werden im System vermerkt. Falls Artikel angeliefert werden, dessen Stammdaten noch nicht im System sind, werden diese aufgenommen. Dazu müssen sie gewogen und vermessen werden, um die Artikelstammdaten vollständig im System hinterlegen zu können. Falls hochwertige Güter angeliefert werden, müssen die Seriennummern erfasst werden, um einzelne Artikel lückenlos verfolgen zu können. Nachdem alle Artikel kontrolliert sind, folgt eine eventuelle Bildung von Lagereinheiten. Da eingehende Güter mit dem Ziel der Versand- und Transportkostenminimierung verpackt wurden oder eigene Ladehilfsmittel und Behälter verwendet werden, ist eine Anpassung notwendig. Wenn die Qualität der angelieferten Ladehilfsmittel niedrig ist, wird die Ware auf firmeneigene, hochwertige Ladehilfsmittel umgesetzt. Wenn die Menge auf einer Palette zu klein wird, werden Mischpaletten gebildet. Die Mischpalettenbildung wird im System erfasst, da es sonst bei folgenden Prozessen zu Fehlern kommen kann. [HS10, vgl. S.23-28]

2.3.2. Einlagerung

Bei der Einlagerung wird zwischen einem manuellem System und einem automatischen System unterschieden. Bei einem manuellen System kann die Einlagerung als ein durchgängiger Prozess dargestellt werden. Die Lagereinheit wird aufgenommen, zum finalen Lagerplatz transportiert und dort abgegeben. Bei automatisierten Systemen wird der Einlagerungsprozess in mehrere Prozesse unterteilt. Nachfolgend werden die einzelnen Prozesse eines automatisierten Systems erklärt. Diese lassen sich jedoch auf ein manuelles System anwenden. [HS10, vgl. S.28]

Der erste Schritt ist die Überprüfung, ob der aktuelle Wareneingang zur Vervollständigung von aktuellen Aufträgen benötigt wird. Wenn dies der Fall ist, folgt der direkte Transport in den Versand, die Kommissionierung oder den Warenausgang. Dieser Vorgang wird als Durchlagerung bezeichnet. Falls das Gut nicht zur umgehenden Vervollständigung von bestehenden Aufträgen benötigt wird, folgt der Transport in den Lagerbereich. Um eine lückenlose Überwachung des Materialflusses zu erreichen, wird das Fördermittel,

auf welchem das Gut transportiert wird, vom System als virtueller Lagerplatz betrachtet. [HS10, vgl. S.29]

Vor dem Transport wird das Ziel im Lager festgelegt. Die Vergabe des Lagerplatzes erfolgt nach unterschiedlichen Parametern.

Erstens werden technische Anforderungen berücksichtigt. Darunter fallen die Beachtung der maximal zulässigen Fach- und Feldlasten, die optimale Nutzung des Fachvolumens und die gleichmäßige Belastung der Regale. Zusätzlich soll eine einseitige Belastung von Regalen verhindert werden.

Zweitens werden bei der Lagerplatzvergabe Aspekte der betrieblichen Optimierung berücksichtigt. Zu diesen Aspekten gehören die Minimierung von Wegstrecken im Lagersystem, die Maximierung der Umschlagsleistung und die maximale Nutzung von Kapazitäten.

Außerdem werden bei der Lagerplatzvergabe die sicherheitstechnischen und rechtlichen Vorgaben berücksichtigt. Diese betreffen die Gruppierung von Chargen, die Getrenntlagerung im Lebensmittelbereich und das Zusammenlagerungsverbot von Gefahrgut.

Nachdem ein passender Lagerplatz gefunden ist, wird das Gut zu diesem transportiert. Um die Einlagerung abzuschließen, werden der Ort (Lagerplatz) und Zeitpunkt an das System zurückgemeldet. [HS10, vgl. S.30-31]

2.3.3. Auslagerung

Bevor die eigentliche Auslagerung beginnt, werden vorliegende Aufträge auf Erfüllbarkeit überprüft. Wenn die geforderte Menge vorhanden ist, werden die jeweiligen Ladeeinheiten und Mengen im System reserviert. Dadurch können Fehlmengen vermieden werden. Das Lagerverwaltungssystem überwacht den Auslagervorgang. Sobald dieser beendet ist, muss eine Rückmeldung an das System erfolgen. Wenn diese durchgeführt ist, wird der Lagerplatz freigegeben, eventuelle Reservierungen werden gelöscht und der Lagerbestand wird um die ausgelagerte Menge reduziert. In diesem Prozess wird aufgrund der Nachvollziehbarkeit des Warenflusses die Lagereinheit auf ein entsprechendes Fördermittel gebucht. Dies repräsentiert den temporären Lagerort. [HS10, vgl. S.32-33]

In diversen Materialflusssystemen folgt der sogenannte Konsolidierungspunkt. Dieser ist eine Schlüsselposition im Materialfluss, an dem die Daten von abgefertigten Einheiten auf Soll- und Ist-Zustand hin überprüft werden. Weiterhin werden der Material- und Informationsfluss verglichen, und gegebenenfalls angepasst. [HS10, vgl. S.34] Eine Anpassung ist beispielsweise eine Änderung der im System abgespeicherten Menge eines Guts.

2.3.4. Kommissionierung

Die Kommissionierung bezeichnet den Vorgang der Zusammenstellung von einem oder mehreren Gütern in eine kundengerechte Bedarfsmenge. Im Zuge dieser, werden Teilmengen größerer Einheiten entnommen, zusammengeführt und für den Versand bereitgestellt. Ein

angewendetes Prinzip in der Kommissionierung ist das "Person-zur-Ware-Prinzip". [HS10, vgl. S.47] Dabei bewegt sich der Kommissionierer zur Ware und greift sich die passenden Einheiten. Die Informationen, welche Einheiten zu entnehmen sind, erhält er dabei einer sogenannten Pickliste.

Aufgrund der thematischen Ausrichtung dieser Arbeit kann auf den sehr umfangreichen Bereich der Kommissionierung im Folgenden nicht detaillierter eingegangen werden. Für weitergehende Informationen zum Thema Kommissionierung sei auf das Buch "Kommissionierung - Materialflusssysteme 2: Planung und Berechnung der Kommissionierung in der Logistik" [HSB11] verwiesen.

2.3.5. Verpackung und Versand

Nachdem der Prozess der Kommissionierung abgeschlossen ist, werden die kommissionierten Güter auf Vollständigkeit geprüft. In diesem Schritt finden gleichzeitig eine finale Qualitätskontrolle und eine Überprüfung des Zustands der Transporteinheiten statt. Anschließend werden die Güter verpackt. Aus informationstechnischer Sicht erfolgt hierbei eine Aktualisierung des Auftragsstatus und eine Ergänzung versandtechnisch relevanter Daten. Dann werden die verpackten Güter an den Versandtoren zur Verladung bereitgestellt. Beim abschließenden Verladen der Güter erfolgt ein letzter Scan der Einheiten, um eine vollständige Transparenz des Warenflusses im System zu erhalten. [HS10, vgl. S.51-53]

2.4. Warehouse Managementsystem

Die Aufgabe eines Warehouse Managementsystems ist die Führung und Optimierung von innerbetrieblichen Lagersystemen. [HS10, vgl. S.54] Dazu gehört die Lagerverwaltung, die Reorganisation des Lagers, die Fördermittelverwaltung, Datenerfassung und die Verwaltung des Inventurprozesses. In Abschnitt 2.4.1 wird der Aufgabenbereich der Lagerverwaltung beschrieben. Anschließend wird in 2.4.2 das Reorganisationsmodul eines Warehouse Managementsystems genauer erklärt. Der Bereich der Fördermittelverwaltung wird in Abschnitt 2.4.3 erläutert. Abschließend wird in 2.4.5 der Vorgang der Inventur beschrieben.

2.4.1. Lagerverwaltung

Ein Warehouse Managementsystem verwaltet die Lagerorte in einem Lagersystem. Dazu gehören ortsspezifische Daten, wie Fachgröße und der Ort im Lager sowie warenspezifische Daten, wie Artikelnummern oder Artikelnamen. Weiter kann das System in der Lagerverwaltung Bestände reservieren und sperren. Ein weiterer Aufgabenbereich der Lagerverwaltung ist die Bestandsführung. In der Bestandsführung werden Artikelmengen verwaltet. Die Lagerverwaltung berücksichtigt gesperrte und reservierte Bestände. Dadurch werden Unter- und Übermengen vermieden. Wenn Bestände im Lager eine maximale Lagerdauer

oder ein Verfallsdatum haben, kann das Warehouse Managementsystem über die Lagerverwaltung diese sperren oder gegebenenfalls den Auslagervorgang initiieren.

Wenn eine Überwachung der Lagerbedingungen auf Temperatur und Luftfeuchtigkeit notwendig ist, wird diese von der Lagerverwaltung übernommen. Die Kontrolle der Sicherheit des Lagers ist in Form von Zugangskontrollen eine weitere Aufgabe der Lagerverwaltung. [HS10, vgl. S.54-56]

2.4.2. Reorganisation

Der zweite Aufgabenbereich eines Warehouse Management Systems ist die Reorganisation des Lagers. Ziel ist es dabei, die Lagereffizienz regelmäßig zu überprüfen und gegebenenfalls zu optimieren. [HS10, vgl. S.58] Mit der Optimierung sind in diesem Fall drei konkrete Aufgaben verbunden.

Die Umbuchung und Zuweisung der Artikel zu geeigneten Zugriffsklassen und Lagerzonen. Eine Umlagerung von Lagereinheiten zu Zeiten des schwachen Zugriffs. Und die Verdichtung von angebrochenen Einheiten, wie zum Beispiel Mischpaletten.

2.4.3. Fördermittelverwaltung

Wenn in einem Lager manuell geführte Unstetigförderer, zum Beispiel Stapler, eingesetzt werden, können diese entweder manuell verwaltet werden, oder vom System automatisch geführt werden. Ein Grund für ein rechnergestütztes Leitsystem ist die Optimierung der Systemleistung, zum Beispiel durch Reduzierung von Leerfahrten. Außerdem werden die Fördermittel von einem rechnergestützten Leitsystem überwacht. [HS10, vgl. S.58-60]

2.4.4. Datenerfassung

Ein Lagersystem zeichnet sich durch eine Vielzahl an generierten Daten und Stammdaten aus. Durch diese lassen sich für die Planung relevante Kennzahlen bilden. Es existieren zwei Verfahren zur Ermittlung von Aussagekräftigen Daten. [HS07, vgl. S.60]

1. Online Erfassung
   Bei der Online Erfassung werden die Daten im Prozess erfasst und unmittelbar in entsprechende Kennzahlen umgesetzt. Welche Daten wie umgesetzt werden, muss im Vorfeld festgelegt werden. Der Vorteil an dieser Methode ist, dass Kennzahlen unmittelbar zur Verfügung stehen. Nachteil ist, dass eine nachträgliche Auswertung von zum Beispiel verwandten Daten nicht mehr möglich ist, da bei der Umsetzung nur die Kennzahl abgespeichert wird. Zusätzliche Analysen müssen erst eingestellt werden und können nicht auf Daten in der Vergangenheit angewendet werden. [HS07, vgl. S.61]

2. Zeitreihenerfassung
   Die Zeitreihenerfassung verwendet eine Logdatei, welche vom System erstellt wird. In die Logdatei werden mit einem Zeitstempel versehene Ereignisse geschrieben. Nach einer bestimmten Zeit wird diese Datei in ein Datenbanksystem geladen. Dann können über entsprechende Abfragen die gewünschten Kennzahlen gebildet werden. Ein Vorteil gegenüber der Online Erfassung ist, dass durch die Abspeicherung aller Daten genauere Analysen ausgeführt werden können. Ein Nachteil ist allerdings, dass die Daten nicht sofort zur Verfügung stehen. [HS07, vgl. S.61]

2.4.5. Inventur

Jeder Kaufmann ist nach §240 HGB gesetzlich dazu verpflichtet, jährlich eine Inventur durchzuführen. Die Art der Inventur kann durch verschiedene Voraussetzungen beeinflusst werden. Diese Voraussetzungen sind der Einsatz einer EDV gestützten Bestandsführung, die Zugänglichkeit der Lagerfläche und der Wert der Güter. So ist zum Beispiel bei einem hohen Wert der Güter kein Stichprobenverfahren mehr möglich. [HS10, vgl. S.62]

Es gibt 3 Arten von Inventuren:

1. Stichtagsinventur
   Die Stichtagsinventur ist die klassische Zählung der Bestände im Lager an einem Stichtag. Diese ist die aufwändigste Art der Inventur. [HS10, vgl. S.62]

2. Permanentinventur
   Bei der Permanentinventur wird der Bestand bei jedem Zugriff gezählt. Also bei jeder Einlagerung und bei jeder Auslagerung. So ist der Bestand dauerhaft mengenmäßig erfasst. [HS10, vgl. S.63]

3. Stichprobeninventur
   Die Stichprobeninventur zeichnet sich durch die stichprobenartige Erfassung des Bestands aus. Diese Stichproben werden unter Zuhilfenahme von mathematisch-statistischen Verfahren ausgewertet. [HS10, vgl. S.63-64]

3. Analyse

Im Hinblick auf die Erstellung des Datenmodells zur effektiven Bestimmung von logistisch relevanten Kennzahlen wird in Abschnitt 3.1 der Kennzahlenbegriff erklärt und es werden Beispiele für Kennzahlen genannt. Anschließen wird in Kapitel 3.2 das Datenmodell des Warehouse-Managementsystems myWMS [Fra14a] analysiert.

3.1. Basisdaten & Kennzahlen

In Abschnitt 3.1.1 wird der Begriff der Basisdaten vorgestellt. Anschließend werden in 3.1.2 relevante Logistikkennzahlen beschrieben.

3.1.1. Basisdaten

Basisdaten sind Absolutzahlen, welche durch Messung, Zählung und Bildung von Summen und Differenzen abgelesen beziehungsweise gebildet werden.

Zu den Basisdaten gehören die Stamm-, Bestands- und Bewegungsdaten.

Stammdaten Die wichtigsten Stammdaten im Kontext eines Lagersystems sind die Artikelstammdaten. Der Fördermittelstamm, der Ladeeinheitenstamm und der Verpackungsdatenstamm sind weitere relevante Stammdaten. [HS10, vgl S. 65] Ein Artikelstamm ist in Tabelle 3.1 dargestellt.

Bestandsdaten Die Bestandsdaten geben über die Menge aller Artikel Auskunft. Zu den Bestandsdaten gehören beispielsweise die Anzahl der Artikel im Sortiment, der momentan verfügbare Bestand, der Mindestbestand und ein Durchschnittsbestand. [HS10, vgl. S.65-66]

Bewegungsdaten Bewegungsdaten beschreiben die Warenbewegungen in den einzelnen Prozessen eines Lagersystems. Dazu gehören die Wareneingänge und die Warenausgänge pro Tag, die Ein- und Auslagerungen pro Tag und die Anzahl der Umlagerungen am Tag. Weitere Bewegungsdaten sind die Anzahl der Aufträge pro Tag und die Menge der Aufträge pro Artikel. [HS10, vgl. S.66]

Artikelstammdaten
Artikelnummer
Bezeichnung
Gewicht
Länge
Breite
Höhe
Einheit
Ladeeinheiten Art
Beladungsfaktor (Packmenge/Ladeeinheit)
Greifeinheit (Packmenge/Entnahmeeinheit)
Sperrkennzeichen
ABC Klassifikation
Chargennummer
Gewicht/Entnahmeeinheit
Gewicht/Ladeeinheit
Mandant
Verfallkennzeichen

Tabelle 3.1.: Beispiel für einen Artikelstamm.
Quelle: [HS10, vgl S. 66]

3.1.2. Kennzahlen

> "Kennzahlen werden als jene Zahlen betrachtet, die quantitativ erfaßbare Sachverhalte in konzentrierter Form erfassen." [Rei01, S.19 ff]

Nach dieser Definition von Reichmann zählen die oben genannten Basisdaten zu den Kennzahlen, da zum Beispiel die Anzahl von Wareneingängen pro Tag einen quantitativ erfassbaren Sachverhalt darstellt. Relevante Kennzahlen sind in Tabelle 3.2 abgebildet.

Zwei zentrale Kennzahlen in der Logistik sind nach Reichmann die Lagerumschlagshäufigkeit und der Lieferbereitschaftsgrad. [Rei01, vgl. S.436] Nachfolgend werden diese beiden Kennzahlen genauer beschrieben.

Lagerumschlagshäufigkeit

$$\frac{\text{Auslagerungen pro Zeiteinheit}}{\text{Ø Lagerbestand in Zeiteinheit}}$$

Die Lagerumschlagshäufigkeit zeigt an, wie oft sich in einer Zeitperiode das komplette Lager umgeschlagen hat. [Rei01, vgl. S.437-439] Eine höhere Umschlaghäufigkeit ist positiv

Kennzahl	Definition
Lieferbereitschaftsgrad	$\frac{\text{termingerechte Abgänge (Stck)}}{\text{angeforderte Abgänge (Stck)}}$
Lagerfüllungsgrad	$\frac{\text{Anzahl belegter Fächer (Stck)}}{\text{Lagerkapazität (Stck)}}$
Umschlagsgrad mengenbezogen	$\frac{\text{Auslagerungen(Stck/a)}}{\text{Lagerkapazität (Stck)}}$
Kosten/Lagerplatz	$\frac{\text{Auslagerungen(Stck/a)}}{\text{Lagerkapazität (Stck)}}$
Pickdichte	$\frac{\text{Positionenzahl(Stck)}}{\text{Zugriffsfläche } (m^2)}$

Tabelle 3.2.: Logistikkennzahlen
Quelle: vgl. [HS10, S. 68]

anzusehen, da dies eine kürzere Lagerdauer bedeutet. Aus einer kürzeren Lagerdauer folgen niedrigere Lagerhaltungskosten. Aus einer niedrigen Lagerumschlagshäufigkeit resultieren hingegen hohe Lagerhaltungskosten, da das Gut meist länger gelagert wird.

Lieferbereitschaftsgrad Der Lieferbereitschaftsgrad berechnet sich aus den termingerechten Abgängen und der Anzahl an angeforderten Abgängen [Rei01, vgl. S.430-431]:

$$\frac{\text{termingemäße Abgänge pro Zeiteinheit}}{\text{angeforderte Abgänge in Zeiteinheit}}$$

Ist der Lieferbereitschaftsgrad zu niedrig, können Folgekosten durch die entstandenen Fehlmengen entstehen. Ein hoher Lieferbereitschaftsgrad ist positiv zu bewerten, muss jedoch in Relation zu anderen Kennzahlen gesetzt werden. So kann es vorkommen, dass die Lagerumschlagshäufigkeit sehr niedrig ist, und deshalb hohe Lagerkosten entstehen. Aufgrund dessen ist zum Beispiel ein Vergleich mit der Lagerumschlagshäufigkeit sinnvoll.

Für genauere Analysen ist es erforderlich Kennzahlen zu untergliedern. Diese Untergliederung kann beispielsweise nach den einzelnen Artikeln erfolgen. Auch kann es sinnvoll sein verschiedene Perioden zu betrachten, um Schwachstellen im Lagersystem zu finden.

3.2. Quelldaten

Der Entwicklung eines mehrdimensionalen Datenmodells geht die Analyse von Quelldaten eines Warehouse Managementsystems voraus.

In diesem Abschnitt wird das Datenmodell eines Warehouse Managementsystems analysiert und genauer betrachtet. Die Quelldaten stammen aus dem myWMS-Warehouse Ma-

nagementsystem. Das myWMS-System ist ein quelloffenes Warehouse Managementsystem, welches vom Fraunhofer IML entwickelt wurde. Dieses wird benutzt, da der Quellcode frei verfügbar ist und das System installiert und gestartet werden kann. Bei der Installation wird die Datenbank automatisch erstellt und auf Wunsch mit Beispielstammdaten befüllt.

Für weitere Informationen zu myWMS siehe [Fra14a].

Die Daten aus den verschiedenen Tabellen des myWMS-Systems lassen sich in zwei Kategorien einteilen.

Zum einen gibt es Stammdatentabellen, in denen die Artikelstammdaten und die Lagerplatzstammdaten gespeichert sind. Zum anderen werden Tabellen generiert, in denen Bewegungsdaten gespeichert sind. Zu den Bewegungsdaten gehören in diesem Fall die Daten des Wareneingangs, des Warenausgangs und protokollierte Daten von Warenbewegungen.

In Abschnitt 3.2.1 werden zuerst die Stammdatentabellen und anschließend in 3.2.2 die Bewegungsdatentabellen genauer betrachtet. Dabei werden nicht alle generierten Tabellen des myWMS-Systems betrachtet, sondern nur die Tabellen, die für die Erstellung des finalen Datenmodells relevant sind.

3.2.1. Stammdaten

Die Stammdaten lassen sich wiederum in den Bereich der Artikeldatenstammdaten und in die Lagerstammdaten unterteilen.

Zuerst werden zwei Tabellen betrachtet, welche dem Bereich der Artikeldaten zugeordnet werden können. Im Folgenden werden die wichtigsten Tabellen der Lagerstammdaten untersucht.

Artikelbezogene Stammdatentabellen

Die zentrale Tabelle der Artikelstammdaten ist die Tabelle *ITEMDATA*. Im Anschluss werden ihre zentralen Attribute erläutert. Das komplette Schema der Tabelle befindet sich im Anhang (siehe A.1).

Die *ITEMDATA*-Tabelle beinhaltet die Artikelstammdaten. Dazu gehören unter anderem ein Name, diverse Abmessungen und das Volumen des Artikels, Felder für EAN-Codes, ein Indikator, ob für den Artikel eine Chargenpflicht besteht, und ein Sicherheitsbestand. Über einen Fremdschlüssel ist jedem Artikel eine Artikeleinheit zugeordnet.

Die Einheiten werden in der Tabelle *ITEMUNIT* abgespeichert. Unter anderem werden der Bezeichner und der Typ (beispielsweise Gewicht oder Stück) abgespeichert. Weiter referenziert sich die *ITEMUNIT*-Tabelle selbst, um Basiseinheiten abspeichern zu können.

Zusätzlich ist jedem Artikel ein Standard-Ladeeinheiten-Typ, wie zum Beispiel eine Palette zugeordnet.

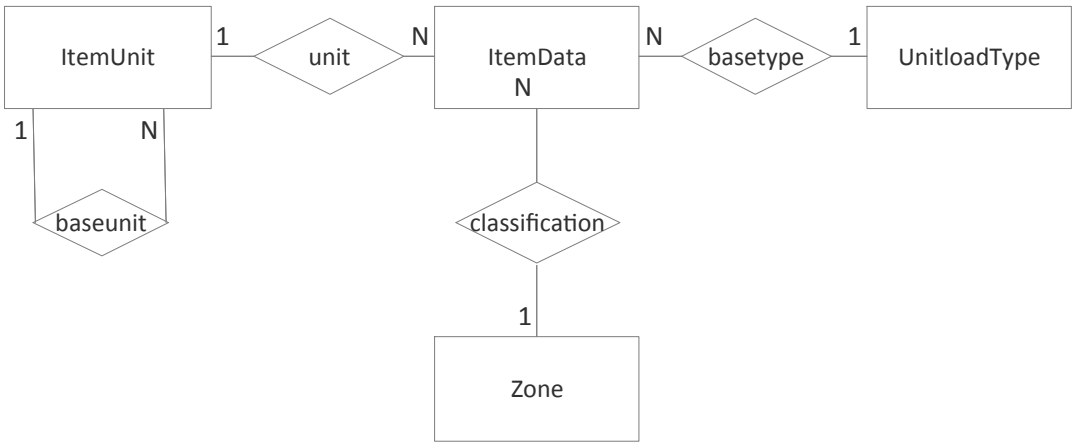

Abbildung 3.1.: ER-Diagramm - ItemData relevanter Ausschnitt

Eine Zone, welche bei einer ABC-Zonung zum Einsatz kommt, ist dem Artikel über die Tabelle *ZONE* zugeordnet. Die Tabellen *ZONE* und *UNITLOADTYPE* werden im Abschnitt der lagerbezogenen Stammdatentabellen erklärt. In Abbildung 3.1 ist der relevante Ausschnitt des ER-Diagramms dargestellt.

Lagerbezogene Stammdatentabellen

Die zentrale Tabelle des gesamten Warehouse Managementsystems ist die *STOCKUNIT*-Tabelle. In dieser werden Bestandseinheiten innerhalb des Lagersystems abgespeichert.

Eine Bestandseinheit ist im Fall vom myWMS-System eine Menge von Artikeln, welche sich im Lagersystem befindet. Das wichtigste Attribut einer Bestandseinheit ist die Menge. Weiter werden bereits vom System reservierte Mengen und eine Seriennummer abgespeichert. Um einer Bestandseinheit einen Artikel zuordnen zu können, referenziert die *STOCKUNIT*-Tabelle über einen Fremdschlüssel einen Artikel (*ITEMDATA*).

Da eine Bestandseinheit immer auf einem Ladehilfsmittel gelagert wird, ist außerdem die Tabelle *UNITLOAD* referenziert. Ein Ladehilfsmittel ist zum Beispiel eine Palette oder ein Behälter. Auf einem Ladehilfsmittel können andere Ladehilfsmittel gelagert werden, beispielsweise eine Box auf einer Palette. Aus diesem Grund referenziert sich die *UNITLOAD*-Tabelle selbst.

Sobald ein Ladehilfsmittel einer oder mehreren Bestandseinheiten zugeordnet ist, lässt sich dessen Gewicht berechnen. Diese wird zu jedem Ladehilfsmittel abgespeichert. Wenn im Lagersystem eine Waage zum Einsatz kommt, kann außerdem das gemessene Gewicht hinzugefügt werden.

Da verschiedene Ladehilfsmittel zum Einsatz kommen, hat jedes Ladehilfsmittel einen Typ. Ladehilfsmitteltypen werden in der Stammdatentabelle *UNITLOADTYPE* abgespei-

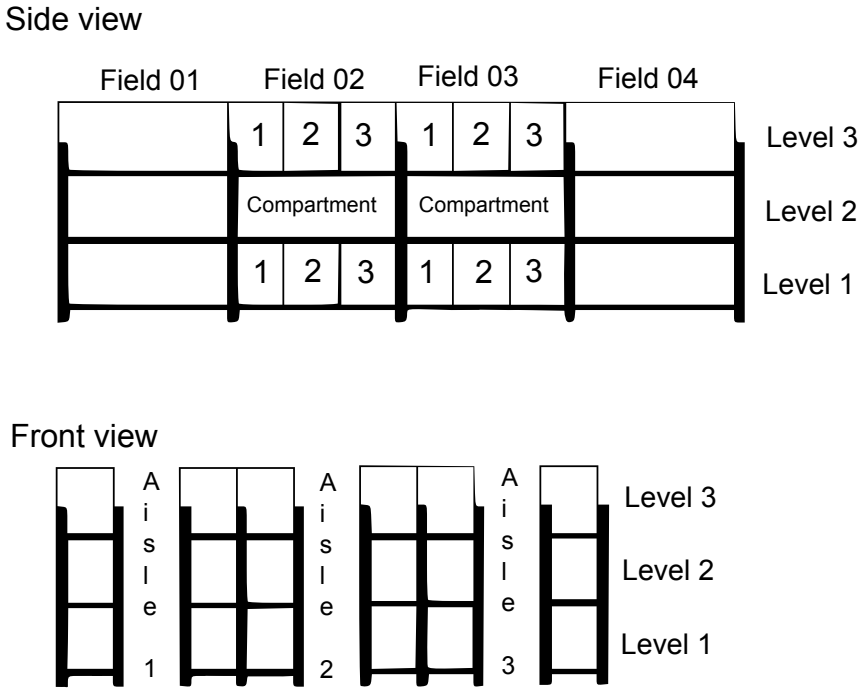

Abbildung 3.2.: Beispielhafter Aufbau eines Regals
Quelle: vgl. [Fra14b]

chert. Dort werden der Name, das Gewicht und eine maximale Tragfähigkeit abgespeichert. Außerdem werden dort alle Abmessungen (Höhe, Länge, Breite, Tiefe) gespeichert.

Jedem Ladehilfsmittel und damit jeder Bestandseinheit ist ein eindeutiger Platz im Lagersystem zugeordnet. Die Tabelle *STORAGELOCATION* beinhaltet alle vorhandenen Lagerplätze. Jedes Ladehilfsmittel referenziert einen Lagerplatz. Damit die Lagerplätze unterschieden werden können, ist zusätzlich zur ID ein menschenlesbarer Name gespeichert. Weiter werden in dieser Tabelle das Datum an dem der Lagerplatz befüllt wurde, die x-, y-, z-Position im Lager und der jeweilige Code zum Scannen abgespeichert.

Der Typ eines Lagerplatzes wird in der Tabelle *STORAGELOCATIONTYPE* gespeichert. Typen können in diesem Fall ein Palettenregal oder Behälterregal sein. Da sich die Typen nicht nur im Namen unterscheiden, hat jeder Lagerplatztyp außer dem Namen noch weitere Attribute. Es werden das maximale Gewicht, welches die Ladehilfsmittel inklusive ihrer Ladeeinheiten haben dürfen, und die Abmessungen des jeweiligen Lagerplatzes festgehalten.

Um den einzelnen Lagerplätzen Regale zuordnen zu können, haben sie eine Referenz auf die Tabelle *RACK*. Als Attribute werden die Position des Labels, welches abgescannt wird, die Anzahl der Spalten und Reihen und eine Gangnummer abgespeichert. In Abbildung 3.2 ist ein beispielhafter Regalaufbau mit Gängen und Lagerplätzen dargestellt.

Über die Tabelle *AREA* wird jedem Lagerplatz ein Bereich zugeteilt. Die Attribute teilen jedem Bereich und damit jedem Lagerplatz eine bestimmte Funktion zu. Beispielsweise

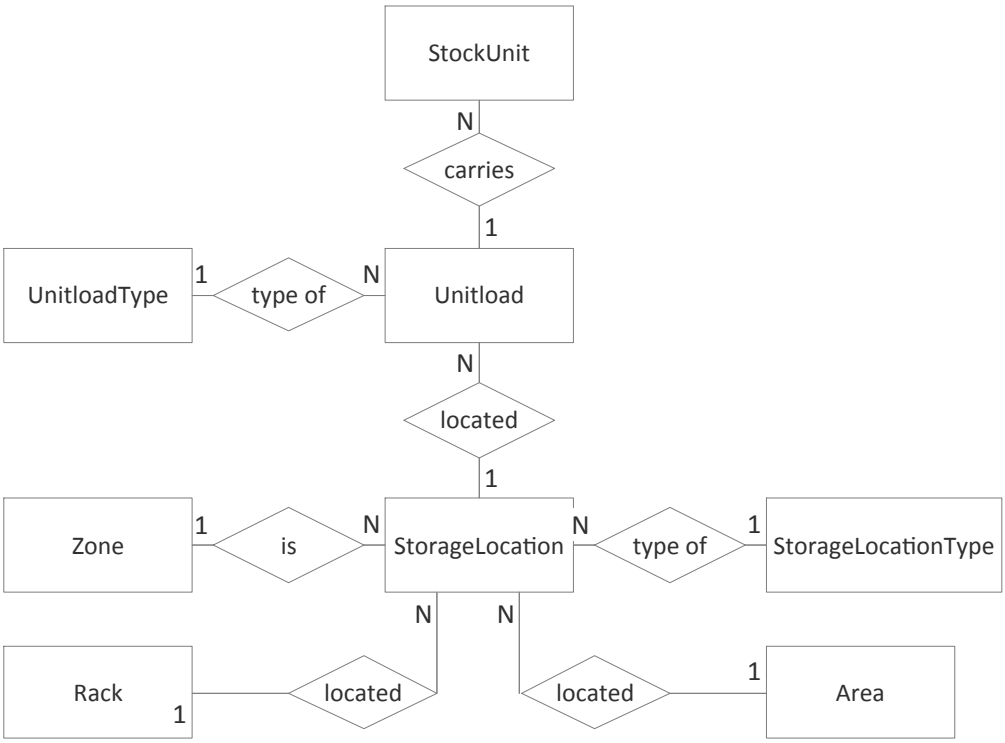

Abbildung 3.3.: ER-Diagramm - Lagerstammdaten relevanter Ausschnitt

wird über das Feld *useForPicking* festgelegt, ob dieser Bereich für die Kommissionierung genutzt wird. Im myWMS-System wird standardmäßig das Lager zur Kommissionierung verwendet.

Falls eine Zonung im Lager zum Einsatz kommt, wird jedem Lagerplatz über die Tabelle *ZONE* eine Zone zugeordnet. Außer einem Namen hat die Zone keine weiteren Attribute.

In Abbildung 3.3 ist der relevante Teil der Lagerstammdaten in einem ER-Diagramm dargestellt.

3.2.2. Bewegungsdaten

Jeder Vorgang im Lagersystem wird in der myWMS-Datenbank in den entsprechenden Tabellen gespeichert. So kann im Nachhinein jede Warenbewegung nachvollzogen werden. Dies ist besonders im Hinblick auf die nachfolgende Modellierung des Datenmodells wichtig, da aus den Bewegungsdaten die Faktentabellen erstellt werden.

Wareneingang

Der Prozess des Wareneingangs beziehungsweise der Warenannahme wird im myWMS-System mit dem *GOODSRECEIPT* quittiert. Jeder Beleg hat dabei eine beliebige Anzahl Anlieferungspositionen. Die Beziehung dieser Entitäten ist im ER-Diagramm in Abbildung 3.4 dargestellt.

Abbildung 3.4.: ER-Diagramm - Bewegungsdaten im Wareneingangsprozess

Pro angelieferten Container/LKW/Wagon wird ein Beleg mit einer eindeutigen Nummer erstellt. Bei der Erstellung des Belegs können verschiedene Daten wie zum Beispiel das Nummernschild des LKW oder der Spediteur festgehalten werden. Falls zusätzlich zum Warehouse Managementsystem ein ERP-System im Unternehmen zum Einsatz kommt, kann der Bearbeiter dem Anlieferungsbeleg ein Lieferavis des ERP-Systems zuteilen. Sobald die Entladung beginnt, wird im System erfasst, wo das entladene Gut zwischengelagert werden soll. Dies ist ein Wareneingangsbereich. Dort wird das Gut auf den Einlagerungsvorgang vorbereitet.

Die Positionen des Belegs werden in der Tabelle *GOODSRECEIPTPOSITION* gespeichert. Jeder Position ist der jeweilige Beleg zugeordnet. Für jede entladene Bestandseinheit wird eine Position erstellt. Da eine Bestandseinheit beliebig viele einzelne Artikel enthalten kann, wird hier die Menge für jede Einheit vermerkt und abgespeichert. Außerdem wird gespeichert, ob eine Position einer Chargen zuzuordnen ist, ob Qualitätsmängel vorliegen und ob eine Position oder ganze Chargen wegen Qualitätsmängeln gesperrt werden müssen.

Nachdem der Vorgang der Entladung/Warenvereinnahmung abgeschlossen ist, existiert im System für jede Bestandseinheit ein Datensatz über welchen fortan alle relevanten Daten ausgelesen werden können. Die Bestandseinheiten, welche auf Ladehilfsmitteln zusammengefasst sind, lagern im Wareneingangsbereich und sind bereit zur Einlagerung.

Umlagerung

Um die Warenbewegung innerhalb eines Lagersystems verfolgen zu können, wird jegliche Bewegung eines Ladehilfsmittels und somit der darauf lagernden Bestandseinheit in der Tabelle *UL_RECORD* erfasst. Folgende Werte werden dabei gespeichert: das Label des bewegten Ladehilfsmittels, zwischen welchen Lagerplätzen es bewegt wurde, der Typ des Eintrags ("CREATED", "TRANSFERRED", "DELETED"), der Typ des Ladehilfsmittels und die Identifikationsnummer des Bearbeiters.

Da die Einlagerung ins Lager eine Warenbewegung ist, lässt sie sich anhand dieser Tabelle nachvollziehen. Aufgrund dessen kann eine eventuelle Bildung von passenden Lagereinheiten in dieser Tabelle nachvollzogen werden. Im Folgenden wird der Einlagerungsvorgang anhand eines Auszugs aus der *UL_RECORD*-Tabelle erklärt. Der Auszug ist in Tabelle 3.3 dargestellt und die verwendete Beispielzeile markiert.

Tabelle 3.3.: Auszug aus der Tabelle *UL_RECORD*

ID	Created	FromLocation	ToLocation	Label	...
4651	2014-06-03 21:05	Wareneingang	000014	000080	...
2653	2014-05-19 20:38	Wareneingang	A1-023-1	000016	...

...	UnitloadType	RecordType	Operator	Client_ID
...	Box 60x40	TRANSFERED	tim	0
...	EURO	TRANSFERED	tim	0

Im Beispiel wird eine Box, welche vorher im Wareneingangsbereich zwischengelagert wurde, auf eine Palette mit der Kennzeichnung 000014 umgelagert.

Das umgelagerte Ladehilfsmittel mit der Kennzeichnung 000080 hat in der Tabelle *UNITLOAD* (siehe markierten Datensatz in Tabelle 3.4) im Feld *CARRIERUNITLOAD*, welches das tragende Hilfsmittel identifiziert, den Schlüssel 1565 eingetragen. Der Datensatz mit dem Schlüssel 1565 ist die Palette, auf welche die Box umgelagert wurde.

Tabelle 3.4.: Auszug aus der Tabelle *UNITLOAD*

ID	LabelID	Carrier	CarrierUnitload	...
1565	000014	true	<null>	...
4416	000080	false	1565	...

...	Weight	WeightCalculated	StorageLocation
...	25	25	1150
...	0.7	0.7	1150

Die Tabelle *UL_RECORD* beinhaltet dabei keine Fremdschlüsselbeziehungen zu den referenzierten Tabellen. Alle Felder sind Textfelder. Dies hat unter anderem den Grund, dass so bei einem eventuellen Wiederverwenden oder Löschen von Datensätzen die Warenbewegung weiter nachvollzogen werden kann. Bei der Benutzung von Fremdschlüsseln, ist bei einem Entfernen von Datensätzen die referentielle Integrität verletzt.

Bestandsverlauf

Der Verlauf der Bestandseinheiten kann in der Tabelle *STOCKRECORD* nachvollzogen werden. Die Felder von dieser Tabelle werden anhand eines Auszugs (Tabelle 3.5) aus der myWMS-Datenbank erklärt.

Im Feld *ACTIVITYCODE* wird die Aktivität gespeichert, welche zu einer Veränderung einer Bestandseinheit geführt hat. In diesem Fall handelt es sich um den Kommissionier-

Tabelle 3.5.: Auszug aus der Tabelle *STOCKRECORD*

ID	Created	ActivityCode	Amount	AmountStock	...
2900	2014-05-20 21:27	PICK 000003	-1	7	...
2901	2014-05-20 21:27	PICK 000003	1	1	...

...	FromStockUnitIdentity	ToStockUnitIdentity	FromStorageLocation	...
...	1605	1605	A1-011-2	...
...	2850	2850	tim	...

...	ToStorageLocation	FromUnitload	ToUnitload	Itemdata	...
...	A1-011-2	000005	000005	14008712	...
...	tim	000017	000017	14008712	...

...	Operator	Type	UnitloadType	Client_ID
...	tim	STOCK_REMOVED	EURO	0
...	tim	STOCK_CREATED	EURO	0

auftrag mit der Nummer 000003. Das Attribut *AMOUNT* gibt die Veränderung der ursprünglichen Menge an. Der Eintrag ist "-1", da ein Stück des Guts entnommen wurde.

AMOUNTSTOCK gibt die Menge des Guts an, die in der Bestandseinheit nach dem Veränderungsvorgang zurückbleibt. Im Beispiel bleiben sieben Einheiten des Guts in der Bestandseinheit zurück. *FROMSTOCKUNITIDENTITY* und *TOSTOCKUNITIDENTITY* beinhalten die von- und nach-Bestandseinheiten des Vorgangs. Analog dazu bezeichnen die Felder *FROMSTORAGELOCATION* und *TOSTORAGELOCATION* den ursprünglichen Lagerort und den Ziellagerort des Vorgangs. *FROMUNITLOAD* und *TOUNITLOAD* geben das Quell- und Zielladehilfsmittel an.

Das Feld *UNITLOADTYPE* gibt an, um welchen Typ von Ladehilfsmittel es sich bei dem Vorgang handelt. Das Feld *TYPE* ist ein Indikator für den Typ der Veränderung. Am Beispiel des Kommissionierauftrags mit der Nummer 000003 ist in der ersten Zeile ein "STOCK_REMOVED" eingetragen, welches indiziert, dass aus einer Bestandseinheit etwas entnommen wurde. Da es sich um den Entnahmevorgang handelt, beinhalten die alle "von"- und "nach"-Felder identische Werte.

In der zweiten Zeile indiziert ein "STOCK_CREATED", dass eine neue Bestandseinheit erstellt wurde. Dieser Vorgang folgt unmittelbar auf den Entnahmevorgang, sodass eine lückenlose Verfolgung des Guts im Warenfluss des Systems möglich bleibt. Hier sind die "von"- und "nach"-Felder identisch. *ITEMDATA* verweist auf die Artikelstammdatentabelle, sodass jedem Vorgang ein Artikel zugeordnet beziehungsweise ohne Umwege über

andere Tabellen herausgefunden werden kann, welche Artikel von der Veränderung der Bestandseinheit betroffen sind.

Zusätzlich zu den genannten Feldern werden ein Zeitstempel, der Benutzer, bei "wertvollen" Gütern eine Seriennummer und bei Chargenpflicht die zugehörige Charge abgespeichert.

Die Tabelle des Bestandsverlaufs beinhaltet keine Fremdschlüsselbeziehungen, sondern nur Verweise in Textfeldern.

Warenausgang

Analog zum Prozess des Wareneingangs wird im Warenausgang pro Bestellung eines Kunden ein Eintrag in der Tabelle *CUSTOMERORDER* angelegt. Bestellungen können von einem übergeordneten ERP-System an das Warehouse Managementsystem weitergeleitet sein. Aus diesem Grund enthält die Tabelle der Bestellungen zusätzlich zur internen Bestellnummer eine externe Bestellnummer und ID, welche auf eine eventuelle Bestellung im ERP-System verweist.

Über einen Fremdschlüssel wird der Zielplatz der Bestellung festgelegt. Beispielsweise kann der Zielplatz die Warenausgangszone vor einem LKW-Tor sein. Der Statusindikator *STATE*, zeigt an in welchem Status sich die Bestellung gerade befindet.

Weitere Felder sind *DOCUMENTURL*, um auf eventuell vorhandene Dokumente zu verlinken, *LABELURL*, um auf ein Label zu verweisen, *CUSTOMERNANME* und *CUSTOMERNR*, um Kunden zu identifizieren.

Die einzelnen Positionen einer Bestellung werden in der Tabelle *CUSTOMERPOS* gespeichert. Dabei kann jede Position über das Feld *ORDER_ ID* einer Bestellung zugeordnet werden. Die Beziehung der Entitäten ist im ER-Diagramm in Abbildung 3.5 dargestellt.

Die Felder *AMOUNT* und *AMOUNTPICKED* geben die bestellte Menge an und wie viel schon kommissioniert wurde. Über einen Fremdschlüssel wird jeder Bestellposition ein eindeutiger Artikel und optional eine Charge zugeordnet. Weiter wird über einen Indikator abgespeichert, ob eine partielle Lieferung erlaubt ist oder nicht.

Abbildung 3.5.: ER-Diagramm - Bewegungsdaten im Warenausgangsprozess

4. Konzept

In diesem Kapitel wird das entwickelte Datenmodell vorgestellt. Die Entwicklung basiert auf den von Kimball beschriebenen vier Schritten. [KR13, vgl S.74]

Der erste Schritt ist die Auswahl eines Geschäftsprozesses. Anschließend wird anhand der Quelldaten die Granularität festgelegt. Nach dem Festlegen der Granularität werden die Dimensionen entwickelt. Der letzte Schritt ist die Auswahl von Fakten, welche in der Faktentabelle dargestellt werden sollen. Die Prozesse eines Lagersystems, welche in Kapitel 2.4 vorgestellt wurden, werden getrennt betrachtet und dazu passend ein Sternschema (siehe 2.1.2) erstellt.

In Abschnitt 4.1 werden die entwickelten Dimensionen vorgestellt. Im darauf folgenden Kapitel 4.2 werden die Faktentabellen der einzelnen Geschäftsprozesse erläutert. Aus der Kombination der Fakten- und Dimensionstabellen ergibt sich je Prozess ein Sternschema.

4.1. Dimensionen

In den folgenden Abschnitten werden die entwickelten Dimensionen dargestellt.

4.1.1. Datumsdimension

Die Datumsdimension wird in beinahe jedem dimensionalen Modell genutzt, da so gut wie jeder Geschäftsprozess eine zeitliche Komponente besitzt. [KR13, vgl. S. 79-80] Der SQL Standard ISO 9075 [DD08] unterstützt kein Filtern nach Attributen wie "Wochentag" oder "Wochennummer". Aufgrund dessen wird eine Datumsdimension genutzt.

In Tabelle 4.1 ist die Datumsdimension dargestellt.

Die Datumsdimension ist in der Tabelle *DateDimension* gespeichert. Jede Zeile in der Tabelle enthält die Daten eines separaten Tages. In Tabelle 4.2 ist ein Auszug aus der Datumsdimension dargestellt. Einige Spalten sind aus Darstellungsgründen entfernt worden.

Der Primärschlüssel der Datumsdimension ist das Attribut *date_key* vom Typ numerisch. Dieser Schlüssel ist trotz seiner Funktion als Primärschlüssel von Menschen lesbar, da er ein Datum im Format "YYYYMMDD" speichert.

Die Spalte *date* beinhaltet das jeweilige Datum im SQL-Datentyp Date. Die Attribute *day_Of_Week* und *day_num_in_month* sind Zahlenfelder und speichern die jeweilige Nummer des Tages in der Woche beziehungsweise im Monat. So sind Vergleiche von gleichen Tagen in verschiedenen Wochen und Monaten möglich.

DateDimension
date_key (PK)
full_date
day_of_week
day_num_in_month
day_num_overall
day_name
day_abbrev
weekday_flag
week_num_in_year
week_num_overall
week_begin_date
week_begin_date_key
month
month_num_overall
month_name
month_abbrev
quarter
year
yearmo
fiscal_month
fiscal_quarter
fiscal_year
last_day_in_month_flag
same_day_year_ago_date

Tabelle 4.1.: DateDimension

In den Feldern *day_name* und *day_abbrev* sind die Wochentagsnamen und die zugehörigen Abkürzungen gespeichert. Mit ihnen ist es für Benutzer einfacher, den Kontext einzelner Tage zu analysieren.

Die Spalten *weekday_flag* und *last_day_in_month_flag* beinhalten Indikatoren, welche das Ende der Woche respektive das Ende des Monat anzeigen. Analog zu *day_of_week* und *day_num_in_month* ist der Spalte *week_num_of_year* die Wochennummer im jeweiligen Jahr gespeichert. Die Spalte *week_begin_date* besitzt den Datentyp *datetime*, welcher das Format YYYY-MM-DD hat. Hier wird abgespeichert, wann eine neue Woche beginnt.

Das Feld *month* beinhaltet die Nummer des Monats, *month_name* und *month_abbr* enthalten analog zu *dayname* und *day_abbr* den Namen und die Abkürzung eines Monats.

date_key	full_date	day_o_w	day_month	day_name	day_abbrev
20140913	2014-09-13	6	13	Saturday	Sa
20140914	2014-09-14	7	14	Sunday	Sun
20140915	2014-09-15	1	15	Monday	Mon

weekday_flag	week_num	month	month_name	abbrev	quarter	year
Weekend	37	9	September	Sep	3	2014
Weekend	37	9	September	Sep	3	2014
Weekday	38	9	September	Sep	3	2014

Tabelle 4.2.: Auszug aus der Tabelle DateDimension

Das Attribut *quarter* speichert die Nummer des Quartals im jeweiligen Jahr ab. In der Spalte *year* ist die Jahresnummer als Zahlenwert gespeichert.

Die letzte Spalte *same_day_a_year_ago* ist vom Typ *datetime* und enthält für jeden Datensatz das Datum des gleichen Tages vor einem Jahr. Mithilfe dieses Wertes sind Vergleiche mit Daten aus dem Vorjahr möglich.

4.1.2. Tageszeitdimension

Die Tageszeitdimension ist ähnlich wie die Datumsdimension eine häufig genutzte Dimension. Datum und Tageszeit werden in zwei unterschiedlichen Dimensionen abgebildet, da die Tabelle der Datumsdimension sonst zusätzlich zu jedem Tag, jede Sekunde eines Tages abspeichern muss. Dies sind 86400 zusätzliche Datensätze pro Tag. Die separate Tageszeitdimension enthält für jede Sekunde eines Tages einen Datensatz. So entstehen 86400 Einträge in der Tabelle. In 4.3 ist die Tabelle der Tageszeitdimension abgebildet.

Der Primärschlüssel der TimeDimension-Tabelle setzt sich aus der Stunden-, Minuten- und Sekundenanzahl nach dem Schema HHMMSS zusammen. Dieser ist vom Typ alphanumerisch. So werden auch führenden Nullen mit abgespeichert. In den Feldern *Hour*, *Minute* und *Second* ist die Anzahl der jeweiligen Stunde, Minute beziehungsweise Sekunde gespei-

TimeDimension
TimeKey (PK)
Hour
Minute
Second
Time

Tabelle 4.3.: TimeDimension

chert. Das Feld *Time* ist vom Datentyp Time und speichert die entsprechende Zeit ab. In Tabelle 4.4 ist ein Auszug mit Beispieldaten aus der Tageszeitdimensionstabelle abgebildet.

TimeKey	Hour	Minute	Second	Time
133659	13	36	59	13:36:59
133700	13	37	00	13:37:00
133701	13	37	01	13:37:01

Tabelle 4.4.: Auszug aus der Tabelle TimeDimension

4.1.3. Artikeldimension

Die Artikelstammdaten werden in die Artikeldimension überführt. Für jeden Artikel existiert ein Datensatz.

Der Artikelstamm liegt in einem operativen System in normalisierter Form vor. Zum Beispiel werden Warengruppen oder die Standardeinheiten in verschiedenen Tabellen gespeichert. Diese Normalisierung wird zugunsten der Produktdimension aus Gründen der Simplizität aufgelöst. Die Artikeldimension ist in der Tabelle *ProductDimension* abgespeichert. Der Aufbau ist in Tabelle 4.5 abgebildet.

Der Primärschlüssel der Produktdimensionstabelle ist der *ProductKey*. Dieser wird im Zuge des ETL-Schrittes als Stellvertreterschlüssel (siehe 2.1.2) neu vergeben.

Die *ProductID* ist die ID des Artikels, welche im operativen System als Primärschlüssel genutzt wurde. Diese wird aus Gründen der Nachvollziehbarkeit und Vollständigkeit weiter mit abgespeichert. Es kann vorkommen, dass Anwender die Schlüssel aus dem operativen System bereits kennen. Ist dies der Fall, kann mit der *ProductID* in der Produktdimension weitergearbeitet werden.

Im Feld *ProductNumber* wird die Artikelnummer des Produkts gespeichert. In den Feldern *Name*, *Description*, *Tradegroup*, *TradegroupDescription* sind entsprechend der Name des Artikels, eine Beschreibung, die Bezeichnung der Warengruppe und eine Beschreibung der Warengruppe gespeichert. Die Spalten *EANCode 1-3* enthalten die eventuell vorhanden EAN-Codes der entsprechenden Artikel.

Die Maßdaten wie Volumen, Gewicht, Breite, Höhe und Tiefe eines Artikels sind in den Feldern *Volume*, *Weight*, *Width*, *Height* und *Depth* gespeichert. Die den Maßdaten zugehörigen Einheiten sind in den Feldern *WHDUnit*, *WeightUnit* und *VolumeUnit* abgespeichert. Durch die Trennung von Einheiten und Werten sind die Aggregationsfunktionen in SQL auf die Maße anwendbar. Falls Einheiten und Werte zusammen in einem Feld abgespeichert werden, so müssen diese Felder vom Typ alphanumerisch sein. Auf alphanumerischen Feldern sind keine Aggregationen möglich.

Die Bezeichnung der Artikeleinheit ist im Feld *ProductUnit* gespeichert. Ein Artikel dessen Menge beispielsweise in Gewicht gemessen wird, hat als Wert im Feld *ProductUnit*

ProductDimension
ProductKey (PK)
ProductID
Name
Description
EANCode1
EANCode2
EANCode3
Weight
WeightUnit
Width
Height
Depth
WHDUnit
Volume
VolumeUnit
SafetyStock
TradeGroup
TradeGroupDescription
ProductUnit
ProductUnitType
BaseUnit
Value
ValueUnit
ABCClassification
ProductNumber

Tabelle 4.5.: ProductDimension

den Wert "kg" gespeichert. Der zur Einheit zugehörige Typ ist im Feld *ProductUnitType* gesichert. Im Falle der Einheit "kg" ist der Typ "weight". Die Einheit "Stk" besitzt als Typen "piece". Falls die Menge eines Artikels durch sein Gewicht bestimmt wird, ist im Feld *BaseUnit* die Basiseinheit der Gewichtseinheit gespeichert.

Das Feld *SafetyStock* ist für den jeweiligen Sicherheitsbestand eines Artikels vorgesehen. Über das Attribut *ABCClassification* wird jedem Artikel seine eventuell vorhandene ABC-Klassifizierung zugeordnet.

Der Wert eines Artikels wird im Feld *Value* und die dem Wert zugehörige Währung im Feld *ValueUnit* gesichert.

4.1.4. Lagerplatzdimension

Die Lagerplatzdimension beschreibt den Ort in einem Lagersystem. Sie ist in der Tabelle *StorageLocationDimension* umgesetzt. Der Lagerort kann zum Beispiel der Wareneingangsbereich oder ein bestimmter Regalplatz im Lager sein. Jeder Datensatz in der Dimension beinhaltet den Datensatz zu genau einem Lagerplatz im Lagersystem. Die Lagerplatzdimension ist in Tabelle 4.6 dargestellt.

StorageLocationDimension
StorageLocationKey (PK)
StorageLocationID
Name
Description
xPos
yPos
zPos
Scancode
Type
Width
Height
Depth
WHDUnit
Volume
VolumeUnit
LiftingCapacity
LiftingCapacityUnit
ZoneName
RackDescription
RackColumns
RackRows

Tabelle 4.6.: StorageLocationDimension

Der *StorageLocationKey* ist der Primärschlüssel der Tabelle. Er ist, wie der Schlüssel der Produktdimension, ein vom ETL-Prozess generierter Stellvertreterschlüssel (siehe 2.1.2). Der Primärschlüssel der Tabelle aus dem operationalen System wird im Feld *StorageLocationID* gespeichert.

Jeder Lagerplatz hat einen Namen als Attribut. Da Lagerplatznamen sich von Warehouse-Managementsystem zu Warehouse-Managementsystem unterscheiden, ist als Datentyp hier Text gewählt. Im Fall vom myWMS-System setzen sich die Namen der Lagerplätze aus einer Nummern- und Zahlenkombination zusammen. Die Bezeichnung "A1-01 2-3" steht für:

"A1" die Regalnummer, "01" das Feld im Regal, "2" für das Fach und "3" für die Ebene im Regal (siehe 3.2.1). Für den Aufbau eines Regals siehe Abbildung 3.2.

Das *Description* Feld ist für eine eventuelle Beschreibung des Lagerplatzes vorgesehen und ist vom Typ alphanumerisch.

Wenn in einem Lagersystem die Lagerplätze in X, Y und Z Koordinaten eingeteilt sind, können für jeden Datensatz die jeweiligen Koordinaten in die Felder *xPos*, *yPos* und *zPos* abgespeichert werden. Weiter besteht die Möglichkeit Scancodes, welche an den Regalfächern angebracht sind, in dem Feld *Scancode* zu hinterlegen.

Die verschiedenen Typen von Lagerplätzen werden im *Type* Feld abgespeichert. Ein Lagerplatz kann zum Beispiel vom Typ Palettenplatz oder Fachboden sein. Zusätzlich zu der Typenbezeichnung sind Felder für die Breite, Höhe, Tiefe, Volumen und Tragfähigkeit der jeweiligen Typen vorgesehen. Zu den Maßeinheiten werden die zugehörigen Einheiten in separaten Feldern gespeichert.

Für eine eventuell vorhandene Lagerzonung ist das Feld *Zone* vorgesehen. Dort kann die Bezeichnung der jeweiligen Zone hineingeladen werden.

Um den einzelnen Lagerplätzen die jeweiligen Regaltypen zuordnen zu können, sind in der Lagerplatzdimension die Felder *RackDescription*, *RackColumns* und *RackRows* vorgesehen. Dort kann eine Beschreibung des Regaltyps und die Anzahl an Spalten und Zeilen des jeweiligen Regaltyps gespeichert werden.

4.1.5. Ladehilfsmitteltypendimension

Um die Verwendung von verschiedenen Ladehilfsmitteltypen im Lagersystem nachvollziehen zu können, wird die Ladehilfsmitteltypendimension genutzt. In ihr werden alle verschiedenen Typen von Ladehilfsmitteln gespeichert. Zu diesen Typen können unter anderem Paletten oder Behälter gehören. In Tabelle 4.7 ist die Ladehilfsmitteltypendimension abgebildet.

Die Ladehilfsmitteltypendimension besitzt einen vom ETL-Prozess erstellten Stellvertreterschlüssel (siehe 2.1.2). Dieser wird im Feld *UnitloadTypeKey* gespeichert und ist der Primärschlüssel der Dimension. Der Primärschlüssel des operativen Systems wird im *UnitloadTypeID* gespeichert.

Um die Ladehilfsmitteltypen unterscheiden zu können, besitzt jeder Typ einen Bezeichner und eine Beschreibung. Für diese Werte sind die Felder Name und *Description* vorgesehen. Beide sind vom Typ alphanumerisch, so dass beliebige Namen oder Werte gespeichert werden können.

Wie in der Produkt- und Lagerplatzdimension können für die Ladehilfsmitteltypen auch Maße gespeichert werden. Die dafür vorgesehenen Felder lauten *Weight*, *Width*, *Height*, *Depth*, *Volume* und *LiftingCapacity*. Zusätzlich sind in dieser Dimension Felder für die Einheiten der Maße vorgesehen.

UnitloadTypeDimension
UnitloadTypeKey (PK)
UnitloadTypeID
Name
Description
Weight
WeightUnit
Width
Height
Depth
WHDUnit
Volume
VolumeUnit
LiftingCapacity
LiftingCapacityUnit

Tabelle 4.7.: UnitloadTypeDimension

So kann zum Beispiel über die maximale Tragkraft eines Ladehilfsmitteltyps eine Überladung festgestellt werden.

4.1.6. Mitarbeiterdimension

In der Mitarbeiterdimension werden alle Mitarbeiterdaten gesichert. Jede Zeile enthält den Datensatz eines Mitarbeiters. Die Tabelle, in der die Mitarbeiterdimension umgesetzt ist, heißt *OperatorDimension*.

Der Primärschlüssel der Tabelle ist im Feld *OperatorKey* abgespeichert. Für jeden Mitarbeiter sind Felder für seinen Vornamen, seinen Nachnamen, seine Emailadresse und seine Telefonnummer vorgesehen. Die Mitarbeiterdimension ist in Tabelle 4.8 abgebildet.

Aus Datenschutzgründen ist es möglich anonymisierte Datensätze einzufügen.

OperatorDimension
OperatorKey (PK)
OperatorID
Firstname
Lastname
Email
Phone

Tabelle 4.8.: OperatorDimension

4.2. Fakten

In den folgenden Abschnitten werden die erstellten Faktentabellen erläutert. Für jeden Geschäftsprozess in einem Lagersystem existieren ein Fakt und damit ein Sternschema. Bei der Erstellung der Fakten wird besonders darauf geachtet, dass sie einfach erweiterbar sind. Es wird weiterhin berücksichtigt, die Sternschemata möglichst simpel zu halten, um einfache Analysen zu ermöglichen.

Zu jedem Geschäftsprozess werden die von Kimball eingeführten Schritte ausgeführt. Zuerst wird anhand der Quelldaten die kleinstmögliche Granularität gewählt. Anschließend wird die Dimensionen ausgewählt und abschließend der eigentliche Fakt bestimmt.

4.2.1. Wareneingangsfakt

Wie in Kapitel 2.4 beschrieben, ist der erste Prozess eines Lagersystems die Warenannahme beziehungsweise der Wareneingang. Die Quelldaten des myWMS-Systems lassen eine kleinstmögliche Granularität von einer Zeile auf dem Anlieferungsbeleg, folglich einer Lieferposition, zu. Aus dieser Granularität resultiert die Faktentabelle, welche die Warenannahme beschreibt. In Abbildung 4.1 ist die Faktentabelle *IncomingGoodsFact* dargestellt.

Um die zeitliche Komponente des Fakts zu beschreiben, wird die Datumsdimension referenziert. Weiter wird die Artikeldimension genutzt, da sich eine Position des Anlieferungsbelegs immer einem Artikel zuordnen lässt. Die Lagerplatzdimension wird als örtliche Komponente des Fakts referenziert, und beschreibt den Lagerplatz, auf den die Artikel entladen werden. Zusätzlich wird die Ladehilfsmitteltypdimension referenziert, um die Typen von entladenen Ladehilfsmitteln zuordnen zu können. Die Mitarbeiterdimension wird referenziert, um jeder Warenannahme einen verantwortlichen Mitarbeiter zuordnen zu können.

Der Wert *DeliveryNumber* ist von der Lieferscheinnummer, somit von einer Lieferung als Ganzes abhängig. Dieser wird als degenerierte Dimension abgespeichert, um Rückschlüsse

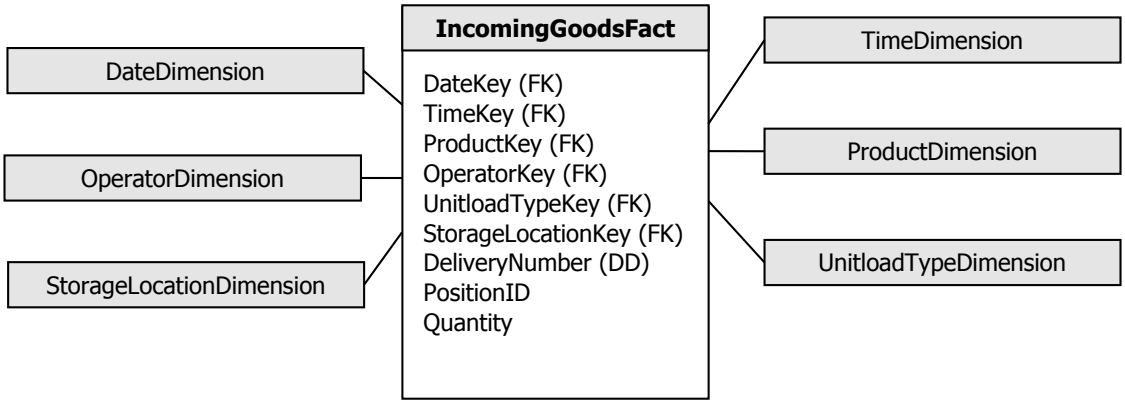

Abbildung 4.1.: Sternschema - Wareneingang

auf komplette Lieferungen ziehen zu können. Da es vorkommen kann, dass zwei Positionen einer Lieferung mit dem gleichen Produkt, zur gleichen Uhrzeit und zum gleichen Ort entladen werden, wird im Feld *PositionID* der Wert der einzelnen Bestellpositionen gespeichert. Dieser ist in einigen Fällen notwendig, um die Eindeutigkeit eines Datensatzes zu gewährleisten.

Im Attribut *Qantity* wird die entladene Menge einer angelieferten Position gespeichert. Die Menge ist ein additiver Fakt, da sie sich über alle Dimensionen aufaddieren lässt.

Der Primärschlüssel der Tabelle *IncomingGoodsFact* setzt sich aus den Fremdschlüsseln aller referenzierten Dimensionen und dem Wert der *PositionID* zusammen.

4.2.2. Einlagerungsfakt

Auf den Prozess des Wareneingangs folgt die Einlagerung. Die Quelldaten lassen in diesem Fakt eine Granularität von einer Bestandseinheit zu. Das heißt, dass in der Einlagerungsfaktentabelle *StoringFact* für jede eingelagerte Bestandseinheit (siehe 3.2.1) ein Datensatz angelegt wird. Der *StoringFact* ist in Tabelle 4.2 abgebildet.

Um bei der Analyse des Einlagerungsprozesses die zeitliche Komponente berücksichtigen zu können, wird die Datumsdimension referenziert. Eine Bestandseinheit besteht aus Artikeln. Demzufolge enthält der Fakt eine Referenz auf die Produktdimension. Der bei der Einlagerung benutzte Ladehilfsmitteltyp ist über eine Referenz auf die Ladehilfsmitteltypdimension und der verantwortliche Mitarbeiter über eine Referenz auf die Mitarbeiterdimension abrufbar.

Bei der Einlagerung wird eine Bestandseinheit von einem Lagerort zu einem anderen Lagerort transportiert wird. Aus diesem Grund kommt das Konzept der Role-Playing-Dimensionen (siehe 2.1.2) zum Einsatz. Eine doppelte Referenz auf eine einzelne Tabelle verhindert SQL-Join-Operationen. Aus diesem Grund werden zwei Views der Tabelle *StorageLocationDimension* erstellt. Zum einen die *FromLocationDimension* und zum an-

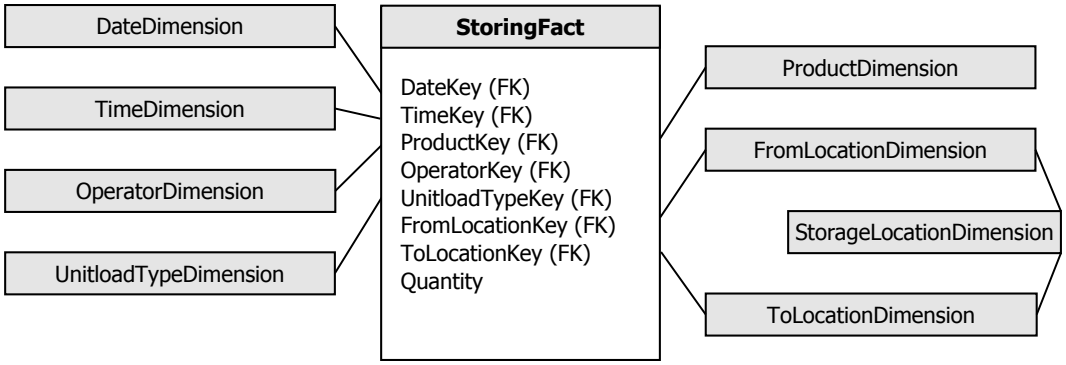

Abbildung 4.2.: Sternschema - Einlagerung

deren die *ToLocationDimension*. So referenziert die Tabelle des Einlagerungsfakts die View *FromLocationDimension* über den Fremdschlüssel *FromLocationKey* und die View *ToLocationDimension* über den Fremdschlüssel *ToLocationKey*. Mithilfe dieser beiden Referenzen sind genaue Analysen der örtlichen Komponente beim Einlagerungsvorgang möglich.

Zusätzlich zu den referenzierten Dimensionen wird die Menge eines Artikels in der transportierten Bestandseinheit im Feld *Quantity* gespeichert. Die Menge der Artikel ist über alle Dimensionen additiv.

Der Primärschlüssel der *StoringFact*-Tabelle setzt sich aus allen Fremdschlüsseln der referenzierten Dimensionstabellen zusammen.

4.2.3. Auslagerungsfakt

Der Vorgang der Auslagerung ist analog zum Einlagerungsvorgang. Aufgrund dessen referenziert der Auslagerungsfakt die gleichen Dimensionstabellen wie die Einlagerungsfaktentabelle. Auch die Granularität der Quelldaten ist identisch. Das Feld *Quantity* enthält wie in der Einlagerungsdimension die Menge eines Artikels einer Bestandseinheit.

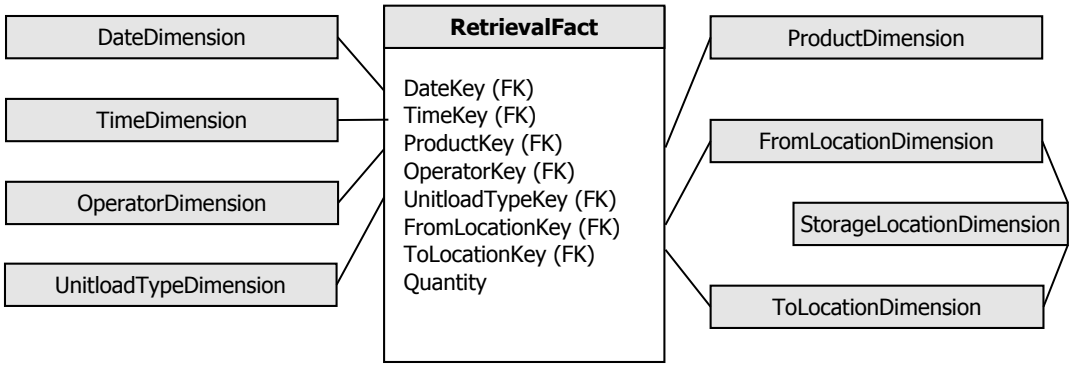

Abbildung 4.3.: Sternschema - Auslagerung

4.2.4. Kommissionierfakt

Der Prozess der Kommissionierung wird mithilfe des Kommissionierungsfakts beschrieben. Der Name der zugehörigen Faktentabelle ist *PickingFact* (siehe Abbildung 4.4).

Ein Kommissionierungsauftrag besteht in der operativen Datenbank des myWMS-Systems aus einzelnen Kommissionierpositionen. Im Fakt der Kommissionierung wird pro Position eine Zeile angelegt, da dies die kleinstmögliche Granularität ist, welche die Quelldaten zulassen.

Als zeitliche Komponente wird die Datumsdimension referenziert. Die Daten des Bearbeiters, der den Kommissionierauftrag bearbeitet, können über die Referenz auf die Mitar-

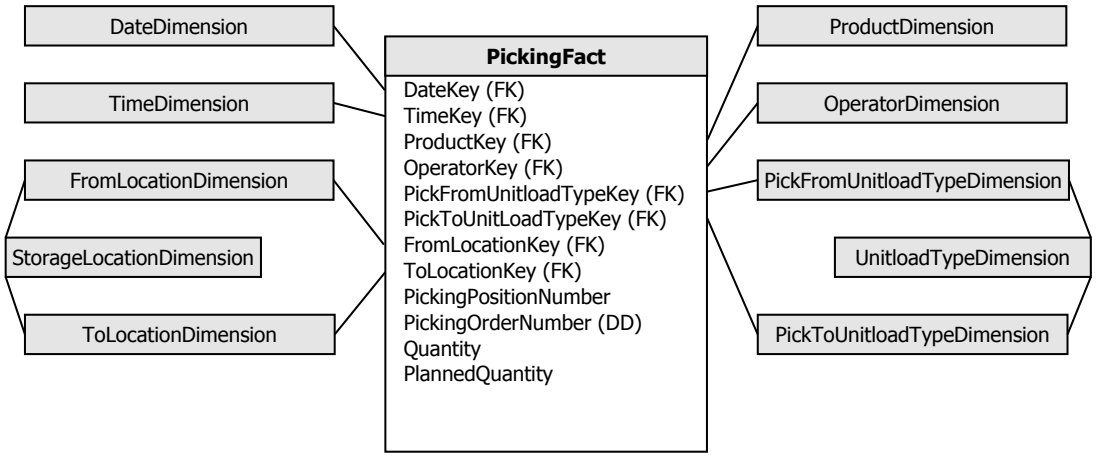

Abbildung 4.4.: Sternschema - Kommissionierung

beiterdimension festgestellt werden. Die Artikel, welche kommissioniert werden, sind durch den Bezug zur Artikeldimension gespeichert.

Bei der Kommissionierung wird, wie auch bei der Ein- und Auslagerung, das Gut von einem Lagerplatz zu einem anderen bewegt. Demnach hat der Kommissionierfakt jeweils eine Referenz auf die erstellten Views *FromLocationDimension* und *ToLocationDimension*. Weiter kann es vorkommen, dass sich der Ladehilfsmitteltyp bei der Kommissionierung ändert. So kann ein Teil einer Bestandseinheit auf einer Palette gelagert sein, jedoch bei der Kommissionierung in einen Behälter umgepackt werden. Um diesen Vorgang abbilden zu können, werden die Views *PickFromUnitloadTypeDimension* und *PickToUnitloadType-Dimension* als Role-Playing-Dimensionen referenziert.

Als degenerierte Dimension wird die Nummer des Kommissionierauftrags im Feld *Picking-OrderNumber* gespeichert. Anhand dieser Nummer können Rückschlüsse auf komplette Kommissionieraufträge geschlossen werden.

Die Menge eines zu kommissionierenden Artikels wird im Feld *Quantity* gespeichert. Aufgrund von Fehlmengen kann es vorkommen, dass einzelne Postionen nicht kommissioniert werden können. Aus diesem Grund wird zusätzlich im Feld *PlannedQuantity* die geplante Kommissioniermenge gespeichert. Der Fakt der Kommissioniermenge ist additiv (siehe 2.1.2), weil er über alle Dimensionen aufaddiert werden kann.

Um die Eindeutigkeit von Datensätzen zu gewährleisten wird zusätzlich zu den Fremdschlüsseln aller referenzierten Dimensionen, die Positionsnummer des Kommissionierauftrags im Feld *PickingPositionNumber* gespeichert. Der Primärschlüssel setzt sich aus dem Feld *PickingPositionNumber* und allen Fremdschlüsseln der von der Tabelle *PickingFact* referenzierten Dimensionen zusammen.

4.2.5. Warenverladungsfakt

Der Prozess der Warenverladung schließt den Durchlauf von Waren durch ein Lagersystem ab.

Dieser Prozess wird im Warenverladungsfakt abgebildet. Dazu wird die Faktentabelle *LoadingFact* erstellt. Diese ist in Abbildung 4.5 dargestellt.

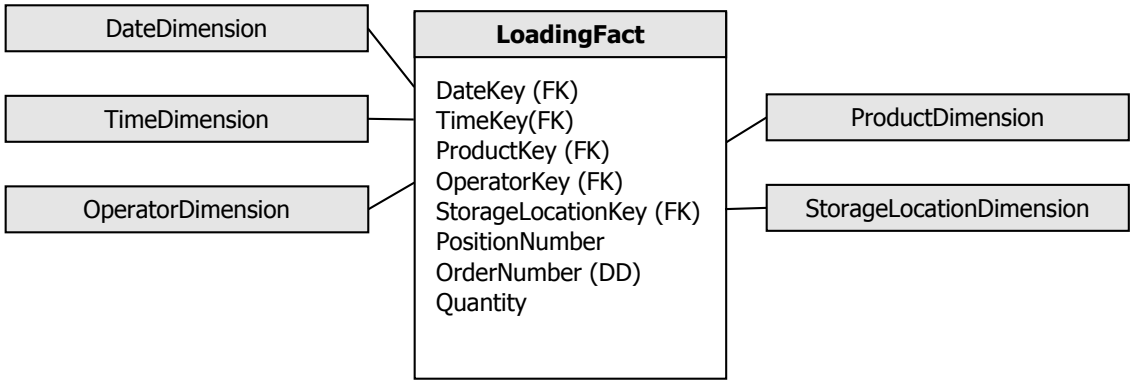

Abbildung 4.5.: Sternschema - Warenverladung

In den operativen Daten des myWMS-Systems wird pro Bestellung eines Kunden ein Eintrag in der Bestellungstabelle angelegt. Jede Bestellung hat dabei mehrere Bestellpositionen. Im Fakt der Warenverladung wird pro Bestellposition ein Datensatz angelegt.

Die Datumsdimension, welche von der Faktentabelle des Warenverladungsprozesses referenziert wird, weist jedem Datensatz das Datum zu. Um Bezug auf verladene Artikel nehmen zu können, wird die Artikeldimension referenziert. Zusätzlich kann der Bearbeiter der Warenverladung über die Mitarbeiterdimension ermittelt werden. Der Standort der Warenverladung im Lagersystem ist in der Tabelle *LoadingFact* über eine Referenz auf die Tabelle der Lagerplatzdimension gesichert.

Um die einzelnen Einträge in der Tabelle der Warenverladungsdimension ihren zugehörigen Aufträgen zuordnen zu können, wird als degenerierte Dimension die Auftragsnummer im Feld *OrderNumber* abgespeichert.

Die Menge des verladenen Guts wird zugehörig zu der Position im Feld *Quantity* gespeichert. Sie ist additiv, da sie sich über alle Dimensionen aufaddieren lässt.

4.3. Mögliche Erweiterungen

In diesem Abschnitt werden mögliche Erweiterungen vorgestellt. In Kapitel 4.3.1 wird erläutert wie zu den in Abschnitt 4.1 vorgestellten Dimensionen Attribute hinzugefügt werden können und wie neue Dimensionen erstellt werden können.

Anschließend werden in Abschnitt 4.3.2 mögliche Faktentabellen zur Erweiterung des Datenmodells vorgestellt.

4.3.1. Dimensionserweiterungen

Wenn das Datenmodell erweitert werden soll, ist ein erster Schritt die Dimensionen um neue Attribute zu erweitern. Zusätzlich ist es möglich komplett neue Dimensionen zu erstellen.

Eine Erweiterung der Dimensionen um einzelne Attribute wird im Abschnitt "Hinzufügen von Attributen" vorgestellt. Das Hinzufügen von komplett neuen Dimensionen wird im darauffolgenden Abschnitt "Erstellen von neuen Dimensionen" erläutert.

Hinzufügen von Attributen

Die bestehenden Dimensionen können um neue Attribute erweitert werden. Die aktuelle Artikeldimension hat mit den Feldern *TradeGroup* und *TradeGroupDescription* nur zwei Attribute, welche Rückschlüsse auf die Warengruppe der Artikel zulassen.

ProductDimensionExtended
ProductKey (PK)
ProductID
Name
...
TradeGroup
TradeGroupDescription
CountryOfOrigin
CityOfOrigin
TradeGroupType
TradeGroupDepartment
...

Tabelle 4.9.: Artikeldimension mit Attributen zur Warengruppe erweitert

Falls weitere Attribute zu verschiedenen Warengruppen in den Quelldaten vorhanden sind, könnten diese der Artikeldimension hinzugefügt werden. Weitere Attribute für die Warengruppe könnten je nach Material oder Artikel die Herkunft, der Verwendungszweck oder Sachbereiche sein. In Tabelle 4.9 ist eine erweiterte Artikeldimension abgebildet.

Erstellen von neuen Dimensionen

Falls eine komplett neue Dimension erstellt und für die Analyse genutzt werden soll, ist das möglich. Das in Kapitel 4 vorgestellte Datenmodell modelliert nur den innerbetrieblichen Warenfluss. Um im Wareneingang Analysen über die verschiedenen Lieferanten erstellen

SupplierDimension
SupplierKey (PK)
NameOfSupplier
ContactPhoneNumber
ContactEmail Address
Contact Name
City
...

Tabelle 4.10.: Mögliche Lieferantendimension

zu können, ist eine weitere Dimension notwendig. Die Lieferantendimension könnte Attribute wie den Namen des Lieferanten, die Telefonnummer, die Email-Adresse, einen Ansprechpartner, die Stadt und die volle Adresse enthalten. Zusätzlich wird ein eindeutiger Primärschlüssel benötigt. Dieser Primärschlüssel wird bei der Erweiterung des Wareneingangsfakts von der Faktentabelle referenziert.

4.3.2. Faktenerweiterung

Im Folgenden wird eine mögliche Erweiterung des Datenmodells durch neue Faktentabellen vorgestellt. Es wird anhand eines Beispiels erläutert wie ein Geschäftsprozess, welcher noch nicht im Modell abgebildet ist, modelliert und somit analysiert werden kann.

Erstellen von neuen Fakten

Der erste Schritt zur Erstellung einer neuen Faktentabelle ist die Auswahl eines Geschäftsprozesses.

Angenommen in einem Lagersystem findet eine häufige Durchlagerung statt. Falls gewünscht ist, diesen Geschäftsprozess genauer zu analysieren, wird eine neue Faktentabelle erstellt.

Nachdem der Geschäftsprozess bestimmt ist, wird anhand der Quelldaten die Granularität der Faktentabelle bestimmt. Im Fall der Durchlagerung wird in der Faktentabelle ein Datensatz pro durchgelagerter Bestandseinheit erstellt.

Wenn der Geschäftsprozess ausgewählt und die Granularität bestimmt ist, werden die Dimensionen ausgewählt. Um die zeitliche Komponente der Durchlagerung abzubilden, werden die Tageszeitdimension und die Datumsdimension vom Durchlagerungsfakt referenziert. Jede Bestandseinheit beinhaltet Artikel. Demzufolge wird außerdem die Artikeldimension genutzt. Die örtliche Komponente wird dem Durchlagerungsfakt durch die beiden Role-Playing-Dimensionen *FromLocationDimension* und *ToLocationDimension* zu-

geordnet. Zusätzlich wird die Ladehilfsmitteltypendimension referenziert, um die bei der Durchlagerung verwendeten Ladehilfsmitteltypen analysieren zu können.

Als additiver Fakt wird die Menge einer durchgelagerten Bestandseinheit gespeichert. In Abbildung 4.6 ist ein möglicher Durchlagerungsfakt dargestellt.

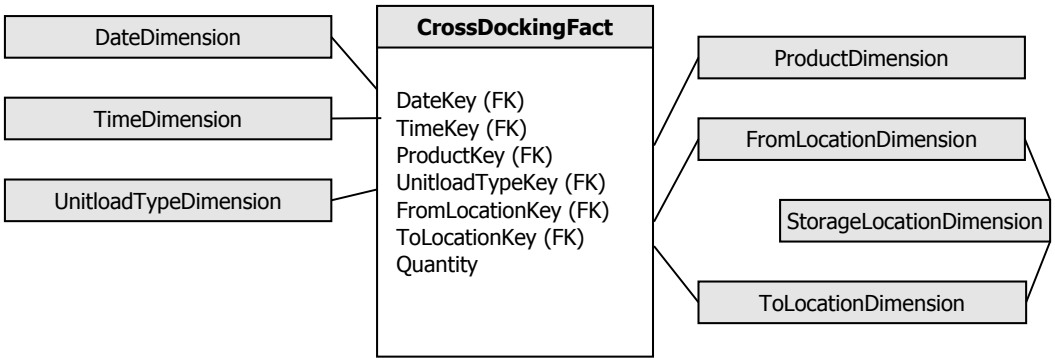

Abbildung 4.6.: Mögliches Sternschema - Durchlagerung

5. Implementierung

Um die Funktion des in Kapitel 4 vorgestellten Datenmodells zu validieren, wird im Zuge dieser Arbeit eine Applikation entwickelt. Diese setzt sich aus zwei separaten Teilen zusammen.

Zum einen wird eine ETL-Anwendung (siehe 2.2) prototypisch realisiert. Diese ermöglicht eine Transformation und das Laden von csv-Dateien in das Datenmodell.

Zum anderen wird das Open-Source Tool Saiku BI [Met14] angepasst und eingebunden. Mithilfe der Saiku Applikation ist es möglich, über eine grafische Benutzeroberfläche Daten zu analysieren und Kennzahlen zu bilden. Diese bereitet Saiku visuell in Form von verschiedenen Graphen auf.

In Abschnitt 5.1 wird die Architektur der Anwendung vorgestellt. Darauffolgend wird in 5.2 die Projektstruktur der Anwendung erläutert. Abschließend werden in den Abschnitten 5.3 und 5.4 die Funktionsweise der ETL-Anwendung und der Saiku-Anwendung vorgestellt.

Auf der beigelegten DVD befindet sich eine unter Windows ausführbare Version der Anwendung. Eine Anleitung zum Starten dieser ist im Ordner "aDatA_Saiku" abgelegt.

5.1. Architektur

Die ETL-Anwendung basiert auf dem Java-EE Standard [Ora14b] und enthält eine Vier-Schichten-Architektur. In Abbildung 5.1 sind die Schichten einer Java EE Anwendung dargestellt.

Der Anwender greift auf die Anwendung über einen Webbrowser zu. Die Präsentationsschicht der Anwendung besteht aus der Java-Server-Faces-Technologie [Ora14d] und dem Komponenten-Framework Primefaces [Pri14]. In der Geschäftslogik der Anwendung wird die EJB- [Ora14a] und JMS-Technologie [Ora14e] verwendet.

Als Datenbank dient MySQL [Ora14f]. Die MySQL-Datenbank wird aufgrund der Verfügbarkeit von Versionen für alle gängigen Betriebssysteme genutzt. Infolge dessen wird eine Plattformunabhängigkeit erreicht. Zusätzlich ist das Datenbanksystem kostenlos nutzbar und weit verbreitet. Die Tabellen des in Kapitel 4 vorgestellten Datenmodells werden mithilfe mehrerer SQL-Skripte erstellt. Die verwendeten Skripte befinden sich auf der beiliegenden DVD im Ordner "SQL-Skripte".

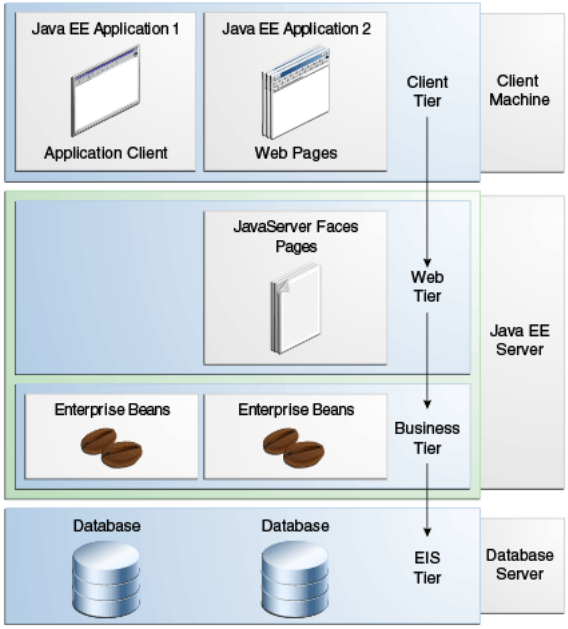

Abbildung 5.1.: Java EE - Schichten
Quelle:[Ora14c]

5.2. Projekt Struktur

Die Anwendung basiert auf einer Projektstruktur. Diese ermöglicht ein Verteilen von Berechtigungen. Jeder Nutzer kann Projekte erstellen, um in diesen seine Daten auszuwerten. Für jedes Projekt wird dabei im Datenbanksystem eine separate Datenbank erstellt. So ist sichergestellt, dass ein Nutzer nur auf seine eigenen Daten zugreifen kann. Ein Nutzer kann beliebig viele Projekte erstellen.

5.3. ETL-Prozess

Der ETL-Prozess (siehe 2.2) teilt sich in drei Unterprozesse auf. Die Extraktion der Quelldaten aus der operativen Datenbank wird in Abschnitt 5.3.1 beschrieben. Das darauffolgende Transformieren und Laden der Daten wird in den Abschnitten 5.3.2 und 5.3.3 erläutert.

5.3.1. Extraktion

Im Extraktionsteil werden die Daten aus der Quelldatenbank extrahiert. Dazu werden im Fall vom myWMS-System (siehe Kapitel 3.2) neun SQL-Abfragen eingesetzt. Von diesen neun Abfragen wählen vier die benötigten Daten für die Artikel-, Lagerplatz-, Mitarbeiter- und Ladehilfsmitteltypdimension aus. Die übrigen fünf extrahieren die Daten für die Fak-

tentabellen des Sternschemas. Exemplarisch werden im Folgenden die SQL-Abfragen für die benötigten Daten der Artikeldimension und des Wareneingangsfakts dargestellt.

Extraktion der Artikelstammdaten

Die Artikelstammdaten speichert das myWMS-System, wie in Kapitel 3 beschrieben, in der *ITEMDATA*-Tabelle. Die Artikeleinheiten sind in der Tabelle *UNITTYPE* gespeichert. Aufgrund dessen ist eine "join"-Operation dieser beiden Tabellen notwendig. Das myWMS-System speichert standardmäßig Artikel und Artikelgruppe in der gleichen Tabelle. Aufgrund dessen ist keine zusätzliche "join"-Operation notwendig, um die Artikelgruppen zu extrahieren. Die resultierende SQL-Abfrage ist in Listing 5.1 dargestellt.

Es werden alle Spalten der *ITEMDATA*-Tabelle sowie der Name und Typ der zugehörigen Artikeleinheit ausgewählt. Das Resultat der Abfrage wird als csv-Datei exportiert und abgespeichert.

Listing 5.1: SQL-Abfrage zur Extraktion der Artikelstammdaten

```
SELECT item.*, unit.UNITNAME, unit.UNITTYPE
FROM MYWMS_ITEMDATA AS item, MYWMS_ITEMUNIT AS unit
WHERE item.HANDLINGUNIT_ID = unit.ID;
```

Es werden die Lager-, Mitarbeiter- und Ladehilfsmitteltypstammdaten aus dem operativen System exportiert und in csv-Dateien abgespeichert.

Extraktion der Wareneingangdaten

Um die nötigen Daten für die Tabelle des Wareneingangsfakts zu extrahieren, werden in Listing 5.2 Daten aus den Tabellen *GOODSRECEIPT*, *GOODSRECEIPTPOSITION*, *STOCKRECORD*, *UNITLOADTYPE* und *ITEMDATA* abgefragt.

Die *GOODSRECEIPTPOSITION*-Tabelle enthält nicht den Primärschlüssel des Artikels, sondern die Artikelnummer. Weiterhin ist nicht der Primärschlüssel des Ladehilfsmitteltyps enthalten. Die "join"-Operationen mit der Artikel- und Ladehilfsmitteltyptabelle ist notwendig, weil der Primärschlüssel dieser beiden Tabellen exportiert werden soll.

Ist der Datenbestand des operativen Systems bereits fehlerhaft, besteht die Möglichkeit, dass mit der in Listing 5.2 dargestellten SQL-Abfrage nicht alle Daten korrekt exportiert werden.

Listing 5.2: SQL-Abfrage zur Extraktion der Wareneingangsdaten

```
SELECT gr.ID AS GoodsInId, grp.ID AS GoodsInPositionID,
        grp.CREATED, gr.FORWARDER, ult.ID AS UnitloadTypeID,
        it.ID AS ItemID , gr.OPERATOR_ID,
        gr.GOODSINLOCATION_ID, grp.AMOUNT
FROM LOS_GRRPOSITION as grp, LOS_GOODSRECEIPT AS gr,
        LOS_STOCKRECORD AS sr, MYWMS_UNITLOADTYPE AS ult,
        MYWMS_ITEMDATA AS it
WHERE gr.ID = grp.GOODSRECEIPT_ID
        AND it.ITEM_NR = grp.ITEMDATA
        AND sr.FROMUNITLOAD = grp.UNITLOAD
        AND sr.TOUNITLOAD = grp.UNITLOAD
        AND sr.UNITLOADTYPE = ult.name
        AND sr.FROMSTORAGELOCATION ='Wareneingang'
        AND sr.TOSTORAGELOCATION = 'Wareneingang';
```

Nachdem alle benötigten Daten aus dem operativen System extrahiert sind, werden die einzelnen csv-Dateien in die entwickelte Applikation geladen.

Generierung der Datums- und Zeitdimension

Die Datensätze der Datums- und Zeitdimension werden mithilfe von Excel-Tabellen generiert. Die zur Generierung benutzen Dateien befinden sich auf der beiliegenden DVD im Ordner "Excel Tabellen". Die generierten Datensätze werden mithilfe von SQL-Skripten in die Tabellen eingefügt. Diese sind im Ordner "SQL-Skripte" zu finden.

5.3.2. Transformation

Der Transformationsschritt beginnt mit dem Laden der exportierten csv-Dateien in das System. Das Laden der csv-Dateien ist nicht mit dem Laden von Datensätzen in die Datenbank, dem dritten Schritt des ETL-Prozesses, gleichzusetzen.

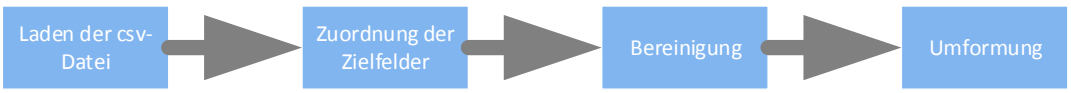

Abbildung 5.2.: Überblick Transformationsvorgang

Tabellentypen
Dimensionen
Artikelstammdaten
Lagerplatzstammdaten
Ladehilfsmitteltypstammdaten
Mitarbeiterstammdaten
Fakten
Wareneingangsdaten
Einlagerungsdaten
Auslagerungsdaten
Kommissionierdaten
Warenverladungsdaten

Tabelle 5.1.: Auswahlmöglichkeiten Tabellentyp

Ein Überblick über die einzelnen Schritte des Transformationsvorgangs ist in Abbildung 5.2 dargestellt.

Über eine Weboberfläche wählt der Anwender aus, dass er eine neue Datei importieren möchte. Dazu muss er den Tabellentyp auswählen. In Tabelle 5.1 sind die verschiedenen Auswahlmöglichkeiten aufgelistet.

Nachdem die Auswahl des Tabellentyps abgeschlossen ist, legt das System im Hintergrund eine neue Entität vom Typ *ImportTable* an. Zusätzlich kann ein Bezeichner zur einfacheren Unterscheidung der Tabellen vergeben werden.

Nun muss vom Benutzer ausgewählt werden, ob die erste Zeile der zu importierenden Datei, Überschriften enthält. Im späteren Verlauf des Prozesses ist es wichtig, dass dies angegeben wird, da andernfalls die erste Zeile der Datei beim Laden der Daten übersprungen wird.

Nachdem der Anwender den Inhalt der ersten Zeile bestätigt hat, kann er eine csv-Datei hochladen. Dazu wird die hochzuladende Datei ausgewählt und bestätigt. Im Hintergrund wird die csv-Datei auf den Server geladen und abgespeichert. Nach dem abgeschlossenen Upload liest die Applikation automatisch die erste Zeile der Datei ein. Beim Einlesen wird für jede Spalte ein *TableObject* erstellt. Diese Objekte werden mit dem bereits erstellten *ImportTable*-Objekt verknüpft. In Abbildung 5.3 ist ein UML-Klassendiagramm der beiden Klassen dargestellt.

Im Feld *colIndex* wird die Nummer der Spalte und in *fieldValue* der zugehörige Wert gespeichert. In *importToTable* ist der vom Anwender ausgewählte Tabellentyp abgespeichert.

Nachdem eine zu importierende Tabelle erstellt und eine csv-Datei hochgeladen wurde, werden die Spalten der csv-Dateien den Feldern der Tabellen des Sternschemas zugeordnet.

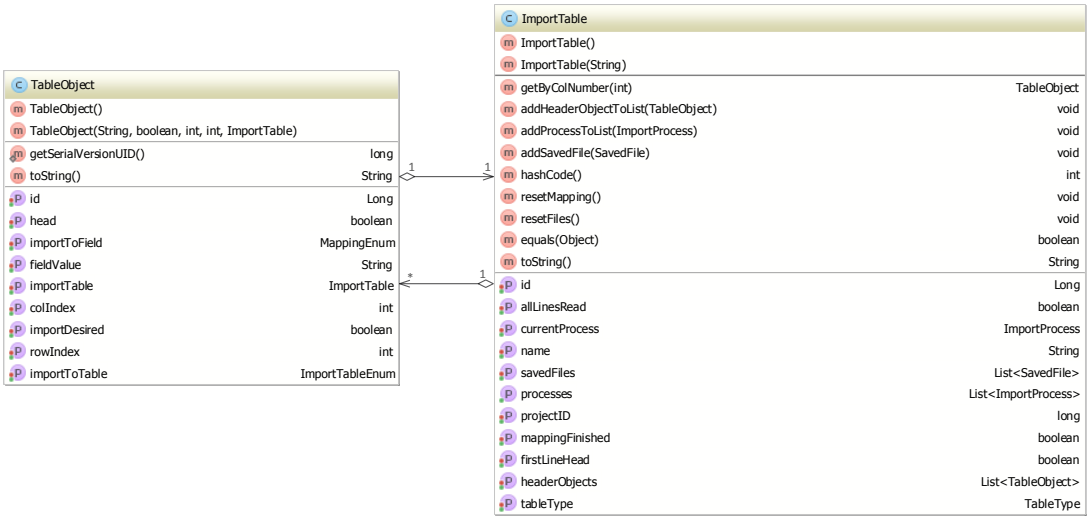

Abbildung 5.3.: Klassendiagramm TableObject und ImportTable

Um die Zuordnung zu beginnen, wird eine Tabelle ausgewählt und anschließend auf "Mapping" geklickt. Dem Anwender werden die Datensätze aus der ersten Zeile der Datei angezeigt. Diese Zeile kann Überschriften oder Datensätze enthalten. Neben jedem Wert befindet sich ein Auswahl-Feld. Aus dieser wird der entsprechende Name der Spalte im Zielschema ausgewählt.

Der Anwender hat die Option seine Auswahl zwischenzuspeichern und zu einem späteren Zeitpunkt fortzuführen, seine aktuelle Auswahl zurückzusetzen, ins vorherige Menü zurückzukehren oder die Zuordnung abzuschließen.

Nach jeder Auswahl eines Zielfelds, wird das ausgewählte Ziel per AJAX-Anfrage im *TableObject* abgespeichert. So geht eine ungesicherte Auswahl nicht verloren. Zusätzlich zu dem Namen wird hier der Datentyp des Zielfelds festgelegt. Falls ein Feld nicht in das Zielschema übernommen werden soll, genügt es nichts auszuwählen. Dieser Ablauf ist in Abbildung 5.4 visualisiert.

Beim Abschluss des Zuordnungsvorgangs ist für jede Spalte des csv-Dokuments im *TableObject* das Zielfeld und die Zieltabelle abgespeichert. Liegt keine Auswahl vor, wird ein leeres Zielfeld gespeichert. Zusätzlich ist in dem Objekt *ImportTable* der Wert *mappingFinished* auf "true" gesetzt.

Im Hauptmenü ist die Schaltfläche "Import starten" jetzt freigegeben. Sobald diese betätigt wird, startet im Hintergrund der Import in die jeweilige Datenbanktabelle. Die Applikation erstellt nach der Betätigung der Schaltfläche eine ObjectMessage mit der *ImportTable* als Inhalt. Diese Nachricht wird an eine JMS-Warteschlange (siehe [Ora14e]) des Applikationsservers gesendet.

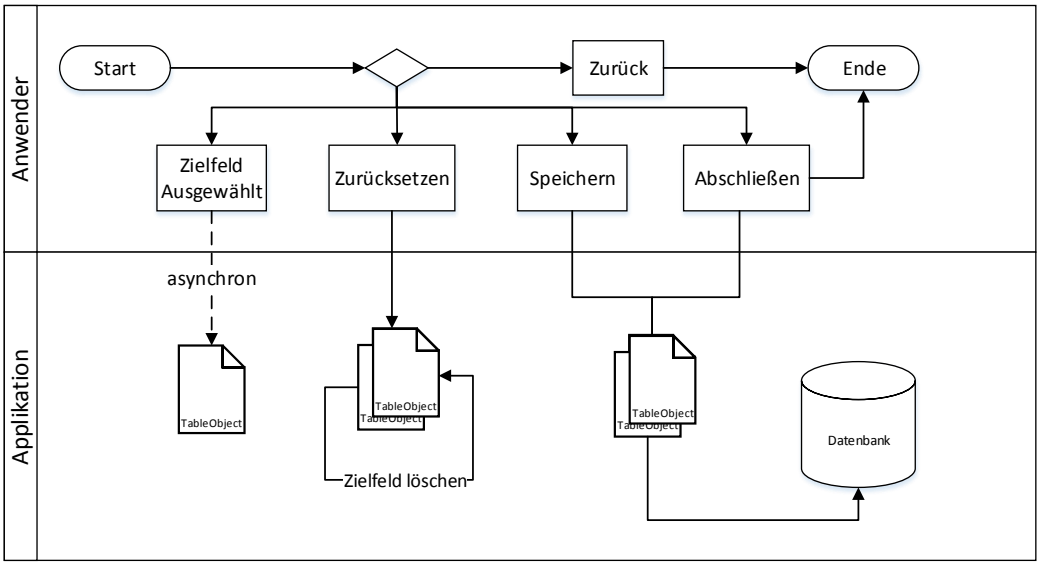

Abbildung 5.4.: Ablauf Mappingprozess

Durch diese Architektur werden eine lose Kopplung der Komponenten und eine asynchrone Bearbeitung von Anfragen möglich. Die JMS-Spezifikation bietet zusätzliche eine verlässliche Kommunikation. [Ora10]

Die Warteschlange wird von der *Message-Driven-Bean Consumer* geleert. Die csv-Datei wird zeilenweise eingelesen. Je Spalte existiert ein, aus dem Zuordnungsprozess befülltes, *TableObject*. Es wird für jedes Feld einer Zeile ein *TransferObject* erstellt. Die Transferobjekte unterscheiden sich je nach Datentyp des Zielfelds. In Abbildung 5.5 ist die Klasse *AbstractTransferObject* dargestellt. Die Transferobjekte werden in einer Liste gespeichert. Sie enthält alle Werte einer eingelesenen Zeile.

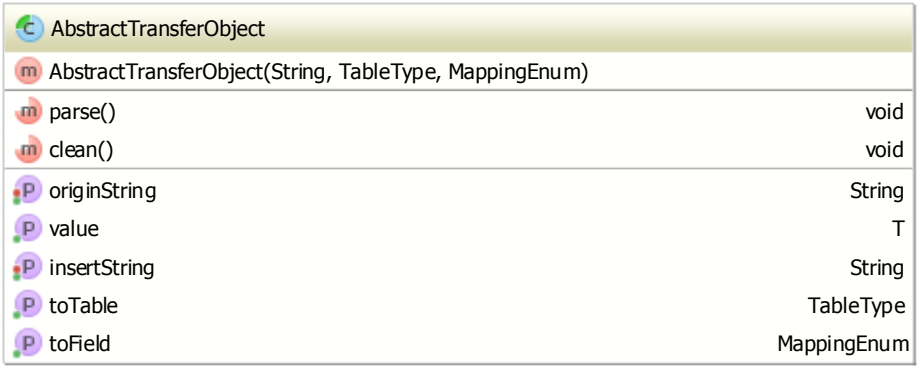

Abbildung 5.5.: Klassendiagramm *AbstractTransferObject*

Nachdem eine komplette Zeile eingelesen ist, werden auf den Transferobjekten der Liste die Methoden *clean* und *parse* aufgerufen. Im Bereinigungsschritt wird auf Wertebene der jeweilige Wert des Objekts bereinigt. Dies bedeutet beispielsweise im Fall eines *DoubleTransferObjects*, dass aus dem eingelesenen *String*-Wert eventuelle Zeichen entfernt werden.

Anschließend folgt der Aufruf der *parse*-Methode. In dieser werden die jeweiligen *String*-Werte in die Zieldarstellung geparst. Zusätzlich wird das *insertString*-Feld mit dem jeweiligen Wert befüllt.

Falls es sich bei dem eingelesenen Dokument um die Quelldaten zum Befüllen einer Faktentabelle handelt, wird im nächsten Schritt eine Methode der *SurrogateKeyFinderBean* aufgerufen. In dieser werden die Stellvertreterschlüssel (siehe 2.1.2) der bereits eingefügten Dimensionen aus der Datenbank abgefragt und im *insertString*-Feld des jeweiligen Objekts gespeichert. Der Primärschlüssel jedes Datensatzes einer Dimension ist zusätzlich zu dem neu vergebenen Stellvertreterschlüssel abgespeichert. Nach diesem Schritt ist jedem Fremdschlüssel der Faktentabelle ein eindeutiger Datensatz aus einer Dimensionstabelle zugeordnet.

Die Liste, welche eine Zeile aus dem Quelldokument repräsentiert, enthält für jeden Wert ein Transferobjekt. Dieses ist beim Abschluss des Transformationsschritts mit den zum Laden benötigten Werten befüllt. Sobald die Liste alle bearbeiteten Transferobjekte enthält, wird die *loadLine* der Java-Klasse *SQLAsync* aufgerufen. Diese schreibt die Daten in die Datenbank.

5.3.3. Laden

Der Ladevorgang beginnt mit dem Aufruf der *loadLine* Methode. Diese bekommt eine Liste mit Transferobjekten übergeben. Sie repräsentiert eine Zeile von Werten aus den Quelldaten. Im Transformationsschritt (siehe 5.3.2) sind diese Werte bereits geparst worden.

Diese Liste wird je nach Tabellentyp (Fakt oder Dimension) an die Methode *insertDimension* oder *insertFact* der Java-Bean *LineInserter* übergeben.

Im Fall einer Dimension wird eine Abfrage erstellt, welche überprüft, ob der Primärschlüssel des einzufügenden Datensatzes schon in der Dimensionstabelle enthalten ist. Falls der Schlüssel schon existiert, wird der Datensatz übersprungen und der Anwender bekommt nach dem Abschluss des Vorgangs eine Benachrichtigung, dass nicht alle Datensätze eingefügt werden konnten. Wenn der Schlüssel nicht existiert, wird von der Java-Bean *QueryBuilder* die Einfüge-Anweisung erstellt. Diese Anweisung wird an die *insert*-Methode des *NativeFacade*-Objekts übergeben. Die *NativeFacade* ist nach dem "Data Access Object"-Entwurfsmuster (siehe [Ora12]) aufgebaut. Dort wird die Einfüge-Anweisung ausgeführt und damit der Datensatz in die Datenbank eingefügt.

Der Ablauf beim Einfügen von Daten in eine Faktentabelle ist ähnlich. Allerdings wird aufgrund der größeren Zeilenanzahl der Faktentabellen die Überprüfung auf Duplikate nicht ausgeführt. Falls es beim Einfügen der Datensätze zu Duplikaten kommt, speichert die Applikation eine Fehlermeldung ab. Auf diese kann der Anwender nach Abschluss des Ladeprozesses reagieren. Die Fehlerbehandlung wird in Abschnitt 5.3.4 beschrieben.

Nachdem alle Datensätze in die Datenbank eingefügt sind, ist der Ladeprozess abgeschlossen.

5.3.4. Fehlerbehandlung

Im ETL-Prozess können verschiedene Fehler auftreten. Im Abschnitt Trennzeichenfehler werden die Fehler beschrieben, die bei dem Einlesen von csv-Dateien auftreten können. Fehler die im Transformationsschritt auftreten können, werden in Abschnitt Formatierungsfehler erläutert. Anschließend werden im Abschnitt Duplikatfehler die Fehler erklärt, welche auftreten falls ein Datensatz beim Laden der Daten bereits in der Datenbank existiert. Abschließend werden im Abschnitt Duplikatfehler die Fehler erklärt. Diese treten auf, wenn geladene Datensätze bereits in der Datenbank existieren.

Trennzeichenfehler

Während des Einlesens hochgeladener Dateien erkennt die Applikation die Trennzeichen "," und ";" automatisch als Trennungen und liest die Felder entsprechend ein. Trennzeichenfehler treten auf, wenn nicht unterstützte Trennzeichen verwendet werden.

Tritt ein Fehler auf, so hat der Anwender nur die Möglichkeit die Trennzeichen seiner csv-Datei anzupassen und diese erneut hochzuladen.

Formatierungsfehler

Formatierungsfehler können beim Parsen der einzelnen Transferobjekte (siehe 5.3.2) auftreten.

Datum- und Zeitwerte müssen im Format "yyyy-MM-dd HH:mm:ss.SSS" oder "HH:mm:ss" gespeichert sein. Falls diese Werte in einem anderen Format vorliegen, müssen sie geändert werden.

Zahlenwerte, wie zum Beispiel die Menge eines Artikels, dürfen keine Buchstaben enthalten. Falls Buchstaben enthalten sind, erstellt die Anwendung beim Parsing der Quelldaten Fehler.

Der Anwender hat bei aufgetretenen Formatierungsfehlern mehrere Möglichkeiten diese zu beheben. Wenn bei einem Importvorgang fünf oder weniger Formatierungsfehler auftreten, lädt die Anwendung die betroffenen fehlerhaften Zeilen in ein Formular, und der Anwender kann Werte manuell ändern. Die maximale Anzahl ist auf fünf Fehler voreinge-

stellt. Diese kann im Quellcode der Anwendung geändert werden. Die geänderten Werte werden anschließend durch eine Bestätigung vom Anwender separat importiert.

Wenn Formatierungsfehler in fünf oder mehr Zeilen auftreten, hat der Anwender entweder die Möglichkeit die Zuordnung zu wiederholen oder eine korrigierte Datei hochzuladen. In beiden Fällen werden die Tabellen, welche von diesem fehlerhaften Importprozess betroffen sind, komplett geleert und das bis dahin durchgeführte Zuordnung zurückgesetzt. Falls der Anwender eine korrigierte Datei hochladen möchte, wird außerdem die bereits hochgeladene Datei gelöscht.

Duplikatfehler

Wenn beim Einfügen von Datensätzen Duplikatfehler auftreten, so hat der Anwender nur die Möglichkeit die Zuordnung zurückzusetzen oder eine neue Datei mit Quelldaten hochzuladen. Die Möglichkeit der Korrektur von Datensätzen, wie sie bei den Formatierungsfehlern möglich ist, entfällt.

5.4. Auswertung

Saiku ist eine Open-Source Analyse-Anwendung. Mit Saiku ist es möglich, Daten zu analysieren und visuell aufzubereiten. Saiku wird aufgrund der freien Verfügbarkeit des Quellcodes, der kostenlosen Nutzbarkeit und der intuitiven Bedienoberfläche genutzt.

Bei der Anpassung von Saiku an die Datenbank und das Datenschema müssen zwei Konfigurationsdateien verändert werden. In Abschnitt 5.4.1 wird die notwendige Konfiguration der Datenquelle beschrieben. Darauf folgend wird in Kapitel 5.4.2 der Aufbau einer Schemadatei erläutert.

5.4.1. Datenquelle

Die Datenquelle wird in einer Konfigurations-Datei konfiguriert. In dieser wird festgelegt, welche Datenbank mit Saiku verwendet werden soll. Außerdem werden dort Sicherheitsrelevante Einstellungen festgelegt. Die verwendete Konfiguration ist in Listing 5.3 dargestellt.

Listing 5.3: Konfigurationsdatei Datenquelle - Saiku

```
# Typ der Datenquelle
type=OLAP
# Bezeichnung der Datenquelle
name=myWMS
# Treibername
driver=mondrian.olap4j.MondrianOlap4jDriver
```

```
# Speicherort des Datenbanktreibers und Datenbank-URL
location=jdbc:mondrian:Jdbc=jdbc:mysql://localhost/adata1;
# Speicherort der Schemadatei
Catalog=res:adata_benutzer/adata.xml;JdbcDrivers=com.mysql.jdbc.Driv
# Datenbank Benutzer und Passwort
username=admin
password=admin
# Sicherheitseinstellungen
security.enabled=true
security.type=one2one
```

Die Authentifizierung und Autorisierung des Anwenders ist mithilfe des Spring-Security Moduls (siehe [PIV10]) umgesetzt. So ist eine gemeinsame Verwendung von Nutzerdaten für die ETL- und Saiku-Anwendung möglich.

Der Sicherheitstyp "one2one" bildet die Benutzerrollen, auf die in der Schemakonfiguration angegebenen Rollen ab (siehe dazu 5.4.2).

5.4.2. Schemadatei

Die Schemadatei definiert ein mehrdimensionales Datenmodell (siehe 2.1). Mithilfe dieser Schemadefinition ist es für die Saiku-Anwendung möglich, zwischen den Dimensions- und Faktentabellen, verschiedenen Datentypen und Benutzern zu unterscheiden.

Im Folgenden werden einzelne Ausschnitte aus der angepassten Schemakonfiguration erklärt. Die komplette Konfigurationsdatei befindet sich auf der beiliegenden DVD im Ordner "Saiku-Konfiguration".

Dimensionen

In Listing 5.4 ist ein Ausschnitt aus der Dimensionskonfiguration dargestellt. In diesem Ausschnitt wird ein Teil der Datumsdimension konfiguriert. Es wird festgelegt, dass es sich um eine Dimension im zeitlichen Kontext handelt. Außerdem wird angegeben, in welcher Tabelle die Daten der Dimension gespeichert sind und wie der Bezeichner des Primärschlüssels lautet.

In der Datei wird die Monatshierarchie der Datumsdimension definiert. Sie ermöglicht dem Anwender der Saiku-Anwendung Datensätze nach "Jahr - Quartal - Monat - Tag im Monat" zu filtern. Diese Hierarchie wird in der Konfigurationsdatei im Hierarchieteil mithilfe der angegebenen Level eingestellt. Jedem Level werden dabei unter anderem ein Name, ein Datentyp und die Spalte in der Dimensionstabelle zugeordnet.

Nach diesem Prinzip werden in der Schemadatei alle in Kapitel 4.1 erstellten Sternschemata konfiguriert.

Listing 5.4: Dimensionsausschnitt aus der Schemakonfiguration

```
<Dimension name="Datum" visible="true"
        highCardinality="false" type="TimeDimension" >
 <Hierarchy name="Monate" visible="true"
        hasAll="true" primaryKey="date_key">

  <Table name="datedimension"/>

  <Level name="Jahr" visible="true" column="year"
        type="String" uniqueMembers="true"
        levelType="TimeYears" hideMemberIf="Never"/>

  <Level name="Quartal" visible="true"
        column="quarter" type="String"
        uniqueMembers="false" levelType="TimeQuarters"
        hideMemberIf="Never"/>

  <Level name="Monat" visible="true"
        column="month" nameColumn="month_name"
        type="String" uniqueMembers="false"
        levelType="TimeMonths" hideMemberIf="Never"
        captionColumn="month_abbrev"/>

  <Level name="Tag" visible="true"
        column="day_num_in_month" type="String"
        uniqueMembers="false" levelType="TimeDays"
        hideMemberIf="Never"/>

 </Hierarchy>
 ...
</Dimension>
```

Fakten

In Listing 5.5 ist ein Ausschnitt aus der Faktenkonfiguration dargestellt. Jede Faktentabelle wird in der Konfigurationsdatei über eine *Cube*-Definition erstellt. Ein *Cube* stellt in der Konfigurationsdatei eine Kombination aus einem Fakt und Dimensionen dar. In die-

ser werden der Bezeichner des Fakts und der Name der Faktentabelle in der Datenbank angegeben.

Anschließend wird über den Bezeichner *DimensionUsage* festgelegt, welche Dimensionen referenziert werden. In Kapitel 4.2 ist dem Wareneingangsfakt die degenerierte Dimension "Lieferscheinnummer" hinzugefügt. Diese degenerierte Dimension wird über einen separaten *Dimension*-Bezeichner festgelegt.

Im Bezeichner *Measure* werden die Fakten festgelegt. Im Fall des Wareneingangsprozesses wird die Anzahl der angelieferten Artikel pro Position eingestellt. Zusätzlich wird im *Measure*-Bezeichner festgelegt, dass die Gesamtzahl der Lieferungen über das Zählen des Attributs *deliveryNumber* möglich ist.

Listing 5.5: Dimensionsausschnitt aus der Schemakonfiguration

```
<!-- Wareneingansfakt -->
<Cube name="Wareneingang" caption="Wareneingang" visible="true"
        cache="true" enabled="true">
  <Table name="incominggoodsfact"/>
  <DimensionUsage source="Datum" name="Datum" visible="true"
        foreignKey="DATEKEY" highCardinality="false"/>
  <Dimension name="Lieferscheinnummer">
    <Hierarchy hasAll="true">
      <Level name="Lieferscheinnummer" column="DELIVERYNUMBER"
        uniqueMembers="true"/>
    </Hierarchy>
  </Dimension>
  <Measure name="Anzahl" column="QUANTITY"
        aggregator="sum" visible="true"/>
  <Measure name="Anzahl Lieferungen"
        column="deliveryNumber" aggregator="distinct-count"/>
  ...
</Cube>
```

Berechtigungen

In der Schemakonfiguration wird festgelegt, welche Berechtigungen ein Benutzer besitzen muss, um auf dieses Schema zugreifen zu können. In Listing 5.6 hat der Benutzer mit der Rolle "ROLE_ benutzer", Zugriff auf alle definierten Schemata.

Listing 5.6: Rollenverwaltung

```
<Role name="ROLE_benutzer">
        <SchemaGrant access="all"/>
</Role>
```

6. Zusammenfassung und Ausblick

Dieses Kapitel ist in zwei Abschnitte gegliedert. In Abschnitt 6.1 werden die vorherigen Kapitel zusammengefasst und in Relation zur Aufgabenstellung gesetzt. Abschließend folgt ein Ausblick auf mögliche Erweiterungen und Verbesserungen.

6.1. Zusammenfassung

Ziel dieser Arbeit war die Entwicklung eines Datenmodells. In einem ersten Schritt wurde der Aufbau eines Warehouse Managementsystems analysiert, wodurch ein Verständnis für die Struktur der Quelldaten erlangt werden konnte (siehe Kapitel 3).

Aus der erstellten Datenbank wurden die für die Konzeption des Datenmodells relevanten Daten und Beziehungen extrahiert, woraufhin das neue Datenmodell erstellt wurde (siehe Kapitel 4). Dabei wurde besonders Wert auf die Erweiterbarkeit gelegt.

Um die Funktion des entwickelten Datenmodells zu validieren, ist eine Software-Lösung entwickelt worden, die einen Import von csv-Dateien ermöglicht (siehe Kapitel 5). Die Dateien wurden aus den Quelldaten des myWMS-Systems exportiert. Nachdem die Daten in das neu entwickelte Schema geladen wurden, konnten Kennzahlen und Analysen gebildet werden.

Im letzten Schritt dieser Arbeit ist die Anwendung Saiku an das Datenmodell und die Datenbank angepasst worden. Mit Saiku ist es möglich Analysen zu erstellen, Daten visuell aufzubereiten und Berichte zu exportieren.

6.2. Ausblick

Im Folgenden wird ein Ausblick auf weiterführende Themen gegeben.

6.2.1. Quelldaten

In Deutschland, Österreich und der Schweiz gibt es über 150 Anbieter von Warehouse Managementsystemen. Der Großteil dieser Systeme unterscheidet sich in der Form und Art der Datenhaltung. [HWD07]

In dieser Arbeit wurde das operative Datenmodell des myWMS-Systems analysiert. Zur Extraktion der zur Analyse benötigten Daten ist ein SQL-Skript entwickelt worden. Um den Datenbestand anderer Systeme analysieren zu können, müssen die benötigten Daten

aus diesen extrahiert werden. Aufgrund dessen ist eine genauere Analyse der Datenmodelle dieser Systeme erforderlich.

6.2.2. Anwendung

Die entwickelte Anwendung ist so konzipiert, dass sie sich erweitern lässt. Im nachfolgenden Abschnitten werden beispielhaft drei mögliche Erweiterungen und Verbesserungen vorgestellt.

Eingabeformat

Zum jetzigen Zeitpunkt ist es lediglich möglich, csv-Dateien einzulesen. Eine zusätzliche Limitierung ist, dass als Trennzeichen der Datei nur "," und ";" erkannt werden. Wie in Kapitel 5.3.2 erklärt, wird außerdem nur eine limitierte Anzahl von Datums- und Uhrzeitformaten erkannt.

Das Hinzufügen der Funktionalität, welche ein Einlesen von zusätzlichen Dateiformaten, wie zum Beispiel Excel-Dateien, ermöglicht, ist bereits in der Anwendung vorgesehen. Zusätzlich ist es mit geringem Aufwand realisierbar weitere Trennzeichen und Datumsformate hinzuzufügen.

Plausibilitätsüberprüfung

Eine Prüfung der hochgeladenen Werte findet in der aktuellen Anwendung auf Wertebene statt. Dies bedeutet, dass Zeichenfehler in Mengenfeldern erkannt werden. Falls jedoch beispielsweise eine unnatürliche Abweichung von Daten über einen bestimmten Zeitraum auftritt, erkennt die Anwendung diese nicht automatisch. Dies birgt die Gefahr, dass die Ergebnisse von Analysen fehlerhaft sind.

Eine Erweiterung, welche die Quelldaten mithilfe von statistischen Methoden auf Plausibilität überprüft, verhindert fehlerhafte Ergebnisse von Analysen.

6.2.3. Kennzahlenbildung

In der Anwendung werden Kennzahlen und Analysen mithilfe der Saiku-Anwendung gebildet. Diese erfordert nach dem Import der Quelldaten eine Nutzerinteraktion.

Da jedoch häufig ähnliche Kennzahlen gebildet werden, könnte die Anwendung so erweitert werden, dass dies automatisch nach dem Import geschieht. Die Saiku-Anwendung bietet das Speichern von erstellten Abfragen an. Dies ermöglicht es, einen Katalog von häufig eingesetzten Abfragen zu erstellen. So ist es dem Anwender möglich, die gewünschten Kennzahlen schnell zu bilden.

A. Weitere Informationen

ITEMDATA
ID (PK)
ADDITIONALCONTENT
CREATED
MODIFIED
VERSION
ADVICEMANDATORY
DEPTH
HEIGHT
LOTMANDATORY
LOTSUBSTITUTIONTYPE
NAME
ITEM_NR
REST_USAGE_GI
SAFETYSTOCK
SCALE
SERIALCTYPE
TRADEGROUP
VOLUME
WEIGHT
WIDTH
DEFULTYPE_ID (FK)
CLIENT_ID (FK)
ZONE_ID (FK)
HANDLINGUNIT_ID (FK)

Tabelle A.1.: Schema *ITEMDATA*-Tabelle

Abbildungsverzeichnis

2.1. Beispiel für ein Sternschema 5
2.2. Schematischer Datenfluß eines ETL-Systems 9
2.3. Prozesse in einem Lagersystem 10

3.1. ER-Diagramm - ItemData relevanter Ausschnitt 20
3.2. Beispielhafter Aufbau eines Regals Quelle: vgl. [Fra14b] 21
3.3. ER-Diagramm - Lagerstammdaten relevanter Ausschnitt 22
3.4. ER-Diagramm - Bewegungsdaten im Wareneingangsprozess 23
3.5. ER-Diagramm - Bewegungsdaten im Warenausgangsprozess 26

4.1. Sternschema - Wareneingang 35
4.2. Sternschema - Einlagerung 36
4.3. Sternschema - Auslagerung 37
4.4. Sternschema - Kommissionierung 38
4.5. Sternschema - Warenverladung 39
4.6. Mögliches Sternschema - Durchlagerung 42

5.1. Java EE - Schichten Quelle:[Ora14c] 44
5.3. Klassendiagramm TableObject und ImportTable 48
5.4. Ablauf Mappingprozess 49
5.5. Klassendiagramm *AbstractTransferObject* 49

Literaturverzeichnis

[Ada10] Adamson, Christopher: Star Schema: The Complete Reference. (2010). ISBN 9780071744331

[DD08] Date, Christopher; Darwen, Hugh: *ISO/IEC 9075-2:2008 (SQL - Part 2: Foundations) , "The SQL Standard", Third Edition (Addison-Wesley Publishing Company, 1993).* 2008

[Fra14a] Fraunhofer IML: *myWMS Warehouse Managementsystem.* `http://www.mywms.org/`. Version: 2014

[Fra14b] Fraunhofer IML: *myWMS Warehouse Managementsystem.* `http://community.mywms.org/typo3temp/pics/70d5e6b670.gif`. Version: 2014

[HS07] ten Hompel, Michael; Schmidt, Thorsten: *Warehouse Management: Organisation und Steuerung von Lager- und Kommissioniersystemen.* 3. Aufl. Berlin: Springer, 2007. – ISBN 783540748755

[HS10] ten Hompel, Michael; Schmidt, Thorsten: *Warehouse Management : Organisation und Steuerung von Lager- und Kommissioniersystemen.* Berlin: Springer, 2010 (VDI-Buch). – ISBN 9783642031854

[HSB11] ten Hompel, Michael; Sadowsky, Volker; Beck, Maria: *Kommissionierung: Materialflusssysteme 2 - Planung und Berechnung der Kommissionierung in der Logistik.* Springer, 2011 (Materialflusssysteme). – ISBN 9783540296225

[HWD07] ten Hompel, Michael; Wolf, Oliver; Daniluk, Damian: WMS-Marktreport 2007-2008. (2007). ISBN 9783899570601

[JPT10] Jensen, Christian; Pedersen, Torben B.; Thomsen, Christian: *Multidimensional Databases and Data Warehousing.* 2010. – ISBN 9781608455379

[KR13] Kimball, Ralph; Ross, Margy: *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling.* 3rd. New York, NY, USA: John Wiley & Sons, Inc., 2013. – ISBN 9781118530801

[Met14] Meteorite BI: *Saiku.* `http://www.meteorite.bi/saiku`. Version: 2014

[Ora10] Oracle: *JMS - Reliable Message Delivery.* http://docs.oracle.com/cd/E19316-01/820-6424/aerbz/index.html. Version: 2010

[Ora12] Oracle: *Core J2EE Patterns - Data Access Object.* http://www.oracle.com/technetwork/java/dataaccessobject-138824.html. Version: 2012

[Ora14a] Oracle: *Enterprise Beans.* http://docs.oracle.com/javaee/7/tutorial/doc/ejb-intro.htm. Version: 2014

[Ora14b] Oracle: *Java EE.* http://www.oracle.com/technetwork/java/javaee/overview/index.html. Version: 2014

[Ora14c] Oracle: *Java EE Documentation.* http://docs.oracle.com/javaee/7/tutorial/doc/overview003.htm. Version: 2014

[Ora14d] Oracle: *JavaServer Faces Technology.* http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html. Version: 2014

[Ora14e] Oracle: *JMS API.* http://docs.oracle.com/javaee/7/tutorial/doc/jms-concepts001.htm. Version: 2014

[Ora14f] Oracle: *MySQL.* http://www.mysql.de/. Version: 2014

[PIV10] PIVOTAL SOFTWARE: *Spring Security.* http://projects.spring.io/spring-security/. Version: 2010

[Pri14] PrimeTek Informatics: *PrimeFaces.* http://primefaces.org/. Version: 2014

[Rei01] Reichmann, Thomas: *Controlling mit Kennzahlen und Managementberichten : Grundlagen einer systemgestützten Controlling-Konzeption.* 6., überarb. und erw. Aufl. München: Vahlen, 2001

ERKLÄRUNG

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 9. September 2014

Tim Lutz