JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

SOWI
Sozial- und Wirtschafts-wissenschaftliche Fakultät

Erweiterung des Data–Warehouse des OÖ Verkehrsverbundes um geographische, demographische und Fahrplandaten

DIPLOMARBEIT

zur Erlangung des akademischen Grades

MAG. RER. SOC. OEC

im Diplomstudium

WIRTSCHAFTSINFORMATIK

Angefertigt am Institut für Wirtschaftsinformatik – Data & Knowledge Engineering

Betreuung / Begutachter:

o.Univ.-Prof. DI Dr. Michael Schrefl

Mitbetreuung:

Mag. Dr. Stefan Berger

Eingereicht von:

Ing. Helmut Hofer, MSc.

Linz, im Juni 2011

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

_____

Linz, Juni 2011 *Helmut Hofer*

DANKSAGUNG

Mein Dank ergeht an …

… meinen Betreuer Mag. Dr. Stefan Berger, Assistent am Institut für Data & Knowledge Engineering, für seine kompetenten und erfrischenden Inputs zur Formung dieser Diplomarbeit

… den OÖ. Verkehrsverbund, der mir diese praxisorientierte Arbeit ermöglicht hat und mit Daten, Fakten und Zahlen unterstützt hat.

… das Institut für Data & Knowledge Engineering unter der Leitung von o.Univ.-Prof. DI Dr. Michael Schrefl für die fachliche und organisatorische Unterstützung während der Ausarbeitung dieser Diplomarbeit.

… meine Frau Gabriele und meine beiden Kinder Sandra und Jan, welche auf viel verzichteten und mir überhaupt erst die Möglichkeit gaben, dieses Studium zu besuchen.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

FORMELVERZEICHNIS

ABSTRACT

For several years the Upper Austrian Transport Association ("Oberösterreichischer Verkehrsverbund", OÖVV) has been operating a data warehouse for public transport ticket sales analysis in the Austrian province Upper Austria. The reports generated from the data warehouse are required for the accounting of transport services and strategic management in transport planning.

In order to facilitate strategic decisions about the public transport offered in Upper Austria, it is important for the OÖVV to obtain better evaluations about current demand for public transport per municipality in relation to demographic characteristics. To this end, the Upper Austrian Transport Association (OÖVV) wishes to deliver to its stakeholders more precise reports better founded on more broad data, to meet the demands of county or municipality, and to justify its mission to politics and the public.

The thesis in hand deals with the integration of new data sources on public transport schedules, demographic data and geographic area data into the existing data warehouse. First we define the basic concepts of data warehousing, data import using ETL processes, data-driven versus user-driven process models for data warehouse design, and modeling of geographical data. Also we familiarize the reader with the background of this thesis by introducing the Upper Austrian Transport Association (OÖVV). Then the existing data warehouse and the current extraction, transformation and loading (ETL) processes are illustrated. The implemented solution integrating the new data sources is compared to and discussed with similar approaches in the literature.

An analysis of the new data sources—public transport schedules, geographic and demographic data—provides an overview of the integration problem. Using the data-driven process model of the Dimensional Fact model according to (Golfarelli & Rizzi, 1998), the existing data structure and the new data structures are transferred into an updated data warehouse. The ETL processes required for the import of the data sources are developed using Microsoft SQL Server Integration Services (SSIS). Finally, a test system illustrates the benefits of the additional data available, and shows the results of a sample analysis.

KURZFASSUNG

Der oberösterreichische Verkehrsverbund (OÖVV) betreibt derzeit ein Data Warehouse zur Auswertung der verkauften Fahrscheine nach gegebenen Verkehrszonen. Diese Auswertungen werden für die Abrechnung von Verkehrsleistungen und auch zur strategischen Steuerung in der Verkehrsplanung benötigt.

Für die Überprüfung gesetzter Maßnahmen, wie die Einführung neuer Buslinien oder Verdichtung des Taktverkehrs auf bestehenden Linien, wird es aber wichtiger, bessere Auswertungen zu erhalten. Erst dann ist es für den OÖ. Verkehrsverbund möglich, den Wünschen und Forderungen seitens des Landes oder der Gemeinden mit präziseren Analysen entgegenzutreten.

Diese Diplomarbeit beschäftigt sich mit der Einbindung der neuen Datenquellen Fahrplandaten, demographische Daten und geographische Flächendaten in das bestehende Data Warehouse. Mit der Erarbeitung der Begriffe Data Warehouse, Datenimport mittels ETL-Prozesse, der Vorstellung datengetriebener und nutzergetriebener Vorgangsmodelle und der Einbindungsmöglichkeiten von geographischen Daten wird in das Thema dieser Diplomarbeit eingeführt.

Zunächst wird der OÖ. Verkehrsverbund vorgestellt. Das bestehende Data Warehouse sowie der Weg der Daten in das bestehende Data Warehouse werden beleuchtet. Eine Analyse der neu einzubindenden Datenquellen Fahrplandaten, geographische und demographische Daten schafft einen Überblick über die Datenstruktur. Der verwendete Lösungsansatz wird mit verwandten Lösungsansätzen aus der Literatur verglichen und diskutiert.

Mittels datengetriebenen Vorgehensmodell des Dimensional Fact Models nach (Golfarelli & Rizzi, 1998) werden die bestehende Datenstruktur sowie die neuen Datenstrukturen in ein erweitertes Data Warehouse übergeführt. Die erforderlichen ETL-Prozesse für das Importieren der zusätzlichen Datenquellen werden mit SQL-Server-Integration-Services (SSIS) entwickelt. Am Ende zeigt ein Testsystem übersichtlich die Ergebnisse einer Beispielanalyse.

1 Einleitung

Diese Arbeit beschäftigt sich mit der Einbindung von neuen Datenquellen in das bestehende Data Warehouse des Unternehmens „Oberösterreichischer Verkehrsverbund“ (OÖVV). Dazu wird im ersten Abschnitt das Themengebiet erläutert und der OÖ. Verkehrsverbund vorgestellt. Der nächste Abschnitt beschäftigt sich mit den Grundlagen eines Data Warehouse. In weiteren Abschnitten werden die Istsituation im OÖVV erfasst, die Problemstellung ausformuliert und verwandten Lösungen aus der Literatur gegenübergestellt.

Ein eigener Lösungsweg wird in den folgenden Abschnitten beschritten, um das bestehende Data Warehouse des OÖVV mit unternehmensinternen Fahrplandaten sowie unternehmensexternen geographischen und demographischen Daten zu erweitern und diese zu importieren. Die Lösungsergebnisse werden im Anschluss daran dargestellt. Ein Fazit schließt die Arbeit ab.

1.1 Einführung in das Themengebiet dieser Arbeit

Der Oberösterreichische Verkehrsverbund (OÖVV) betreibt seit mehreren Jahren ein Data Warehouse zur Analyse der Fahrscheindaten. Die Fahrscheindaten werden derzeit gesammelt und einmal jährlich in das Data Warehouse eingespeist. Die Daten werden auf Plausibilität geprüft, indem die Fahrstrecke mit dem Fahrplan abgeglichen und der tatsächlich verrechnete Fahrpreis mit dem theoretischen Fahrpreis abgeglichen wird. Die analysierten Daten (Verkaufszahlen für die angebotenen Verbindungen) werden internen und externen Empfängern zur Verfügung gestellt.

Seit geraumer Zeit gibt es nun Anforderungen seitens des Landes Oberösterreich oder der Gemeinden, die Analyseergebnisse mit geographischen, demographischen und Fahrplandaten in Verbindung zu setzen und dadurch eine bessere Aussagekraft der Ergebnisse zu erhalten. Daher gibt es Bemühungen, diese Daten in das Data Warehouse einzubinden und mit den derzeitigen Fahrschein-Verkaufsdaten zu verknüpfen. Damit soll der Grundstein für kommende Analyseanforderungen gelegt werden.

Zu den bereits ausgewerteten Fahrschein-Verkaufsdaten wird zunächst die Auswertung der Fahrplandaten gewünscht. Damit sollen sich Auswertungen über die Auslastung und das Angebot an öffentlichen Verkehrsverbindungen ableiten lassen.

Zusätzlich werden geographische und demographische Daten eingebunden, um das Fahrplanangebot und die Verkaufsdaten den Bevölkerungsdaten gegenüberstellen zu können und dem Verkehrsverbund damit die Möglichkeit zu geben, gesetzte Verkehrsmaßnahmen in Form von Fahrangeboten zu untermauern und Entscheidungen stichhaltig zu belegen.

1.2 Zielbeschreibung

Ziel dieser Arbeit ist es, das bestehende Datenmodell so zu erweitern, dass die zusätzlichen Daten eingebunden werden können und Analysen über diese Daten generiert werden können.

Die zusätzlichen Daten sind:

- die Fahrplandaten
- die Flächen (Flächenwidmungspläne aus den geographischen Daten)
- die Bevölkerungszahlen (aus den demographischen Daten)

Die Modellierung der Daten hat so zu erfolgen, dass die nachstehenden beiden Abfragen hinreichend beantwortet werden können.

- Wie verteilen sich die verkauften Fahrscheine auf Regionen/Zonen unter Berücksichtigung der Bevölkerung pro Fläche (spezifischen Bevölkerungsdichte)?
- Wie angemessen ist das Fahrplanangebot bezogen auf die Bevölkerung pro Fläche (spezifischen Bevölkerungsdichte)?

Die Ladeprozesse für die Daten sind so zu gestalten, dass diese nach dem Starten keinen externen Benutzereingriff erfordern. Das Testsystem ist zu erstellen und mit einem Ausschnitt aus Echtdaten zu testen, um die beiden Fragestellungen zu beantworten.

1.3 Der OÖ Verkehrsverbund als Auftraggeber

Der OÖ Verkehrsverbund wurde im Jahre 2000 als hundertprozentige Tochtergesellschaft des Landes Oberösterreich gegründet. Basis für diese Gründung war das Gesetz für Ordnung des Öffentlichen Personennah- und Regionalverkehrs (ÖPNRV-G 1999). In einem weiteren Schritt ging der Verkehrsverbund 2005 in das Eigentum der Oberösterreichischen Landesholding über (Verkehrsverbund, OOE-Verkehrsverbund - Wir über uns, 2011).

Als sein Sinn und Zweck wird vom (Verkehrsverbund, OOE-Verkehrsverbund - Wir über uns, 2011) angegeben *„…auf Basis der planerischen Vorgaben der Gebietskörperschaften (Land OÖ, Gemeinden) die entsprechenden Bestellvorgänge (Tarif- und Leistungsbestellungen) nach den gesetzlichen Vorgaben abzuwickeln, die Durchführung der bestellten Verkehrsdienste durch die Verkehrsunternehmen zu überwachen und sie einer laufenden Evaluierung zu unterziehen.“*

Das Ziel der Tätigkeit des OÖ Verkehrsverbunds wird mit einem qualitativ und quantitativ optimalen und bedarfsgerechten Angebot im öffentlichen Personennah- und Regionalverkehr für die oberösterreichische Bevölkerung definiert.

Um dieses strategische Unternehmensziel zu erreichen, ist es für den OÖ Verkehrsverbund wichtig, quantifizierbare Werte wie Anzahl der verkauften Fahrscheine pro Quartal, Zone und Unternehmer abzuleiten und daraus Zielgrößen (erreichte Anzahl an Fahrscheinen pro Quartal, Zone und Unternehmer) zu formulieren. Für die Überprüfung der Zielvorgaben benötigt der Verkehrsverbund ein Instrument, welches aus den im Unternehmen vorhandenen Daten diese Zielgrößen ableitet und in verschiedenen Granularitäten auswertbar macht. Damit kann gegenüber dem Eigentümer (Land OÖ) und den Kunden, in diesem Fall das Land OÖ und die

Gebietskörperschaften, die Zielerreichung verifiziert werden. Die Kenntnis von Abweichungen zwischen Zielvorgabe und Auswertungsgrößen ermöglicht die planerische Steuerung für den OÖ Verkehrsverbund.

Die strategischen Unternehmensanforderungen implizieren strategische IT-Anforderungen. Diese resultieren in einem Auswertungssystem, welches aus den bestehenden Daten Messgrößen ermittelt, die für strategische Entscheidungen verwendbar sind. Um diese Aufgabe besser erfüllen zu können, soll nun das bereits für diesen Zweck eingesetzte Data Warehouse erweitert werden.

1.4 Die Struktur des OÖ. Verkehrsverbund

Die Fläche des Bundeslandes Oberösterreich ist vom Verkehrsverbund in Zonen aufgeteilt worden. Dieses Zonenmodell beschreibt 450 Zonen, welche über ganz Oberösterreich im Bienenwabenmuster gespannt sind. Das Zonenmodell wurde in Grundzügen vom Verkehrsverbund Tirol übernommen. Abbildung 1 zeigt eine Übersicht über den Zonenplan (Verkehrsverbund, OOE-Verkehrsverbund - Zonenplan, 2011).

Die einzelnen Zonen werden intern zu Konzeptregionen und Mobilitätskreisen zusammengefasst. Es gibt in Oberösterreich derzeit 24 Konzeptregionen. Diese sind nach verkehrsspezifischen Kriterien aufgeteilt. Jede Zone gehört genau zu einer Konzeptregion. Es gibt keine Zone, welche entweder zu keiner oder zu mehreren Konzeptregionen gehört.

Die Konzeptregionen werden weiters in zehn Mobilitätskreise zusammengefasst. Hierbei ist zu erwähnen, dass es für Regionen außerhalb von Oberösterreich zwar ebenfalls Zonen gibt, die Konzeptregionen und Mobilitätskreise decken sich aber. So bilden zum Beispiel alle Zonen in Salzburg eine eigene Konzeptregion und einen eigenen Mobilitätskreis. Somit gibt es für die oberösterreichischen Zonen 19 Konzeptregionen welche weiters in 7 Mobilitätskreise zusammengefasst werden.

Es gibt in Oberösterreich zudem drei Kernzonen: Linz, Wels und Steyr. In diesen Kernzonen werden neben den regionalen Verkehrsmitteln auch örtliche Nahverkehrsmittel zusätzlich angeboten. Für das verdichtete Verkehrsangebot kommen in den Kernzonen eigene Tarife zur Anwendung. Diese werden in Form von Kernzonenzuschlägen zum eigentlichen Tarif zugeschlagen. Kernzonen können sich nur am Anfang und/oder am Ende einer Fahrt befinden. Sie können niemals in der Mitte einer Fahrt als Zwischenstation auftreten.

Die Kosten für eine Fahrt werden aus der Anzahl der zwischen dem Startort und dem Zielort durchfahrenen Zonen berechnet. Kernzonenzuschläge erhöhen gegebenenfalls den Fahrpreis für die drei Kernzonen Linz, Wels und Steyr. Eine Tabelle in den Tarifbestimmungen des OÖ Verkehrsverbunds (Verkehrsverbund, Tarifbestimmungen für den Oberösterreichischen Verkehrsverbund, gültig ab 1.1.2011, 2011) ordnet den anfallenden Fahrpreis der Anzahl der durchfahrenen Zonen für verschiedene Tarifmodelle (Kinder, Schüler, Senioren, …) zu.

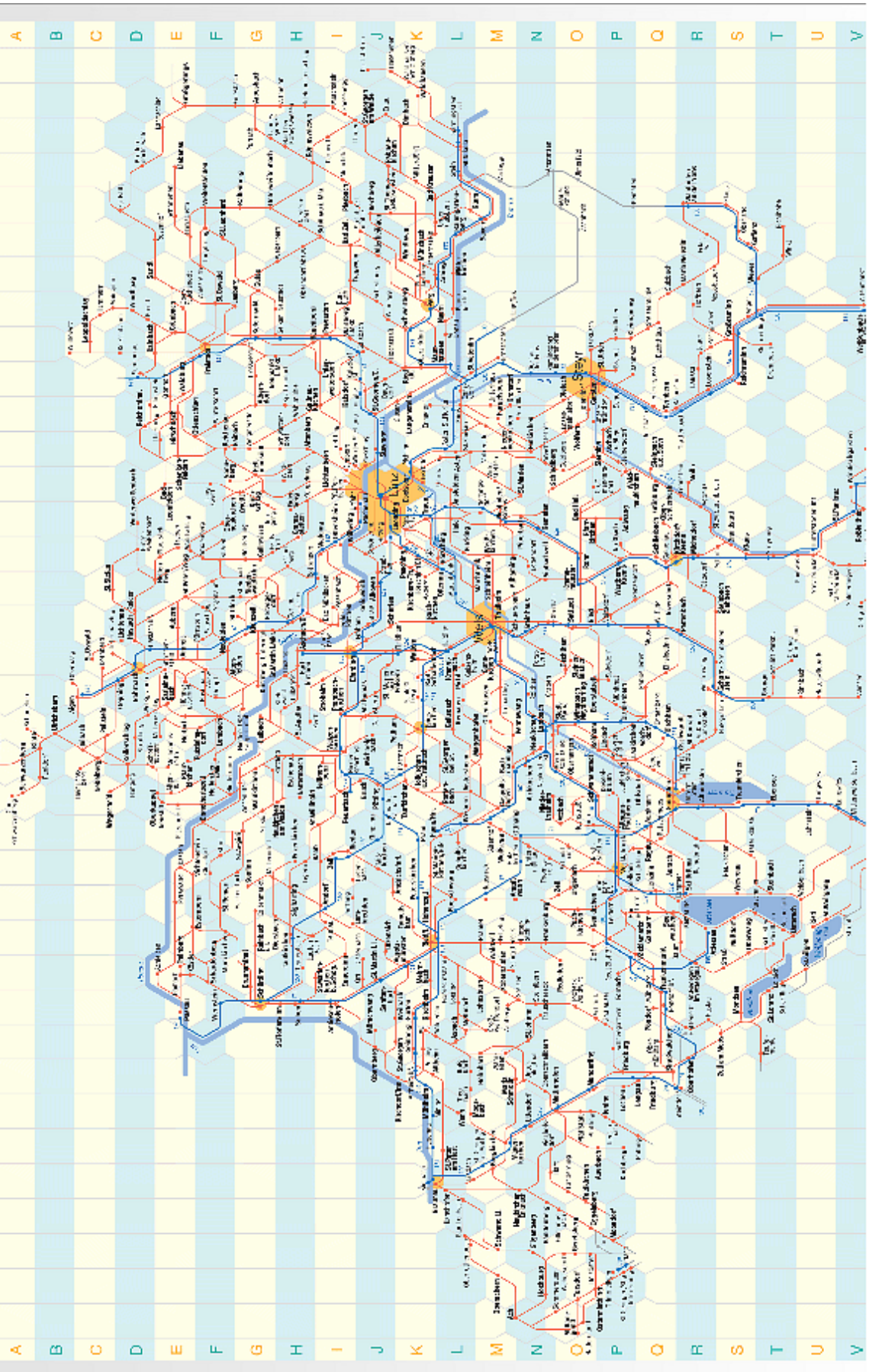

Abbildung 1: Zonenplan für Oberösterreich des OÖ Verkehrsverbundes (Verkehrsverbund 2011)

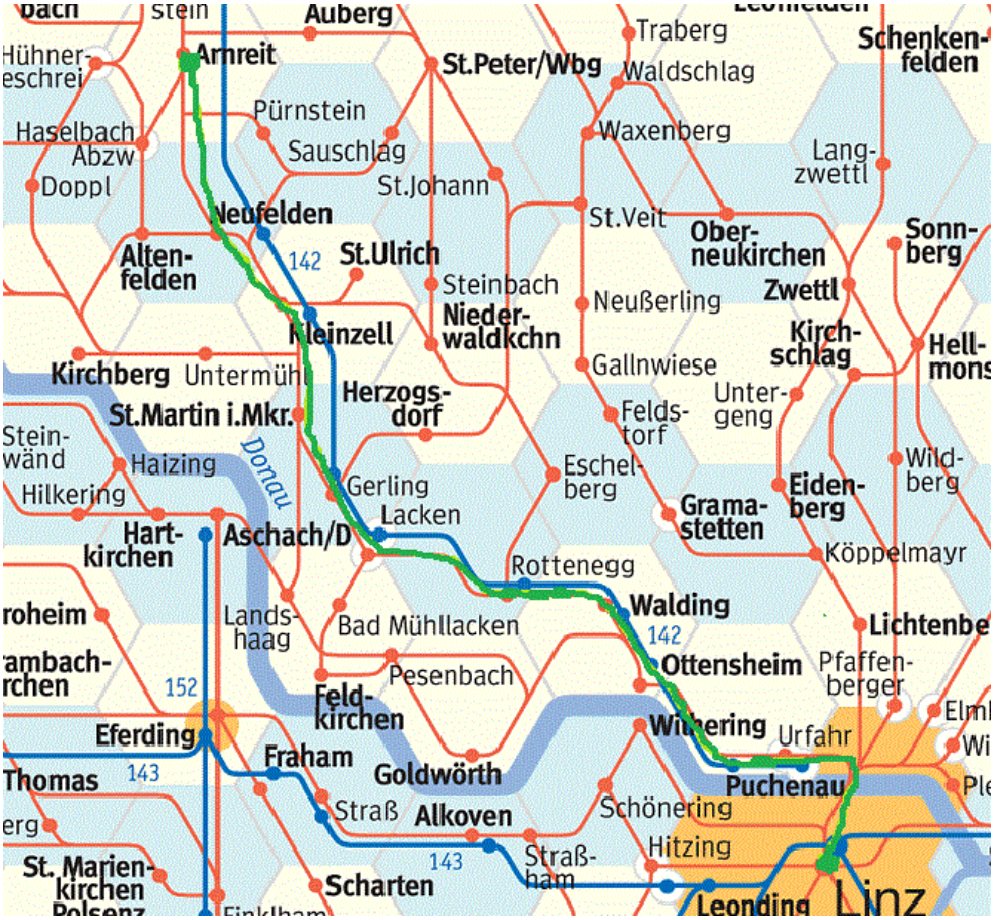

Abbildung 2: Eingetragene Fahrt von Arnreit nach Linz am Zonenplan

Als Beispiel dient die Fahrtstrecke von Linz nach Arnreit. Wie aus Abbildung 2 zu entnehmen ist, werden 8 Zonen durchfahren. Die Fahrtstrecke ist durch eine grüne Linie markiert. Aus den Tarifbestimmungen des OÖVV (Verkehrsverbund, Tarifbestimmungen für den Oberösterreichischen Verkehrsverbund, gültig ab 1.1.2011, 2011) geht hervor, dass diese Fahrt als Einzelfahrt 7,20 EUR kostet.

Der Verkehrsverbund verzeichnet zirka 10 Millionen Fahrten pro Jahr. Dies resultiert in zirka 10 Millionen Fahrscheintupeln pro Jahr für das Data Warehouse.

2 Grundlagen des Data Warehousing

Das Kapitel erläutert die Grundlagen eines Data Warehouse. Im ersten Abschnitt wird das Data Warehouse im engeren Sinne diskutiert. Der zweite Abschnitt informiert über ETL-Prozesse. Der dritte Abschnitt stellt daten- und nutzergetriebene Vorgehensmodelle zur Implementierung eines Data Warehouse vor und vergleicht diese.

2.1 Das Data Warehouse

Ein Data Warehouse ist typischer Weise in einer Umgebung mit anderen Systemen und Prozessen integriert. Wie Abbildung 3 zeigt, selektiert (Bodendorf, 2006) vier Schichten.

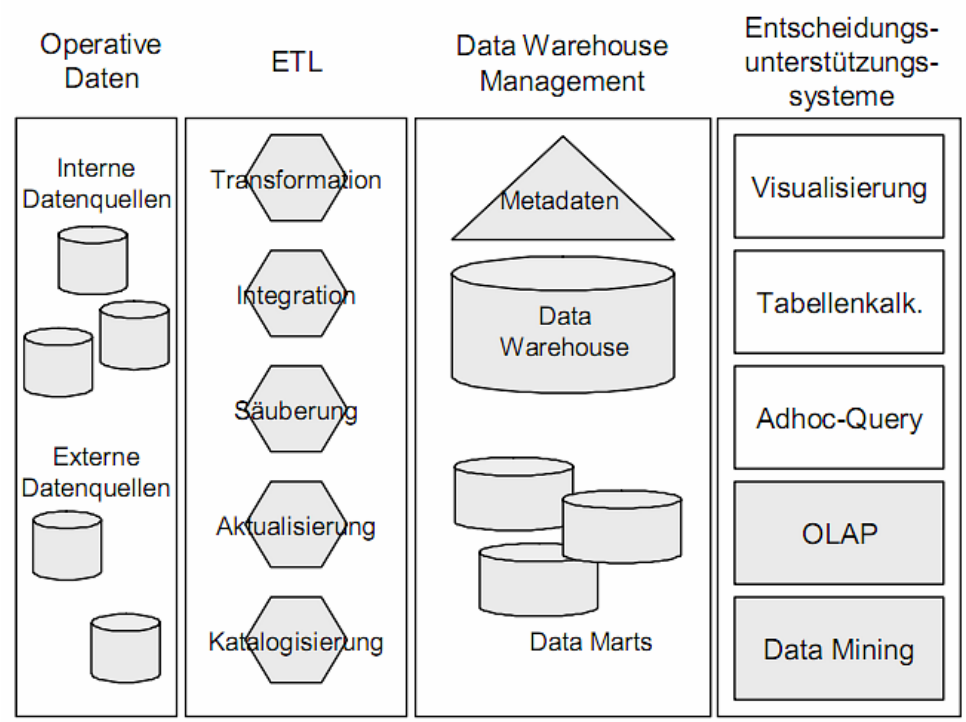

Abbildung 3: Schichtenmodell zum Data Warehouse (Bodendorf, 2006, S. 37)

Die erste Schicht beschreibt nach der Darstellung von (Bodendorf, 2006) die operativen Daten. Diese Schicht bildet die Quelle für die darüber liegenden Schichten. Die operativen Daten werden für den täglichen Geschäftsfall angewendet. Für eine Analyse der Daten sind diese meist nicht vorgesehen.

Die zweite Schicht dominiert der ETL-Prozess. Die Abkürzung ETL steht für **E**xtrahieren, **T**ransformieren, **L**aden (Bodendorf, 2006, S. 38). In dieser Schicht werden die Daten aus den Quellen ausgelesen (Extrahieren), an die notwendigen Anforderungen im Data Warehouse angepasst (Transformation) und dann in das Data Warehouse geladen (Laden). Diese für den Datenfluss wichtige Schicht hat die Aufgabe, die Daten in die für das Data Warehouse geeignete Form zu bringen.

In der dritten Schicht, dem Data Warehouse, werden die Daten gespeichert und die Abfragen des übergeordneten Systems abgearbeitet. Hier gibt es Ausführungen, in denen kleinere zusammenhängende Einheiten in so genannte Data-Marts ausgelagert werden. (Bodendorf, 2006) erkennt darin einen Zusatznutzen während der Entwicklung des Data Warehouse. (Bodendorf, 2006) folgt damit der Ansicht von Kimball, dass Data-Marts eine schrittweise Implementierung eines Gesamtsystems ermöglichen (Kimball, 1996).

Dazu ist es erforderlich, zwischen den beiden Konzepten abhängige und unabhängige Data-Marts zu unterscheiden. (Bauer & Günzel, 2009) begründen den Unterschied mit dem Zugriff des Data-Marts auf die Daten des zugrundeliegenden Data Warehouse. Während unabhängige Data-Marts isolierte Sichten der Daten speichern, werden in einem abhängigen Data-Mart Extrakte aus dem Datenbestand des Data Warehouse abgelegt.

Die vierte Schicht gehört den Entscheidungsunterstützungssystemen. Wie die Abbildung 3 zeigt, gibt es eine Vielzahl an verschiedenen Möglichkeiten und Werkzeugen, auf das Data Warehouse zuzugreifen. Unter anderen findet sich auch OLAP (On-Line Analytical Process) unter den Werkzeugen. Auf OLAP wird in Kapitel 2.1.5 noch weiter eingegangen.

2.1.1 Der Begriff Data Warehouse

Data Warehousing ist ein Begriff, welcher nun seit einigen Jahren auftaucht, wenn aus Nutzdaten Information gewonnen werden soll. Doch vielen der Anwender ist oftmals nicht klar, was der Begriff Data Warehousing wirklich heißt, was dieser aussagt und wie der Begriff abzugrenzen ist. (Bauer & Günzel, 2009, S. 5) zeigen auf, dass die unterschiedliche Interpretation des Begriffs Data Warehouse auf zwei grundlegende Bereiche zurückgeht. Hier ist als erstes der technische Bereich zu nennen. Dieser beschreibt die Datenintegrationsmöglichkeiten und die dahinterliegenden Datenbanken. Zweitens gibt es auch eine betriebswirtschaftliche Sicht, welche ein Data Warehouse als Analysewerkzeug und Werkzeug zur Generierung neuer Information sieht.

Eine Lösung wird von (Bauer & Günzel, 2009, S. 5) nur in der Kombination dieser beiden Eigenschaften gesehen. Ein Data Warehouse ist ein Werkzeug, welches aus verschiedenen Datenquellen Daten integriert und dem Anwender in einheitlicher Form zur Verfügung stellt.

Der Begriff Data Warehouse wird seit Anfang der neunziger Jahre verwendet (Humm & Wietek, 2005). (Inmon & Kelley, 1993) haben aber erst durch die Vereinfachung des Zugangs zum Data Warehouse eine Verbreitung dieser Technologie ermöglicht. Daher wird Inmon auch gerne als „Vater“ des Data Warehouse bezeichnet. (Zeh, 2003) erweitert die Begrifflichkeit und streicht damit die Einschränkung auf die nur entscheidungsrelevante Managementebene.

Ein Data Warehouse wird von (Inmon & Kelley, 1993) folgendermaßen spezifiziert:

> „*A data warehouse is a subject-oriented, integrated, time-variant, nonvolatile, collection of data, in support of management's decision-making process*“ (Inmon & Kelley, Rdb - VMS: Developing a Data Warehouse, 1993)

Diese Spezifikation beinhaltet die wichtigsten Grundsätze eines Data Warehouse. Die Subjektorientierung sagt aus, dass sich Daten nach dem Subjekt, also nach verschiedenen bewertbaren Elementen (Entitäten des Geschäftsbereichs/Geschäftsmodells) richten. Als klassische Beispiele dienen Kunden, Aufträge, Bestellungen, Produkte und so weiter.

Unter Integrierung oder auch Vereinheitlichung verstehen (Inmon & Kelley, 1993) die Homogenisierung der Daten, damit diese einheitliche Metadaten und einheitliche Granularität aufweisen. Hier kann eine Bevölkerungszahl als Beispiel herangezogen werden. Eine Angabe für eine Bevölkerungszahl ist zum Beispiel in Integer (Ganzzahl) gespeichert und bezieht sich auf einen bestimmten Ort. Damit neue Bevölkerungszahlen integriert werden können, müssen diese ebenfalls im Integerformat vorliegen und sich auf die Granularitätsstufe Ort beziehen.

Mit der Zeitorientierung werden Analysen über längere Zeiträume ermöglicht. Erst damit machen Auswertungen auf Basis eines Data Warehouse Sinn, da dadurch Trends ableitbar und

unterschiedliche Zeitperioden vergleichbar werden. Die Auswertung erfolgt mit Data Mining-Werkzeugen. Für weiterführende Information zu Data-Mining wird auf die einschlägige Literatur verwiesen.

Ein Data Warehouse dient zur langfristigen und dauerhaften Speicherung der Daten („non-volatile“). Daten werden in ein Data Warehouse normalerweise eingefügt und gespeichert, jedoch nicht gelöscht. Damit wird ein Zugriff auf historische Daten möglich, was die Auswertung langfristiger Trendanalysen zulässt.

In jüngerer Zeit taucht weiters der Begriff „Business Intelligence“ auf. Dieser wird in der Literatur oft kontroversiell betrachtet. (Bauer & Günzel, 2009) stellen den Begriff Business Intelligence als Erweiterung zum Begriff Data Warehouse dar. Zum eigentlichen Data Warehouse werden mit dem Begriff Business Intelligence auch eine um Strategien, Prozesse oder auch Technologien erweiterte Integration und die Ausweitung der Analysewerkzeuge auf Wissensmanagement gezählt. Das Wissensmanagement deckt in diesem Zusammenhang Fragen um den derzeitigen Wissensstatus, mögliches verstecktes Wissenspotential oder Zukunftsperspektiven ab.

Damit umfasst der Begriff Business Intelligence ein größeres Spektrum und der Begriff Data Warehouse stellt den wichtigsten Aspekt darunter dar.

(Kemper, Mehenna, & Unger, 2006, S. 8) definieren Business Intelligence als einen *„integrierten unternehmensspezifischen IT-basierten Gesamtansatz“*. Sie trennen dabei strikt die Bezeichnung Business Intelligence von der Verwendung für Werkzeuge, da auf dem Markt erhältliche Werkzeuge nur einen Teilaspekt aus der Gesamtheit von Business Intelligence darstellen.

Die Struktur eines Data Warehouse kann, wie Abbildung 4 zeigt, als Baum dargestellt werden. Hier sieht man den Aufbau und die Zusammenhänge zwischen den einzelnen Elementen. Diese werden nun in weiterer Folge näher erläutert.

Der Würfel setzt sich aus mehreren Datenzellen zusammen. Eine Datenzelle, auch Fakt genannt, beinhaltet ein oder mehrere Kennzahlen (Measures). Eine Datenzelle wird durch die Dimensionen referenziert. Die Auswahl einer bestimmten Datenzelle erfolgt durch Bestimmung einzelner Werte in allen Dimensionen. Geht man von einer dreidimensionalen Darstellung aus, so ergibt sich daraus ein Würfel. Werden mehr als drei Dimensionen verarbeitet, so lässt sich dies graphisch nur mehr durch Anordnen von mehreren Würfeln darstellen. Ab einer bestimmten Anzahl von Dimensionen ist eine graphische Darstellung nicht mehr sinnvoll.

2.1.2 Begriffe aus dem Data Warehouse

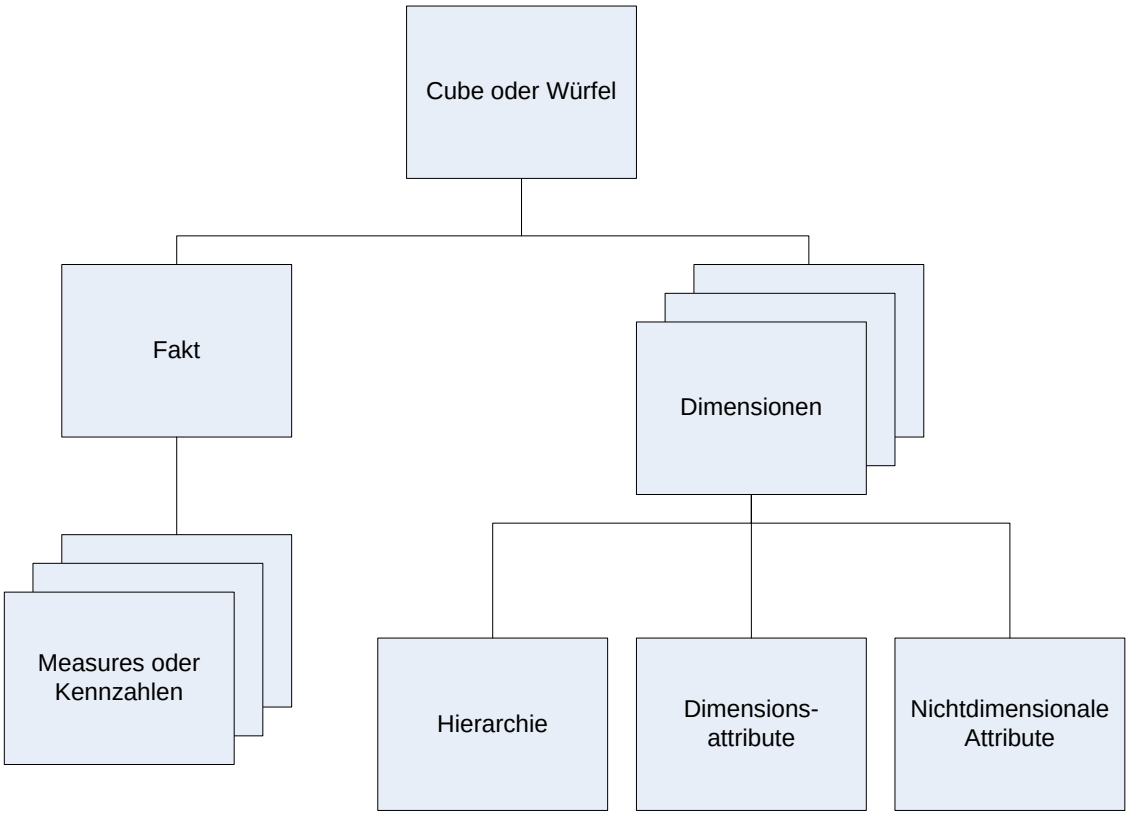

Abbildung 4: Strukturierung des multidimensionalen Schemas eines Data Warehouse

Daten werden zur Analyse oft nach bestimmten Vorgaben (Attributen) gruppiert und berechnet. Die Darstellung nach zwei unterschiedlichen Vorgaben kann leicht in Tabellen dargestellt werden. Hierbei wird die erste Vorgabe auf der waagrechten Achse, die zweite Vorgabe auf der senkrechten Achse dargestellt, sodass eine Matrix entsteht. In die entstehenden Spalten werden die den Vorgaben entsprechenden Werte eingetragen. Die Vorgaben werden auch Dimensionen genannt. In diesem Fall spricht man von einer zweidimensionalen Darstellung. Ab der dritten Dimension spricht man von multidimensionalen Darstellungen.

(Bauer & Günzel, 2009, S. 109) beschreiben eine Dimension als *„eine ausgewählte Entität[…], mit der eine Auswertungssicht eines Anwendungsbereichs definiert wird und die der eindeutigen orthogonalen Strukturierung dient“*. Die Ausprägungen der Dimensionen werden in so genannten Dimensionstabellen gespeichert. (Bauer & Günzel, 2009, S. 189) beschreiben eine Dimension auch als ausgewählte Entität in einem multidimensionalen Datenmodell, welche eine Auswertungssicht eines Anwendungsbereiches definiert.

Hierarchien beschreiben die Gliederung der Dimensionen in verschiedene Aggregationsstufen. Dies ermöglicht die Betrachtung von Werten in unterschiedlich stark zusammengefassten Sichten. Ein Beispiel ist die Aggregation von Produkten in Produktgruppen und weiter in Kategorien.

Als Dimensionsattribute werden jene Attribute bezeichnet, welche für eine Referenzierung der Dimension aus der Faktentabelle oder für die Gliederung der Hierarchie erforderlich sind.

Im Gegensatz dazu gibt es in Dimensionen nichtdimensionale Attribute („Dekorationen“), welche nur informellen Charakter besitzen.

Die Faktentabelle speichert die Werte für die Spalten, auch Kennzahlen genannt, in der oben erwähnten, durch Dimensionen gebildeten Matrix. Damit diese Werte entsprechend des oben beschriebenen Modells zu den Dimensionen zugeordnet werden können, beinhaltet die Faktentabelle auch Referenzen in Form von Fremdschlüsseln zu den oben beschriebenen Dimensionen.

Als Measures bezeichnet man die Kennzahlen innerhalb der Faktentabelle. Mit der Aggregation der Kennzahlen über die Dimensionen werden aus den Measures Summenwerte, Zählwerte, Mittelwerte oder ähnliches berechnet. In den Measures sind jene Werte abgespeichert, welche die für den Benutzer wertvolle relevante Information darstellen.

2.1.3 Das multidimensionale Modell

Ein multidimensionales Modell unterscheiden (Bauer & Günzel, 2009) von üblichen Datenbankmodellen, weil es der Denkweise des Menschen mehr entspricht als herkömmliche Datenbankanwendungen. Die Möglichkeit, Abstrahierungen zu bilden und damit Daten auf wenige Kennzahlen zu verdichten, oder auf der anderen Seite in die Tiefe zu gehen und bestimmte Ausschnitte zu detaillieren, verschafft Analysten auf einfache Weise jene Information, welche für diese nützlich und erforderlich ist.

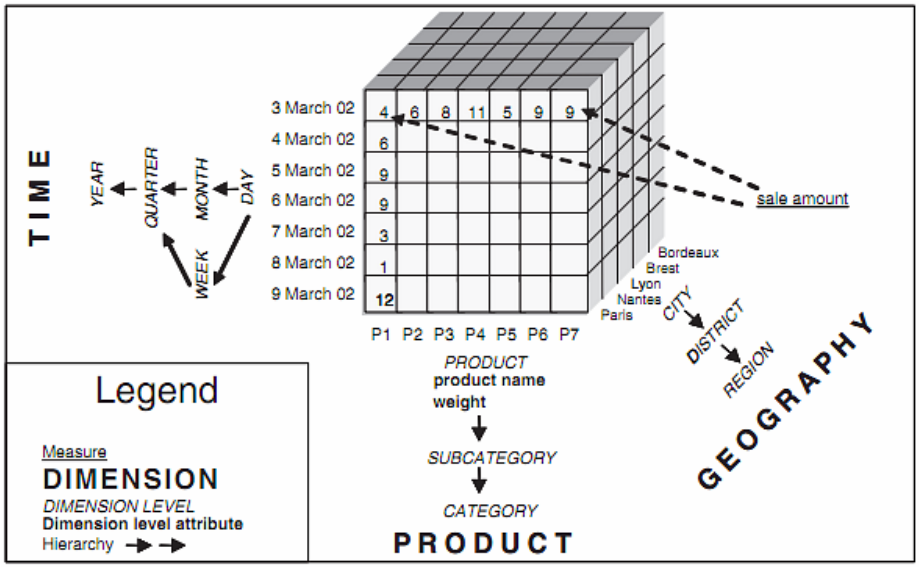

Abbildung 5: Graphische Darstellung des multidimensionalen Modells (Prat, Akoka, & Comyn-Wattiau, 2006)

Abbildung 5 zeigt beispielhaft eine graphische Darstellung eines multidimensionalen Modells als Würfel mit drei Dimensionen. Die drei Dimensionen „`Time`“, „`Product`“ und „`Geography`“ entsprechen den Kanten des Würfels.

Das multidimensionale Modell wird nun anhand eines einfachen Beispiels formalisiert (Bauer & Günzel, 2009). Dazu wird folgendes Szenario eingeführt.

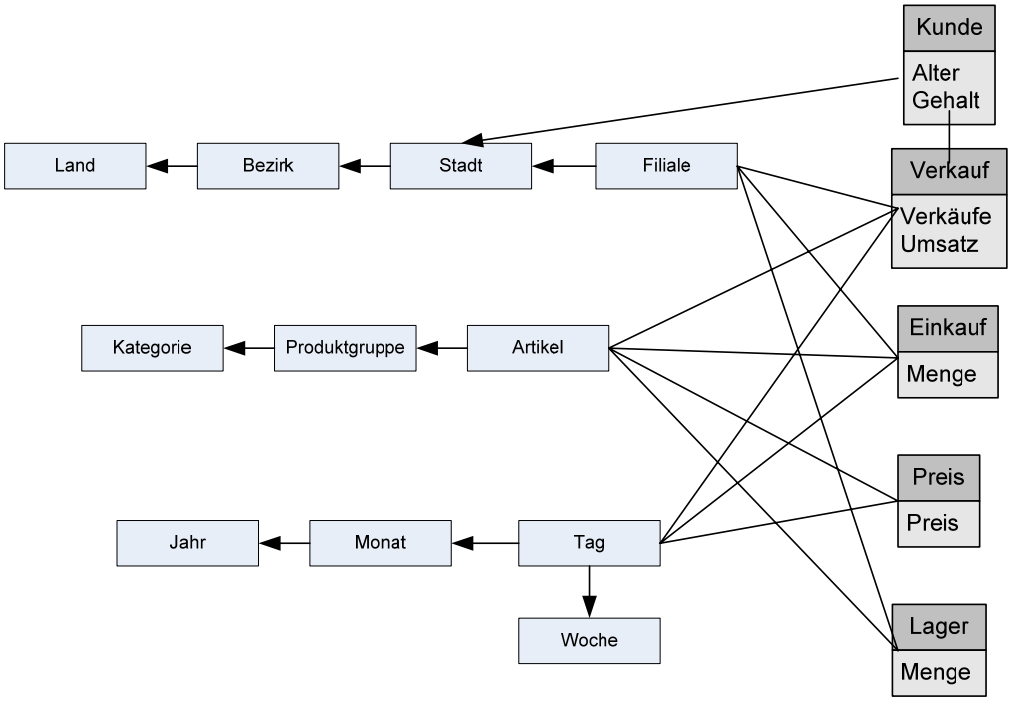

Abbildung 6 Multidimensionales Beispielschema (Bauer & Günzel, 2009)

Das multidimensionale Modell in Abbildung 6 stellt ein Unternehmen mit mehreren Filialen dar, in welchen Produkte an Kunden verkauft werden. Die Filiale verfügt über ein Lager, in welchem die Waren gelagert und anschließend zu einem bestimmten Preis verkauft werden. Wie ersichtlich ist, stellen die Entitäten `Filiale`, `Artikel` und `Tag` die Dimensionen des Modells dar. Die über den Entitäten angesiedelten Hierarchien wie Produktgruppe oder Kategorie sind ebenfalls den Dimensionen zuzuordnen, da diese eine Aggregation der zugrunde liegenden Dimensionen darstellen.

Die hierarchische Abstufung der dimensionalen Bausteine erfolgt durch Klassifikationsstufen (Bauer & Günzel, 2009, S. 189ff).

In einem Klassifikationsschema werden die Abhängigkeiten zusammengefasst als funktionelle Abhängigkeiten beschrieben. Ein Attribut A impliziert ein Attribut B (Formel 1).

$$A \rightarrow B$$

Formel 1: Funktionelle Abhängigkeit von Dimensionen

Korrekterweise ist zu berichtigen, dass Klassifikationsschemata nur halbgeordnete Mengen darstellen, da diese erst als so genannter Pfad als geordnete Mengen auftreten. Dazu wird die Klassifikationsstufe formal um das Klassifikationsschema der Dimension, dargestellt in Formel 2, erweitert.

$$D.K_1 \rightarrow D.K_2$$

Formel 2: Klassifikationsschema Dimension

Ein Pfad ist die vollständige Auflistung aller Klassifikationsstufen. Legt man dies nun um auf die oben dargestellten Hierarchien, so ergibt sich für die Dimension `Filialen` der in Formel 3 dargestellte Pfad.

$$F.Filiale \rightarrow F.Stadt \rightarrow F.Bezirk \rightarrow F.Land$$

Formel 3: Pfad der Klassifikationsstufen für Beispieldimension `Filialen`

Die Dimension `Zeit`, bestehend aus `Tag`, `Monat`, `Jahr` und `Woche` weist eine Besonderheit auf, welche in Formel 4 dargestellt wird.

$$Z.Tag \rightarrow Z.Monat \rightarrow Z.Jahr$$

$$Z.Tag \rightarrow Z.Woche \rightarrow Z.Jahr$$

Formel 4: Pfad der Klassifikationsstufen für Beispieldimension `Zeit`

Wie bei der Zeitdimension zu sehen ist, gibt es zwei parallele Hierarchien. Wochenwerte können nicht einfach in Monatswerte umgewandelt werden. In der obersten Klassifikationsstufe treffen sich beide Hierarchien wieder, da die Summe der Wochen wieder in Jahren repliziert werden kann.

Mittels Klassifikationshierarchie kann dann aus dem Pfad ein Baum dargestellt werden. Die Kanten des Baumes stellen die Dimensionshierarchie dar. Der Baum ist grundsätzlich ausbalanciert, d.h. entlang aller Kanten gibt es die gleiche Anzahl an Hierarchien (Bauer & Günzel, 2009, S. 190ff).

In den Blättern des Baumes treffen sich die verschiedenen Dimensionen auf den Achsen des multidimensionalen Raumes und bilden einen Berechnungspunkt. In diesem Berechnungspunkt ist die einzelne Datenzelle angeordnet. Darin existieren dann die Kennzahlen der Faktentabelle. Die Gesamtheit der Darstellung wird Würfel oder in englischer Übersetzung „Cube“ genannt.

Auch in den inneren Knoten des Baumes bilden sich Berechnungspunkte mit anderen Dimensionen. Hier werden die darunterliegenden Blätter nach der Formel 5 aggregiert.

$$\{D.K_1 \dots D.K_n\}$$

Formel 5: Aggregation der Daten in Knoten

(Bauer & Günzel, 2009) definieren das Würfelschema W mit der Granularität G und der Menge M, den Kenngrößen. Das Würfelschema W besteht aus einer Menge von Würfelzellen. Diese berechnet sich nach der in Formel 6 dargestellten Funktion.

$$W \subseteq G(z) \times M$$

Formel 6: Formel zur Berechnung des Würfels

Teilmengen werden durch die Koordinaten respektive den Dimensionen definiert, welche der Funktion in Formel 7 folgen.

$$z \in W$$

Formel 7: Einfluss der Dimensionen auf den Würfel

Aus der Darstellung in Abbildung 6 ergeben sich dann in meinem Beispiel-Szenario die nachstehenden Abhängigkeiten.

- `Verkauf` ((`Artikel`, `Tag`, `Filiale`, `Kunde`), (`Verkäufe`, `Umsatz`))
- `Einkauf` ((`Artikel`, `Tag`, `Filiale`), (`Menge`))
- `Preis` ((`Artikel`, `Tag`), (`Preis`))
- `Lager` ((`Artikel`, `Woche`), (`Menge`))

Damit ist aus den Formeln ableitbar, dass eine Veränderung einer Dimension D eine Veränderung des Würfelschemas W impliziert. Über die in Formel 5 dargestellte Aggregation werden dann die entsprechenden Werte für den Verkauf, den Einkauf, den Preis oder auch das Lager gebildet.

Auf eine weitere theoretische Abhandlung wird verzichtet. Für eine weitere Vertiefung wird auf die einschlägige Literatur verwiesen.

2.1.4 Speicherung des multidimensionalen Schemas

Das oben besprochene multidimensionale Schema stellt ein konzeptuelles Schema dar, das nicht direkt in einem Informationssystem implementierbar ist. Daher muss eine Übersetzung in ein logisches und dann in ein physisches Schema erfolgen.

Es gibt verschiedene Ansätze zur Abbildung der Daten. Man findet in der Literatur unter anderen die Begriffe ROLAP (Relational OLAP = Relational On-Line Analytical Processing), MOLAP (Multidimensional OLAP) oder auch HOLAP (Hybrid OLAP) (Bauer & Günzel, 2009) (Azevedo, Brosius, Dehnert, Neumann, & Scheerer, 2009). Diese Aufzählung stellt keinen Anspruch auf Vollständigkeit. Weitere Modelle wie SOLAP (Spatial OLAP) (Rivest, Bedard, Proulx, Nadeau, Hubert, & Pastor, 2005) oder O3LAP (Object-Oriented OLAP) (Buzydlowski, Song, & Hassell, 1998) sind in der Literatur bekannt.

Weiters könnte man noch die beiden Modelle DOLAP (Desktop OLAP) und FFOLAP (Flat-File OLAP, d.h. OLAP ohne Datenbank im Dateisystem) erwähnen. Diese stellen aber nur vereinfachte Ausführungen für kleinere Datenmengen dar. Ein eher neueres Modell wurde im Jahre 2002 von (Reuven Bakalash, Guy Shaked, & Joseph Caspi, 2002) patentiert. Dieser MDDB-OLAP (Multidimensional Database OLAP) genannte Ansatz basiert auf einer relationalen Datenbank mit innerem multidimensionalen Kern für Aggregationen.

Objektorientierte Ansätze werden in der Wissenschaft seit längerem diskutiert. Viele der Publikationen wurden in den Jahren 1998 bis 2000 geschrieben. (Huynh, Mangisengi, & Tjoa, 2000), (Buzydlowski, Song, & Hassell, 1998), (Vivekanand, L., & Kamalakar, 1999) und (Ravat & Teste, A Temporal Object-Oriented Data Warehouse Model, 2000) können stellvertretend genannt werden. Im objektorientierten Modell werden anstatt der Kennwerte in Faktentabellen oder Dimensionen objektorientierte Klassen – welche Dimensionen darstellen – eingefügt. Die Klassen können komplexere Datentypen halten und kapseln die Daten und Instanz-Methoden (Buzydlowski, Song, & Hassell, 1998, S. 11). Damit können nicht nur Zahlenwerte oder Zählwerte verarbeitet werden. Da aber für die Analyse und Entscheidungsfindung die Daten und Fakten den Ausschlag geben, sind andere nicht standardmäßig aggregierbare Kennwerte eher die Ausnahme. Es gibt aber mögliche Anwendungsfelder für diesen interessanten Ansatz. Die Autoren nennen als Beispiel Anwendungen im medizinischen Bereich mit Röntgenbildern als Fakten.

In diesem Kapitel werden die beiden Modelle ROLAP und MOLAP gegenübergestellt, da diese derzeit in der Praxis am gebräuchlichsten sind. Zur Abrundung wird noch auf das Mischsystem HOLAP eingegangen.

Das relationale Speicherkonzept – ROLAP – wird von (Azevedo, Brosius, Dehnert, Neumann, & Scheerer, 2009) mit der Speicherung der Daten in einer relationalen Datenbank beschrieben. Nur die multidimensionalen Definitionen werden im Cube gehalten. Aggregierte Kennzahlen werden ebenfalls vorab berechnet und in dafür angelegte Tabellen im Hintergrund auf das relationale Datenbankschema abgelegt.

(Bauer & Günzel, 2009) diskutieren den Begriff der relationalen Speicherung dagegen intensiver. Dabei werden Daten in Tabellen mit relationalem Bezug zueinander abgespeichert. Die Autoren gehen insbesondere auf die Kriterien zur optimalen Umsetzung ein. Dabei sollte möglichst wenig Semantik bei der Umwandlung von multidimensionalen Daten auf relationale Daten verloren gehen. Gefährdet sehen (Bauer & Günzel, 2009) die Klassifikationshierarchien, speziell in Verbindung mit dem Star-Schema. Das Star-Schema, auf welches in Kapitel 2.1.4 näher eingegangen wird, vereint alle Hierarchien einer Dimension in einer einzigen Dimensionstabelle. Die Hierarchien werden dadurch nur mehr über Ebenenattribute erkennbar.

Ein weiterer wichtiger Punkt ist die effiziente Übersetzung multidimensionaler Abfragen. Da diese Übersetzung mit jeder neuen Abfrage durchlaufen wird, ist besondere Vorsicht geboten, damit sich kein Engpass ergibt, welcher den gesamten Abfrageprozess nachhaltig bremst. Auch die Gegenrichtung, das heißt die Abarbeitung und die Ausgabe der Daten, ist von dieser Übersetzung betroffen.

Für die Abfrage von Daten im relationalen Modell kann die standardisierte Sprache SQL (ISO/IEC 9075-1:2008 04 14) herangezogen werden. Die Abfragen lassen sich einfach formulieren und können auch von weniger spezialisiertem Personal gewartet werden. Die Referenzierung zwischen Faktentabelle und Dimensionen erfolgt über die Angabe der Verbundbedingungen in der WHERE-Klausel. Die Aggregation wird durch die Angabe von GROUP BY Bedingungen erzielt.

Da es mit steigender Anzahl an Dimensionen immer komplexer wird, Gruppierungen für alle Aggregationsstufen anzupassen, wurden bereits mit dem Standard SQL-99 (ISO/IEC 9075-1:1999 12 01) und in Folge mit dem Standard SQL-2003 (ISO/IEC 9075-1:2003 12 15), mächtigere Befehle wie die Grouping-Sets, rollende oder gewichtete Mittelwerte eingeführt (Bauer & Günzel, 2009). Die Gruppierungsfunktionen benennt (Bodendorf, 2006, S. 43) mit „Group by grouping sets“, „Group by Rollup“ und „Group by Cube“.

Auch die Verbindung der einzelnen Dimensionstabellen mit der Faktentabelle wurde vereinfacht. Ein CUBE-Operator übernimmt alle möglichen Verbindungsvarianten der Dimensionstabellen mit der Faktentabelle und dient daher als Kurzschreibweise für den Benutzer/Anwender.

Die Vorteile des relationalen Schemas sind die ressourcenschonende Ablage der Daten und die bekannte Abfragestruktur mittels SQL. Als Nachteil ist die notwendige Übersetzung der Abfragen in das multidimensionale Schema und die Übersetzung der abgefragten Daten zurück in das multidimensionale Abfragetool zu nennen.

Für die Speicherung von Hierarchien gibt es in den relationalen Schemata mehrere Lösungen. Diese werden nach ihrer Art der Darstellung klassifiziert. Hier finden sich in (Azevedo, Brosius, Dehnert, Neumann, & Scheerer, 2009) die folgenden Schemata:

- Star-Schema
- Snowflake-Schema
- Galaxy-Schema

Allen Schemata gemein ist, dass die Faktentabelle als zentrales Element und als Ausgangspunkt der Betrachtung festgelegt wird.

Das Star-Schema stellt die einfachste Möglichkeit zur Speicherung der Dimensionen in einer relationalen Datenbank dar. Hier wird für jede Dimension nur eine Tabelle angelegt. (Bauer & Günzel, 2009) erwähnen als Vorteil die hohe Performanz des Systems sowie das einfachere und übersichtlichere Datenbankschema.

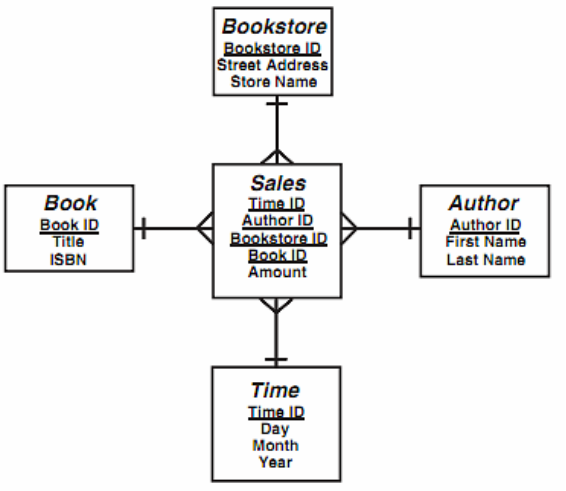

Abbildung 7: Einfaches Star-Schema (Corral, Schuff, & Louis, 2006)

(Azevedo, Brosius, Dehnert, Neumann, & Scheerer, 2009) gehen darauf ein, dass die Werte der Dimensionstabellen aber dann in denormalisierter Form vorliegen. Dies induziert die Mehrfachablage von Attributwerten und damit eine Erhöhung der Speichermenge. Das Star-Schema wird von (Levene & Loizou, 2003) auch als Spezialisierung eines Snowflake-Schemas gesehen. Der Name „Star-Schema“ rührt aus der bildlichen Darstellung des Datenbankschemas her, da dieses einem strahlenden Stern ähnelt (siehe Abbildung 7).

Im Snowflake-Schema werden die Tabellen normalisiert abgelegt (Azevedo, Brosius, Dehnert, Neumann, & Scheerer, 2009). Damit ergibt sich für jede Klassifikationsstufe in einer Dimension eine eigene Tabelle. Dies führt zu komplexeren Datenbankschemata und 1:n-Verbindungen zwischen den einzelnen zusammenhängenden Tabellen in den Dimensionen. Damit erklären (Bauer & Günzel, 2009) den Erhalt eines Fremdschlüssels für die Tabelle der höheren Klassifikationsstufe. Der Fremdschlüssel referenziert die benachbarte darunterliegende Tabelle. Der Vorteil des Snowflake-Schemas ergibt sich in der direkten Abbildung der Hierarchien aus dem konzeptuellen Schema. Weiters werden durch die Normalisierung Redundanzen in den Tupeln der Dimensionstabellen vermieden. Als Nachteil wird angeführt, dass es im Snowflake-Schema für eine Abfrage zu mehreren Verknüpfungen über die Fremdschlüssel kommen muss und damit die Komplexität der Abfragen wesentlich steigt. Der Name rührt aus der Darstellungsform des Datenbankschemas. Sie erinnert an die Form einer Schneeflocke (siehe Abbildung 8)

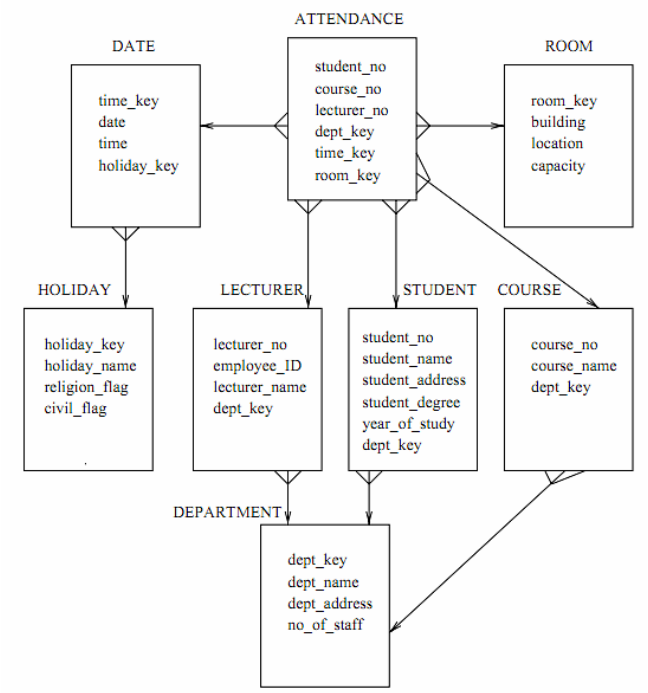

Abbildung 8: Snowflake-Schema (Levene & Loizou, 2003)

Ein Galaxy-Schema wird von (Bauer & Günzel, 2009) nach Kimball als Multifaktentabellen-Schema, Galaxie oder auch als Multi-Cube oder Hyper-Cube bezeichnet. Hier gibt es im Würfel nicht nur eine Faktentabelle. Benötigen mehrere verschiedene Faktentabellen teilweise gemeinsame Dimensionen, so ist es sinnvoll, diese Faktentabellen in einem gemeinsamen

Würfel zu projektieren. Damit können sehr einfach verschiedene Measures in einer Darstellung gegenübergestellt werden („Drill-across“). Als Grundlage für ein Galaxy-Schema kann ein Star-Schema oder auch ein Snowflake-Schema dienen. Bezeichnet wird das Schema wegen der einzelnen Faktentabelle, welche wie verschiedene Galaxien im Datenbankschema existieren.

Dieses Schema ist als Spezialfall auch unter dem Begriff Fact-Constellation-Schema (Bauer & Günzel, 2009, S. 221) zu finden. Das Fact-Constellation-Schema speichert neben der eigentlichen Faktentabelle auch aggregierte Daten als eigene Faktentabellen. Am konzeptuellen Schema ändert sich dadurch nichts, das heißt, konzeptuell bilden alle Faktentabellen im Fakt-Constellation-Schema lediglich ein einziges Fakt ab. Als Grundformen des Fakt-Constellation-Schemas dient wiederum das Snowflake-Schema oder das Star-Schema.

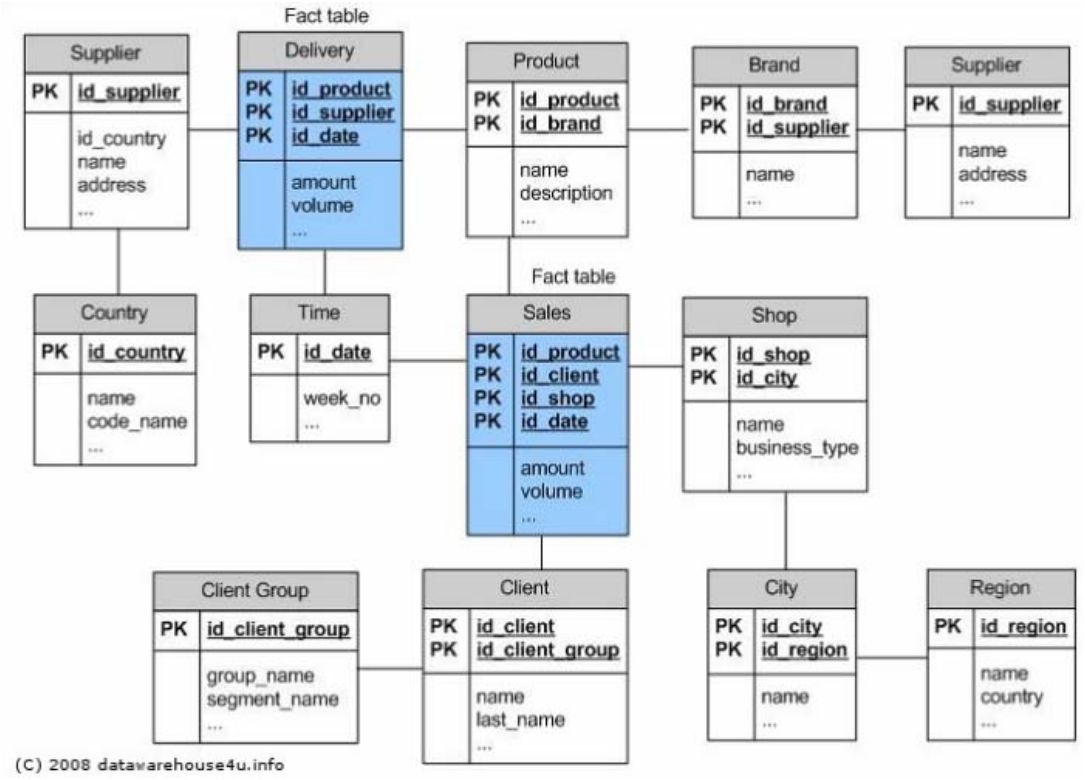

Abbildung 9: Galaxy- oder Fact Constellation Schema (Datawarehouse4u.Info, 2009)

Das multidimensionale Konzept – MOLAP – speichert die Daten direkt in multidimensionaler Form im Cube. Da eine Umwandlung wie beim relationalen Modell wegfällt, ergeben sich dadurch in der Regel Effizienzsteigerungen bei der Beantwortung von Abfragen. In multidimensionalen Speicherstrukturen wird die Speicherreihenfolge der Dimensionen festgelegt (Linearisierung der Zellen). Dadurch kann es sein, dass die Effizienzsteigerungen minimal ausfallen oder sich gar umkehren (Bauer & Günzel, 2009). In jedem Fall einfacher ist der Abbildungsprozess des konzeptuellen auf das logische Schema, da dieses 1:1 umgesetzt werden kann.

Die physikalische Abspeicherung der Daten erfolgt getrennt in Fakten und Dimensionen. Die Dimensionen werden in einem Array geordnet abgelegt. Dazu werden auch die jeweiligen Klassifikationsknoten mit abgelegt. (Bauer & Günzel, 2009) zählen die möglichen Datenty-

pen mit Text, Ganzzahl oder Datum auf. Die Dimensionen im multidimensionalen Konzept benötigen mehr Speicherplatz zum Ablegen der gleichen Datenmenge als relationale Dimensionen (Azevedo, Brosius, Dehnert, Neumann, & Scheerer, 2009).

Da Dimensionsarrays geordnet abgelegt werden, können sie auch indexiert werden. Der maximale Wert eines Datentyps der Indexwerte stellt die Begrenzung für die Menge der Ausprägungen einer Dimension dar. Damit wird der Zahlenwert entsprechend endlich begrenzt. Bei einem Integer mit 16 Bit wären das 2^{16} ergibt 65536 Ausprägungen.

Mit der Anzahl der Dimensionen wird auch die Anzahl der Fakten festgelegt. Durch Multiplikation der jeweiligen maximalen Anzahl an Elementen je Dimension ergibt sich die maximale Gesamtanzahl an adressierbaren Fakteneinträgen.

$$2^{32} x\ 2^{32} x\ 2^{32} x\ 2^{32}$$

Formel 8: Berechnung der maximalen Anzahl an Fakten für vier Dimensionen

Daher ergibt sich als Beispiel bei vier Dimensionen eine Kalkulation wie in Formel 8 dargestellt. Die Indexwerte für die Dimensionen sind für dieses Beispiel als Integer mit 32 Bit angenommen.

Für multidimensionale Modelle erwähnen (Bauer & Günzel, 2009) weiters die Unterscheidung bei der Ablegung der Fakten. So gibt es multidimensionale Würfel, welche nur ein Fakt speichern können. Sollten mehrere Fakten gespeichert werden, resultiert dies in mehreren parallel verwalteten Würfeln. Es gibt aber auch so genannte Single-Cube-Systeme, in welchen mehrere Fakten parallel abgespeichert werden.

Um die Abfragegeschwindigkeit in multidimensionalen Modellen zu erhöhen, werden Aggregationen für die jeweiligen Knoten bereits beim Laden in das Data Warehouse berechnet. Damit liegen zum Abfragezeitpunkt vorkalkulierte Werte vor, welche schnell zurückgegeben werden können. Mögliche Probleme für diese Methode treten dann auf, wenn Daten häufig in das System geladen werden. Die Berechnung der entsprechenden Aggregationen belastet das System und führt in dieser Zeit zu einer hohen Antwortzeit.

Um solchen Fällen vorzubeugen, gibt es Lösungen, um nur Teilbereiche eines Würfels vorzuberechnen und bestimmte Teile erst direkt bei der Abfrage zu rechnen. Als weitere Möglichkeit wird die Berechnung und Speicherung bei der ersten Abfrage erwähnt. Hier hat der erste Anwender mehr Wartezeit zu verbuchen, da dieser auf die Echtzeitauswertung warten muss, der Folgeanwender aber auf einen vorberechneten Wert zugreifen kann.

Wenn man die beiden vorangegangenen Modelle gegenüberstellt, ergeben sich genau gegensätzliche Stärken und Schwächen. Das multidimensionale Schema hat die Stärken in der direkten analytischen Umsetzung und bei kleineren Datenvolumen, das relationale Schema seine Stärken in der Skalierbarkeit und der Offenheit.

Das hybride Modell teilt nun die Fakten und Dimensionen auf und legt die Detaildaten in einem relationalen Modell ab, wohingegen die aggregierten Daten im multidimensionalen

Schema abgelegt werden. Damit werden die größeren Datenmengen relational, die am meisten verwendeten aggregierten Daten dagegen multidimensional gespeichert.

Der Nachteil des hybriden Modells ist die Implementierung und Modellierung, da auf beide Technologien Rücksicht genommen werden muss.

2.1.5 Online Analytical Processing

Der Begriff OLAP wird gerne in Verbindung mit dem Begriff Data Warehouse gebracht und bedeutet **O**n-**L**ine **A**nalytical **P**rocessing. Dieser Begriff beschreibt einen Ansatz zur Analyse von Daten in einem multidimensionalen Umfeld. Der OLAP-Ansatz umfasst 18 Kriterien (Kemper, Mehenna, & Unger, 2006, S. 57), (Bauer & Günzel, 2009, S. 105), an denen Werkzeuge für ihre Eignung in Bezug auf die Auswertungen von Daten in einem Data Warehouse bewertet werden können. Die Kriterien werden zu Gruppen zusammengefasst. Nachstehend folgt eine Auflistung der verschiedenen Kriterien:

- Grundlegende Kriterien
    1. Mehrdimensionale konzeptuelle Sichten
    2. Intuitive Datenmanipulation
    3. Vereinheitlichte Zugriffsmöglichkeit
    4. Batch-Verarbeitung
    5. Verschiedene Analyse-Modelle
    6. Client-Server Architektur
    7. Transparenz
    8. Mehrbenutzerbetrieb
- Spezielle Eigenschaften
    1. Behandlung nicht normalisierter Daten
    2. Eigene Verwaltung für geänderte Daten
    3. Berechnung fehlender Werte
    4. Differenzierung zwischen Werten und Nullwerten
- Berichtswesen - Dimensionsverarbeitung
    1. Freie Berichtgestaltung
    2. Antwortzeiten gleichbleibend
    3. Ausgleichung physischer Ungleichheiten in der Lastverteilung

4. Generische Dimensionen
5. Unbegrenzte Anzahl von Dimensionen und Ebenen
6. Unbeschränkte Join-Operationen

Diese 18 Kriterien können auch weiter zusammengefast werden unter dem Begriff „FASMI“ (Pendse & Creeth, 1995). Hierunter versteht man ein Akronym aus den Einzelworten „Fast – Analysis (of) – Shared – Multidimensional – Information“. Übersetzt man die Worte aus dem Englischen, so lässt sich der Begriff OLAP und dessen Kriterien in einem Satz zusammenfassen. OLAP bedeutet eine schnelle komplexe Analysefunktion in einem multidimensionalen Datenmodell für mehrere Benutzer gleichzeitig, wobei das OLAP-System eine uneingeschränkte Menge von Daten verwalten kann.

OLAP wird fälschlicherweise gerne mit dem Begriff Data Warehouse verwechselt oder als eins gesehen. Dabei stellt OLAP das Analysewerkzeug für die Abfragen und Verwaltung dar. Das Data Warehouse hingegen ist die Basis und stellt die Daten für OLAP zur Verfügung (Bauer & Günzel, 2009).

2.2 Datenimport in das Data Warehouse – Der ETL-Prozess

Die Grundlagen des Data Warehouse und dessen mögliche Ausprägungen wurden in Kapitel 2.1 eingehend diskutiert. Um das Data Warehouse später mittels OLAP nützen zu können, bedarf es auch der Daten aus den zugrundeliegenden Datenquellen. Ging man früher davon aus, dass Datenquellen gleich Datenbanken sind, hat man heute erkannt, dass auch abseits der Datenbanken verschiedene Datenquellen wie Excel-Tabellen, Textdateien oder Webinhalte existieren.

Der Import der Datenquellen wird in einem Prozess abgehandelt, für den sich in der Literatur die Bezeichnung ETL-Prozess etabliert hat. Die Abkürzung bedeutet Extrahieren, Transformieren und Laden. Dieser Prozess ist der zweiten Schicht in der Schichtenarchitektur (siehe Seite 19 Abbildung 3) rund um das Data Warehouse zugeordnet. Der ETL-Prozess dient zum Extrahieren der Daten aus einer Datenquelle, zum Transformieren der Daten und zum Laden der Daten in das Data Warehouse.

(Bodendorf, 2006) erkennt fünf Kategorien an zu lösenden Problemen in diesem Prozess.

- Transformation
- Integration
- Säuberung
- Aktualisierung
- Katalogisierung

Diese fünf Problemkategorien werden in den drei Phasen des ETL-Prozesses abgearbeitet. In den folgenden Unterkapiteln werden die Phasen des ETL-Prozesses näher erläutert und dabei

auf die oben genannten Probleme eingegangen, welche vor allem den Transformations- und Ladephasen zuzuordnen sind.

2.2.1 Extrahieren der Daten

Die Extraktionsphase dient zum Auslesen der verschiedenen Datenquellen. Die Entscheidung, welche Datenquellen mit welchen Ausschnitten einzulesen sind, hängt für (Bauer & Günzel, 2009, S. 85) dabei von den Fragestellungen an das Data Warehouse ab. Die Daten werden durch Transformation in eine gemeinsame Arbeitsstruktur eingelesen. Dabei werden Formate und Datentypen an die gemeinsame Arbeitsstruktur angepasst (vergleiche 2.2.2).

Der Zeitpunkt der Extraktion wird von (Bauer & Günzel, 2009, S. 86) in vier verschiedenen Ausprägungen angegeben. Erstens gibt es die Möglichkeit der periodischen Extraktion. Weiters kann die Extraktion anfragegesteuert oder ereignisgesteuert erfolgen. Hier kommt es darauf an, ob die Quelle oder der Nutzer die Extraktion aktualisierter Daten anstößt. Die vierte Ausprägung steht für eine sofortige Extraktion. Diese kann erforderlich sein, wenn die Daten im Data Warehouse sehr aktuell sein sollten.

2.2.2 Transformieren der Daten

Sind die Daten in einer gemeinsamen Struktur abgelegt, so ist es in der Transformationsphase nun erforderlich, diese Daten zu einem gemeinsamen Schema zu integrieren. Dabei werden Attributsüberlappungen oder Äquivalenzen behandelt. (Bodendorf, 2006) bringt als Beispiel die Schlüsselbeziehung Kundennummer versus Kundenname / Geburtsdatum. Weitere Anpassungen betreffen Währungsunterschiede, unterschiedliche Maßeinheiten oder auch die Konvertierung von Kodierungen. Eine der häufigsten Anpassungen sind die Formatunterschiede von Zeit und Datum. Es gibt mittlerweile sehr viele unterschiedliche Formate für die Angabe von Uhrzeit und Datum. Für die Datenintegration ist die Behebung möglicher Differenzen immer wieder eine Herausforderung. (Bauer & Günzel, 2009) legen den Schwerpunkt für die Transformationsphase auf Integration und Transformation von Daten.

Nachdem die einzelnen Datenfelder abgeglichen wurden, werden in einem nächsten Schritt die Daten von nicht benötigten Attributen oder leeren Tupeln gesäubert. (Bauer & Günzel, 2009) gliedern die Bereinigung nach den folgenden Datenqualitätsmerkmalen:

- Korrektheit
- Konsistenz
- Vollständigkeit
- Redundanzfreiheit
- Einheitlichkeit

Die einfachste Variante der Überprüfung der Korrektheit von Daten ist die Abgleichung mit vordefinierten Grenzwerten. Sind die Grenzwerte überschritten, so kann der Wert mit einem festen Wert oder einem berechneten Wert aus der Menge der guten Proben ergänzt werden. Hier gibt es viele Methoden der Überprüfung und der Korrektur von fehlerhaften oder fehlen-

den Werten. Im Bereich der Statistik und Qualitätssicherung wurden bereits gute Erfahrungen (Bergamaschi, Guerra, Orsini, Sartori, & Vincini, 2011, S. 5) im Umgang mit inkorrekten Werten gesammelt.

Konsistenz kann nur durch das Wissen von Zusammenhängen in den Daten überprüft werden. Hierunter fallen zum Beispiel die Überprüfung der Plausibilität zwischen Lieferdatum und Bestelldatum. Das Bestelldatum muss vor dem Lieferdatum liegen. Die Postleitzahl eines Kunden, der einem Betreuer zugeordnet ist, muss in einem bestimmten Postleitzahlenkreis liegen. Die Schlüsselkonsistenz ist für die Datenintegrität besonders wichtig.

Vollständigkeit kann durch das Berechnen fehlender Werte aus den bestehenden Werten erzielt werden. Zu unterscheiden ist, ob fehlende Werte in der realen Welt zulässig sind oder nicht. Für Elektroartikel gibt es zum Beispiel bei einer Sicherung keine Leistungsangabe. Wenn in der realen Welt Werte dafür existieren, so ist es zum Beispiel möglich, durch Berechnung des Mittelwerts oder gleitenden Mittelwerts diese fehlenden Werte durch angepasste Werte zu ersetzen. Ist nicht bekannt, ob es in der realen Welt einen Wert dafür gibt, so muss von Fall zu Fall unterschieden werden, wie diese fehlende Information behandelt wird.

Redundanzfreiheit bedeutet, dass keine gleichen Datensätze im Data Warehouse gespeichert werden. Sind die Quelldaten entsprechend mit eindeutigem Schlüssel codiert, so kann die Redundanzfreiheit leicht überprüft werden. Für die Redundanzfreiheit erwähnen (Bauer & Günzel, 2009) die Verfahren probabilistisches Record-Linkage-System, Sorted Neighbourhood- Verfahren und Neighbourhood Hash-Join.

Für die Einheitlichkeit werden Daten überprüft, ob deren Format stimmt. Als Beispiel kann das Datumsformat angesehen werden. Ist das Datum falsch formatiert, so wird dies richtiggestellt. Fehlt das Datum, kann entweder ein Einheitsdatum oder das Datum des vorhergehenden Datensatzes eingefügt werden oder der Datensatz wird aussortiert und für eine manuelle Nachbearbeitung bereitgestellt.

Sowohl in der Literatur als auch von Werkzeugherstellern in der Praxis wurden zahlreiche Ansätze entwickelt, die Datenqualität nach den oben genannten Kriterien zu gewährleisten. Es gibt sogar Anbieter von Werkzeugen zur Datenbereinigung, die sich ausschließlich auf diesen Nischenmarkt spezialisiert haben (man spricht hier von „Data Cleansing“ bzw. „Data Auditing“). Die heute gebräuchlichen Ansätze können an dieser Stelle nicht in der Tiefe behandelt werden. Mit einem Überblick über entsprechende Ansätze und Werkzeuge beschäftigt sich zum Beispiel (Leitner, 2008).

2.2.3 Laden der Daten

Das Laden der Daten in das Data Warehouse ist ein ressourcenintensiver und zeitintensiver Prozess. Aufgrund der zu erwartenden großen Datenmengen wird der Ladeprozess normalerweise in Zeitfenster mit geringer Auslastung (zum Beispiel die Nachtstunden) geschoben. Diese Vorgehensweise wird aber mit zunehmender Internationalisierung immer schwieriger, da die freien Zeitfenster immer kleiner werden.

Daher führt (Bodendorf, 2006, S. 38f) die inkrementelle und die parallele Lademöglichkeit für Data Warehouses an. Damit sollte der Ladeprozess so ressourcenschonend wie möglich ablaufen. Bei einer inkrementellen Lademöglichkeit werden Daten mit einer niedrigeren Auslastung des Gesamtsystems geladen. Der Ladeprozess dauert länger, das System bleibt aber aktiv. Eine parallele Lademöglichkeit geht von einer Abschaltung des Systems aus, wo in parallelen Prozessen mit maximaler Systemleistung geladen wird. Die Systemabschaltzeit sollte in letzterem Fall so kurz wie möglich gehalten werden.

Eine wichtige Aufgabe des Ladeprozesses ist die Historisierung der Daten. Werden Daten im Data Warehouse aktualisiert, so werden nicht einfach die bestehenden Daten überschrieben, sondern die neuen Daten zusätzlich zum bestehenden Datensatz persistiert.

Im Zuge des Ladeprozesses werden Indexe aktualisiert und materialisierte Sichten auf den letzten Stand gebracht. Um die Belastung des Gesamtsystems so gering wie möglich zu halten, werden die Sichten normalerweise nicht neu generiert, sondern inkrementell aktualisiert.

Die Metadaten der aktualisierten Daten werden mit der Katalogisierung in das Warehouse geschrieben. Im Metadaten-Katalog (auch Repository genannt) werden Informationen über die Daten, wie Datum der Extrahierung, Bereinigungsvorgänge oder auch Alleinstellungsmerkmale aufgezeichnet. Damit ist der ETL Prozess abgeschlossen.

2.3 Vorgehen zur Modellierung eines Data Warehouse

In diesem Abschnitt werden die Vorgehensmodelle zur Modellierung eines Data Warehouse in zwei Gruppen (daten- oder angebotsorientiert beziehungsweise abfragen- oder nachfrageorientiert) kategorisiert, erläutert und gegenübergestellt.

(Abello & Romero, 2010) teilen die Vorgehensmodelle in die beiden Gruppen *„demand-driven“* und *„supply-driven“* ein. Dabei beschreiben sie mit dem Begriff demand-driven die Modellierungssicht des Nutzers der Daten aus der Analyse. Hier wird grundsätzlich von möglichen Fragestellungen an das künftige Data Warehouse ausgegangen und davon Anforderungen an die Modellierung erzeugt. Erst in späteren Schritten der Modellierung werden die Quelldaten mit dem Datenschema abgestimmt.

Supply-driven stellt dagegen das Modellierungsvorgehen aus Sicht der Quelldaten dar. Hier wird von den Möglichkeiten der Quelldaten ausgegangen und aus diesen Quelldaten ein „maximales“ multidimensionales Schema generiert. Erst später wird auf mögliche Fragestellungen Rücksicht genommen. Dieser Ansatz ist auch unter dem Begriff *„data-driven“* in der Literatur zu finden.

Bei der Gegenüberstellung der beiden Gruppen gehen (Abello & Romero, 2010) auf die unterschiedlichen Sichtweisen und deren Eigenschaften ein. Nutzergetriebene Modelle (= demand-driven) verzichten oftmals auf Teile der Information, welche die Quelldaten beinhaltet, da das Vorgehensmodell nur die erwarteten Fragestellungen modelliert und beantwortet. Als Vorteil dieser Vorgehensweise ist zu sagen, dass das Ergebnis der Analyse für den Nutzer einfach zu interpretieren ist und überflüssige Information bereits im Vorfeld ausgefiltert wird.

Dagegen bieten datengetriebene oder quellgetriebene Vorgehensmodelle (= supply-driven) die Möglichkeit, neue Zusammenhänge zu erkennen und neue Erkenntnisse über den Bezug der Daten untereinander zu gewinnen. Als nachteilig wird angesehen, dass diese Vorgehensmodelle von einer erschöpfenden Analyse der Daten ausgehen und damit oft unnötige Komplexität in die Ergebnisse einbringen. Ein weiterer Kritikpunkt ist auch die Menge an Daten im Data Warehouse. Diese führt zu einem erhöhten Ressourcenbedarf und natürlich auch zu komplexeren und rechenintensiveren Abfragen. Dies resultiert unweigerlich in längeren Antwortzeiten für Abfragen. Vorteil der datengetriebenen Vorgehensmodelle ist, dass auch nicht vorhersehbare Ergebnisse und Zusammenhänge erfasst werden. Ist die Fragestellung an die Daten nicht genau spezifiziert, so ermöglichen diese Vorgehensmodelle eine Maximalanalyse der Quelldaten und damit die Schaffung neuer Erkenntnisse.

Eine weitere Einteilung findet man in der Literatur unter den Begriffen *„bottom-up“* (Moody & Kortink, 2000) oder *„top-down“* (Kimball, 1996). Diese Gliederung ist aber gleichzusetzen mit der bereits weiter oben beschriebenen Einteilung nach (Abello & Romero, 2010) in „demand-driven“ und „supply-driven“. Der Begriff „bottom-up“ ist somit äquivalent zum Begriff „supply-driven“ und der Begriff „top-down“ ist dem Begriff „demand-driven“ gleichzusetzen.

Als dritte Art von Vorgehensmodellen wird von (List, Bruckner, Machaczek, & Schiefer, 2002) die *„Goal-driven“* Methode erwähnt. Bei dieser Art werden in einer ersten Stufe die Ziele und Leistungen der Nutzer bestimmt. Aus diesen werden die Geschäftsprozesse abgeleitet und dann durch die Anwendung von Analysen, welche die Nutzer und deren Transaktionen beleuchten, die Transaktionen gefiltert. In einem dritten Schritt werden die Transaktionen in Abhängigkeiten umgewandelt, welche in einem Data Warehouse abgebildet werden können. Da diese Methode sehr kompliziert und abstrakt ist, wird diese von (List, Bruckner, Machaczek, & Schiefer, 2002) eher kritisch betrachtet. Daher wird diese Methode in der weiteren Betrachtungsweise nicht mehr berücksichtigt.

Vorgehensmodelle können normalerweise nicht rein einer der vorgestellten Gruppen zugeordnet werden. Die meisten Vorgehensmodelle leiten sich von beiden Gruppen ab und stellen damit hybride Vorgehensmodelle dar. Schwerpunktmäßig können sie aber einer der beiden Gruppen zugeordnet werden. (Vassiliadis, Quix, Vassiliou, & Jarke, 2001, S. 206) versuchen bereits, den hybriden Ansatz in ein einheitliches Vorgehensmodell zu bringen.

(List, Bruckner, Machaczek, & Schiefer, 2002) haben die verschiedenen Gruppen an Vorgehensmodellen untersucht und gegenübergestellt. Dabei wurde eine Tabelle mit Bewertungskriterien entworfen. Die Autoren sehen das datengetriebene Vorgehen vordergründig für Data Mining-Anwendungen und das Schaffen von Wissen als geeignet an. Angesprochen wird eher der operationale Unternehmensbereich. Das nutzergetriebene Vorgehen eignet sich eher für strategische Anwendungen, ähnlich dem zielgetriebenen Vorgehen.

Eine andere Einteilung erfolgt durch (Bauer & Günzel, 2009, S. 176ff). Anders als bei (Abello & Romero, 2010) gilt als Klassifizierungskriterium nicht der Ausgangspunkt bzw. Fokus der Modellierung, sondern die vorwiegend angewandte Methodik zur Modellierung.

Bauer und Günzel bilden nach der Methodik zur Modellierung eines Data Warehouse die beiden Klassen der evolutionären und der revolutionären Modelle.

Die evolutionären Vorgehensmodelle werden aus bereits bestehenden Vorgehensmodellen der Datenbankmodellierung entwickelt. Es wird auf die bestehende Semantik und Notation aufgebaut und diese um Elemente zur Modellierung der spezifischen Eigenheiten von multidimensionalen Schemata erweitert. Dies geschieht dahingehend, dass neue Konstrukte immer Spezialisierungen von bestehenden Konstrukten darstellen. Damit einhergehend wird auch das Vorgehensmodell zur Modellierung an diese Herangehensweise angepasst. Der Vorteil dieser Methode wird von (Bauer & Günzel, 2009, S. 176) dadurch begründet, dass für den Modellierer durch die Bekanntheit des Grundwerkzeuges eine leichtere Handhabung von diesem erfolgt.

Revolutionäre Vorgehensmodelle orientieren sich hingegen nicht an bestehenden Werkzeugen und Modellen. Sie stellen eine auf die Modellierung von multidimensionalen Datenschemata abgestimmte Methodik zur Verfügung. Der Modelldesigner hat bei revolutionären Vorgehensmodellen den Vorteil, nicht auf bereits bestehende Konstrukte Rücksicht nehmen zu müssen.

Das Vorgehensmodell für die Modellierung eines Data Warehouse sollte grundsätzlich in drei Abstrahierungsebenen erfolgen. In den Richtlinien von ANSI/X3/SPARC (Brodie & Schmidt, 1982) werden diese mit Konzeptebene, logische Ebene und physikalische Ebene benannt. Diese Richtlinien sind weit verbreitet und akzeptiert, wenn diese auch diskutiert werden und es keine Einigung über den Inhalt der Ebenen gibt (Prat, Akoka, & Comyn-Wattiau, 2006).

2.3.1 Das Vorgehensmodell von Golfarelli zur Modellierung eines Data Warehouse

Das Vorgehensmodell zur Modellierung eines Data Warehouse von (Golfarelli & Rizzi, 1998) wird als „Dimensional Fact Model“ (DFM) beschrieben. Die Autoren teilen die Modellierung in 6 Phasen ein, welche in Tabelle 1 dargestellt werden.

Schritt	*Input*	*Output*
Analyse des bestehenden Datenbanksystems	Bestehende Dokumentation	Datenbankschema
Anforderungsanalyse	Datenbankschema	Fakten, Datenmengengerüst
Konzeptuelles Design	Datenbankschema, Fakten, Datenmengengerüst	Dimensionsschema
Datenmengen, Validierung Dimensionsschema	Datenbankschema, Fakten, Datenmengengerüst	Datenmengen
Logisches Design	Dimensionsschema Logisches Zielschema, Datenmengen	Logisches Data Warehouse-Schema
Physisches Design	Logische Data Warehouse-Schema, Datenbankanforderungen, Datenmengen	Physisches Schema Data Warehouse

Tabelle 1: Sechs Phasen der Data Warehouse Modellierung nach (Golfarelli & Rizzi, 1998)

Wie die Tabelle 1 zeigt, finden sich die weiter oben bereits erwähnten drei Abstrahierungsebenen nach den Richtlinien von ANSI/X3/SPARC in den einzelnen Schritten wieder.

In der entnommenen Tabelle (Golfarelli & Rizzi, 1998, S. 4) befindet sich eine weitere Spalte, welche beschreibt, wer in den einzelnen Schritten involviert ist. Zusammenfassend kann gesagt werden, dass der Designer des Data Warehouse den gesamten Prozess begleitet. Bei der Analyse der bestehenden Struktur im ersten Schritt werden die IT-Spezialisten für die zugrunde liegenden Datenquellen, in den Schritten Anforderungsanalyse und Verfeinerung hingegen die späteren Nutzer einbezogen.

Im ersten Schritt werden die bestehenden Datenstrukturen, in Folge die Datenquellen analysiert. Dabei werden Datenbankschemata oder Datenschemata allgemein aus den Quelldaten definiert. Da (Golfarelli & Rizzi, 1998) nur auf Datenbanken als Quelle abstellen, ist es doch erforderlich, diese Sichtweise zu erweitern, da auch andere Datenquellen in einer Welt der heterogenen Information existieren. Namentlich anzuführen sind unter anderem Tabellen aus Tabellenkalkulationsprogrammen, Web Services im Internet, XML-Daten oder textorientierte Daten, letztere auch unter dem Begriff Flatfile zu finden.

Der zweite Schritt ist ein Blick auf die Erfordernisse der zukünftigen Nutzer. Hier werden spätere Fakten definiert und Diskussionen über nachkommend zu erwartende Datenmengen geführt. Die Ableitung erfolgt normalerweise aus einem relationalem Schema, welches im Schritt eins bereits erstellt oder erhoben wurde. Dabei werden die einzelnen Tabellen nach ihrer Häufigkeit der Änderung betrachtet. Tabellen im Datenbankschema, welche oft veränderte bzw. instanziierte Werte halten, eignen sich als Fakten. Im Gegensatz dazu eignen sich Tabellen, welche fast statische Werte halten, eher zu Dimensionen. Aus dieser Entscheidung für Fakten und Dimensionen heraus kann auf ein späteres Datenmengengerüst geschlossen werden.

Das konzeptuelle Design im Schritt drei hat als Grundlage das Datenbankschema der darunterliegenden Quellen. Hier werden die Fakten festgelegt und zu jedem Fakt die Dimensionen abgeleitet. Dazu werden von (Golfarelli & Rizzi, 1998) die folgenden Punkte angegeben

- Erstellen eines Attributbaumes
- Bereinigung und Zusammenfügen
- Definieren der Dimensionen
- Definieren der Measures
- Definieren der Hierarchien

Im Dimensional Fact Model wird von (Golfarelli & Rizzi, 1998) das dimensionale Schema eingeführt. Dieses beinhaltet die Faktenschemata, welche aus Dimensionen, Fakten und Hierarchien bestehen. Für jedes Fakt wird ein eigenes Faktenschema erstellt. Mehrere Faktenschemata ergeben das dimensionale Schema.

Graphisch gesehen stellt ein Fakt einen Wurzelknoten in einem Baum dar. Ein Fakt wird als Kasten repräsentiert (siehe in Abbildung 10 das Fakt `Verkauf`), analog zu einer Klasse im UML-Klassendiagramm. Die Beschriftung befindet sich oben im Kasten. Darunter werden die im letzten Schritt erhobenen Measures aus dieser Tabelle angeordnet. Gemäß dem ersten

Punkt „Erstellen eines Attributbaums“ werden nun die weiteren Tabellen aus dem Datenbankschema entlang ihrer Verbindungen (Fremdschlüssel-Beziehungen) angeordnet. Dazu werden die Verbindungen analysiert und jene Verbindungen weiter betrachtet, welche eine 1:n-Beziehung aufweisen. Diese sind im relationalen Schema leicht erkennbar, da sie eine Fremdschlüsselabhängigkeit zur Faktentabelle aufweisen. Somit sind die Knoten der untersten Klassifikationsebenen gefunden.

Als nächstes wird von diesen Knoten aus vorgegangen und weitere Abhängigkeiten aufgespürt. Dabei gilt wieder die gleiche Vorgangsweise wie für die erste Klassifikationsebene. Es werden alle 1:n-Beziehungen ausfindig gemacht und entsprechend mittels Kanten mit dem bestehenden Knoten verbunden. Mit dieser Methode wird durch das gesamte Datenbankschema traversiert. Jene Tabellen, welche nicht erreicht wurden, können ausgeschieden werden, da diese nicht im Zusammenhang mit dem jeweiligen Fakt stehen.

Im nächsten Punkt wird der Attributbaum bereinigt und zusammengeführt. Dies bedeutet, dass Elemente in den verschiedenen Tabellen, welche mehrfach vorkommen oder welche für den Nutzer keine Aussagekraft haben und auch nicht für Fremdschlüssel-Beziehungen dienen, ausgeschieden werden. In diesem Punkt werden auch Pfade, welche durch die rekursive Betrachtung der einzelnen Beziehungen entstanden sind, entsprechend zusammengeführt, wenn gleichbedeutende Elemente betroffen sind.

Danach werden die Dimensionen beschrieben. Alle direkt mit dem Fakt (der Wurzel des Baumes) verbundenen Knoten, die sich gut als Klassifikationsattribute, dagegen weniger als Kennzahlen eignen (meist nicht-numerische Attribute), werden als Dimensionen deklariert. Von der Wurzel ausgehend werden dann die Hierarchien der Dimensionen durch Verallgemeinerung gebildet. Ein Beispiel: `Produkt` – `Produktgruppe` – `Kategorie`.

Für den nächsten Punkt wird wieder zurück an die Wurzel gegangen und der dortige Faktenknoten untersucht. Alle jene Elemente, welche nun durch eine Fremdschlüsselbeziehung beansprucht werden, scheiden aus der Betrachtung für die Definition der Measures aus. Die restlichen Elemente werden geprüft, ob eine Möglichkeit der Aggregation besteht. Letztere Elemente werden als Measures deklariert. Diese beinhalten die zu extrahierenden Daten.

Der letzte Punkt widmet sich der Definition der Hierarchien. Diese ergeben sich grundsätzlich durch die mittels Kanten verbundenen Klassifikationsebenen. Hier können aber zusätzliche Hierarchien geschaffen werden. Diese werden dann aus den Daten abgeleitet. Als Beispiel kann man die Zeitdimension heranziehen. In Abbildung 10 wird die Zeitdimension durch `Tag`, `Monat`, `Jahr` und `Woche` dargestellt. Diese Dimensionsattribute werden aus dem Datum der Quelldaten abgeleitet. Wie auch ersichtlich ist, werden dann die beiden Pfade mit der `Jahr`-Ebene (-Level) wieder zusammengefasst. Die Verbindungen werden mit Pfeilen gekennzeichnet, damit klargestellt ist, dass die Hierarchie nur von innen nach außen fortläuft und beispielsweise im Pfad „`Tag` → `Monat` → `Jahr`“ nicht weiter nach „`Woche`“ laufen kann. Zirkelbezüge oder ternäre Beziehungen werden dadurch verhindert.

Hierarchien beschreiben die Zusammenhänge der einzelnen Klassifikationsstufen in den Dimensionen und stellen damit auch Zusammenhänge der Daten im Datenschema dar. Über jede

Dimension werden die Fakten bei einer Abfrage aggregiert. Je näher eine Dimension am Fakt liegt, desto feiner ist die Granularität der Daten. Die äußersten Dimensionen an den Blättern des Baumes weisen so die höchste Aggregationsstufe auf.

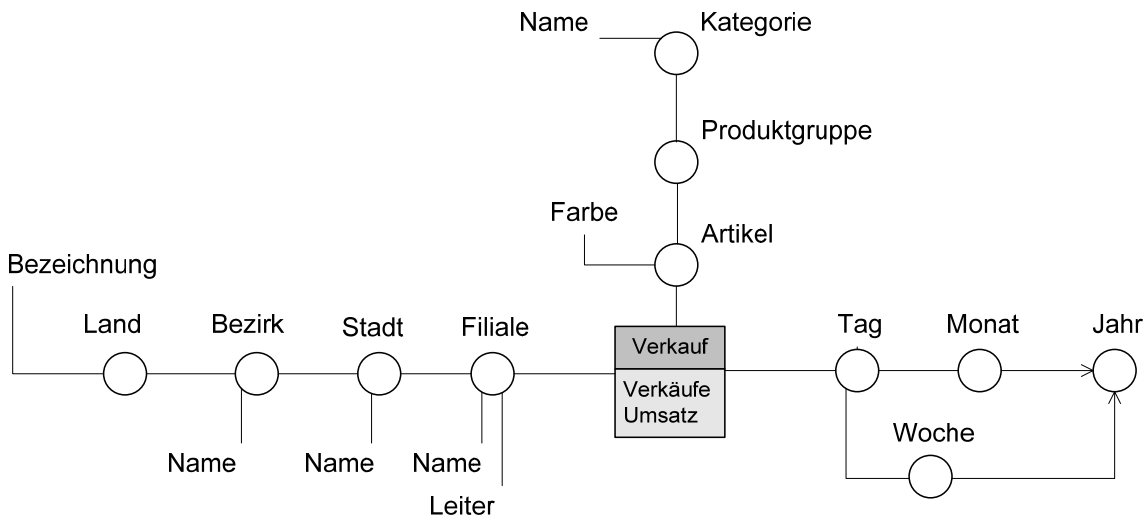

Abbildung 10: Multidimensionales Beispielschema als Faktenschema nach Golfarelli

In Abbildung 10 wird das zuvor in Abbildung 6 dargestellte multidimensionale Beispielschema als Faktenschema gemäß DFM (Golfarelli & Rizzi, 1998) dargestellt. Dazu wird die Tabelle `Verkauf` als Fakt angenommen und die Dimensionen entsprechend im Verhältnis zum Fakt dargestellt. Wie aus Abbildung 10 entnommen werden kann, ist in der Darstellung als Faktenschema auch das multidimensionale Modell aus den Grundlagen von (Bauer & Günzel, 2009) wieder erkennbar. Die in Abbildung 6 geschaffenen Kanten stellen nun ebenfalls die Verbindungskanten zwischen den einzelnen Klassifikationsstufen dar.

Die Klassifikationsstufen der Dimensionen werden in der Abbildung 10 als Kreis abgebildet. Die an diese Kreise mit Linien angefügten Namen werden als nicht-dimensionale Attribute („Dekorationen“) bezeichnet und haben auf die Struktur keinen Einfluss. Diese bieten nur zusätzliche Information zur Beschreibung der Dimensionsinstanzen.

$$Dom(f) = \{a_i \in A | \exists (a_0, a_i) \in R\}$$

Formel 9: Dimensionsbeziehung nach (Golfarelli & Rizzi, 1998, S. 4)

Der mathematisch-formale Zusammenhang zwischen den Klassifikationsstufen der Dimensionen wird von (Golfarelli & Rizzi, 1998, S. 4) mittels der Gleichung in Formel 9 definiert. „A“ ist die Menge an Dimensionsattributen, „a“ das einzelne Dimensionsattribut. „R“ ist die Beziehung zwischen den einzelnen Attributen in der Form von „(a_i,a_j)“ als 1:n Beziehung. Gibt es eine Klassifikationsstufe a_i der Dimension A und existiert eine Abhängigkeit zur Basisklassifikationsstufe a_0, so handelt es sich um eine hierarchische Dimensionsbeziehung von a_0 zu a_i.

Eine Faktentabelle beschreibt ein Fakt. Existieren in einem Datenbankschema mehrere Fakten, respektive mehrere Faktentabellen, so werden diese, jede für sich, in ein Faktenschema

extrahiert. Dabei können die daraus gewonnenen Dimensionen von mehreren Faktenschemata genutzt werden.

Das Vorgehensmodell von (Golfarelli & Rizzi, 1998) kann nach der Klassifikation von (Abello & Romero, 2010) in die Gruppe der datengetriebenen Vorgehensmodelle eingeordnet werden. Nach dem Ansatz von (Golfarelli & Rizzi, 1998) wird aus der bestehenden Datenstruktur in sechs Schritten das Data Warehouse-Schema generiert. Der Nutzereinfluss erfolgt erst im fortgeschrittenen Stadium mit der Bereinigung und Anpassung der Knoten.

Für die Klassifikation des Vorgehensmodells von (Golfarelli & Rizzi, 1998) nach (Bauer & Günzel, 2009) kann dieses zu den revolutionären Modellen gezählt werden. Die Notation unterscheidet sich wesentlich von jenem nach der Unified Modelling Language, kurz UML, erstellten Klassendiagramm oder auch eines Entity-Relationship Modells (E/R-Modell).

2.3.2 Das Unified multidimensional UML – Vorgehensmodell

Das Vorgehensmodell nach der vereinheitlichten multidimensionalen UML, in weiterer Folge mit UmUML (Unified multidimensional UML) abgekürzt, wird von (Prat, Akoka, & Comyn-Wattiau, 2006) in vier Phasen gegliedert. Diese lauten

- Konzeptuelles Design
- Logisches Design
- Physisches Design
- Dateneinbindung

Wie aus den Namen der einzelnen Phasen abgeleitet werden kann, lehnt sich auch dieses Vorgehensmodell an die Richtlinien von ANSI/X3/SPARC (Brodie & Schmidt, 1982) an.

In der nachstehenden Graphik von Abbildung 11 sind sämtliche Schritte aufgelistet und die Zugehörigkeit zu den jeweiligen Phasen seitlich links mit geschwungenen Klammern markiert. Die ovalen Felder in Abbildung 11 beschreiben durchzuführende Aufgaben. In den rechteckigen Feldern wird das Ergebnis der vorangegangenen Aufgabe beschrieben.

Das Vorgehensmodell startet mit den Anforderungen der Nutzer. Da die Anforderungen der Nutzer keinen eigenen Schritt für das Vorgehensmodell bedeuten, sondern als gegeben angenommen werden, sind diese auch nicht in der seitlichen Klammer der ersten Phase inkludiert.

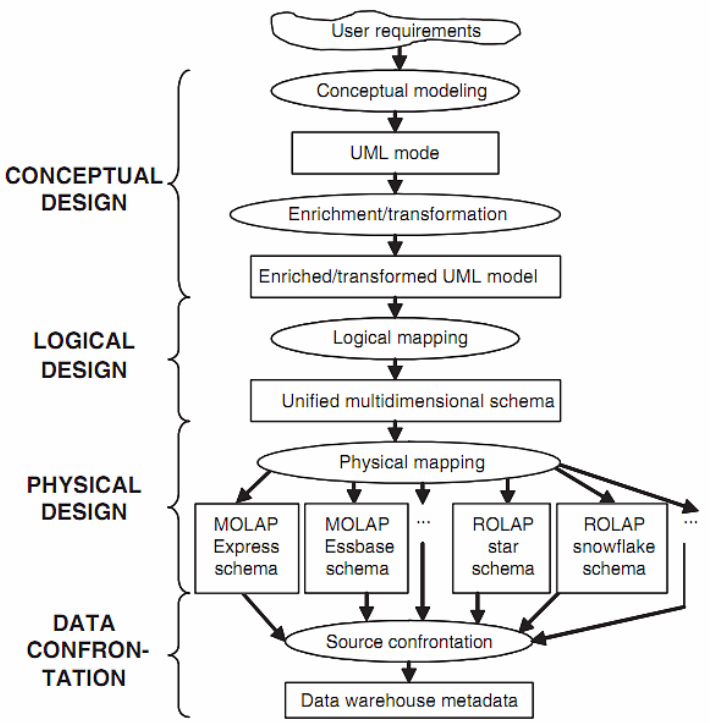

Abbildung 11: Ebenen für das Unified Multidimensional UML (UmUML) – Vorgehensmodell (Prat, Akoka, & Comyn-Wattiau, 2006)

(Prat, Akoka, & Comyn-Wattiau, 2006, S. 1456) starten das UmUML-Vorgehensmodell mit dem konzeptuellen Design. Diese Phase dient zur Spezifizierung des konzeptuellen Data Warehouse Schemas. Hierfür wird im konzeptuellen Design ein UML-Modell erstellt. Das UML-Modell bietet den Vorteil, für Entwickler und Nutzer leicht lesbar zu sein. Weiters bietet UML eine weitreichendere Semantik als zum Beispiel Entity-Relationship-Notationen. Schließlich eröffnet UML die Möglichkeit, das konzeptuelle Modell mit Werkzeugen in ein multidimensionales Schema zu transformieren.

Das konzeptuelle Design wird in zwei Teilen abgearbeitet. Nach dem ersten Teil entsteht ein UML-Klassendiagramm ohne Operationen. Dann wird das Klassendiagramm im zweiten Teil so transformiert, dass ein Mapping auf ein multidimensionales Schema ermöglicht wird.

Die Transformation wird von (Prat, Akoka, & Comyn-Wattiau, 2006) dann in vier Schritte gegliedert:

- Bestimmung der identifizierenden Attribute
- Festlegung der Measures
- Migration der Assoziationsattribute
- Transformierung der Generalisierungen

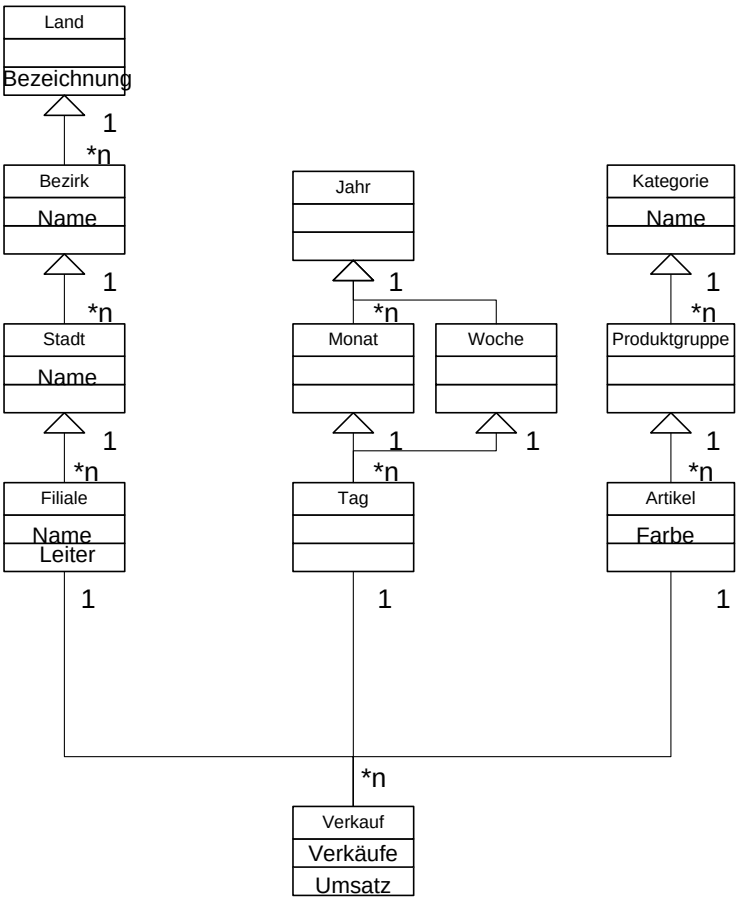

Abbildung 12: Konzeptuelles Unified Multidimensional UML-Modell

Abbildung 12 zeigt für mein Musterbeispiel das konzeptuelle UML-Modell. Hier sind alle Klassen eingezeichnet, welche später im multidimensionalen Schema enthalten sein sollen. Auch die Beziehungen zwischen den Klassen sind spezifiziert. Dieses UML-Diagramm stellt die Nutzersicht dar.

Für mein Beispiel sind keine identifizierenden Attribute zu bestimmen. Alle Klassen lassen sich durch einen eindeutigen Schlüssel bestimmen. Im zweiten Schritt werden die Attribute bestimmt, welche Measures darstellen. Diese sind in meinem Fall `Verkäufe` und `Umsätze` aus der Klasse `Verkauf`. Die beiden nächsten Schritte betreffen Assoziationsklassen und

Generalisierungen. Beides kommt in meinem Musterbeispiel nicht vor, so können auch diese Teile übersprungen werden.

Damit ist mein transformiertes konzeptuelles Schema aus dem zweiten Teil gleich dem konzeptuellen Schema aus dem ersten Teil (siehe Abbildung 12).

In der nächsten Phase wird das transformierte konzeptuelle Schema als Ergebnis aus der konzeptuellen Phase in ein logisches Schema gemappt. Dazu werden fünf Schritte durchlaufen. Diese werden von (Prat, Akoka, & Comyn-Wattiau, 2006, S. 1459ff) wie folgt benannt:

- Definition der Fakten, der Measures und der verbundenen Dimensionen
- Definition der Hierarchien
- Definition der Dimensionen
- Definition der Klassifikationsstufen
- Definitionen der Aggregationen über die Dimensionshierarchien

Für den ersten Schritt werden alle n:m-Assoziationsklassen und alle n-ären Beziehungsklassen zu Fakten gemappt. Die Attribute, welche nicht zur Referenzierung der verbundenen Klassen dienen, werden zu Measures dieser Fakten gemappt. Die verbundenen Klassen werden als Dimensionen gemappt. Weiters werden alle einfachen Klassen, das sind Klassen, welche für keine weitere Klasse als Referenz dienen, ebenfalls als Fakten gemappt. Hier werden die Attribute, welche nicht als Fremdschlüssel dienen, ebenfalls zu Measures.

Für mein Beispiel kommt als Fakt nur die Klasse „`Verkauf`“ in Frage. Das UML-Schema in Abbildung 12 weist keine n:m- oder n-ären Beziehungen auf. Lediglich die Klasse „`Verkauf`“ ist eine ordinäre Klasse. Daher wird diese als Fakt und deren Attribute „`Verkäufe`“ und „`Umsatz`“ als Measures gemappt.

Die Hierarchien werden im zweiten Schritt nun durch Verfolgung der Abhängigkeiten der einzelnen Dimensionen gebildet. Hier werden alle Verbindungen von den im ersten Schritt gefundenen Dimensionen weiter verfolgt. Damit entsteht eine Baumstruktur der Abhängigkeiten. Jede gefundene Klasse beschreibt eine weitere Hierarchie in diesem Pfad. Verfolgt werden aber nur Beziehungen des Typs 1:n oder 1:1. Damit vermeiden (Prat, Akoka, & Comyn-Wattiau, 2006) eventuell mögliche Zirkelbezüge und damit Rekursionen später bei der Aggregation.

Für mein Beispiel ergeben sich die angeführten Dimensionspfade:

- `Filiale→Stadt→Bezirk→Land→All`
- `Tag→Monat→Jahr→All`
- `Tag→Woche→Jahr→All`
- `Artikel→Produktgruppe→Kategorie→All`

Im nächsten Schritt werden die Dimensionen deklariert. Damit wird fixiert, welche Dimensionen auf die Fakten zugreifen und aus wie vielen Dimensionen das multidimensionale Schema besteht. Weiters werden die Dimensionen mit sprechenden Namen benannt.

In meinem Beispiel ergeben sich drei Dimensionen, da die beiden Zeitdimensionen über ein Wurzelelement (`Tag`) verfügen. Diese sind `Filialen` mit der Dimension `Filiale`, `Zeit` mit der Dimension `Tag` und `Produkt` mit der Dimension `Artikel`.

Im vierten Schritt werden die nicht-identifizierenden (nicht-dimensionalen) Attribute definiert. Diese haben dann erklärenden Charakter. In meinem Beispiel ergeben sich für die Dimensionsklasse `Artikel` das Attribut `Farbe`, für `Kategorie` das Attribut `Name`, für `Filiale` die Attribute `Name` und `Leiter`, für `Stadt` und `Bezirk` je das Attribut `Name` und für die Dimensionsklasse `Land` das Attribut `Bezeichnung`.

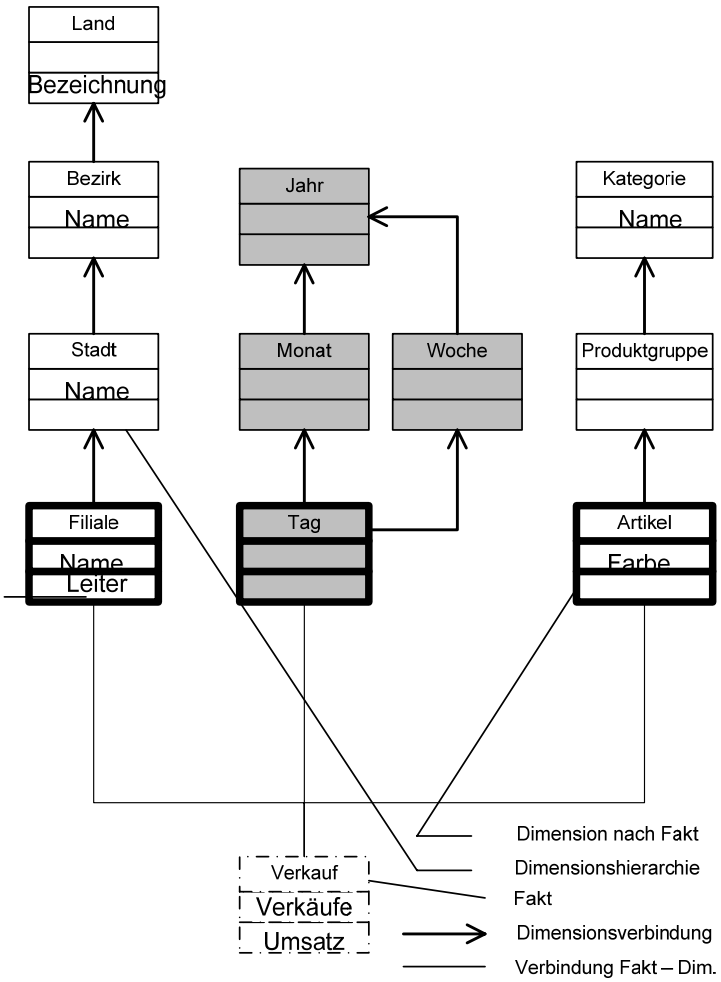

Abbildung 13: Logisches multidimensionales Schema mit UmUML

Als letzten Schritt deklarieren (Prat, Akoka, & Comyn-Wattiau, 2006, S. 1466) die semantisch sinnvollen Möglichkeiten der Aggregation der einzelnen Measures über die Dimensionshierarchien. Hier ist das wichtigste Kriterium die Summierbarkeit. Dazu sind die zu erwartenden Daten zu prüfen, ob eine Summierbarkeit gegeben ist oder nicht. (Bauer & Günzel, 2009) geben als wichtigstes Kriterium die Richtigkeit und Sinnhaftigkeit der Ergebnisse an.

`Umsatz` und `Anzahl` der verkauften Artikel können ohne weiteres über alle Dimensionen summiert werden. Andere Werte wie `Lagerstände` machen keinen Sinn, wenn diese über die Zeit summiert werden. Die Summe aller Monatslagerbestände ergibt nicht den Lagerstand am Jahresende. `Lagerstände` über die Dimension `Filialen` zu summieren macht dagegen wieder Sinn. Preise von Artikeln zu summieren ergibt nur über die Dimension `Produkte` Sinn. Über die anderen Dimensionen sind nur zum Beispiel der Mittelwert oder die Standardabweichung wieder sinnvoll. Daher ist von Measure zu Measure eine gesonderte Betrachtung erforderlich.

Wichtig in diesem Zusammenhang ist eine richtige Ableitung der Dimensionen, damit jedes Fakt sich in jeder verbundenen Dimension genau einmal wiederfindet. Aggregiert man zum Beispiel über `Produktgruppen`, so darf es keine `Produkte` geben, welche in mehr als eine `Produktgruppe` fallen. Diese würden sonst doppelt gewertet und somit die Summen nicht übereinstimmen.

Daraus entwickeln (Prat, Akoka, & Comyn-Wattiau, 2006) nun das logische Schema, welches in Abbildung 13 für mein Beispiel abgebildet ist.

Die dritte Phase nach (Prat, Akoka, & Comyn-Wattiau, 2006) leitet das aus Phase zwei gewonnene logische multidimensionale Schema ab in ein physisches Datenbankschema. Diese Phase hängt von der physischen Gegebenheit der Datenbank ab. Hier muss unterschieden werden, welche Datenbank als Grundlage dienen soll. Geschieht die Implementierung auf einer relationale Datenbank, so hat das Mappen anders zu erfolgen als bei einer Datenbank mit multidimensionalem Speicherkonzept (vgl. 2.1.4 Speicherung des multidimensionalen Schemas).

Das Vorgehensmodell ist von den physischen Gegebenheiten der darunterliegenden Datenbank aber unabhängig. Es werden sukzessive

- die logischen Dimensionsebenen
- die logischen Measures und Fakten
- die logischen Hierarchien
- die logischen Dimensionsattribute

in das physische Datenbankschema gemappt. Dabei ergeben sich Tabellen mit dem Namen des Attributs und dessen Datentyp. (Prat, Akoka, & Comyn-Wattiau, 2006) verwenden in ihrer Arbeit eine Oracle Datenbank mit multidimensionalem Schema. Daher werden im Beispiel (Tabelle 2) auch die Mappings entsprechend abgehandelt.

Name	*Datentyp*
Filiale	NUMBER
Filiale_Name	NVARCHAR2
Filiale_Leiter	NVARCHAR2
Stadt	NUMBER
Stadt_Name	NVARCHAR2
Bezirk	NUMBER
Bezirk_Name	NVARCHAR2
Land	NUMBER
Land_Bezeichnung	NVARCHAR2
Tag	NUMBER
Monat	NUMBER
Jahr	NUMBER
Woche	NUMBER
Artikel	NUMBER
Artikel_Farbe	NUMBER
Produktgruppe	NUMBER
Kategorie	NUMBER
Kategorie_Name	NVARCHAR2

Tabelle 2: Zuordnung der Dimensionen zum physischen Datenbankschema mit UmUML

Im nächsten Schritt werden die Measures und Fakten zum physischen Datenbankmodell gemappt. In meinem Beispiel wird für die Measures eine Tabelle mit den Dimensionsattributen angelegt, welche die Measures referenzieren. Dabei ist zu beachten, dass die Referenzattribute bereits in Tabelle 2 angelegt wurden. Mit diesen Attributen müssen sich die Metadaten des Measures decken. In einer dritten Spalte in Tabelle 3 wird das referenzierte Dimensionsattribut aufgenommen.

Name	*Datentyp*	*Dimension*
Verkäufe	NUMBER	<Filiale Tag Artikel>
Umsatz	NUMBER	<Filiale Tag Artikel>

Tabelle 3: Zuordnung der Fakten zu den Dimensionen mit UmUML

Damit sind die die Fakten und Measures gemappt. Als nächstes werden im Folgeschritt die Hierarchien für das physische Datenbankmodell gemappt.

Name	*Ausgangsdimension*	*Zieldimension*
Filiale.Stadt	<Filiale>	<Stadt>
Stadt.Bezirk	<Stadt>	<Bezirk>
Bezirk.Land	<Bezirk>	<Land>
Land.All	<Land>	<All>
Tag.Monat	<Tag>	<Monat>
Monat.Jahr	<Monat>	<Jahr>
Tag.Woche	<Tag>	<Woche>
Woche.Jahr	<Woche>	<Jahr>
Jahr.All	<Jahr>	<All>
Artikel.Produktgruppe	<Artikel>	<Produktgruppe>
Produktgruppe.Kategorie	<Produktgruppe>	<Kategorie>
Kategorie.All	<Kategorie>	<All>

Tabelle 4: Mapping für die Hierarchien der Dimensionen mit UmUML

Als letztes werden noch die nicht-identifizierenden Attribute gemappt. Diese werden ebenfalls wieder in einer Tabelle zusammengestellt.

Name	*Datentyp*	*Zugehörige Dimension*
Name	NVARCHAR2	<Filiale>
Leiter	NVARCHAR2	<Filiale>
Name	NVARCHAR2	<Stadt>
Name	NVARCHAR2	<Bezirk>
Bezeichnung	NVARCHAR2	<Land>
Farbe	NVARCHAR2	<Artikel>
Name	NVARCHAR2	<Kategorie>

Tabelle 5: Mapping der nichtidentifizierenden Attribute zu den Dimensionen mit UmUML

Damit ist das Mapping vollzogen und die wichtigsten Phasen für das Vorgehensmodell abgehandelt. Die vierte Phase befasst sich mit der Einbindung von Daten in das Data Warehouse. Diese Aufgabe wird erst zu einem späteren Zeitpunkt diskutiert.

Zusammenfassend zu diesem Vorgehensmodell von (Prat, Akoka, & Comyn-Wattiau, 2006) lässt sich feststellen, dass der Aufwand etwas höher ist, als bei Einsatz des weiter oben diskutierten Dimensional Fact Model von (Golfarelli & Rizzi, 1998). Das UmUML-Vorgehensmodell beleuchtet aber den Weg zur Erstellung eines Datenbankmodells für ein Data Warehouse in sehr überschaubaren Schritten, welche auch eine gewisse Automatisierung erfahren können.

Das Unified multidimensional UML-Vorgehensmodell kann in die Gruppe der *„demand-driven“* nach (Abello & Romero, 2010) eingeteilt werden. Der Ansatz geht von einer Analyse der Nutzersicht aus und erarbeitet daraus in verschiedenen Phasen das Datenbankschema. Die Quelldaten werden für dieses Vorgehensmodell nicht verwendet. (Prat, Akoka, & Comyn-Wattiau, 2006) gehen davon aus, dass die Daten in einem ETL-Prozess (Extrahieren, Transformieren, Laden) aufbereitet und geladen werden. Eine Erläuterung für diese Prozesse erfolgt später in dieser Arbeit.

Nach der Einteilung von (Bauer & Günzel, 2009) dagegen wird das UmUML-Vorgehensmodell den evolutionären Vorgehensmodellen zugeordnet. Als Grundlage für UmUML wird die Notation nach UML verwendet, welche für die Modellierung von Datenbankschemata, Ablaufschemata und ähnlichem breiten Einsatz findet.

2.4 Einbindung geographischer Daten

Geographische Daten stellen an ein Data Warehouse besondere Anforderungen, da sie graphische Elemente darstellen. In den folgenden Unterkapiteln werden die Begriffe Geographie und geographische Daten erläutert. Danach werden zwei Möglichkeiten zur Einbindung geographischer Daten in ein Data Warehouse vorgestellt. Das erste Modell speichert die geographischen Daten im Data Warehouse, das zweite Modell speichert diese extern.

2.4.1 Definition Geographie

Die allgemeine Geographie ist eine klassische Wissenschaft, welche früher die Beschreibung der Erde als zentrales Feld hatte. Heute wird die Landschaft von (Leser, Haas, Mosimann, & Paesler, 1997, S. 252) als integratives Element gesehen, welches ein Wirkungsgefüge aus physischen, anthropogenen und biotischen Sachverhalten betrachtet.

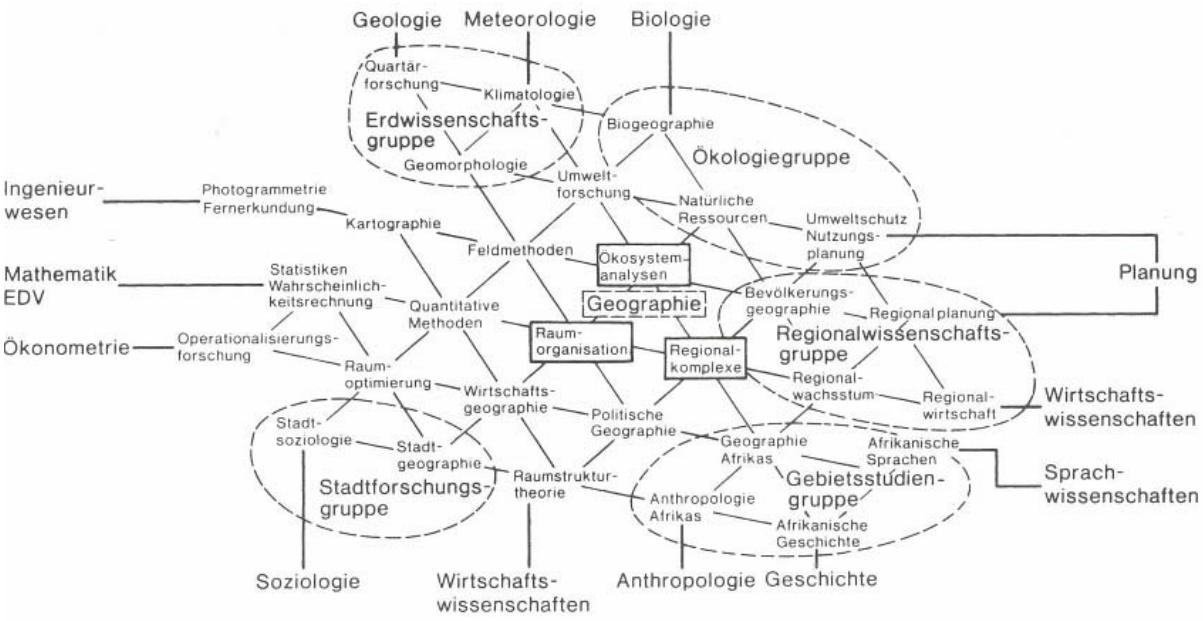

Abbildung 14: Einteilung der Geographie (Leser, Haas, Mosimann, & Paesler, 1997, S. 252)

Neben der allgemeinen Geographie beschäftigt sich die regionale Geographie mit der Länderkunde (Astor & Bussian, 1997, S. 119). Durch die Spezialisierung in den letzten Jahren sind zahlreiche Spezialgebiete aus diesen beiden Grundtypen hervorgegangen. Abbildung 14 zeigt eine Übersicht über die Wissenschaftsgebiete und deren Schnittstelle zu anderen Wissenschaften.

Die für die Anwendung in dieser Diplomarbeit maßgebliche Kartographie leitet sich von der Geographie über Raumorganisation und Feldmethoden ab. Die Kartographie wird von (Hake & Grünreich, 1994, S. 3) als Fachgebiet beschrieben, welches für das Sammeln, Verarbeiten, Auswerten, Speichern und Darstellen von raumbezogener Information steht. Die Aufteilung der Stoffgebiete in der Kartographie beinhaltet in der angewandten Kartographie unter anderen auch Geo-Informationssysteme.

2.4.2 Definition der geographischen Daten

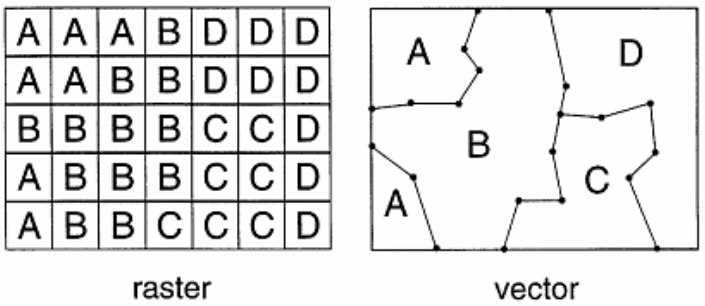

Abbildung 15: Raster- versus Vektorgraphik (Church, 2002, S. 543)

Geographische Daten werden von (Bimonte, Tchounikine, & Miquel, 2005, S. 39) als eine Sammlung von Attributen, welche geographische und beschreibende Information umfassen, bezeichnet. Geographische Information stellt geographische Datenobjekte dar. (Church, 2002) teilt geographische Datenobjekte nach der zur Repräsentation verwendeten Technologie in Rasterdatenobjekte und Vektordatenobjekte.

Rasterdatenobjekte (Abbildung 15 links) teilen ein bestimmtes begrenztes Betrachtungsfeld in gleichmäßig eingeteilte Felder, den so genannten Raster. Diese Rasterfelder besitzen einen Darstellungswert, zum Beispiel grün für einen Wald. Felder mit gleichem Darstellungswert werden gruppiert zu Layern (ergibt zum Beispiel eine Waldfläche). Verschiedene Layer werden zu einem geographischen Rasterdatenobjekt zusammengefasst.

Vektordatenobjekte (Abbildung 15 rechts) beschreibt (Church, 2002) als geometrisch deklarierte Linien (Vektoren) oder Punkte. Aus diesen Vektoren und Punkten ergeben sich die primitiven geographischen Datentypen Punkt, Linie und Polygon.

Verwendung finden Rasterdatenobjekte vor allem für die Darstellung von Ansichten für Flächen oder Landschaftsbilder. Vektordatenobjekte werden hingegen zur Verortung von Gegenständen im geometrischen Raum verwendet. Als Beispiel nennt (Church, 2002) Transportnetzwerke oder Erhebungsflächen.

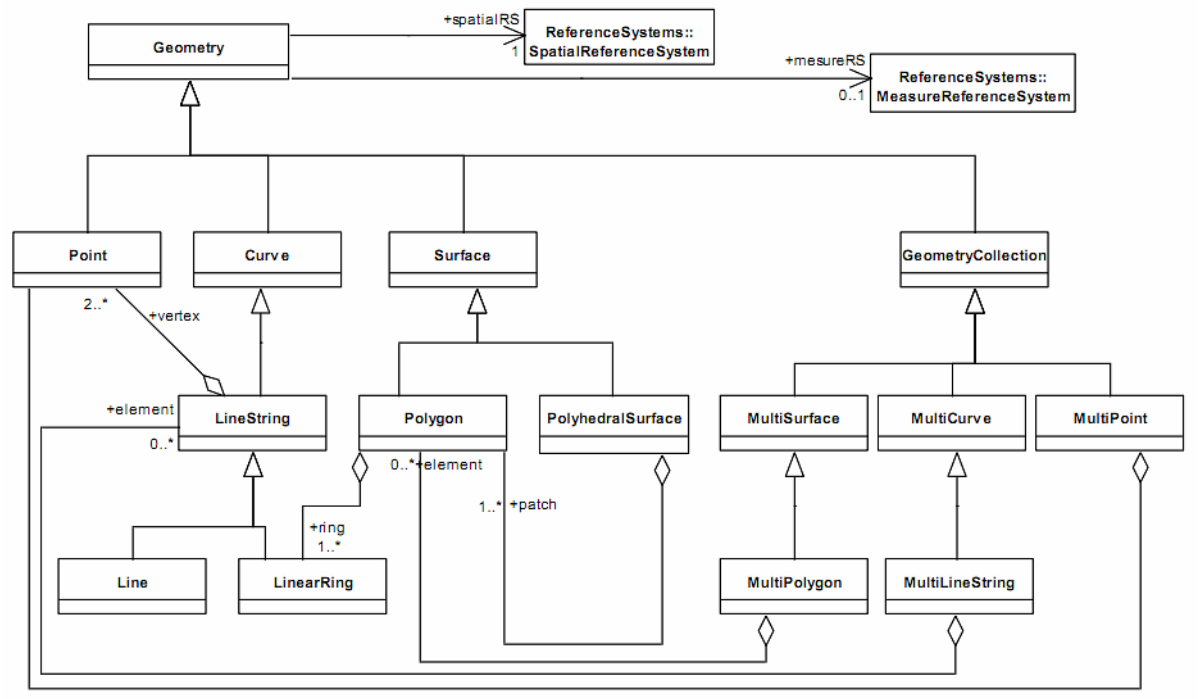

Abbildung 16: Geometrietypenhierarchie der Geometrietypen (Herring, 2010, S. 16)

(Herring, 2010) vom Open Geospatial Consortium (OGC) beschreibt die primitiven geographischen Datentypen in den „*Implementation Standards for Geographic Information*“ als Geometrietyphierarchie. Diese wird in Abbildung 16 dargestellt.

Geometrische Typen bestehen aus den primitiven Typen Punkt (Point), Kurve (Curve) und Fläche (Surface) oder Zusammenfassungen der einzelnen primitiven Typen. Mehrere geometrische Punkte ergeben Linien, wahlweise einfache Linien oder zusammenhängende Linien. Zusammenhängende Linien sind auch als Teil eines Polygons, wie in Abbildung 16 modelliert, verwendet. Linien sind als spezielle Kurven ohne Krümmung modelliert, für die ebenfalls die Aufteilung in einfache und zusammengesetzte Linien gilt. Oberflächen werden von (Herring, 2010, S. 16) in Polygone und vielflächige (polyhedrale) Oberflächen eingeteilt. In zusammengefassten Typen werden mehrere einzelne primitive Typen verwendet. Diese stellen dann größere geometrische Zusammenhänge dar.

Der SQL-Server von Microsoft unterstützt ab der Version 2008 direkt die Datentypen „Geometry“ und „Geography“. Wie (Konopasek, 2010) beschreibt, referenziert der Datentyp „Geometry“ ein kartesisches Koordinatensystem in der ebenen Fläche, wohingegen der Datentyp „Geography“ ein auf die Krümmung der Erde abgestimmtes geodätisches Modell verwendet. „Geometry“ kann somit auch für herkömmliche Koordinaten in einem Gebäude oder in einer Stadt verwendet werden.

Als Geo-Typen kennt auch der SQL-Server die drei primitiven Typen Punkt, Linie und Polygon. Auswertungen nach diesen Datentypen werden durch SQL unterstützt. Eine graphische Darstellung der Ergebnisse wird ebenfalls unterstützt.

2.4.3 Einbindung der geographischen Daten in ein Data Warehouse

Zur Darstellung geographischer Daten wird von (Da Silva, De Oliveira, Fidalgo, Salgado, & Times, 2007) ein Metamodell vorgestellt. Für das Metamodell wurden die Richtlinien der Object-Constraint-Langugage (OCL) der (Object Modeling Group, 2005) auf Basis von UML zum besseren Verständnis herangezogen.

Im oberen Bereich von Abbildung 17 wird der Aufbau eines Data Warehouse modelliert. Das Schema beinhaltet eine Menge an Tabellen. Diese können vom Typ Dimensionstabelle oder Faktentabelle sein. Drei Arten von Dimensionstabellen leiten (Da Silva, De Oliveira, Fidalgo, Salgado, & Times, 2007) ab: geographische, konventionelle und hybride Dimensionstabelle. Die geographischen Attribute in einer geographischen Dimensionstabelle werden von den Autoren auf die weiter oben vorgestellten primitiven und zusammengesetzten geographischen Datentypen zurückgeführt.

Measures leiten sich von Tabellenspalten ab und werden von den Autoren in räumlich und normal unterteilt. Die räumlichen Measures teilen sich wie die Attribute der Dimensionen ebenfalls in die oben erwähnten primitiven Datentypen auf.

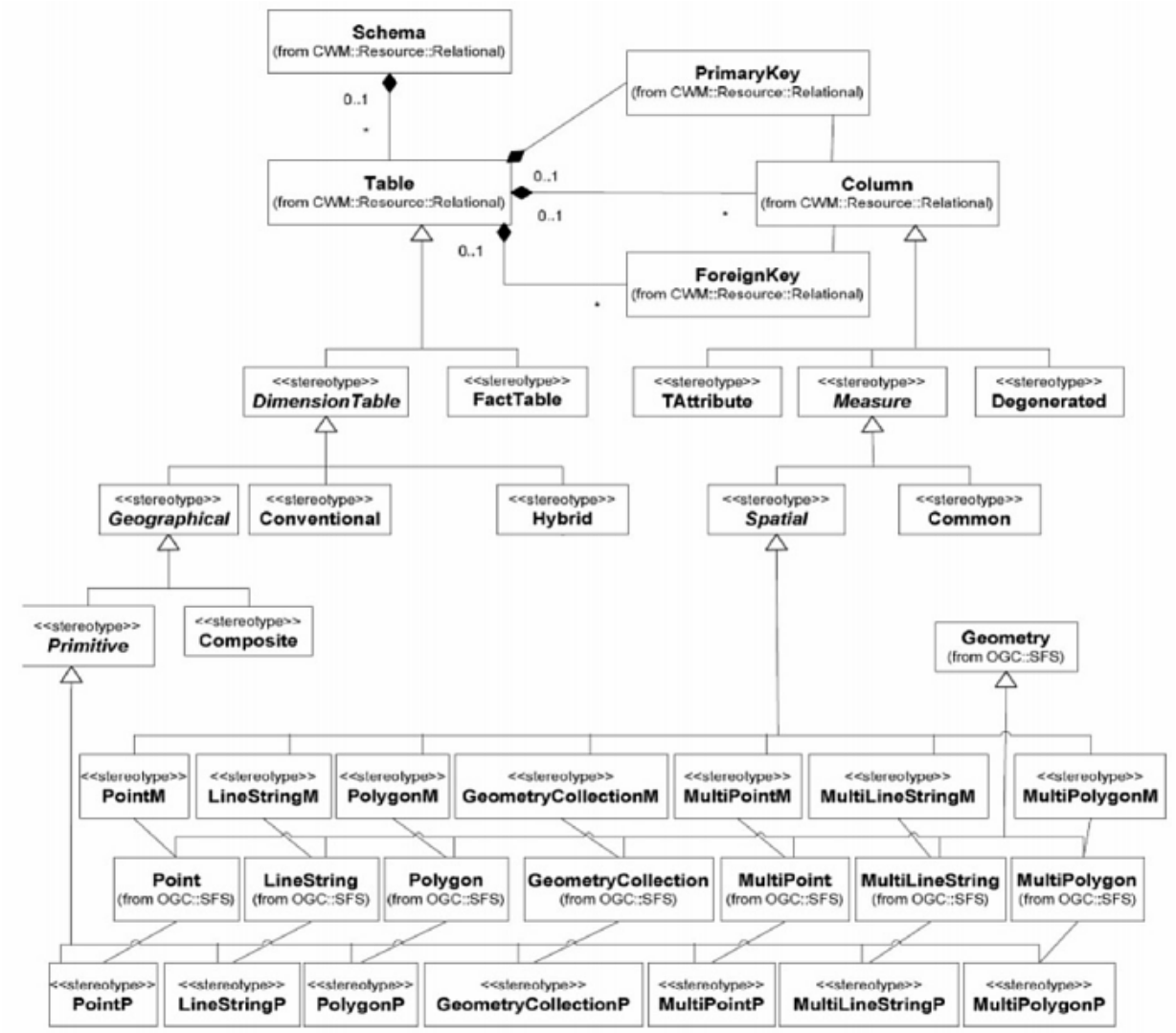

Abbildung 17: Metamodell für geographische Daten im Data Warehouse (Da Silva, De Oliveira, Fidalgo, Salgado, & Times, 2007, S. 595f)

Alternativ zum Metamodell von (Da Silva, De Oliveira, Fidalgo, Salgado, & Times, 2007) stellen (Rivest, Bedard, Proulx, Nadeau, Hubert, & Pastor, 2005) die SOLAP Technologie für Data Warehouse vor. SOLAP bedeutet Spatial On-Line Analytical Processing. Ein SOLAP-System führt zu den normalen Dimensionen drei räumliche Dimensionen ein. Diese sind die nichtgeometrische räumliche Dimension, die geometrische räumliche Dimension und die gemischte räumliche Dimension. Abbildung 18 stellt diese bildlich dar.

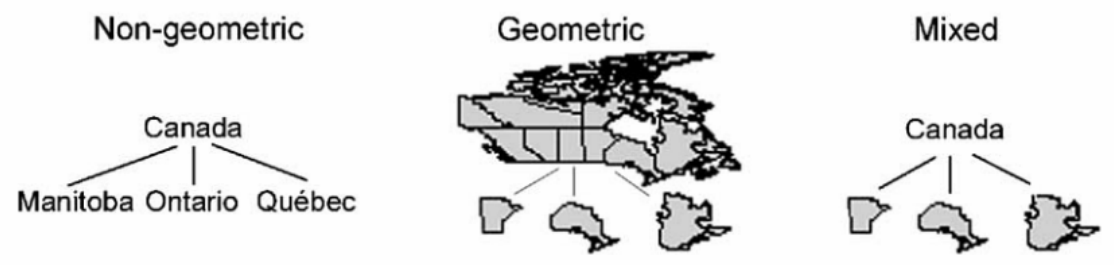

Abbildung 18: Räumliche Dimensionen in SOLAP (Rivest, Bedard, Proulx, Nadeau, Hubert, & Pastor, 2005)

Die nichtgeometrische räumliche Dimension erfasst beschreibende Attribute für geographische Elemente. Dies können `Ortsname`, `Höhe über dem Meeresspiegel`, `Einwohneranzahl` oder dergleichen mehr sein.

Eine geometrische räumliche Dimension kann nur geographische Attribute speichern. Die Fläche der Postleitzahlengebiete für Deutschland kann als Multipolygonquelle genannt werden. Diese kann frei heruntergeladen werden (www.sourceforge.net/project/showfiles.php?group_id=88554&package_id=118698&release_is=287362).

Als letzte räumliche Dimension wird die gemischte räumliche Dimension angeführt. Diese ist in der Lage, beide Arten von Information in einer Dimension zu halten. Das bedeutet, die Einbindung geographischer Attribute ist neben beschreibenden Attributen möglich.

Nur die nichtgeometrische räumliche Dimension kann mit den herkömmlichen OLAP-Werkzeugen bearbeitet werden. Für die geometrische und gemischte geometrische Dimension sind eigene Werkzeuge erforderlich, welche in SOLAP verfügbar sind.

Bei Measures werden zwei Arten von räumlichen Measures unterschieden. Die erste Art von Measures kombiniert alle in den Dimensionen vorhandenen geographischen Formen, die zweite Art von Measures ergibt sich durch Berechnung oder Aggregation der räumlichen geographischen Attribute.

Für die Auswertung geographischer Daten unterstützt SOLAP „Roll-up“, „Drill-down“ aber auch „slice“ oder „dice“ in der Ausprägung geographische Daten. Eine geovisuelle Darstellung unterstreicht diese Auswertemöglichkeiten.

(Rifaie, et al., 2008, S. 179) beschreiben einen etwas anderen Ansatz zur Modellierung geographischer Daten in einem Data Warehouse. Hier liegt der Blickwinkel auf den beschreibenden Attributen für geographische Daten.

Die räumlichen geographischen Daten werden aus dem Datenmodell ausgeschieden und in eigene Datenarrays abgelegt. In das Datenmodell werden Referenzen zu den geographischen Daten gehalten.

Die Modellierung des Data Warehouse erfolgt unter Einhaltung der Vorgehensmodelle, welche in Kapitel 2.3 diskutiert wurden. Die beschreibenden Attribute für die geographischen Daten werden wie nichtgeographische dimensionale Attribute oder Measures behandelt. Die Einbindung in das Data Warehouse kann demzufolge als Dimensionsattribute oder als Fakten in einer Faktentabelle erfolgen.

Abbildung 19 liefert eine Ansicht des Datenmodells von (Rifaie, et al., 2008) mit den eingebundenen geographischen Beschreibungsattributen. Aus Übersichtsgründen wurde auf die Darstellung aller weiteren dimensionalen und nichtdimensionalen Attribute sowie Measures verzichtet.

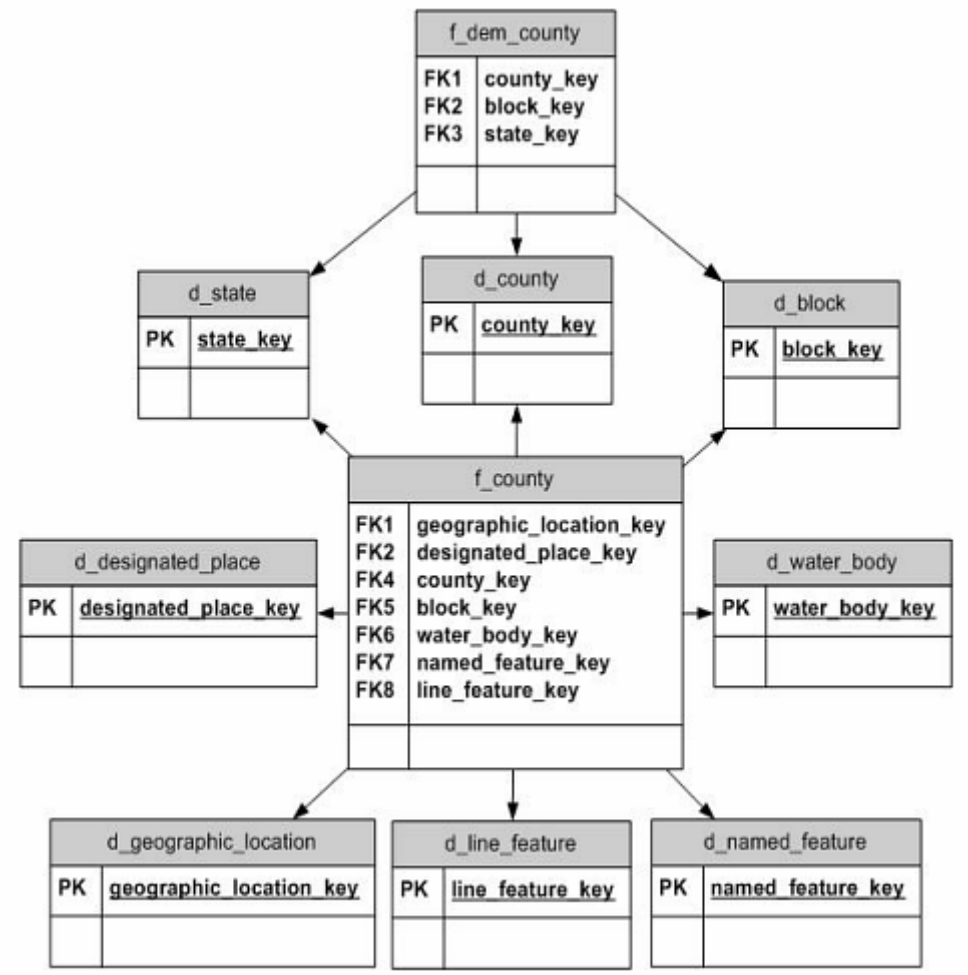

Abbildung 19: Modell eines Data Warehouse mit geographischen Daten nach (Rifaie, et al., 2008, S. 184)

In Abbildung 19 kann man erkennen, dass es für alle geographischen Attribute Platzhalter gibt, welche auf das externe Datenarray verlinken. Den Vorteil dieses Ansatzes beschreibt (Rifaie, et al., 2008) mit der Einfachheit des Aufbaus und der effizienten Arbeitsweise des Data Warehouse.

3 Die derzeitige Istsituation im Unternehmen OÖVV

In diesem Kapitel wird auf die Ausgangssituation dieser Diplomarbeit eingegangen. Als erstes wird der Aufbau des bestehenden Data Warehouse im Unternehmen OÖVV beschrieben. Die einzelnen Dimensionen und die Faktentabelle mit den Measures werden erläutert. Der Weg der Daten von den Quellen bis in das Data Warehouse (d.h. die ETL-Prozesse im OÖVV) wird in einem weiteren Abschnitt erklärt.

Der Verkehrsverbund Oberösterreich betreibt, wie in der Einleitung bereits erwähnt, ein Data Warehouse zur Analyse der Fahrscheindaten. Darin werden die Verkaufsdaten der Fahrscheine erfasst und für verschiedene Analysen, wie z.B. Verkaufszahlen gruppiert nach Mobilitätskreisen für die letzen 5 Jahre, aufbereitet.

Das bestehende Data Warehouse soll für weitere Fragestellungen um die Fahrplandaten sowie um geographische und demographische Daten erweitert werden. Die Daten liegen dem Verkehrsverbund zwar vor, jedoch sind diese in verschiedenen Quellen verteilt. Dazu muss erst die Istsituation erläutert werden, bevor die Analyse, der Entwurf und die Implementierung der Änderungen am bestehenden System erläutert werden.

3.1 Das bestehende Data Warehouse

Im bestehenden Data Warehouse werden derzeit die Daten des Verkaufs der Fahrkarten repliziert. Das Datenschema wird in Abbildung 20 dargestellt.

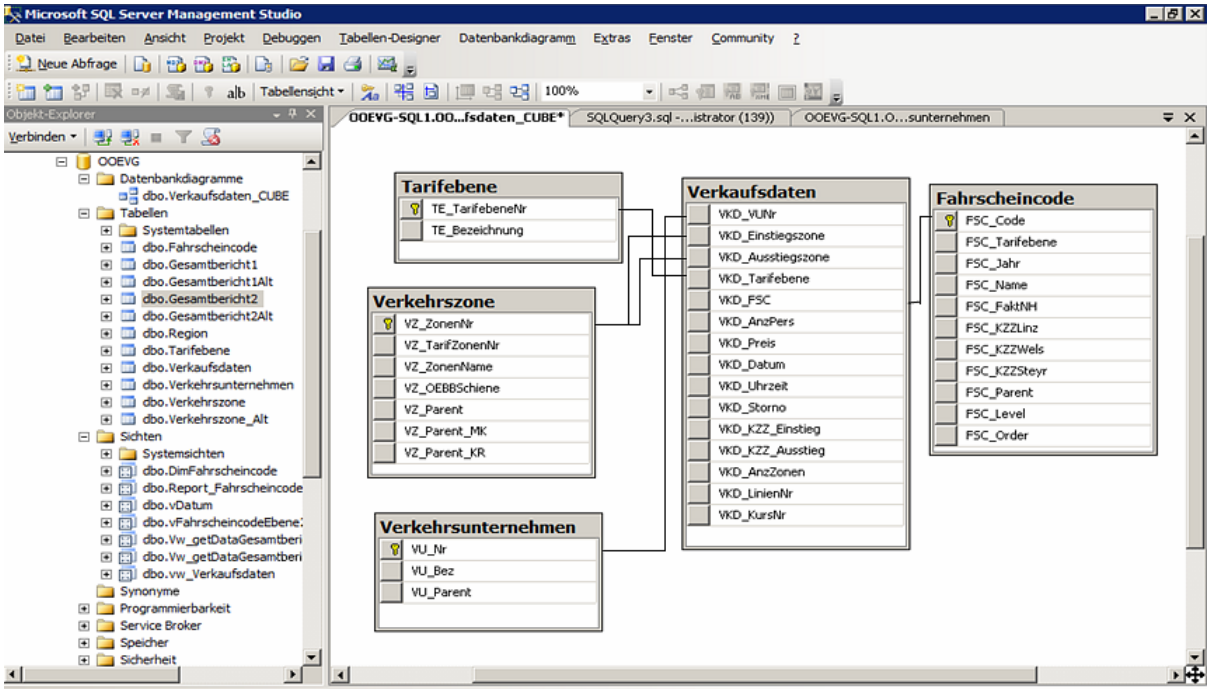

Abbildung 20: Datenschema des bestehenden Data Warehouse im OÖ Verkehrsverbund

Dieser Screenshot zeigt die Tabellensicht des bestehenden Data Warehouse. In der Mitte befindet sich die Faktentabelle `Verkaufsdaten`. Diese referenziert die vier Dimensionen `Tarifebene`, `Verkehrszone`, `Verkehrsunternehmen` und `Fahrscheincode`. Jede dieser Dimensionen ist als Tabelle hinterlegt. Das Datenschema ist als Starschema ausgebildet, wie bereits unter 2.1.4 vorgestellt.

3.1.1 Die Dimensionen im bestehenden Data Warehouse

Die Dimension `Tarifebene` ist die einfachste Dimension und besteht nur aus der `Tarifebenennummer` und einer `Tarifebenenbezeichnung`. Dabei stellt die `Tarifebenennummer` den Primärschlüssel zur Verfügung.

Die Dimension `Verkehrszone` beschreibt die vorhandenen Zonen aus Sicht der Verkaufsdaten. Als Schlüsselelement dient die eindeutige `Zonennummer`. Weiters ist eine `Tarifzonennummer` vergeben. Diese wird als Dimensionsattribut zur Darstellung der Measures nach den `Tarifzonennummern` benötigt. Die Zonen verfügen ebenfalls über einen `Zonennamen`, welcher die einzelne Zone lesbar macht. Für die Anbindung an das Fahrplannetz

der ÖBB dient das Attribut „OEBBSchiene“, das eine Referenznummer zu Zugverbindungsnummern der ÖBB enthält.

Die Hierarchisierung der Zonen erfolgt über die drei weiteren Felder Parent, Parent_MK und Parent_KR. Das Feld Parent beinhaltet eine für diese Zone übergeordnete Zone. Damit kann eine rekursive Hierarchisierung erstellt werden, welche ohne Veränderung des Datenschemas leicht anpassbar ist. Das Feld Parent_KR ordnet die Zone einer Konzeptregion zu. Das Feld Parent_MK gibt die Zuordnung der Zone zu einem Mobilitätskreis bekannt.

In der Dimension Verkehrsunternehmen findet sich eine VU_Nr, anhand derer ein Verkehrsunternehmen eindeutig referenziert werden kann. Darunter befindet sich die Bezeichnung des Verkehrsunternehmens. Weiters gibt es wieder ein Feld VU_Parent. Hier können Hierarchien über zusammenhängende Verkehrsunternehmen und Unternehmensgruppen als Parent-Child-Beziehung abgebildet werden.

Die Fahrschein-Dimension schließlich stellt die Details über den Fahrschein zur Verfügung. Das Feld für den Fahrscheincode beschreibt eine Nummer des Fahrscheintyps. Mögliche Ausprägungen umfassen zum Beispiel Einzelfahrschein, Tageskarte, Wochenkarte, Einzelfahrt ermäßigt und so weiter. Diese Fahrscheintypen gibt es für verschiedene Tarifebenen. Im nächsten Feld FSC_Tarifebene wird diese Tarifebene registriert. Das Feld Jahr beinhaltet die Jahreszahl für diesen Fahrscheintyp. Da es jährlich zu Tarifänderungen und Anpassungen in den Fahrscheinmodellen kommen kann, wird zusätzlich eine Zeitzuordnung zur Historisierung gespeichert. Im nächsten Feld findet sich eine Klartextbeschreibung zum Fahrscheincode. Mit dem Fakt_NH wird die Nutzungshäufigkeit für den Fahrschein angegeben. Während für den Einzelfahrschein klar ist, wie oft dieser genutzt wird, so ist es für Wochenkarten, Monatskarten oder gar Jahreskarten nicht mehr selbstverständlich, wie oft diese genutzt werden. Die Werte resultieren aus Erfahrungswerten für verschiedene Fahrscheintypen. Tatsächliche Werte zur Nutzungshäufigkeit sind derzeit nicht verfügbar, da diese nur aufwändig zu erheben wären. Die nächsten drei Felder (KZZ_Linz, KZZ_Wels und KZZ_Steyr) geben an, ob eine der drei Kernzonen mit gekauft wurden oder nicht. Mit diesen Feldern ist es möglich, das Verhältnis von Fahrkarten gesamt zu Fahrkarten mit Kernzone auszuwerten. Das Feld Parent ermöglicht so wie bei den anderen Dimensionen wieder eine Parent-Child-Beziehung zwischen den einzelnen Fahrscheincodes. Die Felder Level und Order beinhalten eigene Gruppierungen, welche sich nicht durch eine klassische Parent-Child-Beziehung abbilden lassen.

3.1.2 Die Faktentabelle im bestehenden Data Warehouse

Die Verkaufsdaten in der Abbildung 20 stellen die Faktentabelle dar. Hier dient als Referenz zur Dimension Verkehrsunternehmen die VUNr. Die Einstiegszone gibt die Zone an, ab welcher der Fahrschein gilt. Die Ausstiegszone gibt an, bis zu welcher Zone die Fahrkarte gilt. Die Einstiegszone wie auch die Ausstiegszone referenzieren die Dimension Verkehrszone.

Das Feld `Tarifebene` gibt die `Tarifebenennummer` an und bildet damit die Referenz zur Dimension `Tarifebene`. FSC steht für `Fahrscheincode` und stellt die Beziehung zur Dimension `Fahrscheincode` her. Mit der Angabe von `AnzPers` wird die Personenanzahl, für die der Fahrschein gilt, angegeben. Im nächsten Feld findet sich der `Preis` in Euro-Cent für den Fahrschein. Werden mehrere Personen angegeben, so gilt der Preis für alle Personen. Der Preis für die Einzelperson muss dann durch die Anzahl der Personen geteilt werden.

`Datum` und `Uhrzeit` geben den Gültigkeitsbeginn des Fahrscheins an. Dieser ist normalerweise gleich mit dem Verkaufsdatum, muss es aber nicht sein. Im nächsten Feld `Storno` wird angegeben, ob der Fahrschein nach dem Ausstellen wieder storniert wurde. Dieser Eintrag dient zur Überprüfung, ob es zu vermehrten Stornierungen in einem bestimmten Gebiet oder bei einem bestimmten Unternehmer kommt. Wird hier eine Auffälligkeit festgestellt, so kann gezielt nachgeforscht werden, warum dort Häufungen der Stornierungen auftreten.

Das Feld `KZZ_Einstieg` und das Feld `KZZ_Ausstieg` beschreiben den Kernzonenzuschlag je für die Einstiegsstelle oder die Ausstiegsstelle. Mit der Angabe `AnzZonen` wird die Gesamtanzahl der durchfahrenen Zonen gespeichert. Hier zählen die Startzone und die Endzone ebenfalls dazu. Die `Liniennummer` gibt an, für welche Buslinie der Fahrschein gekauft wurde. Im letzten Feld wird die `Kursnummer` angegeben, auf welcher der Fahrschein gekauft wurde. Die `Kursnummer` dient dazu, in Verbindung mit der `Liniennummer` jede Busverbindung eindeutig zu identifizieren.

Bestimmte Strecken werden von bestimmten Linien befahren. Zum Beispiel führt die Linie 230 von Rohrbach nach Linz und wieder zurück. Die Kursnummer gibt an, welcher Bus auf dieser Linie betroffen ist. Diese werden tageweise von 1 weg nach oben gezählt. Als Beispiel kann der Bus um 05:15 Abfahrt in Rohrbach als Kurs 02 für die Linie 230 angeführt werden. Die Kombination von Kurs und Linie ergibt eine eindeutige Zuordnung für jede Verbindung.

3.2 Der Weg der Daten von der Datenquelle in das Data Warehouse

Die Verkaufsdaten werden von verschiedenen Fahrscheinautomaten (Abbildung 21 unten) während der Ausgabe eines Fahrscheins generiert und gespeichert. Diese Daten werden dann in bestimmten individuell vereinbarten Intervallen vom Verkehrsunternehmen an den Verkehrsverbund gesammelt übermittelt.

Für die Übermittlung der Daten wurde ein Standardabrechnungsdatensatz, kurz SAD, eingeführt. Dieser ist textbasiert, umfasst 230 Byte und bietet Felder für die `Fahrscheinart`, den `Zonenweg`, den `Preis`, das `Verkaufsdatum`, die `Fahrernummer`, die `Gerätenummer`, die `Anzahl` der Personen und mehr. Der SAD wurde für mehrere Verkehrsverbünde entwickelt.

Die Standardabrechnungsdatensätze (SAD) werden aus der Übermittlungsdatei in eine Oracle-Datenbank eingelesen und dort unter Vergleich mit den Fahrplandaten migriert. Die migrierten Daten werden dann verschiedenen Empfängern, wie zum Beispiel dem Bund zur

Abrechnung oder den Gemeinden zur Darstellung der Einnahmenaufteilung, zur Verfügung gestellt. Abbildung 21 bildet im oberen Teil einen Auszug der Empfänger ab.

Einige der Empfänger erwarten das Ergebnis der Auswertung in Form von einer Microsoft Access Datenbank (Abbildung 21 unter Datenzugriff). Das Ergebnis der Auswertung ist so aufbereitet, dass dieses mit dem Datenbankschema für das Data Warehouse korreliert. Für das Data Warehouse werden die Daten dann aus der Access-Datenbank manuell mittels „Drag and Drop“ in die Faktentabelle kopiert.

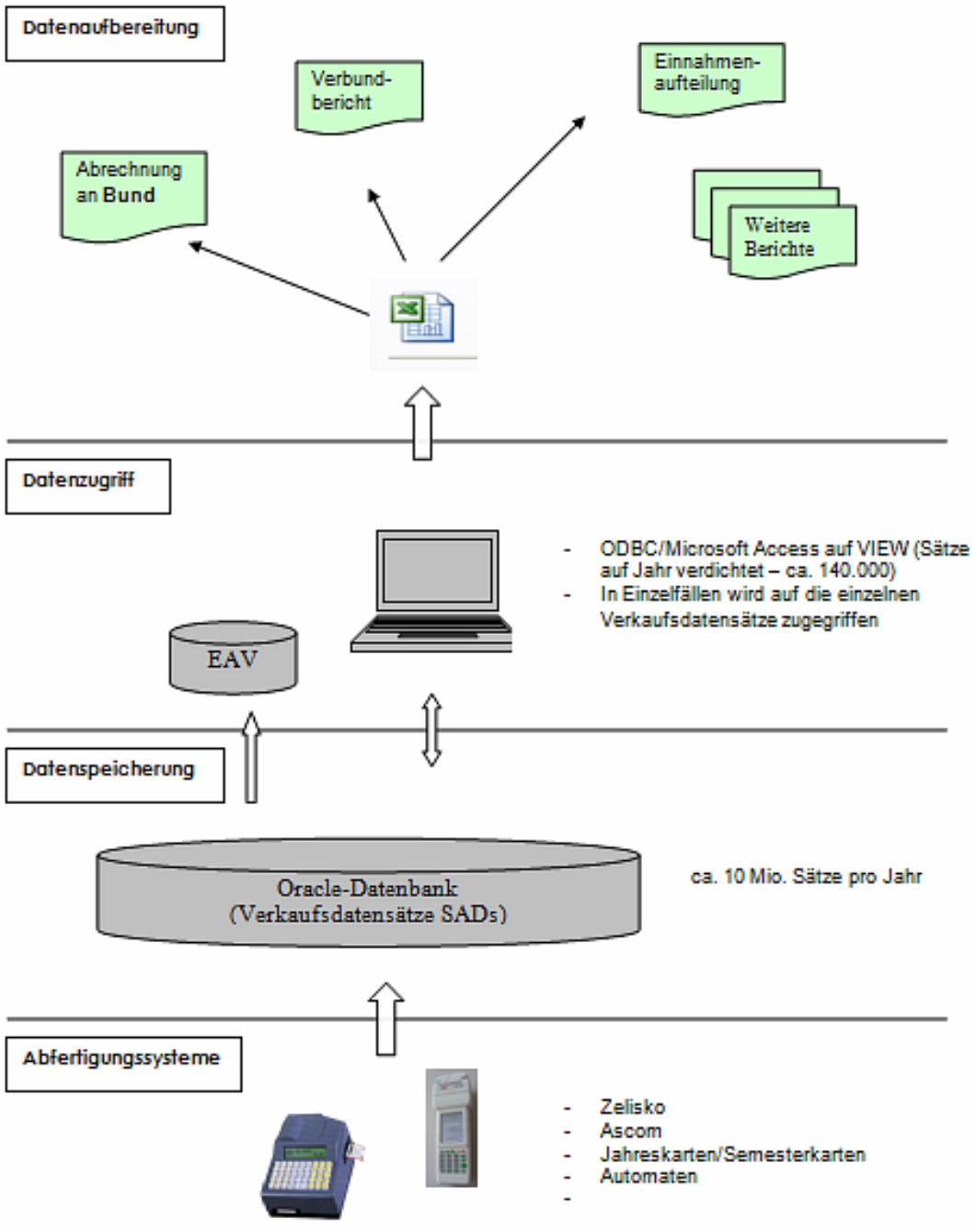

Abbildung 21: Graphische Darstellung des Imports der Daten aus den Abfertigungssystemen

Die Dimensionen werden aus der Tabelle „ZentraleTab.xls“ ebenfalls manuell kopiert und in die Dimensionstabellen eingefügt.

4 Problembeschreibung

Die neuen Anforderungen an die Analyse (als Beispiel Verkaufsverteilung unter Berücksichtigung der Bevölkerung pro Fläche) erfordern auch neue Datenquellen. Hier gibt es grundsätzlich drei neue Quellen und deren Speichersystem zu unterscheiden:

- Fahrplandaten
- demographische Daten
- geographische Daten (Flächenwidmungspläne).

Die Fahrplandaten kommen aus einer Oracle-Datenbank. Diese werden über Textdateien exportiert. Ein direkter Zugriff auf diese Datenbank ist aus organisatorischen Gründen nicht vorgesehen.

Die Bevölkerungszahlen werden in Form einer Excel-Tabelle des statistischen Amtes der Landesregierung Oberösterreich zur Verfügung gestellt. Ein Zugang über Web-Service wäre möglich. Da dieser aber über das zentrale Melderegister abgewickelt würde, wäre dieser kostenpflichtig. Daher wurde diese Möglichkeit von vornherein ausgeschieden.

Die Flächendaten kommen ebenfalls vom statistischen Amt der Landesregierung Oberösterreich und liegen auch im Excel-Format vor. Online-Quellen sind wiederum kostenpflichtig und damit nicht relevant für dieses Projekt.

Für die Fahrplandaten liegt eine Spezifikation in schriftlicher Form vor, die statistischen Daten in den Excel-Tabellen sind selbstbeschreibend. Im weiteren Verlauf des folgenden Kapitels werden diese Daten erläutert.

4.1 Aufbau der Fahrplandaten

Die Fahrplandaten bestehen aus 13 Textdateien, in denen die Daten abgelegt sind. Der Aufbau und das Format wurden von den österreichischen Bundesbahnen (ÖBB) übernommen. Damit existieren in den Daten mehr Felder als für den Verkehrsverbund benötigt. Auch die Namensgebung verschiedener Elemente und Dateien erinnert sehr an die bei den österreichischen Bundesbahnen verwendeten Namen. (HaCon, 2004) beschreibt den Inhalt der einzelnen Dateien. Die Dateien werden dabei in notwendige und optionale Dateien geteilt.

Notwendige Dateien:

- Das Haltestellenverzeichnis `BAHNHOF`
- Die Koordinaten der Haltestellen `BFKOORD`
- Der Fahrplan `FPLAN`
- Fahrplangültigkeit `ECKDATEN`
- Verkehrstagebeschränkungen `BITFELD`

- Verkehrsmittel bzw. Gattung ZUGART
- Verbindungen zwischen Haltestellen METABHF
- Haltestellenbezogene Umsteigezeiten UMSTEIGB

Optionale Dateien:

- Zusätzliche Attribute ATTRIBUT
- Durchbindungen DURCHBI
- Zusätzliche Fahrplaninformationen FPLAN_PLUS
- Linieninformationen LINIEN
- Linienbezogene Umsteigezeiten UMSTEIGL

Der Aufbau dieser Dateien wird beschrieben, da dieser für die Modellierung des erweiterten Datenschemas des Data Warehouse im OÖ Verkehrsverbund wichtig ist.

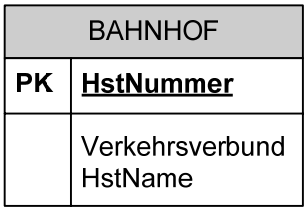

Abbildung 22: Darstellung der Datenquelle BAHNHOF als Klassendiagramm

Die Datei BAHNHOF beinhaltet die Haltestellennummer, eine Nummer für den zugehörigen Verkehrsverbund und eine Haltestellenbezeichnung.

Beispiel:

```
0050617     Abtenau Heimhofsiedlung
```

HstNummer: 0050617
Verkehrsverbund:
HstName: Abtenau Heimhofsiedlung

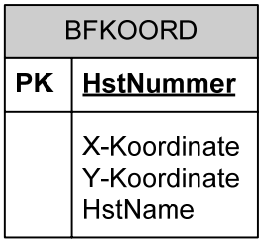

Abbildung 23: Darstellung der Datenquelle BFKOORD als Klassendiagramm

In der Datei `BFKOORD` befinden sich neben der `Haltestellennummer` noch die beiden geographischen Koordinaten x und y. Weiters findet sich zur besseren Lesbarkeit der Name der Haltestelle.

Die Koordinaten x und y geben den Haltestellenort in Format eines kartesischen Koordinatensystems nach dem UTM-System (Universal-Transversal-Mercator-Projection) an. Das UTM-System teilt die Erde zwischen 80° südliche Breite und 84° nördlicher Breite in 60 Meridianstreifen mit je 6° Ausdehnung. Weiters erfolgt eine Trennung nach Nordhalbkugel und Südhalbkugel. Innerhalb der entstandenen Rechtecke findet eine Punktfestlegung mittels rechtwinkeligem kartesischen Koordinatensystem statt (Hake & Grünreich, 1994).

Oberösterreich befindet sich im Meridianstreifen 33 nördliche Halbkugel bei einem Mittelmeridian 15° östlicher Länge (Amt der oö. Landesregierung Abteilung Geoinformation und Liegenschaft, 2010). Eine genaue Bezeichnung nach dem UTM-System lautet UTM 33N. Die Angaben in diesem Fall erfolgen in Meter.

Beispiel:

```
0050617 375183,608 5269976,71        % Abtenau Heimhofsiedlung
```

HstNummer:	0050617
X-Koordinate:	375183,608
Y-Koordinate:	5269976,71
HstName:	Abtenau Heimhofsiedlung

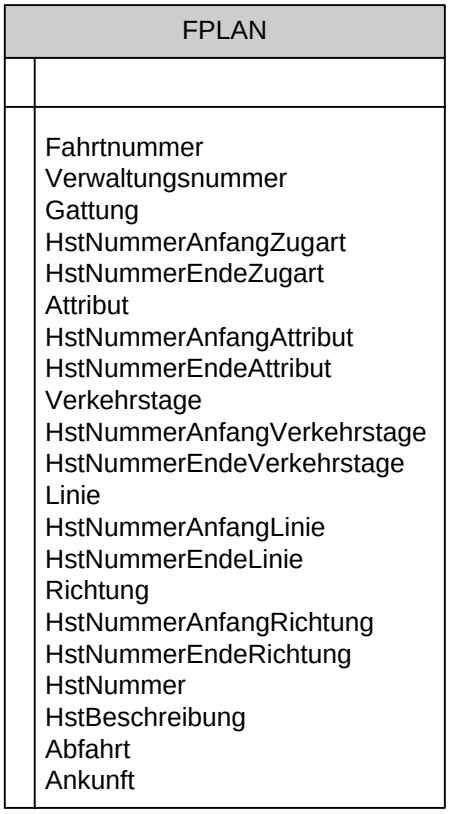

Abbildung 24: Darstellung der Datenquelle FPLAN als Klassendiagramm

Die Datei `FPLAN` beinhaltet den kompletten Fahrplan für alle Linien und alle Kurse. Der Fahrplan teilt sich in einen Header und einen Body. Im Header stehen allgemeine Daten wie `Fahrtnummer`, `Fahrplantage` oder `Gattung` der Verbindung. Im Body sind die einzelnen Haltestellen mit Abfahrtszeit und Ankunftszeit angeführt. Die vier Felder `Haltestelle`, `Haltestellenbezeichnung`, `Haltestelle Ankunft` und `Haltestelle Abfahrt` wiederholen sich nach Bedarf. Nach dem Prozentzeichen ist ein Bereich für Kommentar vorgesehen, der aber ebenfalls noch nützliche Information beinhaltet. Leider ist zur in den Kommentaren enthaltenen Information keinerlei Dokumentation mehr verfügbar, auch nicht seitens des OÖVV. Daher werden nur jene Felder aus dem Kommentar beschrieben, welche durch Referenzen zu anderen Dateien geprüft und deren Bedeutung erhoben wurde.

Beispiel:

```
*Z 00202 EE663                                                    % 202
*G Bus 0042958 0042640                                              %
*A VE 0042958 0042640 000005                                % KAT 1 KRZ FØ
*L 663SV 0042958 0042640                                    % LID 710 KSN 3
*R            1            0042640            0042958            0042640
0042958 Finklham Abzw St.Mar          0623                          %
0042961 Unterfreundorf Reha- 0625 0625                              %
0045116 St.Marienkirchen/Pol 0631 0631                              %
0042640 St.Marienkirchen/Pol 0633                    %
```

Fahrtnummer:	202
Verwaltung Fahrt 1:	EE663
Gattung:	Bus
Gattung von Haltestelle:	0042958
Gattung bis Haltestelle:	0042640
Bitfeld von Haltestelle:	0042958
Bitfeld bis Haltestelle:	0042640
Bitfeld:	000005
Linie	663SV
Linie von Haltestelle:	0042958
Linie bis Haltestelle:	0042640
Linien ID (LID im Kommentar)	710
Kursnr. (KSN im Kommentar)	3
Richtung (0 = hin 1 = zurück)	1
Richtung von Haltestelle:	0042958
Richtung bis Haltestelle:	0042640

Haltestelle	0042958
Haltestellenbezeichnung:	Finklham Abzw St.Mar
Haltestelle Ankunft:	
Haltestelle Abfahrt:	0623
Haltestelle	0042961
Haltestellenbezeichnung:	Unterfreundorf Reha-
Haltestelle Ankunft:	0625
Haltestelle Abfahrt:	0625
Haltestelle	0045116
Haltestellenbezeichnung:	St.Marienkirchen/Pol

Haltestelle Ankunft:	0631
Haltestelle Abfahrt:	0631
Haltestelle	0042640
Haltestellenbezeichnung:	St.Marienkirchen/Pol
Haltestelle Ankunft:	0633
Haltestelle Abfahrt:	0633

Abbildung 25: Darstellung der Datenquelle ECKDATEN als Klassendiagramm

In der Datei `ECKDATEN` sind der `Fahrplanbeginn` und das `Fahrplanende` eingetragen. Diese beiden Attribute begrenzen den Gültigkeitszeitraum des Fahrplans.

Beispiel:

```
13.12.2010 Fahrplanstart
10.12.2011 Fahrplanende
2010-12-13
```

Fahrplanstart:	13.12.2010
Fahrplanende:	10.12.2011
Fahrplanbezeichnung:	2010-12-13

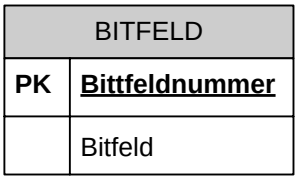

Abbildung 26: Darstellung der Datenquelle BITFELD als Klassendiagramm

Der Inhalt der Datei `BITFELD` beschreibt die Tage, für welche die `Bitfeldnummer` gilt. Die `Bitfeldnummer` referenziert das `Bitfeld`. Im `Bitfeld` wird ein Feld mit 380 Tagen bestimmt, 366 Tage für ein Schaltjahr plus 14 Tage Überlappungszeit, sollte der nächste Fahrplanstart nicht genau am gleichen Tag im Folgejahr starten. Ist die `Bitfeldnummer` für den Tag gültig, so wird dieser mit 1 belegt, ist die `Bitfeldnummer` für diesen Tag ungültig, so wird eine 0 eingetragen. Diese bitcodierte Abfolge wird HEX-codiert (4Bit = 1HEX) im Bitfeld abgelegt.

Beispiel:

```
000001 FE78F1D3E7CF9F3E7CF9F3E7CF9F3E7CF8F3E7CF9F3A7C79D3E7CF9F3E7CF8F3E7CF
9F3E7CF9F3E6CB9F3E7CF9D60000
```

Bitfeldnummer:	000001

Bitfeld: `FE78F1D3E7CF9F3E7CF9F3E7CF9F3E7CF8F3E7CF9F3A7C79D`
`3E7CF9F3E7CF8F3E7CF9F3E7CF9F3E6CB9F3E7CF9D60000`

Wandelt man die ersten vier Zeichen aus dem `Bitfeld` in deren binäre Darstellung um, so ergibt sich aus „FE78“ die binäre Darstellung „1111 1110 0111 1000“. Der `Fahrplan-start` (Eckdaten) beginnt mit 13.12.2010. Tabelle 6 stellt den Zusammenhang zwischen `Bitfeld`, der äquivalenten binären Darstellung und dem `Fahrplandatum` gegenüber.

Zeichen aus Bitfeld	*Binärer Wert*	*Referenziertes Datum*	*Aktion gültig*
F	1	13,12.2010	Ja
	1	14,12.2010	Ja
	1	15,12.2010	Ja
	1	16,12.2010	Ja
E	1	17,12.2010	Ja
	1	18,12.2010	Ja
	1	19,12.2010	Ja
	0	20,12.2010	Nein
7	0	21,12.2010	Nein
	1	22,12.2010	Ja
	1	23,12.2010	Ja
	1	24,12.2010	Ja
8	1	25,12.2010	Ja
	0	26,12.2010	Nein
	0	27,12.2010	Nein
	0	28,12.2010	Nein

Tabelle 6: Zusammenhang zwischen referenziertem Datum und Bitfeld

Tabelle 6 listet in der ersten Spalte das Zeichen aus dem `Bitfeld` auf. In der zweiten Spalte wird der Binärcode aus dem `Bitfeld`-Zeichen aufgelöst. Das zum `Bitfeld` passende Datum wird in Spalte drei angegeben. Dieses beginnt mit dem Datum `Fahrplanstart` und erhöht sich mit jedem Bit um einen Tag. In der Spalte vier wird angegeben, ob die zum `Bitfeld` zugeordnete Aktion gültig ist oder nicht. Wie leicht zu erkennen ist, sind die Spalten zwei und vier äquivalent. Eine Aktion kann sein, dass eine Fahrt an diesem Tag stattfindet oder nicht.

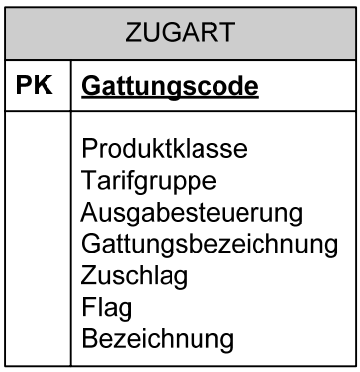

Abbildung 27: Darstellung der Datenquelle ZUGART als Klassendiagramm

Die Datei ZUGART beschreibt die Art des Verkehrsmittels. Als Schlüssel dient das Attribut Gattungscode. In diesem steht die Kurzform des Verkehrsmittels. Als nächstes wird die Produktklasse angegeben. Hier können die Verkehrsmittel zu Gruppen zusammengefasst werden. Die nächste Angabe ist die Tarifgruppe, die zur Fahrpreisberechnung benötigt wird. Dann folgt das Feld Ausgabesteuerung, das spezifiziert, ob am Plan nur die Gattung, nur die Nummer oder beides ausgegeben wird. Als nächstes erfolgt die Gattungsbezeichnung für die Ausgabe. Ist für das Verkehrsmittel ein Zuschlag fällig, so wird dies im nächsten Feld vermerkt. Mit dem folgenden Flag können weitere Eigenschaften deklariert werden. In der Beschreibung (HaCon, 2004, S. 42) wurde als Beispiel N für Nahverkehr angegeben. Schließlich kann noch eine Beschreibung im Feld Bezeichnung angegeben werden.

Beispiel:

```
SCH 1 1 1 SCH      0 N
```

Gattungscode:	SCH
Produktklasse:	1
Tarifgruppe:	1
Ausgabesteuerung:	1
Gattungsbezeichnung:	SCH
Zuschlag:	0
Flag:	N
Bezeichnung:	

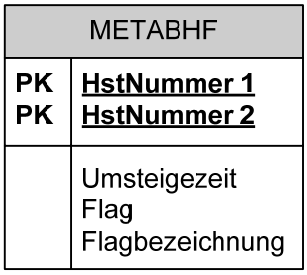

Abbildung 28: Darstellung der Datenquelle METABHF als Klassendiagramm

Die Datei METABHF beschreibt Umsteigemöglichkeiten zwischen zwei Haltestellen. Hier werden die beiden Haltestellennummern angegeben. Dazu wird im Feld Umsteigezeit die Zeitdauer für das Umsteigen angegeben. Die Bedeutung der beiden Einträge Flag und Flagbezeichnung konnte mangels Verfügbarkeit einer geeigneten Dokumentation nicht geklärt werden. Da das Schema von den ÖBB übernommen wurde, werden nicht alle Felder vom Verkehrsverbund genutzt. Nach den Einträgen zu urteilen, gibt es eine Verbindung zu den Liniennummern. Leider ist der Zusammenhang nicht schlüssig.

```
0040105 0042062 02    %
*L 0000172
```

Haltestelle 1:	0040105
Fahrplanende:	0042062

Umsteigezeit in Minuten: 02
Flag: *L
Flag-Bezeichnung: 0000172

UMSTEIGB	
PK,FK1	**HstNummer**
	UmsteigezeitIC Umsteigezeit HstName

Abbildung 29: Darstellung der Datenquelle UMSTEIGB als Klassendiagramm

In der Datei `UMSTEIGB` werden die Umstiegszeiten in Minuten zwischen verschiedenen Linien auf einem Bahnhof festgelegt. Da das Schema von den ÖBB übernommen wurde, gibt es zwei unterschiedliche Umstiegszeiten. Die erste Umstiegszeit betrifft den Umstieg zwischen zwei IC-Zügen. Dieser Wert wird auch für den Verkehrsverbund übernommen. Alle anderen Verbindungen bei den ÖBB verwenden den zweiten Wert, der jedoch für den Verkehrsverbund nicht von Bedeutung ist und deshalb standardmäßig mit 15 Minuten festgesetzt ist.

Beispiel:

```
0050620 1 15 Abtenau Fischbach
```

HstNummer: 0050620
Umsteigezeit IC: 1
Umsteigezeit: 15
HstName: Abtenau Fischbach

ATTRIBUT	
PK	**Attributscode**
	Haltestellenzugehoerigkeit Prioritaet Sortierung Attributstext

Abbildung 30: Darstellung der Datenquelle ATTRIBUT als Klassendiagramm

Die Datei `ATTRIBUT` liefert Zusatzangaben zum Fahrplan, die meist am Fahrplan unten angedruckt sind. Hier findet man unter anderem die Information, dass es sich bei einer Busverbindung zum Beispiel um einen Rufbus handelt.

Beispiel:

```
01 0 001 00 Rufbus: Tel.+43 664 9114244, Fahrtanmeldung 30 Min. vor Abfahrt, Betrieb durch Fa. Brunner, 4490 St.Florian
```

Attributscode: 01
Zuordnung zu Haltestelle: 0

Priorität (0=höchste Priorität): 001
Sortierung (0 = 1. Platz): 0
Attributstext: `Rufbus: Tel.+43 664 9114244, Fahrtanmeldung 30 Min. vor Abfahrt, Betrieb durch Fa. Brunner, 4490 St.Florian`

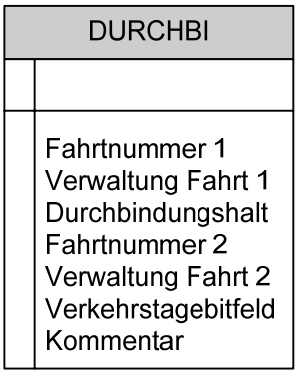

Abbildung 31: Darstellung der Datenquelle DURCHBI als Klassendiagramm

In der Datei `DURCHBI` finden sich Daten für Durchbindungen von Fahrten. Eine Durchbindung liegt dann vor, wenn zwei Fahrten so zusammengefasst werden, dass logisch eine Fahrt daraus wird. Dazu werden die beiden Fahrten mit `Fahrtnummer` und `Verwaltung` sowie der `Durchbindungshalt` und die Verkehrsdaten registriert. Im Kommentar steht der Haltestellenname.

Beispiel:

```
174 EE 0050617 41 EE 1C Abtenau Heimhofsiedlung
```

Fahrtnummer 1: 174
Verwaltung Fahrt 1: EE
Durchbindungshalt: 0050617
Fahrtnummer 2: 41
Verwaltung Fahrt 2: EE
Verkehrstagebitfeld: 1C
Kommentar: Abtenau Heimhofsiedlung

Abbildung 32: Darstellung der Datenquelle FPLAN_PLUS als Klassendiagramm

Die Datei FPLAN_PLUS gibt Auskunft über die Reisezeit zwischen zwei Haltestellen. Dazu wird als erstes Feld die Linien-ID angegeben. Als zweites Feld folgt die Kursnummer. Danach kommen die Felder KAT für Kategorie und KRZ für Konzeptregionszeichen. Anschließend werden zwei Haltestellen in den Feldern Haltestelle1 (Von-Halt) und Haltestelle2 (Bis-Halt) angegeben. Das Feld Reisezeit gibt die Reisezeit zwischen den beiden Haltestellen an. Das Feld AFD wurde leider nicht beschrieben. Der Inhalt des Feldes AFD ist meistens 0.

Beispiel:

```
133   1     1     ¹     46024 46088 1     0
```

Linien-ID:	133
Kursnummer:	1
Kategorie:	1
Konzeptregionszeichen:	¹
Haltestelle 1:	46024
Haltestelle 2:	46088
Reisezeit in Minuten:	1
AFD:	0

LINIEN	
PK	**ID**
	LinienNr ProRegioNr Text

Abbildung 33: Darstellung der Datenquelle LINIEN als Klassendiagramm

Die Datei LINIEN in Abbildung 33 stellt weitere Information über die Linien zur Verfügung. Dabei wird als erstes eine Linien-ID angegeben. Als nächstes folgt eine weitere Liniennummer mit Richtungsangabe. Das nächste Feld beschreibt die ProRegio-Nummer. Eine ProRegio-Nummer stellt eine eigene Kennzeichnung der Buslinien für das Land Oberösterreich dar. Als letzter Eintrag folgt eine textuelle Beschreibung der Linie.

```
ID: 133, NR: 174h, ProRegioNR: 174, TEXT: Straßenbahn Gmunden: Franz-Josef-
Platz – Bahnhof
```

Linien ID:	133
Liniennummer mit Richtung:	174h
ProRegio Nummer:	672
Linienbeschreibung:	Straßenbahn Gmunden: Franz-Josef-Platz – Bahnhof

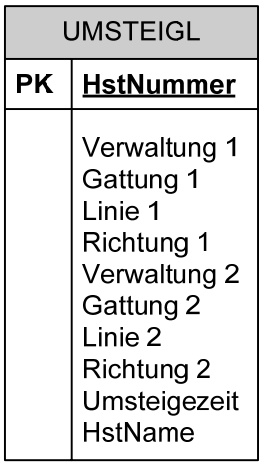

Abbildung 34: Darstellung der Datenquelle UMSTEIGL als Klassendiagramm

In der Datei `UMSTEIGL` werden die Umstiegszeiten zwischen zwei Linien festgelegt. Das erste Feld gibt die Umstiegshaltestelle an. In den nächsten vier Feldern `Verwaltung 1`, `Gattung 1`, `Linie 1` und `Richtung 1` wird die erste Linie definiert. Weitere vier Felder (`Verwaltung 2`, `Gattung 2`, `Linie 2` und `Richtung 2`) definieren die zweite Linie. Unter `Verwaltung` kann ein Verkehrsverbund angegeben werden. `Gattung` gibt das weiter oben in diesem Kapitel beschriebene Verkehrsmittel an. Die `Linie` bezeichnet die Buslinie (siehe Linien) und `Richtung` bezeichnet die Fahrtrichtung, 1 bedeutet Hinfahrt, 2 bedeutet Rückfahrt. Dann wird die Umstiegszeit angegeben. Das letzte Feld `HstName` beinhaltet die Haltestellenbeschreibung zur besseren Lesbarkeit.

Beispiel:

```
0045554 000000e Bus 000277 2 000000e Bus 000277 2 001 ! Haslach/Mühl Alten-
heim
```

HstNummer:	0050620
Verwaltung 1:	000000e
Gattung 1:	BUS
Linie 1:	000277
Richtung 1:	2
Verwaltung 2:	000000e
Gattung 2:	BUS
Linie 2:	000277
Richtung 2:	2
Umsteigezeit:	001
HstName:	Haslach/Mühl Altenheim

4.2 Aufbau der demographischen Daten

Die demographischen Daten sind in einer Excel-Tabelle (Zentralmelderegister, 2002-2009) gespeichert. Auf der Ordinate sind die Einzeljahre (= Alter in Lebensjahren) aufgetragen, auf der Abszisse die Gemeinden. Übergeordnet zu den Gemeinden gibt es eine Gruppierung nach Geschlecht. Tabelle 7 zeigt einen Ausschnitt aus der Tabelle. In den Zellen sind die Anzahl der Einwohner mit dem angegebenen Alter pro Gemeinde und Geschlecht, die in der jeweiligen Zeile eingetragen sind, angegeben. Für jedes Kalenderjahr gibt es eine eigene Tabelle.

Bevölkerung in den oö. Gemeinden und Bezirken nach Geschlecht, zusammen und Alter (Einzeljahre und 5-Jahres-Gruppen) per 1.1.2009

			Alter in Einzeljahren													
Geschl.	Gemnr.	Region	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Männer	40101	Linz (Stadt)	941	891	863	867	850	818	843	804	860	756	845	823	902	865
Männer	40201	Steyr (Stadt)	180	190	170	199	187	178	169	152	167	183	196	229	207	191
Männer	40301	Wels (Stadt)	339	326	318	322	318	339	329	289	284	341	308	341	379	373
Männer	40401	Altheim	27	19	15	24	20	22	25	9	24	23	31	30	32	27
Männer	40402	Aspach	13	14	14	11	18	10	17	15	12	10	15	22	22	15
Männer	40403	Auerbach	6	2	4	4	4	3	3	1	1	4	3	4	2	2
Männer	40404	Braunau am Inn	73	68	77	61	83	67	63	60	89	91	78	100	85	98
Männer	40405	Burgkirchen	16	11	17	16	12	10	12	18	14	16	16	12	15	10
Männer	40406	Eggelsberg	16	8	12	12	23	13	20	18	10	17	11	13	18	16
Männer	40407	Feldkirchen bei Mattighofen	5	7	10	3	5	10	14	14	15	15	17	13	10	13
Männer	40408	Franking	5	5	7	6	3	9	5	2	9	9	8	10	6	4
Männer	40409	Geretsberg	7	5	7	3	6	4	4	2	8	10	10	7	6	4
Männer	40410	Gilgenberg am Weilhart	5	4	5	7	7	1	4	6	6	10	7	12	8	10
Männer	40411	Haigermoos	3	3	1	3	3	1	2	4	1	7	9	1	3	3
Männer	40412	Handenberg	8	9	8	8	11	4	5	8	12	10	8	7	12	11
Männer	40413	Helpfau-Uttendorf	11	26	19	19	21	26	18	20	16	14	28	23	21	21
Männer	40414	Hochburg-Ach	8	14	11	9	11	11	14	15	14	17	18	18	12	17
Männer	40415	Höhnhart	7	9	8	8	9	4	7	9	6	6	7	9	14	12
Männer	40416	Jeging	2	4	7	5	6	2	2	6	3	4	3	6	3	6
Männer	40417	Kirchberg bei Mattighofen	3	7	4	5	6	5	3	10	5	1	3	7	6	6
Männer	40418	Lengau	16	18	18	20	18	31	28	12	23	22	25	34	33	25

Tabelle 7: Auszug aus den demographischen Quelldaten (Zentralmelderegister, 2002-2009)

4.3 Aufbau der geographischen Daten

Die geographischen Daten sind ebenfalls in Excel-Tabellen abgelegt. Für den OÖ Verkehrsverbund sind die Flächendaten der oberösterreichischen Gemeinden von Interesse, welche in zwei separaten Excel-Dateien zur Verfügung stehen. Die (relativ dynamischen) Flächenwidmungspläne, die die Nutzung von Gemeindeflächen als Agrargebiete, Bauland, Gewerbegebiete, usw. ausweisen, werden so von den (relativ statischen) Stammdaten der Gemeindeflächen getrennt.

In der ersten Datei sind die Flächen nach der Widmung aufgeteilt. Hier findet sich zum Beispiel `Wohngebiet genutzt`, `Wohngebiet gewidmet`, `gemischtes Baugebiet gewidmet` und so weiter. Die Widmung ist auf der Ordinate aufgetragen. Die Gemeinden sind auf der Abszisse aufgetragen. Zusätzlich sind die Gemeinden nach Bezirk gruppiert. Einen Auszug daraus zeigt Tabelle 8.

	◁ fehlende bzw. unvollständige Flächenbilanz im aktuellen ÖEK (Örtliches Entwicklungskonzept)					*Wohn-gebiet gewidmet*	*Wohn-gebiet genutzt*	*Dorf-gebiet gewidmet*	*Dorf-gebiet genutzt*	*gemischtes Baugebiet gewidmet*	*gemischtes Baugebiet genutzt*	*Betriebs-baugebiet gewidmet*	*Betriebs-baugebiet genutzt*	*Industrie-gebiet gewidmet*
	◁ Flächenberechnung mittels Messung im FWPL													
Gde.Nr.	BAULAND - FLÄCHENBILANZ Gemeinde - *Angaben in ha* -	Stand:	Stand bei FWPL Nr.	Raumtyp		W.gew	w.gen	D.gew	d.gen	M.gew MB.gew	m.gen mb.gen	B.gew	b.gen	I.gew
	Bezirk Braunau													
40401	Altheim	2009	4.37	3		132,80	84,57	13,73	10,68	13,93	11,44	48,26	30,53	0,00
40402	Aspach	2009	2.28	3		55,38	30,45	34,76	25,40	17,20	14,20	23,50	10,45	0,00
40403	Auerbach	2009	3.12	3		0,83	0,44	32,51	23,38	1,76	1,59	2,85	0,53	0,00
40404	Braunau am Inn	2009	4.28	3		319,28	225,00	9,34	8,22	29,38	17,92	75,16	56,36	83,91
40405	Burgkirchen	2009	3.20	3		55,15	34,26	38,47	31,05	21,94	17,22	14,68	6,38	0,00
40406	Eggelsberg	2009	3.15	3		59,75	33,61	52,97	41,97	10,16	7,64	28,67	15,82	0,00
40407	Feldkirchen bei Mattighofen	2009	3.35	3		19,39	10,24	64,57	34,57	9,93	5,00	8,94	6,79	0,00
40408	Franking	2009	4.27	3		13,64	9,34	15,74	12,00	6,21	5,00	6,66	6,50	0,00
40409	Geretsberg	2009	5.4	3		14,71	10,61	15,16	11,27	9,76	6,97	10,25	7,30	0,00
40410	Gilgenberg am Weilhart	2009	3.16	3		19,40	12,04	20,24	18,80	1,94	1,94	6,72	6,72	0,00
40411	Haigermoos	2009	3.4	3		11,49	8,25	12,01	6,06	3,11	2,17	0,33	0,33	0,00
40412	Handenberg	2009	4.14	3		15,11	10,41	13,30	11,64	9,34	4,34	13,42	8,02	0,00
40413	Helpfau-Uttendorf	2009	4.18	3		81,22	52,64	18,13	12,21	5,23	2,86	31,35	19,10	0,00
40414	Hochburg-Ach	2009	4.35	3		99,38	62,43	18,41	17,50	16,69	14,00	15,10	12,30	0,00

Tabelle 8: Auszug aus den Flächenwidmungsdaten (Statistikamt, 2009)

In einer weiteren Excel-Tabelle liegen Angaben zum Flächenumfang der Gemeinden vor. Hier sind auf der Ordinate die `Gemeinenummer`, der `Gemeindename`, eine Kurzform des Gemeindenamens, der zugehörige Bezirk und die `Fläche` sowie der `Umfang` der Gemeinde angegeben.

A	B	C	D	E	F
GEM_NR	GEM_NAME	GEM_KURZNAME	GEM_BEZREF	GEM_FLAECHE IN M2	GEM_UMFANG IN M
40101	Linz	LINZ	401	96047365,69	75047,76578
40201	Steyr	STEYR	402	26560290,7	30497,61869
40301	Wels	WELS	403	45909961,21	40260,13428
40401	Altheim	Altheim	404	22666722,35	38085,80715
40402	Aspach	Aspach	404	31477472,5	41370,18313
40403	Auerbach	Auerbach	404	10787331,31	20491,97728
40404	Braunau am Inn	Braunau	404	24859367,31	32289,43038
40405	Burgkirchen	Burgkirchen	404	45920616,9	54290,10581
40406	Eggelsberg	Eggelsberg	404	24131997,8	36485,31394
40407	Feldkirchen bei Mattighofen	Feldkirchen	404	34662094,23	40060,05342
40408	Franking	Franking	404	10463599,9	24634,33787
40409	Geretsberg	Geretsberg	404	37454190,46	38620,89766
40410	Gilgenberg am Weilhart	Gilgenberg	404	26630756,98	29827,38176
40411	Haigermoos	Haigermoos	404	7443409,922	15548,33936
40412	Handenberg	Handenberg	404	27607716,51	33162,94352
40413	Helpfau-Uttendorf	Helpfau-Uttdf.	404	26355195,22	36594,08227
40414	Hochburg-Ach	Hochburg-Ach	404	40109579,47	35156,49427

Tabelle 9: Auszug aus der Quelltabelle *„Flächen der OÖ Gemeinden“* (Statistikamt, 2009)

4.4 Erläuterung der Problemstellung

Aufgabe ist es nun, die Fahrplan-, geographischen und demographischen Daten zu analysieren und in ein einheitliches und durchgängiges Datenmodell zu bringen, um die künftigen OLAP-Fragestellungen (als Beispiel *„Wie verteilen sich die verkauften Fahrscheine auf Regionen/Zonen unter Berücksichtigung der Bevölkerung pro Fläche (spezifische Bevölkerungsdichte)?“*) bedienen zu können.

Die Problemstellung dieser Arbeit ist das Einbinden der neuen Datenquellen Fahrplandaten, geographische und demographische Daten in das bestehende Data Warehouse. Dazu ist das bestehende Datenmodell zu überprüfen und um die neuen Datenquellen Fahrplandaten, geographische Daten und demographische Daten zu erweitern. In Kapitel 2.1 wurden bereits die grundlegenden Begriffe und Konzepte für das Data Warehouse erörtert. Die Einbindung der neuen Daten erfordert eine Erweiterung des multidimensionalen Modells und eine Anpassung der Auswertemöglichkeiten über OLAP.

Für geographische Daten gibt es mehrere Möglichkeiten, diese in ein Data Warehouse einzubinden. Wie Kapitel 2.4.3 erläutert, ist es auch für geographische Daten wichtig, das richtige Modell zur Einbindung zu wählen.

Eine strukturierte Herangehensweise an diese Aufgaben setzt ein geeignetes Vorgehensmodell voraus. Kapitel 2.3 verschaffte bereits einen Überblick über mögliche Vorgehensmodelle. Im folgenden Kapitel 5 werden dazu verwandte Lösungsansätze aus der Literatur diskutiert und ein geeignetes Vorgehensmodell aus den Grundlagen gewählt.

In einem weiteren Schritt gilt es, die ETL-Prozesse für die neuen Anforderungen zu generieren und die bereits bestehenden ETL-Prozesse zu optimieren. Die Schritte Extrahieren der Daten aus den Quellen, Transformieren der Daten für das Zielformat und Laden der Daten in das Data Warehouse wurden bereits in Kapitel 2.2 ausführlich beschrieben.

5 Verwandte Lösungsansätze aus der Literatur

In der Literatur finden sich einige Ansätze, die ähnliche Problemstellungen lösen wie die vorliegende Diplomarbeit. In diesem Kapitel erfolgt eine Gegenüberstellung des Standes der Technik aus der Literatur mit den Lösungsansätzen und Vorgehensweisen, die im Kapitel 2.3 ausführlich beschrieben wurden.

Diese Arbeit beschäftigt sich mit der Integration von neuen Datenquellen in ein bestehendes Data Warehouse. Die neuen Datenquellen Fahrplandaten und geographische beziehungsweise demographische Daten kann man in unternehmensinterne und unternehmensexterne Daten gliedern. Fahrplandaten werden seitens des OÖ. Verkehrsverbundes selbst generiert. Daher spricht man von internen Daten. Die geographischen und demographischen Daten hingegen werden von externen Quellen bezogen. Als externe Quellen allgemein können Excel-Tabellen der Statistik Austria (Statistikamt, 2009), Fremddatenbanken oder auch Web Services wie http://maps.google.com/staticmap dienen.

Auf den folgenden Seiten werden verschiedene Möglichkeiten zur Integration externer Daten vorgestellt. Als erstes werden Web-Services als Datenquelle erläutert. Weiters wird die Integration von XML-Daten diskutiert und XML (Extensible-Markup-Language) im traditionellen Data Warehouse dem XML-Warehouse gegenübergestellt.

Das W3C (World Wide Web Consortium) beschreibt ein Web Service oder auch Web of Service als eine nachrichtenbasierte Verbindung, welche im Web häufig verwendet wird (W3C, Web of Service, 2010). Das Web of Service basiert auf verschiedenen Technologien wie HTTP, XML, SOAP, WSDL, SPARQL und andere. Es umfasst nach (W3C, Web of Service, 2010) die Bereiche der Protokolle, der Servicebeschreibungen, der Sicherheit und der internationalen Anpassungen.

Ein Web-Service ist nach dem (W3C, Service Description W3C, 2010) eine nachrichtenbasierte Verbindung, welche dem Standard WSDL (Web-Service-Description-Language) oder dem semantischen Standard SAWSDL (Semantic-Annotation-WSDL) entspricht. Die Standards WSDL und SAWSDL bedienen sich der Markup- Sprache XML. Einstieg in XML findet man in der Literatur wie zum Beispiel (Vonhoegen, 2006). Durch Verwendung von XML sind Web-Services maschinencodierbar und trotzdem für den Anwender lesbar. Dabei liegt für Web-Services das Hauptaugenmerk auf der Unabhängigkeit der verwendeten Technologien sowohl auf der Serverseite als auch auf der Clientseite.

Mit Web-Services und WSDL importierte Daten liegen im XML-Format vor. (Ravat, Teste, Tournier, & Zurfluh, 2009) stellen mehrere Ansätze für die Einbindung von XML-basierten Daten in ein Data Warehouse vor. Die erste Möglichkeit beschreiben die Autoren mit dem Einbinden von XML-Daten durch Einlesen mittels ETL-Prozess in ein datenbankbasiertes Data Warehouse.

Microsoft SQL-Server stellt für Web Services eine eigene Import-Schnittstelle zur Verfügung. (Knight, Veerman, Dickinson, Hinson, & Herbold, 2008) beschreiben diese Schnittstelle im Zuge der Integration-Services (SSIS). Über einen Web-Service-Task-Editor kann in einfachen

Schritten die Quelladresse angelegt werden. In einem nächsten Schritt wird die zugehörige WSDL-Datei geladen. Diese informiert über die verfügbaren Services. Über eine Menüauswahl kann das entsprechend gewünschte Service ausgewählt werden. Durch Angabe von Parametern kann das Ergebnis des Web-Services beeinflusst werden. Die Daten werden in SSIS importiert und zur Weiterverarbeitung zur Verfügung gestellt (siehe Vorstellung SSIS in Kapitel 7.1).

Eine XML-Schnittstelle des Microsoft SQL-Server bietet die Möglichkeit, XML-Dateien zu importieren. Für den Import muss eine XSD-Datei (XML-Schema-Definition) vorhanden sein. Diese beschreibt den Aufbau der XML-Datei und die darin verwendeten Datentypen (Vonhoegen, 2006). Ist ein XSD-Schema angegeben, dann können in Folge die einzelnen Felder aus der XML-Datei für den Import selektiert werden.

Als zweite Möglichkeit für die Verarbeitung von XML-Daten stellen (Ravat, Teste, Tournier, & Zurfluh, 2009) ein XML-Data Warehouse vor. Der ETL-Prozess wird in diesem Fall wesentlich vereinfacht, da keine Typumwandlung stattfinden muss. Die Daten werden im Format XML aus der Datenquelle gelesen und im gleichen Format im XML-Warehouse abgelegt. Die Autoren unterscheiden, ob die Daten direkt im XML-Warehouse integriert sind oder ob diese nur logisch integriert sind, aber extern des XML-Warehouse gespeichert werden.

Als dritte Möglichkeit beschreiben (Ravat, Teste, Tournier, & Zurfluh, 2009) ein XML-Warehouse, welches numerische wie auch textorientierte Analysen ausführen kann. Textorientierte XML-Warehouse werden zum Beispiel für die Analyse von HTML-Seiten oder Texten eingesetzt. Für diese Arbeit ist eine textorientierte Analyse nicht erforderlich, daher wird diese Variante nur der Vollständigkeit halber aufgelistet.

Die Einbindung geographischer Daten in ein Data Warehouse wurde bereits in Kapitel 2.4.3 eingehend beleuchtet. (Wang, Pang, Zhang, & Zhang, 2010) haben sich mit der Einbindung geographischer Daten für das Verkehrsmonitoring beschäftigt. Die Arbeit der Autoren umfasst Daten über die Wege und Straßen in einer Graph-Struktur, kombiniert mit der Unfallstatistik und geographischen Koordinaten. Die Modellierung der geographischen Daten in der Arbeit von (Wang, Pang, Zhang, & Zhang, 2010) erfolgen als Punktangaben für die Messstationen. Diese drei Datenquellen werden in ein Data Warehouse zusammengeführt.

Das von (Wang, Pang, Zhang, & Zhang, 2010) gewählte Vorgehensmodell ist „bottom-up“ und lehnt sich somit an die Vorgehensweise von (Golfarelli & Rizzi, 1998) an. Die Autoren argumentieren mit der relativ klaren Datenstruktur, die ein datengetriebenes Vorgehensmodell bevorzugt. Liegt hingegen ein sehr genau beschriebenes Anforderungsprofil in Form von exakt definierten und abgegrenzten Fragen vor, so empfiehlt es sich, ein nutzergetriebenes Vorgehensmodell zu bevorzugen (Wang, Pang, Zhang, & Zhang, 2010).

Mit der Einbindung von geographischen Daten haben sich (Rifaie, et al., 2008) eingehend beschäftigt. Deren Vorgehensmodell wurde bereits in Kapitel 2.4.3 vorgestellt. Die zugrundeliegende Arbeit beschäftigt sich mit der Einbindung von geographischen Elementen wie Bahnlinien, Straßen, Bauwerke, Bezirke, Städte sowie auch demographischen Daten von Städten, Bezirken und mehr.

Für die geographischen Daten gibt es neben den geographischen Umrissdaten einer Fläche die beschreibenden Daten, welche Flächenangaben, Bezeichnungen und ähnliches beinhalten. Die geographischen Umrissdaten werden im Ansatz der Autoren als externe Daten verknüpft.

Die demographischen Daten in der Arbeit von (Rifaie, et al., 2008) umfassen eine Summe der Einwohner, eine Summe der Einwohner über 18 Jahre, Aufteilungen hinsichtlich der Rasse und noch weitere Information. In einer zweiten Quelle wird zusätzlich noch Information über den Haushalt berücksichtigt. Als Beispiel werden hier die Anzahl der männlichen und weiblichen Personen im Haushalt angeführt.

Als Vorgehensmodell wählen die Autoren eine Entwicklung des Datenmodells aus den bestehenden Datenquellen. Diese Vorgangsweise kann unter den Begriff des datengetriebenen Vorgehensmodells eingereiht werden.

Eine weitere Anwendung mit geographischen Daten erläutern (Zhu, Zhang, & Zhao, 2008). Inhalt dieser Arbeit ist die Verwaltung von Schiffsverkehrsdaten in einem Data Warehouse. Die verarbeiteten Daten werden aus zwei Quellen extrahiert, dem Schiffsplanreport und der Schiffsplan-Email, welches als Vorgabe für die Schiffsunternehmen gesendet wird. Diese beiden Quellen umfassen geographische Daten der Häfen, Schiffsdaten und Frachtdaten. Die geographischen Daten der Häfen wurden erweitert um Bereichsdaten und Länderdaten. Ziel des Data Warehouse von (Zhu, Zhang, & Zhao, 2008) ist die Auswertung der Frachtverteilung und der Auslastung der Häfen. Geographische Daten stellen nur eine Hilfsgröße dar. Daher werden diese als Dimensionsattribute in den Hafendaten manifestiert.

Bei der Entwicklung des Data Warehouse folgen (Zhu, Zhang, & Zhao, 2008) den Schritten Datenanalyse, Modelldesign, Datenextrahierung und Implementierung. Dieses Vorgehen geht, wie schon die bereits vorgestellten Anwendungen, wieder von einem datengetriebenen Vorgehensmodell aus.

6 Anpassung und Erweiterung des bestehenden Data Warehouse Schemas

Kapitel 4 informierte über die neuen Datenquellen und deren Struktur. Mögliche Vorgehensweisen für ähnliche Anwendungsfälle wurden in Kapitel 5 vorgestellt. Um die neuen Datenquellen mit dem bestehenden Data Warehouse analysieren zu können, ist es erforderlich, ein Datenmodell zu entwickeln, welches alle nun verfügbaren Quellen einbindet und eine gemeinsame Analyse ermöglicht.

Die geographischen und demographischen Daten werden derzeit als Excel-Listen bereitgestellt. Der Import von inhaltlich vergleichbaren Web-Services ist kostenpflichtig. Daher wurde diese Möglichkeit nicht in Betracht gezogen. Eine Einbindung von Web-Services (in Kapitel 5 erläutert) in das bestehende Data Warehouse ist mit geringem Aufwand möglich. Sollten sich in späterer Zukunft Möglichkeiten ergeben, Daten via Web-Service zu integrieren, so kann hier dank SSIS durch Austauschen der ETL-Prozessquelle eine einfache Anpassung erreicht werden.

Die Vorgabe des Datenbanksystems (Microsoft SQL-Server 2008-R2) seitens des OÖ. Verkehrsverbundes schließt eine vollständige XML-Lösung, wie in Kapitel 5 vorgestellt, in diesem Fall dezidiert aus. Als weiteres Argument steht die überwiegende Menge an Daten aus internen Quellen, welche kein XML-Format aufweisen, dagegen.

Vergleicht man die Anforderung von (Wang, Pang, Zhang, & Zhang, 2010) mit den Anforderungen in dieser Arbeit, so ergeben sich Parallelen. Beide Ansätze umfassen geographische Daten, wobei die Linien im Datenmodell der vorliegenden Arbeit mit den Wegen und Straßen im Ansatz der oben genannten Autoren vergleichbar sind. Es bestehen jedoch bedeutende Unterschiede in der Modellierung der geographischen Daten. Die geographischen Angaben, die in das Data Warehouse des OÖ Verkehrsverbunds integriert werden, sind in Gemeindegebiete granuliert, wohingegen die geographischen Daten in der Arbeit von (Wang, Pang, Zhang, & Zhang, 2010) als Punktangaben für die Messstationen dienen. Die Autoren wählen ein datengetriebenes Vorgehensmodell. Als Begründung gelten die bekannten Quelldatenstrukturen. Die vorliegende Arbeit weist ebenfalls bekannte Datenstrukturen auf, welche ein datengetriebenes Vorgehensmodell nach (Golfarelli & Rizzi, 1998) befürworten.

Die geographischen und demographischen Daten, welche (Rifaie, et al., 2008) anführen, sind vom Typ her ähnlich denen in der vorliegenden Arbeit. Die demographischen Daten umfassen Bevölkerungsdaten in verschiedenen Ausprägungen. Für die geographischen Daten gibt es neben den geographischen Umrissdaten einer Fläche die beschreibenden Daten, welche Flächenangaben, Bezeichnungen und ähnliches beinhalten. Die geographischen Daten, welche dieser Arbeit zugrundeliegen, umfassen ebenfalls Flächenangaben und Beschreibungsdaten. Geographische Umrissdaten sind nicht Teil der Quelldaten. Da diese im Ansatz der Autoren aber als externe Daten verknüpft werden, kann diese Abweichung durch Weglassen der Verknüpfung relativ einfach überbrückt werden.

Ein datengetriebenes Vorgehensmodell in der Entwicklung seitens (Rifaie, et al., 2008) wird mit der bekannten Datenstruktur gerechtfertigt. Die in dieser Arbeit bekannte Datenstruktur und die Ähnlichkeit der Problemstellung lassen eine gleiche Wahl des Vorgehensmodells zu.

Die Datenquellen, welche der vorliegenden Arbeit zugrunde liegen, sind ausreichend definiert und weitgehend exakt dokumentiert. Ziel des Data Warehouse ist es, Auswertungen zu generieren, welche einen Neuheitswert aufweisen oder strategische Entscheidungen belegen. Daher eignet sich nach der Gegenüberstellung verschiedener Vorgehensmetamodelle von (List, Bruckner, Machaczek, & Schiefer, 2002) für diese Ausarbeitung ein datengetriebenes Vorgehen. Als Vorgehensmodell wird das bereits in Kapitel 2.3.1 vorgestellte Vorgehen im Dimensional Fact Model nach (Golfarelli & Rizzi, 1998) angewendet.

Das Modell nach (Da Silva, De Oliveira, Fidalgo, Salgado, & Times, 2007) beschreibt im Gegensatz eine Einbindung der geographischen räumlichen Daten als primitive oder zusammengesetzte geographische Datentypen. Die Datenquellen dieser Arbeit beinhalten keine relevanten Daten für eine Einbindung der geographischen räumlichen Daten. Daher findet das Modell von (Da Silva, De Oliveira, Fidalgo, Salgado, & Times, 2007) keine Möglichkeit der Anwendung.

Das Auslagern der geographischen räumlichen Daten und die alleinige Einbindung der beschreibenden geographischen Daten nach (Rifaie, et al., 2008) ermöglicht eine Anwendung in diesem Kontext. Die verfügbaren Datenquellen beinhalten nur beschreibende Attribute der geographischen Daten. Daher ist eine Auslagerung der geographischen räumlichen Daten nicht erforderlich. Die beschreibenden Daten können wie gewöhnliche nicht-geographische Attribute im Vorgehensmodell behandelt und abgebildet werden. Wie auch die referenzierten Arbeiten von (Zhu, Zhang, & Zhao, 2008), (Rifaie, et al., 2008) oder (Wang, Pang, Zhang, & Zhang, 2010) zeigen, gibt es bereits Erfahrungen mit der Einbindung von geographischen Flächendaten.

Sollte es später zu einer Erweiterung um geographische räumliche Daten kommen, kann das Datenmodell jederzeit leicht angepasst werden. Zu diesem Zeitpunkt besteht immer noch die Möglichkeit, nach dem Ansatz von (Da Silva, De Oliveira, Fidalgo, Salgado, & Times, 2007) die geographischen räumlichen Daten direkt in das Data Warehouse einzubinden oder auf diese nach dem Ansatz von (Rifaie, et al., 2008) extern zu referenzieren.

Die vorliegende Arbeit wird nach den obigen Erkenntnissen aufgrund der bekannten Datenstruktur mit einem datengetriebenen Vorgehensmodell entwickelt. Als Vorgehensmodell wird das in Kapitel 2.3.1 vorgestellte Vorgehensmodell „Dimensional Fact Model“ (DFM) von (Golfarelli & Rizzi, 1998) gewählt, welches sich in die fünf Phasen Analyse der Datenquellen, dimensionale Analyse, konzeptuelles Design, Validierung und logisches Design gliedert.

6.1 Phase 1 – Analyse der Datenquellen

In der ersten Phase werden die Datenquellen analysiert und Datenbankschemata wie folgt daraus generiert.

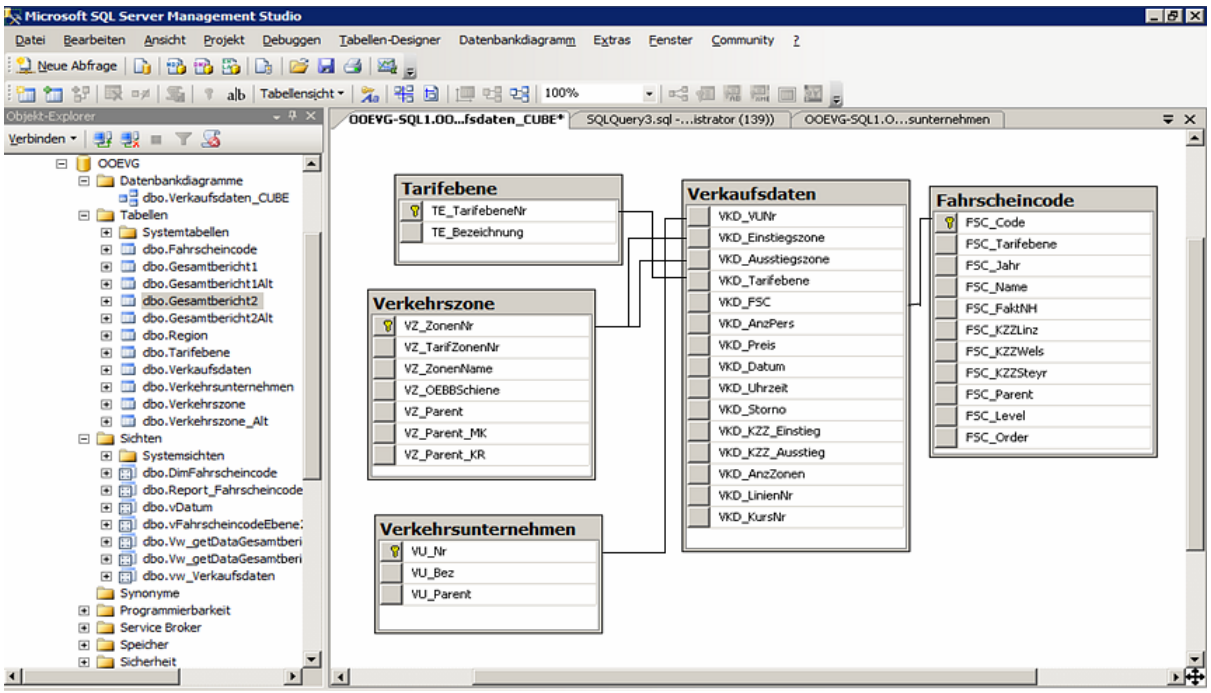

Abbildung 35: Datenbankschema für Verkaufsdaten

Für das bestehende Data Warehouse gibt es bereits ein Datenschema (Abbildung 20 in Kapitel 3.1). Dieses wird zur weiteren Analyse verwendet. Als Datenquelle können die zugrunde liegenden Tabellen und Datenbanken gesehen werden. Da diese Datenquellen äquivalent zum Datenschema aus dem Data Warehouse aufgebaut sind, wird dieses Datenschema 1:1 als Datenbankschema (Abbildung 35) übernommen.

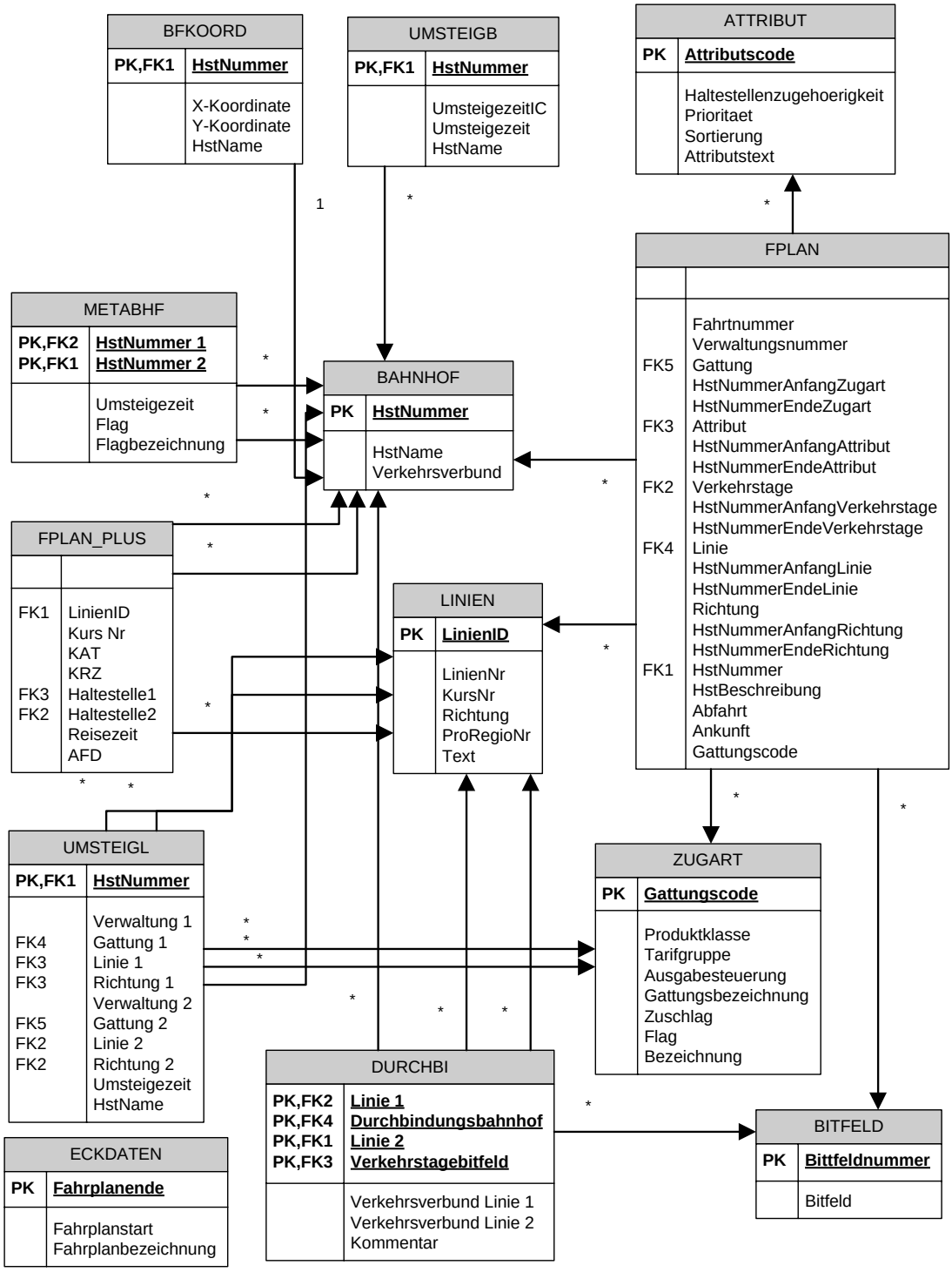

Abbildung 36: Datenbankschema für Fahrplandaten

Das Datenbankschema in Abbildung 36 für die Fahrplandaten wurde aus den Textdateien mit Hilfe der Beschreibung und Information seitens des Verkehrsverbunds generiert. Zwischen

der Tabelle `FPLAN` und der Tabelle `BAHNHOF` wurde zwecks Übersichtlichkeit auf die Darstellung aller Fremdschlüssel verzichtet, welche für die Start- und Zielhaltestellen der einzelnen Werte `Attribut`, `Gattung`, `Verkehrstage`, `Linie` und `Richtung` bestehen. Wie eine Analyse der Daten später zeigt (Kapitel 6.4), können diese Fremdschlüssel vernachlässigt werden.

Das Datenmodell in Abbildung 37 für die demographischen Daten kann sehr leicht dargestellt werden, da es sich nur um eine einzige Tabelle handelt. Als zusammengesetzter Schlüssel ergeben sich das `Geschlecht`, die `Gemeindenummer`, das `Jahr` und das `Alter in Jahren`.

Abbildung 37: Datenschema für demographische Daten

Das Datenmodell in Abbildung 38 für die geographischen Daten wird von zwei Tabellen bestimmt. In der ersten Tabelle befinden sich der `Name`, die `Bezirksreferenz` sowie `Fläche` und `Umfang`. Die zweite Tabelle beinhaltet Flächenwidmungsdaten der Gemeinden, welche sich jährlich ändern.

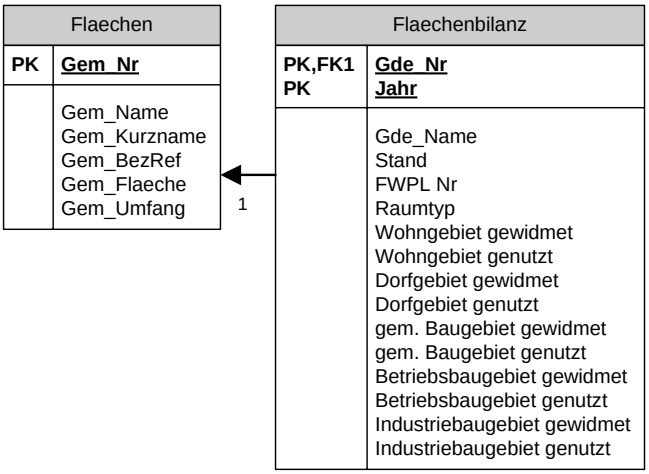

Abbildung 38: Datenschema für geographische Daten

Da sich aus Datenschemata mit einer oder zwei Tabellen nur schlecht ein Data Warehouse ableiten lässt, werden die Datenschemata der Fahrplan-, demographischen sowie geographischen Daten mit dem Datenschema der Fahrplandaten zusammengeführt.

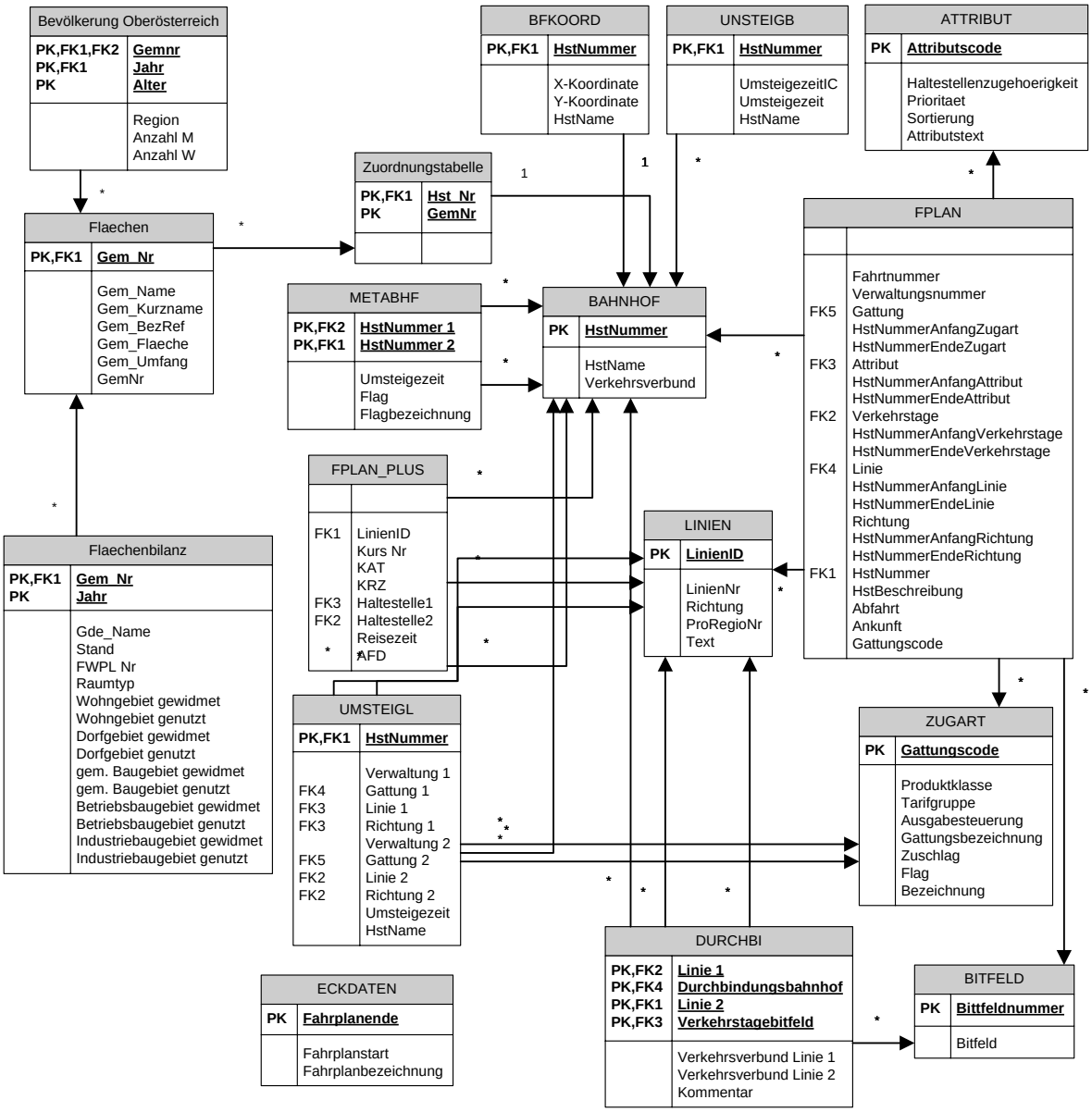

Abbildung 39: Zusammengefasstes Fahrplanschema mit geographischen und demographischen Daten

Die Zusammenführung der Datenschemata für Fahrplan-, demographische und geographische Daten mit dem Datenschema der Fahrplandaten kann leichter erfolgen, da eine direkte Zuordnung zwischen Gemeinden und Haltestellen besteht. Wie Abbildung 39 zeigt, wird dazu eine Zuordnungstabelle zwischen Haltestellen und Gemeinden geschaffen.

6.2 Phase 2 – dimensionale Analyse

Für die dimensionale Analyse („Anforderungsanalyse“ im Vorgehensmodell des Dimensional Fact Model nach (Golfarelli & Rizzi, 1998)) werden nun die beiden Schemata analysiert und nach dem Vorgehensmodell bewertet. Dazu werden die einzelnen Tabellen betrachtet und nach deren Änderungshäufigkeit bewertet. Dies zeigt die Anwärter für die Faktentabelle auf.

Im Datenschema `Verkaufsdaten` aus Abbildung 35 gibt es fünf Tabellen. Für die Tabelle `Tarifebene` sind im Normalfall keine Änderungen zu erwarten. Für die Tabelle `Verkehrszone` sind auch keine Veränderungen zu erwarten, da davon auszugehen ist, dass sich Zonen nicht ändern.

Die Tabelle `Verkehrsunternehmen` wird sich nur dann ändern, wenn neue Verkehrsunternehmen dazukommen oder Daten von bestehenden Verkehrsunternehmen sich ändern. Dies wird aber nicht sehr häufig passieren. Die Tabelle `Verkaufsdaten` wird sich häufig ändern, da die Fahrscheindaten abgelegt werden. Häufige Änderungen werden dagegen bei den `Fahrscheincodes` nicht zu erwarten sein.

Als Faktentabelle wird somit – wie im bisherigen Data Warehouse-Schema des OÖVV – die Tabelle `Verkaufsdaten` festgelegt.

Die Datenschemata der Fahrplan-, demographischen und geographischen Daten umfassen insgesamt 17 Tabellen. Diese werden in einer Tabelle zusammengestellt und systematisch ausgewertet.

Tabelle	*Häufige Änderung*	*Interessant*	*Begründung*
Bevölkerung OÖ	Ja	Ja	Ändern sich mindestens jährlich
Flächen	Nein	Nein	Die Gemeindeflächen ändern sich praktisch nicht
Flächenbilanz	Ja	Ja	Ändern sich mindestens jährlich
Zuordnungstabelle	Nein	Nein	Haltestellen verschieben sich kaum
Bahnhof	Nein	Nein	Haltestellen ändern sich kaum
MetaBhf	Nein	Nein	Haltestellen ändern sich kaum
BFKoord	Nein	Nein	Haltestellen ändern sich kaum
UmsteigB	Nein	Nein	Haltestellen ändern sich kaum
FPlan_Plus	Ja	Ja	Ändern sich mit jeder Fahrplanänderung
UmsteigL	Nein	Nein	Ändern sich nur wenn sich Linien ändert
Eckdaten	Ja	Nein	Signalisiert Anfang und Ende des Fahrplanes
DurchBi	Nein	Nein	Ändern sich nur, wenn sich Linien ändern
Bitfeld	Nein	Nein	Gibt Fahrplantage an, ändert sich nur, wenn neue Fahrplanintervalle kommen
Zugart	Nein	Nein	Reine Definition der Verkehrsmittel
Linien	Nein	Nein	Komplette Linien werden kaum erstellt oder aufgelassen
FPlan	Ja	Ja	Fahrplandaten mit Abfahrts- und Ankunftszeiten
Attribut	Nein	Nein	Attribute ändern sich kaum

Tabelle 10: Zusammenstellung der Änderungshäufigkeit für Tabellen im Fahrplanschema (incl. demographischen und geographischen Daten)

Hier ergeben sich vier Tabellen, welche für eine Faktentabelle in Fragen kommen. Betrachtet man diese Tabellen im Datenschema näher, so lassen sich je zwei Tabellen zusammenfassen.

Daher werden die Tabellen `Bevölkerung OÖ` und `Flächenbilanz` zusammengefasst zur Tabelle `Strukturdaten`. Um diese Tabellen zusammenfassen zu können, werden die beiden Werte „`Anzahl_W`“ und „`Anzahl_M`“ aus der Tabelle „`Bevölkerung Oberösterreich`“ in die beiden Werte „`EinwohnerW`“ und „`EinwohnerM`“ in der Tabelle

„Flaechenbilanz“ übergeführt. Weiters wird der Primärschlüssel der Tabellen vereinheitlicht. Der Wert „Region“ aus der Tabelle „Bevölkerung Oberösterreich“ entfällt. Die Tabelle erhält die Bezeichnung „Strukturdaten“ und ist in Abbildung 40 abgebildet.

Strukturdaten	
PK **PK** **PK**	**Gem_Nr** **Jahr** **Alter**
	Gde_Name Stand FWPL Nr Raumtyp Wohngebiet gewidmet Wohngebiet genutzt Dorfgebiet gewidmet Dorfgebiet genutzt gem. Baugebiet gewidmet gem. Baugebiet genutzt Betriebsbaugebiet gewidmet Betriebsbaugebiet genutzt Industriebaugebiet gewidmet Industriebaugebiet genutzt EinwohnerW EinwohnerM

Abbildung 40: Tabelle Strukturdaten nach Zusammenfassung aus Bevölkerung und Flächenbilanz

Weiters werden die Felder FPLAN und FPLAN_PLUS in der Tabelle FPLAN zusammengefasst.

Die Datenmengen ergeben somit die einzelnen Fahrplaneinträge sowie die einzelnen Positionen bei den Gemeindedaten.

6.3 Phase 3 – Konzeptuelles Design

Für die Erstellung des konzeptuellen Schemas wurden in Phase zwei bereits mögliche Faktentabellen erörtert. Diese werden nach dem Vorgehensmodell von (Golfarelli & Rizzi, 1998) abgeleitet und das konzeptuelle Schema entworfen.

Das Faktenschema für die Verkaufsdaten wird aus dem Datenschema der Verkaufsdaten abgeleitet.

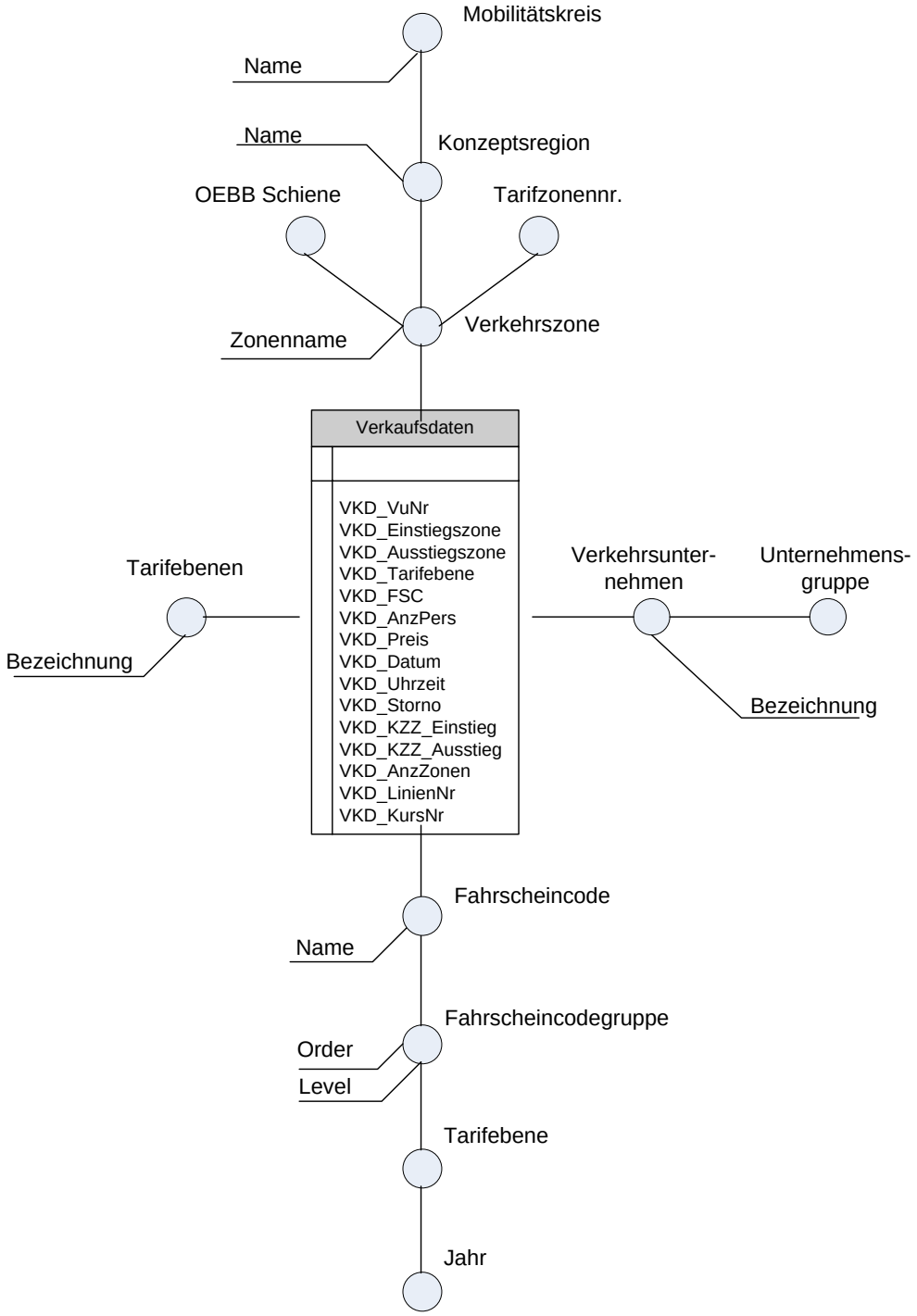

Abbildung 41: Faktenschema Verkaufsdaten

Wie Abbildung 41 zeigt, ergeben sich keine Veränderungen gegenüber dem Datenschema. Da dieses Datenschema aus einem Data Warehouse kommt, ist es schon einmal durch diesen Prozess gelaufen. Damit wird auf eine erneute Validierung verzichtet.

Das Faktenschema für die Gemeindedaten (mit Einwohnerzahlen und Flächenbilanzen aus den Flächenwidmungsplänen) wird von der Faktentabelle `Strukturdaten` abgeleitet.

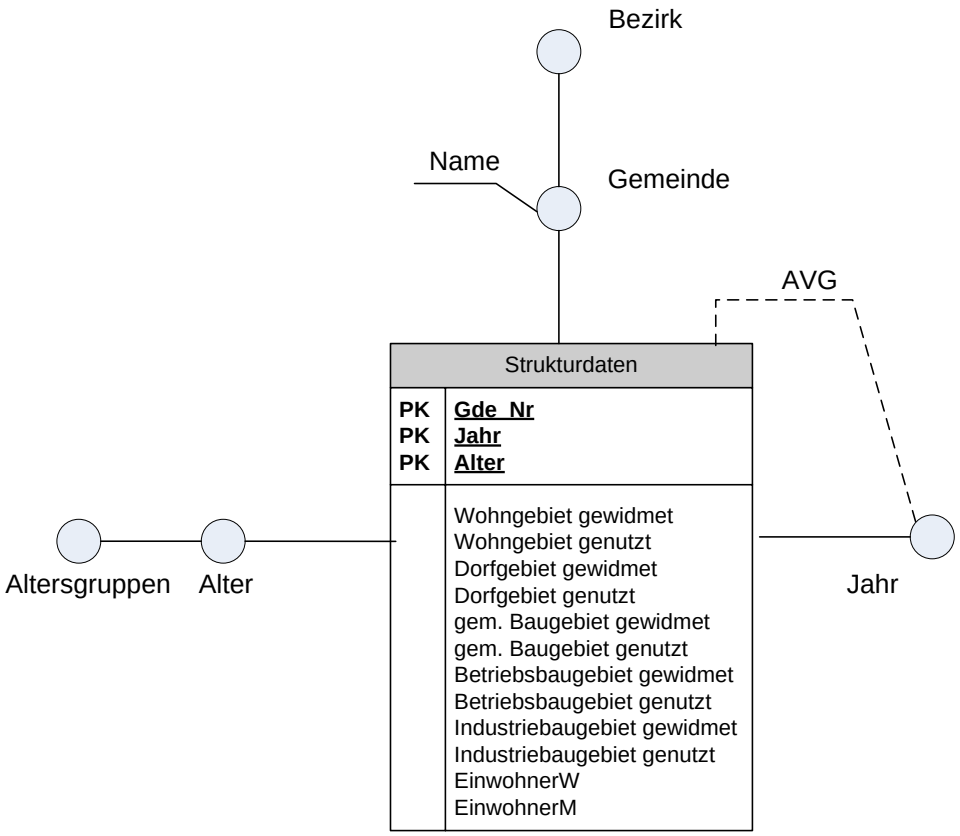

Abbildung 42: Faktenschema Gemeindedaten

Zwischen der Tabelle `Strukturdaten` und der Tabelle `Gemeinde` gibt es eine 1:1-Beziehung. Damit wird nach dem Vorgehen von (Golfarelli & Rizzi, 1998) die Tabelle `Gemeinde` zur Dimension von `Strukturdaten`. Von der Tabelle `Gemeinde` führt kein Fremdschlüssel weiter zu einer verknüpften Tabelle. Es kann daraus keine weitere Hierarchie abgeleitet werden. Die Tabelle `Gemeinde` enthält ein Feld `Bezirk`. Dieses eignet sich als Hierarchiestufe, da mehrere Gemeinden in einen Bezirk zusammengefasst werden können.

Als weitere Dimensionen werden im Faktenschema in Abbildung 42 eine Zeitdimension für historische Daten und eine Dimension `Alter` zur Unterscheidung nach dem Lebensalter eingeführt. Die Zeitdimension lässt nur Aggregationen mit Mittelwertbildung zu. Eine Summenbildung ergibt unrealistische Werte für die Measures. Bauland über die Zeit zu summieren würde heißen, dass jedes Jahr diese Fläche erneut zur Verfügung stünde.

Die Felder `Gemeindename`, `Stand`, `Flächenwidmungsplannummer` und `Raumtyp` wurden aus der Faktentabelle gestrichen, da diese nicht aggregiert werden können und somit keine Measures darstellen. Für die Referenzierung der Dimensionen werden diese Felder ebenfalls nicht benötigt.

Das Faktenschemas `Fahrplan` wird von der zusammengeführten Tabelle aus `FPLAN` und `FPLAN_PLUS` abgeleitet.

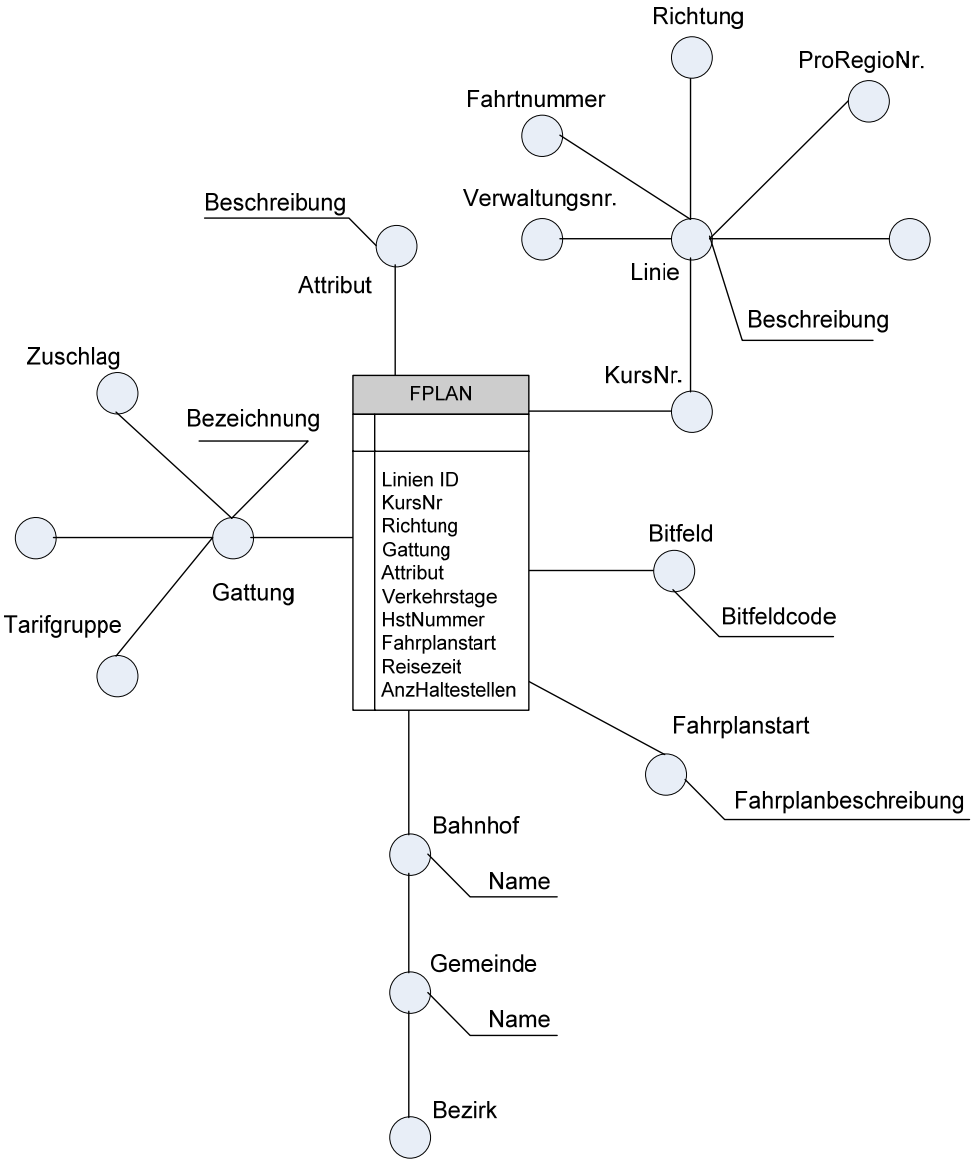

Abbildung 43: Faktenschema Fahrplandaten

Für die Ableitung des Faktenschemas `Fahrplan` wurde von der Tabelle `FPLAN` mit der integrierten Tabelle `FPLAN_PLUS` ausgegangen. Als erstes wurde die Dimension `Attribut`, Abbildung 43 oben, bestimmt. Die Dimension `Attribut` besitzt keine weiteren Fremdschlüssel und infolgedessen auch keine weiterführende Hierarchie.

Die nächste Fremdschlüsselverbindung `Linie` wird ebenfalls zur Dimension. Als Dimensionsattribut wird die `Kursnummer` festgelegt. Die Kursnummer ist mit `Linien_ID` ein zusammengesetzter Primärschlüssel. `Linie` ist die nächste Hierarchiestufe und besitzt ein zusätzliches nichtdimensionales Attribut `Beschreibung`. Nichtdimensionale Attribute können nicht zu Aggregationszwecken herangezogen werden und haben beschreibenden Charakter. Weiters gibt es die fünf Dimensionsattribute `ProRegioNr`, `Richtung`, `Fahrtnummer`, `Verwaltungsnummer` und `LinienNr`. Diese Dimensionsattribute bilden eigene Hierarchien. Daher sind diese in Abbildung 43 auch parallel angeordnet. Die beiden Dimensionsattribute `Fahrtnummer` und `Verwaltungsnummer` werden aus der Tabelle `FPLAN`

als Dimensionsattribute extrahiert, da nach diesen Dimensionsattributen aggregiert werden soll.

`Bitfeld` stellt die nächste Dimension dar. `Bitfeld` hat keine weiterführenden Fremdschlüssel und damit keine weiterführende Hierarchie.

Die Dimension `Fahrplanstart` wurde eingebunden, indem die Tabelle `Eckdaten` aus Abbildung 39 mit der Tabelle `FPLAN` kombiniert wurde. Damit wird eine zeitliche Komponente für die Fahrplandaten implementiert und historische Daten mittels Zeitstempelung ermöglicht.

Die Dimension `Bahnhof` in Abbildung 43 unten beschreibt die Haltestellen. Hier gibt es eine weiterführende Verbindung über die Zuordnungstabelle zur Tabelle `Gemeinde`. Diese ermöglicht eine eindeutige Zuordnung der Haltestellen zu den Gemeinden. Die Tabelle `Gemeinde` besitzt zwar eine 1:1- Beziehung zur Tabelle `Gemeindedaten`, jedoch beinhaltet letztere Tabelle keinerlei Hierarchisierungsfelder. Aus der Tabelle `Gemeinde` wird eine Klassifikationsstufe `Bezirk` extrahiert.

Die letzte Dimension in Abbildung 43 ist die Dimension `Gattung`. Die Dimension `Gattung` lässt sich weiter in Klassen, Tarifgruppen und Art des Zuschlages aggregieren. Das nichtdimensionale Attribut Bezeichnung verschafft der Dimension eine leichtere Lesbarkeit.

Nach dem Ableiten des Faktenschemas wurde ersichtlich, dass die Tabellen `BFKOORD`, `UMSTEIGB`, `UMSTEIGL`, `METABHF` und `DURCHBI` keine Verbindungen mit dem Faktenschema aufweisen. Damit wurden diese Tabellen ausgeschieden.

Von der Tabelle `Attribut` werden die Attribute `Haltestellenzugehörigkeit`, `Priorität` und `Sortierung` gestrichen, da diese weder als Dimensions- noch als Informationsattribute `dienen`. Aus der Tabelle `Gattung` wurden ebenfalls die Attribute gestrichen, welche für das Faktenschema keine Bedeutung haben.

Die Faktentabelle `Fahrplan` wurde jeweils um die beiden Haltestellen `Einstieg` und `Ausstieg` für die Felder `Attribut`, `Zugart`, `Verkehrstage`, `Linie` und `Richtung` bereinigt. Dies konnte deshalb geschehen, da es in den ganzen Fahrplandaten keinen einzigen Fall gab, in denen eigene Zuordnungen für `Starthaltestelle` oder `Endehaltestelle` vorlagen. Das Feld `Haltestellenbeschreibung` stellt eine Redundanz mit der Dimension `Bahnhof` dar. Daher wird das Feld ebenfalls gestrichen. Weiters werden die beiden Felder `Abfahrt` und `Ankunft` gestrichen. Damit bleibt nur das Feld `Reisezeit` als Measure. Dieses Feld kann summiert und gezählt werden.

6.4 Phase 4 – Validierung des Dimensionsschemas

Für ein gemeinsames Data Warehouse ist es interessant, eine oder mehrere Dimensionen für jedes Faktenschema zu erhalten, welche für alle drei Faktenschemata Gültigkeit haben. Tabelle 11 stellt die Dimensionen der drei Faktenschemata `Verkaufsdaten`, `Gemeindedaten` und `Fahrplandaten` gegenüber.

Verkaufsdaten	*Gemeindedaten*	*Fahrplandaten*
Tarifebene		
Verkehrszone	Gemeinde	Bahnhof
Verkehrsunternehmen		
Fahrscheincode		
(Zeit)	Jahr	Bitfeld (Fahrplanstart)
	Geschlecht	
		Linie
		Attribut
		Gattung

Tabelle 11: Gegenüberstellung der Dimensionen aus den drei Faktenschemata

Aus Tabelle 11 ist zu entnehmen, dass es zwei Dimensionen gibt, welche in allen drei Faktenschemata vorkommen. Eine Übereinstimmung der Dimensionen ist nur möglich, wenn die Dimensionsattribute eine einheitliche Semantik aufweisen. Diese Übereinstimmung der Dimensionen kann nach einer Analyse der Daten vollzogen werden.

In Kapitel 1.4, Abbildung 1: wird der Zonenplan auf die oberösterreichische Landkarte projiziert. Die Gemeindegrenzen können ebenfalls auf die oberösterreichische Landkarte projiziert werden. So können die Gemeinden den einzelnen Zonen zugeordnet werden. Für die Bahnhöfe im Faktenschema `Fahrplan` kann ebenfalls so vorgegangen werden. Die Bahnhöfe werden zur jeweiligen Verkehrszone zugeordnet und aggregiert. Eine feinere Auflösung als die Verkehrszonen wird seitens des OÖ Verkehrsverbundes nicht angestrebt.

Abbildung 35 zeigt das Datenbankschema des bestehenden Data Warehouse. In der Faktentabelle `Verkaufsdaten` gibt es bereits ein Feld `Datum`. Damit kann das Faktenschema für die Verkaufsdaten um eine Dimension `Zeit` erweitert werden. Die Gemeindedaten beinhalten einen Jahreseintrag. Im Faktenschema `Strukturdaten` ist dafür eine Zeitdimension vorhanden.

Das Faktenschema `Fahrplan` in Abbildung 43 zeigt die Dimension `Fahrplanstart` und die Dimension `Bitfeld`. Diese beiden Dimensionen können durch eine Dimension `Zeit` ersetzt werden. Mit der Datenquelle `Eckdaten` kennt man den Gültigkeitszeitraum des Fahrplans und damit das Datum, an dem der Fahrplan in bzw. außer Kraft tritt. In der Datenquelle `Bitfeld` steckt die Information, an welchen Tagen die Buslinie bedient wird. Aus diesen beiden Attributen kann ein Zeitstempel für die Faktentabelle generiert werden, welcher von der Dimension `Zeit` referenziert wird.

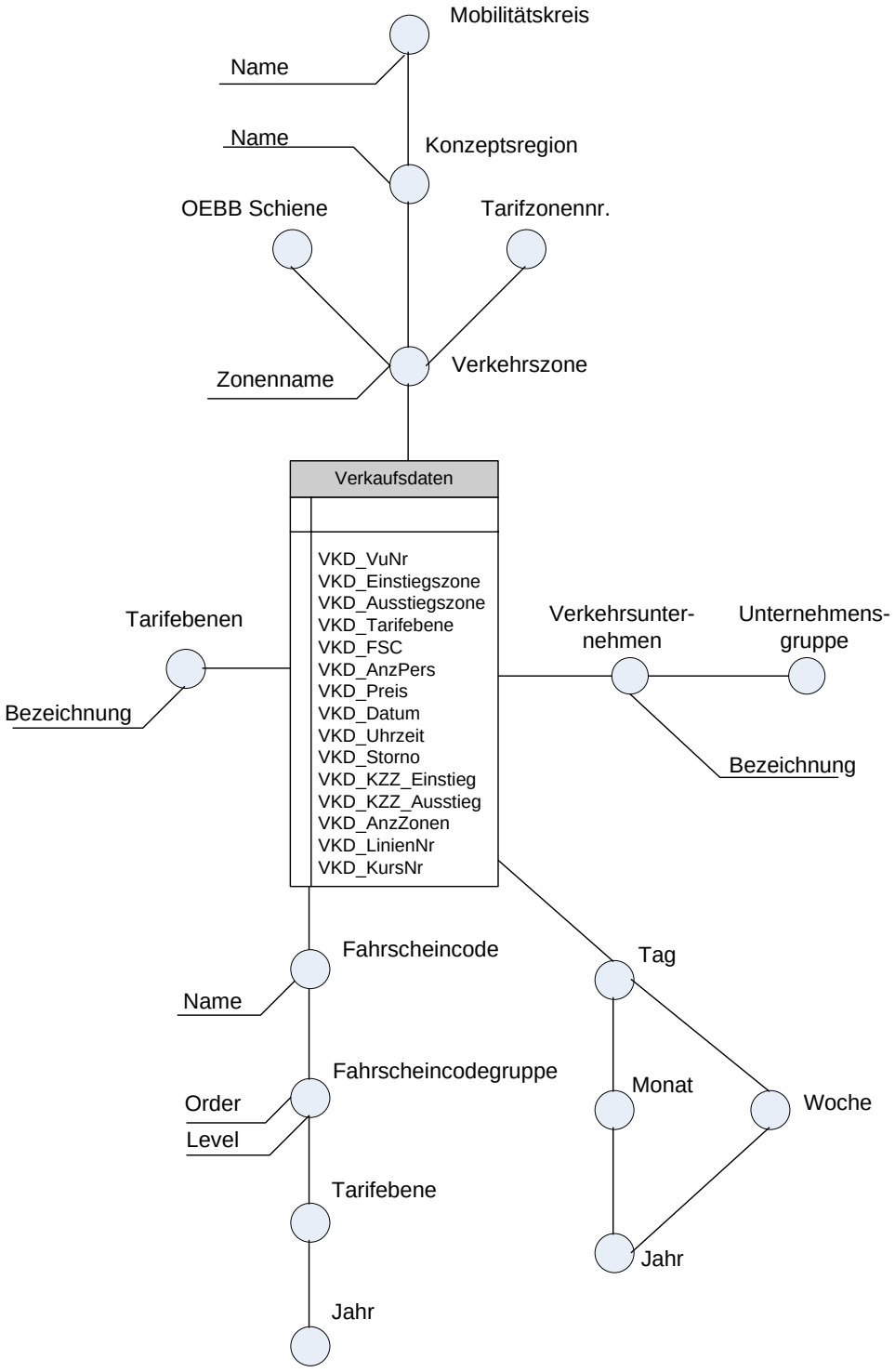

Abbildung 44: Modifiziertes Faktenschema Verkaufsdaten

Das Faktenschema `Verkaufsdaten` wurde um die Dimension `Zeit` erweitert. Diese ermöglicht eine Aggregation der Daten nach Tagen, Wochen, Monaten und Jahren. Die Faktentabelle erfuhr keine Änderung.

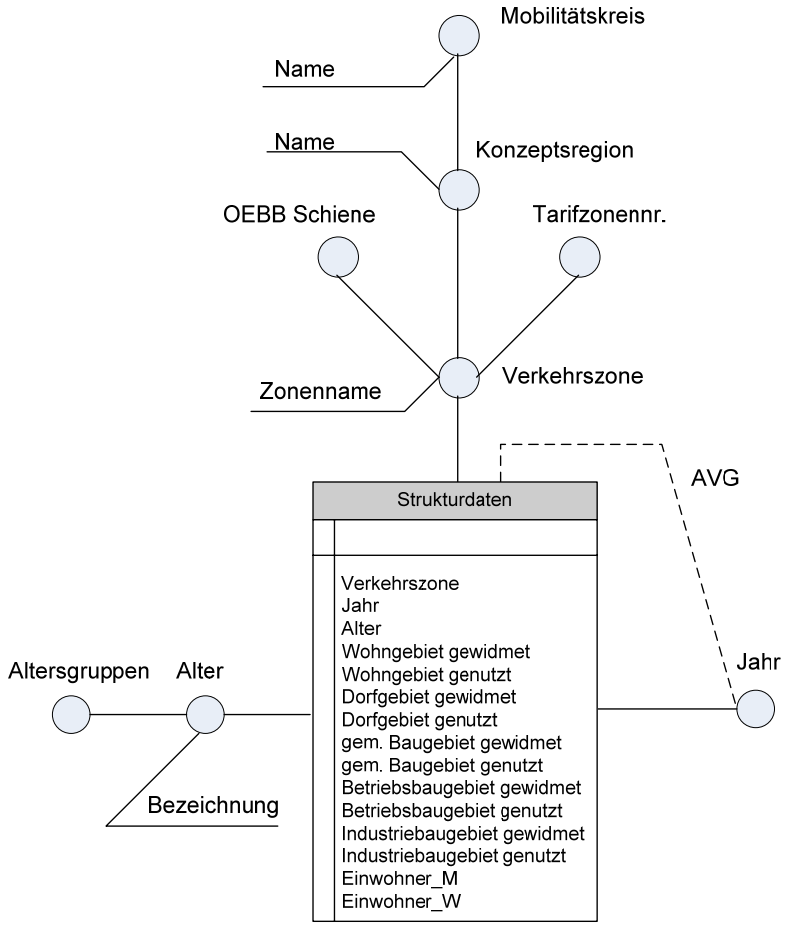

Abbildung 45: Modifiziertes Faktenschema Strukturdaten

Die Dimension `Gemeinde` wurde aus dem Faktenschema `Strukturdaten` in Abbildung 45 ersetzt durch die Dimension `Verkehrszone` mit all ihren Ausprägungen.

Abbildung 46 zeigt das modifizierte Faktenschema `Fahrplan`. Die Dimension `Bahnhof` wurde ersetzt durch die Dimension `Verkehrszone` mit all ihren Ausprägungen. Die Ausprägungen wurden aus dem Faktenschema `Verkaufsdaten` übernommen. In der Faktentabelle wurde ein Fremdschlüssel `Verkehrszone` eingeführt. Der Fremdschlüssel `Gemeindenummer` wurde entfernt. Die Zeitdimension ersetzt die Dimensionen `Bitfeld` und `Fahrplanstart`. In der Faktentabelle wurden die `Fahrplantage` zum `Datum` geändert.

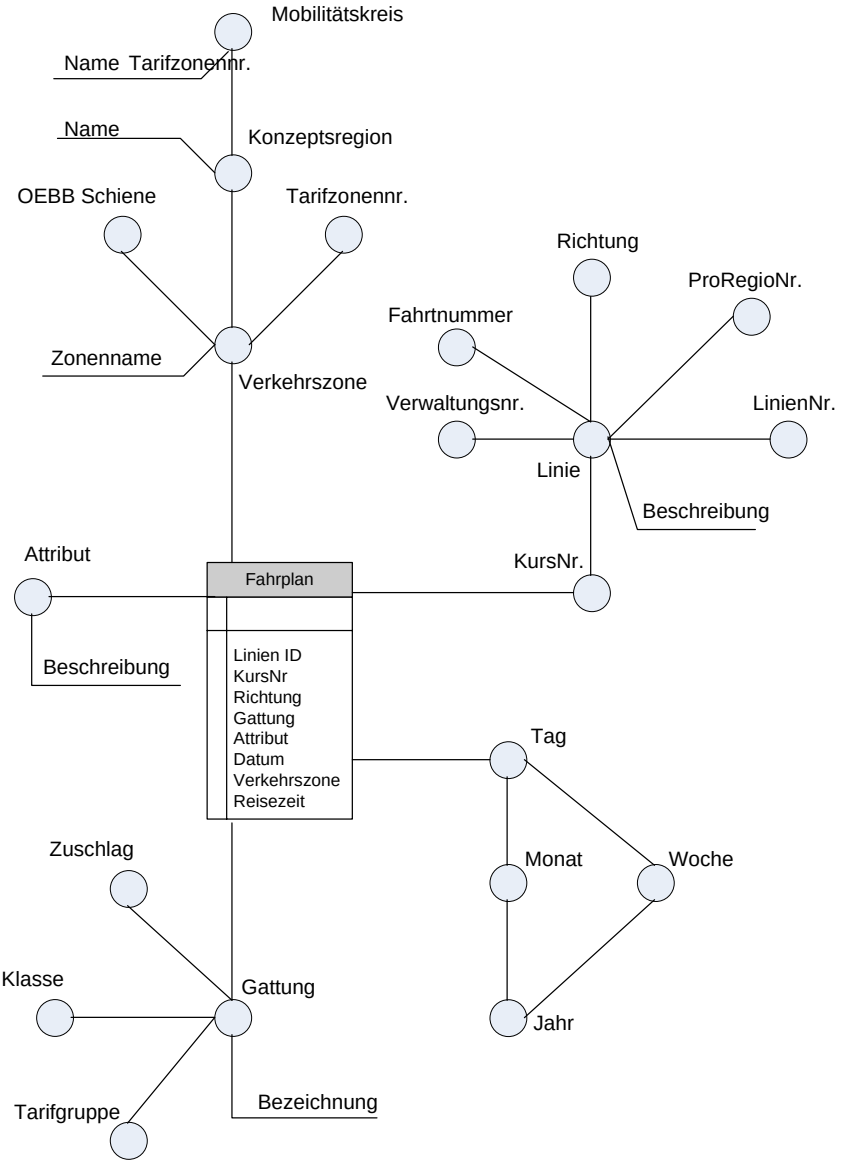

Abbildung 46: Modifiziertes Faktenschema Fahrplandaten

Damit ergibt sich die Möglichkeit, diese Daten über die gemeinsamen Dimensionen `Zeit` und `Verkehrszone` auszuwerten („Drill-across“).

6.5 Phase 5 – Logisches Design

Im logischen Design werden die Dimensionen und die Faktentabelle aus dem konzeptuellen Data Warehouse-Schema generiert. Im ersten Teil werden die Faktentabellen und Dimensionstabellen angelegt. Danach werden für das Verständnis die einzelnen Faktenschemata nach der Notation von Oracle (Lorentz, 2005) abgeleitet. In einem weiteren Schritt wird die Implementierung des Data Warehouse auf SQL-Server veranschaulicht.

Als Speicherschema wird das Star-Schema, welches in Kapitel 2.1.4 vorgestellt wurde, verwendet. Da mehrere Faktentabellen mit teilweise sich überlappenden Dimensionen zu speichern sind, liegt ein Galaxy-Schema oder Multifaktenschema vor (vergleiche Kapitel 2.1.4). Die Hierarchien der einzelnen Dimensionen können durch Parent-Child-Beziehungen abgeleitet werden. Es kommt kaum zu duplizierten Werten aufgrund der Hierarchie. Die Dimension

`Zeit` wird durch eine Funktion im SQL-Server realisiert. Diese erstellt eine Dimension `Zeit` im Starschema und befüllt diese automatisch mit Werten.

Für die Erstellung der Tabellen und das Ableiten der Dimensionen wird die Notation von Oracle herangezogen (Lorentz, 2005).

Die Dimensionen `Zeit` und `Verkehrszone` kommen in allen Faktenschemata vor. Daher werden diese herausgehoben und das logische Design abgeleitet. Im ersten Schritt werden die Dimensionstabellen abgeleitet.

```
CREATE TABLE DIM_Zonen
(
VZ_ZonenNr INTEGER NOT NULL PRIMARY KEY,
VZ_TarifZonenNr INTEGER,
VZ_ZonenName NVARCHAR(50),
VZ_OEBBSchiene INTEGER,
VZ_Parent INTEGER
)
```

In einem weiteren Schritt werden die Dimensionen `Verkehrszone` und `Zeit` abgeleitet. Für die Erstellung der Dimensionen enthält die Notation von Oracle (Lorentz, 2005, S. 868ff) eigene Befehle. Der Befehl `CREATE DIMENSION` legt eine Tabelle in einem multidimensionalen Schema an. Das Anlegen der einzelnen Dimensionsattribute erfolgt mit dem Befehl `LEVEL … IS (…)`. Die hierarchische Anordnung der Dimensionsattribute wird mit dem Befehl `HIERARCHY … (… CHILD OF …)` realisiert. Am Ende werden noch die nichtdimensionalen Attribute durch den Befehl `ATTRIBUTE … DETERMINES (…)` gesetzt.

```
CREATE DIMENSION Dim_Verkehrszone
(
LEVEL VZ_ZonenNr IS (DIM_Zonen.ZonenNr),
LEVEL VZ_TarifZonenNr IS (DIM_Zonen.TarifZonenNr),
LEVEL VZ_OEBBSchiene IS (DIM_Zonen.OEBBSchiene),
LEVEL VZ_Parent_KR IS (DIM_Zonen.Parent),
LEVEL VZ_Parent_MK IS (DIM_Zonen.Parent),
HIERARCHY tarifzonenZweig (VZ_ZonenNr CHILD OF VZ_TarifZonenNr),
HIERARCHY MK_Zweig (VZ_ZonenNr CHILD OF VZ_Parent_KR CHILD OF VZ_Parent_MK),
HIERARCHY OEBBSchieneZweig (VZ_ZonenNr CHILD OF VZ_OEBBSchiene),
ATTRIBUTE VZ_ZonenNr DETERMINES (ZonenName),
ATTRIBUTE VZ_Parent_KR DETERMINES (KR_Name),
ATTRIBUTE VZ_Parent_MK DETERMINES (MK_Name)
)
```

```
Create DIMENSION Dim_Zeit
(
LEVEL T_Tag IS (DIM_Zeit.PS_Datum),
LEVEL T_Monat IS (DIM_Zeit.Monat),
LEVEL T_Jahr IS (DIM_Zeit.Jahr),
LEVEL T_Woche IS (DIM_Zeit.Woche),
LEVEL T_TagJahr IS (DIM_Zeit.Tag_Des_Jahres),
LEVEL T_TagMonat IS (DIM_Zeit.Tag_Des_Monats),
LEVEL T_TagWoche IS (DIM_Zeit.Tag_Der_Woche),
LEVEL T_WocheJahr IS (DIM_Zeit.Woche_Des_Jahres),
LEVEL T_MonatJahr IS (DIM_Zeit.Monat_Des_Jahres),
HIERARCHY T_HiMonat (T_Tag CHILD OF T_Monat CHILD OF T_Jahr),
HIERARCHY T_HiWoche (T_Tag CHILD OF T_Woche CHILD OF T_Jahr),
ATTRIBUTE T_Tag DETERMINES (Datum_Name),
ATTRIBUTE T_Monat DETERMINES (Monat_Name),
ATTRIBUTE T_Jahr DETERMINES (Jahr_Name),
ATTRIBUTE T_Woche DETERMINES (Woche_Name),
ATTRIBUTE T_TagJahr DETERMINES (Tag_Des_Jahres_Name),
ATTRIBUTE T_TagMonat DETERMINES (Tag_Des_Monats_Name),
ATTRIBUTE T_TagWoche DETERMINES (Tag_Der_Woche_Name),
ATTRIBUTE T_WocheJahr DETERMINES (Woche_Des_Monats_Name),
ATTRIBUTE T_MonatJahr DETERMINES (Monat_Des_Jahres_Name)
)
```

Die Verkaufsdaten bestehen aus einer Faktentabelle `Verkaufsdaten` und fünf Dimensionen. Als erstes wird die Faktentabelle generiert. Diese unterscheidet sich in keiner Weise von einer Tabelle in einer normalen relationalen Datenbank.

```
CREATE TABLE Verkaufsdaten
(
VKD_ImportLog NVARCHAR(15) NOT NULL,
VKD_Zeilennummer INTEGER NOT NULL,
VKD_VuNr INTEGER NOT NULL,
VKD_Einstiegszone INTEGER NOT NULL,
VKD_Ausstiegszone INTEGER NOT NULL,
VKD_Tarifebene INTEGER NOT NULL,
VKD_FSC INTEGER NOT NULL,
VKD_AnzPers SMALLINT,
VKD_Preis INTEGER,
VKD_Datum DATETIME,
VKD_Uhrzeit INTEGER,
VKD_Jahr SMALLINT NOT NULL,
VKD_Storno SMALLINT,
VKD_KZZ_Einstieg BIT,
VKD_KZZ_Ausstieg BIT,
VKD_AnzZonen SMALLINT,
VKD_LinienNr INTEGER NOT NULL,
VKD_KursNr INTEGER NOT NULL
)
```

```
ALTER TABLE Verkaufsdaten
ADD CONSTRAINT [PK_Verkaufsdaten] PRIMARY KEY
([VKD_ImportLog],[VKD_Zeilennummer])
```

Die Dimensionstabellen werden ebenfalls in der normalen relationalen Umgebung angelegt.

```
CREATE TABLE DIM_Verkehrsunternehmen
(
VU_Nr INTEGER NOT NULL PRIMARY KEY,
VU_Bez NVARCHAR(50),
VU_Parent INTEGER
)

CREATE TABLE DIM_Tarifebenen
(
TE_TarifebeneNr INTEGER NOT NULL PRIMARY KEY,
TE_Bezeichnung NVARCHAR(100),
TE_Parent INTEGER
)

CREATE TABLE DIM_Fahrscheincode
(
FSC_Code INTEGER NOT NULL,
FSC_Tarifebene INTEGER NOT NULL,
FSC_Jahr SMALLINT NOT NULL,
FSC_Name NVARCHAR(50),
FSC_FaktNH FLOAT,
FSC_KZZLinz FLOAT,
FSC_KZZWels FLOAT,
FSC_KZZSteyr FLOAT,
FSC_Parent INTEGER,
FSC_Level INTEGER,
FSC_Order INTEGER
)

ALTER TABLE DIM_Fahrscheincode
ADD CONSTRAINT [PK_FSC]
PRIMARY KEY ([FSC_Code],[FSC_Tarifebene],[FSC_Jahr])
```

Nachstehend die Dimensionen `Verkehrsunternehmen`, `Tarifebene` und `Fahrscheincode` in der Notation von Oracle (Lorentz, 2005, S. 868ff).

```
CREATE DIMENSION Dim_Verkehrsunternehmen
(
LEVEL VU_Nr IS (DIM_Verkehrsunternehmen.VuNr),
LEVEL VU_Parent IS (DIM_Verkehrsunternehmen.VuParent),
HIERARCHY VU_Zweig (VU_Nr CHILD OF VU_Parent),
ATTRIBUTE VU_Nr DETERMINES (VuName)
)
```

```
CREATE DIMENSION Dim_Tarifebenen
(
LEVEL TE_TarifebeneNr IS (DIM_Tarifebene.TarifebeneNr),
ATTRIBUTE TE_TarifebeneNr DETERMINES (Bezeichnung)
)

CREATE DIMENSION Dim_Fahrscheincode
(
LEVEL FSC_Code IS (DIM_Fahrscheincode.Code),
LEVEL FSC_Parent IS (DIM_Fahrscheincode.Parent),
HIERARCHY FSC_Zweig IS (FSC_Code CHILD OF FSC_Parent),
ATTRIBUTE FSC_Code DETERMINES (Tarifebene, Jahr, Name),
ATTRIBUTE FSC_Parent DETERMINES (Order, Level),
)
```

Die Strukturdaten bestehen aus einer Faktentabelle `Strukturdaten` und drei Dimensionen. Als erstes wird die Faktentabelle `Strukturdaten` generiert.

```
CREATE TABLE Strukturdaten
(
SD_Verkehrszone INTEGER,
SD_Jahr DATETIME,
SD_Alter INTEGER,
SD_Wohngebiet_gewidmet FLOAT,
SD_Wohngebiet_genutzt FLOAT,
SD_Dorfgebiet_gewidmet FLOAT,
SD_Dorfgebiet_genutzt FLOAT,
SD_gemBaugebiet_gewidmet FLOAT,
SD_bemBaugebiet_genutzt FLOAT,
SD_Betriebsbaugebiet_gewidmet FLOAT,
SD_Betriebsbaugebiet_genutzt FLOAT,
SD_Industriebaugebiet_gewidmet FLOAT,
SD_Industriebaugebiet_genutzt FLOAT,
SD_EinwohnerW INTEGER,
SD_EinwohnerM INTEGER
)
```

Die Dimensionstabellen werden ebenfalls in der normalen relationalen Umgebung angelegt.

```
CREATE TABLE DIM_Alter
(
AL_Jahre INTEGER NOT NULL PRIMARY KEY,
AL_Bezeichnung NVARCHAR(15),
AL_Altersgruppe INTEGER
)
```

Nachstehend die Dimension `Alter` in der Notation von Oracle (Lorentz, 2005, S. 868ff).

```
CREATE DIMENSION Dim_Alter
(
LEVEL AL_Jahre IS (DIM_Alter.Jahre)
LEVEL AL_Altersgruppe IS (DIM_Alter.Altersgruppe)
)
```

Die Fahrplandaten bestehen aus einer Faktentabelle `Fahrplan` und fünf Dimensionen. Als erstes wird die Faktentabelle generiert.

```
CREATE TABLE Fahrplan
(
FP_LID INTEGER,
FP_KSN INTEGER,
FP_Gattung NVARCHAR(3),
FP_Attribut NVARCHAR(3),
FP_Datum DATETIME,
FP_Verkehrszone INTEGER,
FP_Reisezeit SMALLINT,
FP_AnzHaltestellen SMALLINT
)
```

Die Dimensionstabellen werden ebenfalls in der normalen relationalen Umgebung angelegt.

```
CREATE TABLE DIM_Linien
(
LI_LID INTEGER NOT NULL,
LI_KSN INTEGER NOT NULL,
LI_Verwaltungsnr INTEGER,
LI_Fahrtnummer INTEGER,
LI_Richtung SMALLINT,
LI_ProRegioNr INTEGER,
LI_LinienNr NVARCHAR(15),
LI_Beschreibung NVARCHAR(100)
)

ALTER TABLE DIM_Linien
ADD CONSTRAINT [PK_linie]
PRIMARY KEY ([LI_LID],[LI_KSN])

CREATE TABLE DIM_Attribut
(
AT_Attributscode NVARCHAR(3) NOT NULL PRIMARY KEY,
AT_Beschreibung NVARCHAR(100),
)
CREATE TABLE DIM_Gattung
(
GA_Gattung NVARCHAR(3) NOT NULL PRIMARY KEY,
GA_Zuschlag SMALLINT,
GA_Klasse SMALLINT,
GA_Tarifgruppe SMALLINT,
GA_Bezeichnung NVARCHAR(15)
)
```

Nachstehend die Dimensionen `Linie`, `Attribut` und `Gattung` in der Notation von Oracle (Lorentz, 2005, S. 868ff).

```
CREATE DIMENSION Dim_Linie
(
LEVEL LI_LID IS (DIM_Linien.LID),
LEVEL LI_KSN IS (DIM_Linien.KSN),
LEVEL LI_Verwaltungsnr IS (DIM_Linien.Verwaltungsnr),
LEVEL LI_Fahrtnummer IS (DIM_Linien.Fahrtnummer),
LEVEL LI_Richtung IS (DIM_Linien.Richtung),
LEVEL LI_ProRegioNr IS (DIM_Linien.ProRegioNr),
LEVEL LI_LinienNr IS (DIM_Linien.LinienNr),
HIERARCHY LI_HiRichtung (LI_KSN CHILD OF LI_LID CHILD OF LI_LI_Richtung),
HIERARCHY LI_HiVerwaltungsNr (LI_LID CHILD OF LI_VerwaltungsNr),
HIERARCHY LI_HiFahrtnummer (LI_LID CHILD OF LI_Fahrtnummer),
HIERARCHY LI_HiProRegioNr (LI_LID CHILD OF LI_ProRegioNr),
HIERARCHY LI_HiLinienNr (LI_LID CHILD OF LI_LinienNr)
ATTRIBUTE LI_LID DETERMINES (Beschreibung)
)

CREATE DIMENSION Dim_Attribut
(
LEVEL AT_Attribut IS (DIM_Attribut.Attributscode),
ATTRIBUTE AT_Attribut DETERMINES (Beschreibung)
)

CREATE DIMENSION Dim_Gattung
(
LEVEL GA_Gattung IS (DIM_Gattung.Gattung),
LEVEL GA_Zuschlag IS (DIM_Gattung.Zuschlag),
LEVEL GA_Klasse IS (DIM_Gattung.Klasse),
LEVEL GA_Tarifgruppe IS (DIM_Gattung.Tarifgruppe),
HIERARCHY GA_HiZuschlag IS (GA_Gattung CHILD OF GA_Zuschlag),
HIERARCHY GA_HiKlasse IS (GA_Gattung CHILD OF GA_Klasse),
HIERARCHY GA_HiTarifgruppe IS (GA_Gattung CHILD OF GA_Tarifgruppe),
ATTRIBUTE GA_Gattung DETERMINES (Bezeichnung)
)
```

Das Datenbanksystem SQL Server von Microsoft bringt ein Werkzeug zum Generieren des logischen Schemas mit. Dieses Werkzeug wird mit SQL Server Analysis Service, kurz SSAS bezeichnet. Die Umsetzung in ein logisches Schema erfolgt mit Hilfe eines graphischen Werkzeugs. Dazu müssen die Dimensionstabellen und die Faktentabelle auf dem Server in einem relationalen Datenmodell gespeichert sein. Werden die Fremdschlüssel auf die Faktentabelle bereits im relationalen Datenbankschema gesetzt, so werden die Verbindungen zu den Dimensionen ebenfalls erkannt und generiert.

Zur Vorbereitung für den Import werden die Fremdschlüssel von den Dimensionstabellen zur zugehörigen Faktentabelle angelegt. Der erste Fremdschlüssel verbindet die Tabelle `Ver-`

kaufsdaten mit der Dimensionstabelle Verkehrszonen über das Attribut Einstiegszone. Eine zweite Fremdschlüsselverbindung wird mit der Ausstiegszone verknüpft. Die nächste Verbindung generiert einen Fremdschlüssel für die Dimensionstabelle Verkehrsunternehmen, weiters wird ein Fremdschlüssel für die Tarifebenen erstellt. Als nächstes folgt dann der Fremdschlüssel für den Fahrscheincode. Der letzte Fremdschlüssel wird für die Zeitdimension erstellt.

```
ALTER TABLE Verkaufsdaten
ADD CONSTRAINT [FK_VKD_VZE]
FOREIGN KEY ([VKD_Einstiegszone]) REFERENCES DIM_Zonen ([VZ_ZonenNr])

ALTER TABLE Verkaufsdaten
ADD CONSTRAINT [FK_VKD_VZA]
FOREIGN KEY ([VKD_Ausstiegszone]) REFERENCES DIM_Zonen ([VZ_ZonenNr])

ALTER TABLE Verkaufsdaten
ADD CONSTRAINT [FK_VKD_VU]
FOREIGN KEY ([VKD_VuNr]) REFERENCES DIM_Verkehrsunternehmen ([VU_Nr])

ALTER TABLE Verkaufsdaten
ADD CONSTRAINT [FK_VKD_TE]
FOREIGN KEY ([VKD_Tarifebene]) REFERENCES DIM_Tarifebenen ([TE_TarifebeneNr])

ALTER TABLE Verkaufsdaten
ADD CONSTRAINT [FK_VKD_FSC]
FOREIGN KEY ([VKD_FSC],[VKD_Tarifebene],[VKD_Jahr])
REFERENCES DIM_Fahrscheincode ([FSC_Code],[FSC_Tarifebene],[FSC_Jahr])

ALTER TABLE Verkaufsdaten
ADD CONSTRAINT [FK_VKD_T]
FOREIGN KEY ([VKD_Datum]) REFERENCES DIM_Zeit([PS_Datum])
```

Für die Faktentabelle Strukturdaten werden die Fremdschlüssel in der Reihenfolge Verkehrszone, Jahr und Alter angelegt.

```
ALTER TABLE Strukturdaten
ADD CONSTRAINT [FK_SD_VZ]
FOREIGN KEY [SD_Verkehrszone] REFERENCES DIM_Zonen ([ZonenNr])

ALTER TABLE Strukturdaten
ADD CONSTRAINT [FK_SD_T]
FOREIGN KEY [SD_Jahr] REFERENCES DIM_Zeit ([PS_DATUM])

ALTER TABLE Strukturdaten
ADD CONSTRAINT [FK_SD_AL]
FOREIGN KEY [SD_Alter] REFERENCES DIM_Alter ([Jahre])
```

Die Faktentabelle `Fahrplan` erhält ihre Fremdschlüssel in der Reihenfolge `Verkehrszone`, `Linie`, `Zeit`, `Gattung` und `Attribut`.

```
ALTER TABLE Fahrplan
ADD CONSTRAINT [FK_FP_VZ]
FOREIGN KEY [FP_Verkehrszone] REFERENCES DIM_Zonen ([ZonenNr])

ALTER TABLE Fahrplan
ADD CONSTRAINT [FK_FP_LI]
FOREIGN KEY [FP_LID],[FP_KSN] REFERENCES DIM_Linien([LID],[KSN])

ALTER TABLE Fahrplan
ADD CONSTRAINT [FK_FP_T]
FOREIGN KEY [FP_Datum] REFERENCES DIM_Zeit ([PS_Datum])

ALTER TABLE Fahrplan
ADD CONSTRAINT [FK_FP_GA]
FOREIGN KEY [FP_Gattung] REFERENCES DIM_Gattung ([Gattung])

ALTER TABLE Fahrplan
ADD CONSTRAINT [FK_FP_AT]
FOREIGN KEY [FP_Attribut] REFERENCES DIM_Attribut([Attributscode])
```

Sind die Tabellen in der Datenbank angelegt und die Fremdschlüssel gesetzt, so kann der Import mit SSAS begonnen werden. Das Werkzeug ist Teil des SQL Server Business Intelligence Development Studio von Microsoft.

Nach dem Anlegen eines SQL-Server Analysis-Services-Projektes zeigt das SSAS im Projektmappen-Explorer den vergebenen Projektnamen und darunter eine Speicherstruktur, welche die Datenquellen, Datenquellensichten, die Cubes, die Dimensionen und mehrere weitere Elemente beinhaltet (siehe Abbildung 47). Die Speicherstruktur ist derzeit noch leer. Zur Erstellung eines logischen Data Warehouse-Schemas müssen je zumindest ein(e) Datenquelle, Datenquellensicht, Cubes und Dimensionen in der genannten Reihenfolge erstellt werden. Die einzelnen Schritte werden in der Folge genauer erläutert.

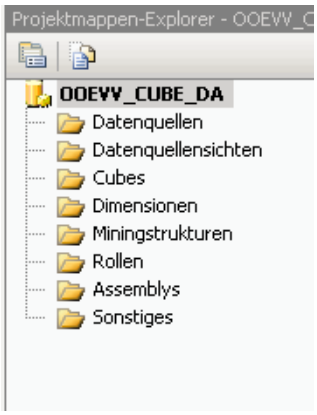

Abbildung 47: Ansicht des Projektmappen-Explorers im SSAS

Das Anlegen einer neuen Datenquelle ist der erste Schritt für den Import. Das Erstellen einer neuen Datenquelle ruft einen Assistenten auf, mithilfe dessen es leicht ist, die entsprechende Datenbank zu wählen, in welcher die Faktentabellen und die Dimensionstabellen angelegt wurden. Als Identitätswechselinformation ist die Auswahl Dienstkonto zu bevorzugen.

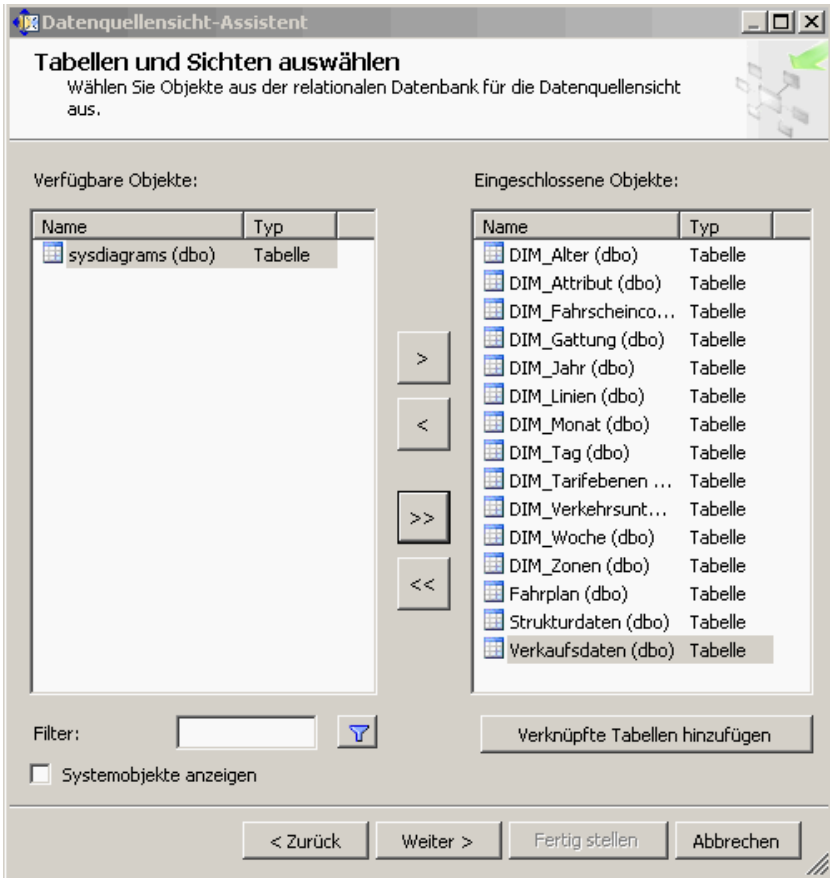

Abbildung 48: Auswahl der einzelnen Tabellen zum Import in das Data Warehouse

Nach Abschluss des Assistenten kann weiter zum Erstellen der Datenquellensichten gegangen werden. Hier gibt es wieder einen Assistenten. Die erste Auswahl gilt der Datenquelle, in der die eben angelegte Datenquelle aufscheinen sollte. Nach Auswahl dieser Datenquelle werden die verfügbaren Objekte angezeigt. Wie Abbildung 48 zeigt, werden alle in diesem Kapitel angelegten Tabellen ausgewählt. Nach Abschluss des Assistenten zeigt eine Datenquellensicht alle Tabellen inklusive der angelegten Verbindungen. Ordnet man die Tabellen wie in Abbildung 49 an, so kann man die entwickelten Faktenschemata deutlich erkennen. Auch die Zusammenhänge über die Fremdschlüssel sind leicht erkennbar.

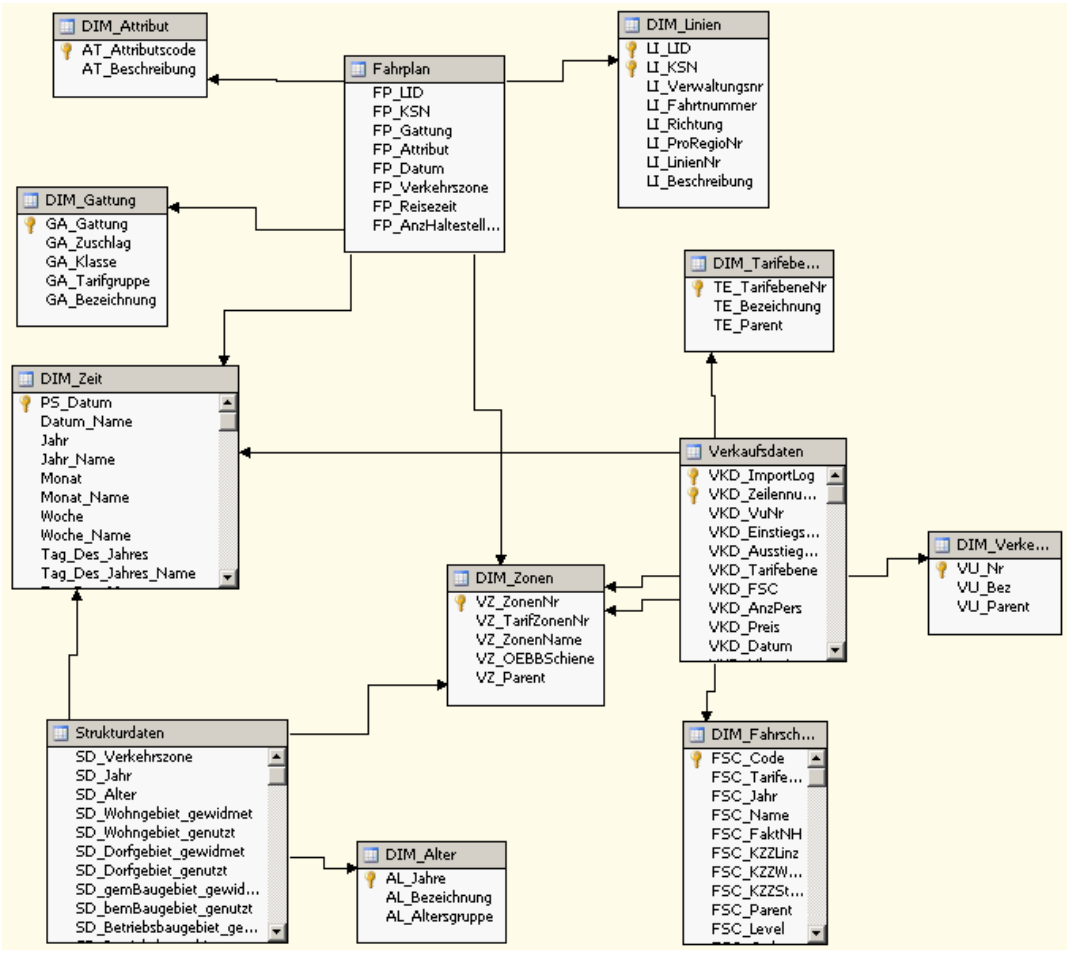

Abbildung 49: Datenquellensicht nach Import der ausgewählten Tabellen in das SSAS

Im nächsten Schritt wird der Cube erstellt. Ein Assistent begleitet durch den Vorgang. Als Erstellungsmethode in Abbildung 50 wird ein leerer Cube gewählt. Es gibt keine vorhandenen Tabellen im Cube, welche verwendet werden könnten. Tabellen müssen in der Datenquelle auch nicht erstellt werden, da dies bereits erledigt wurde. Die Auswahl der Datenquellensicht sollte die weiter oben erstellte Datenquellensicht zur Auswahl anzeigen. Diese wird ausgewählt. Damit kann der Assistent abgeschlossen werden und es wird ein leerer Cube generiert. In die Datenquellensicht des Cubes können nun mit der rechten Maustaste alle eingebundenen Tabellen zur Anzeige gewählt werden.

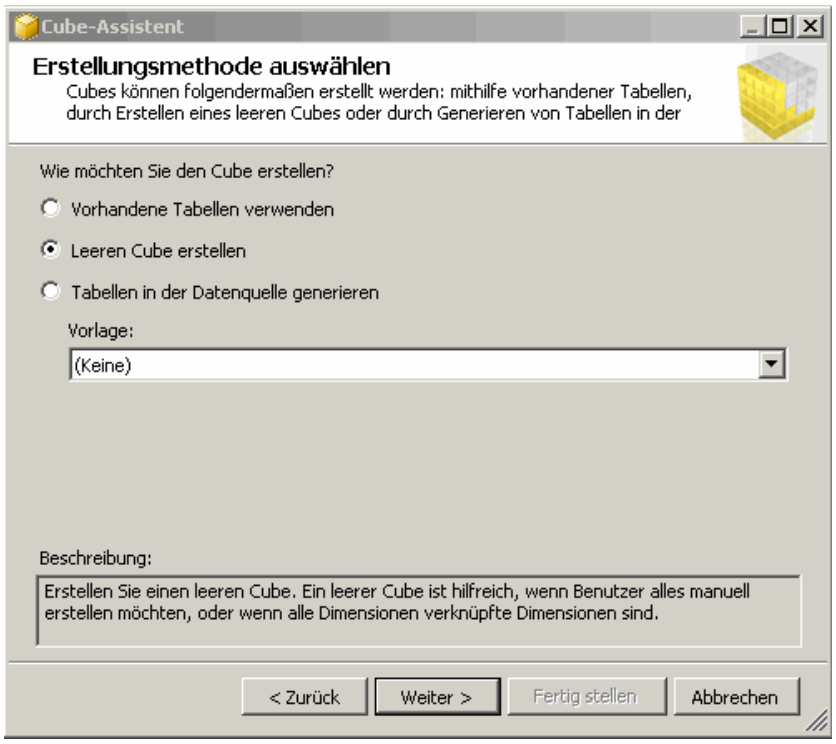

Abbildung 50: Erstellungsmethode für den Cube im SSAS

Zur Definition der Dimensionen wird unter Dimensionen der Dimensionsassistent ausgewählt. Für die Erstellungsmethode wird „Vorhandene Tabelle verwenden“ gewählt. Das nächste Auswahlmenü erhebt die Quellinformationen (= Metadaten zur Dimensionstabelle). Die Auswahlfelder Datenquellensicht und Haupttabelle definieren die Dimensionstabelle in der relationalen Datenbank.

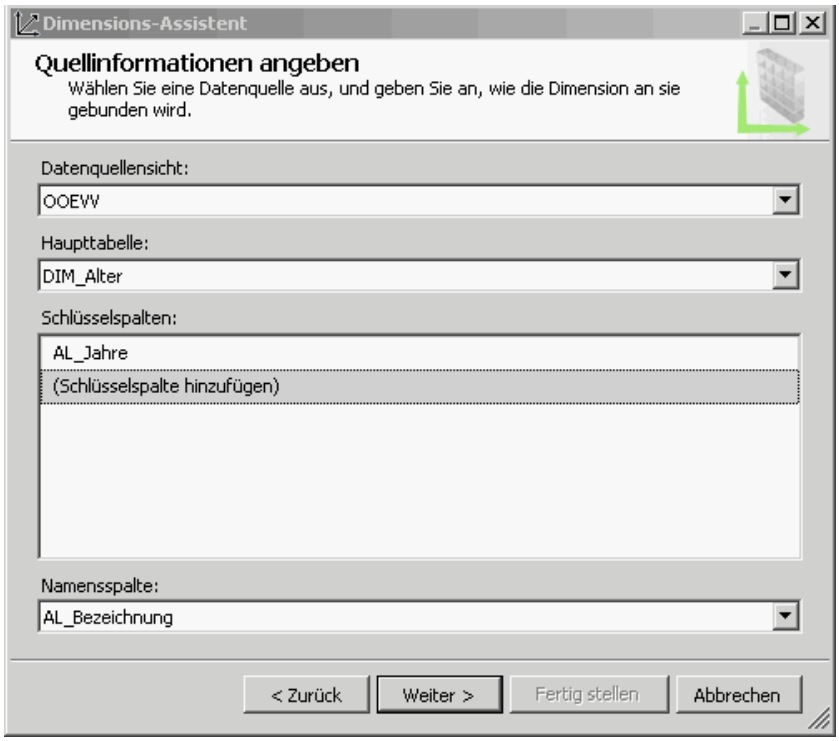

Abbildung 51: Auswahlfeld für Dimensionsschlüssel im Dimensionsassistenten

Abbildung 51 zeigt die ausgewählte Dimensionstabelle `DIM_Alter`. Als Schlüsselspalte wird die Primärschlüsselspalte `AL_Jahre` gewählt. Hier könnten weitere Spalten angegeben werden, wenn ein zusammengesetzter Primärschlüssel vorliegt. Als Namensspalte kann ein nichtdimensionales Attribut zur Namensgebung für die Schlüsselobjekte festgelegt werden. Die Auswahl der Dimensionsattribute zeigt dann nicht mehr die Dimensionsattribute selbst, sondern den zugeordneten Inhalt der Namensspalte, was die Lesbarkeit erhöht. Das nächste Auswahlfenster lässt eine Angabe weiterer Dimensionsattribute zu. Abbildung 52 zeigt die Auswahl für die Dimension `Alter`. Zusätzlich zum Attribut `AL_Jahre` wird das Attribut `AL_Altersgruppen` selektiert. Die Dimension kann nun unter Angabe eines sprechenden Namens angelegt werden. Mit sämtlichen anderen Dimensionstabellen wird nach dem gleichen Vorgehen verfahren. Am Ende sollten insgesamt neun Dimensionen deklariert sein.

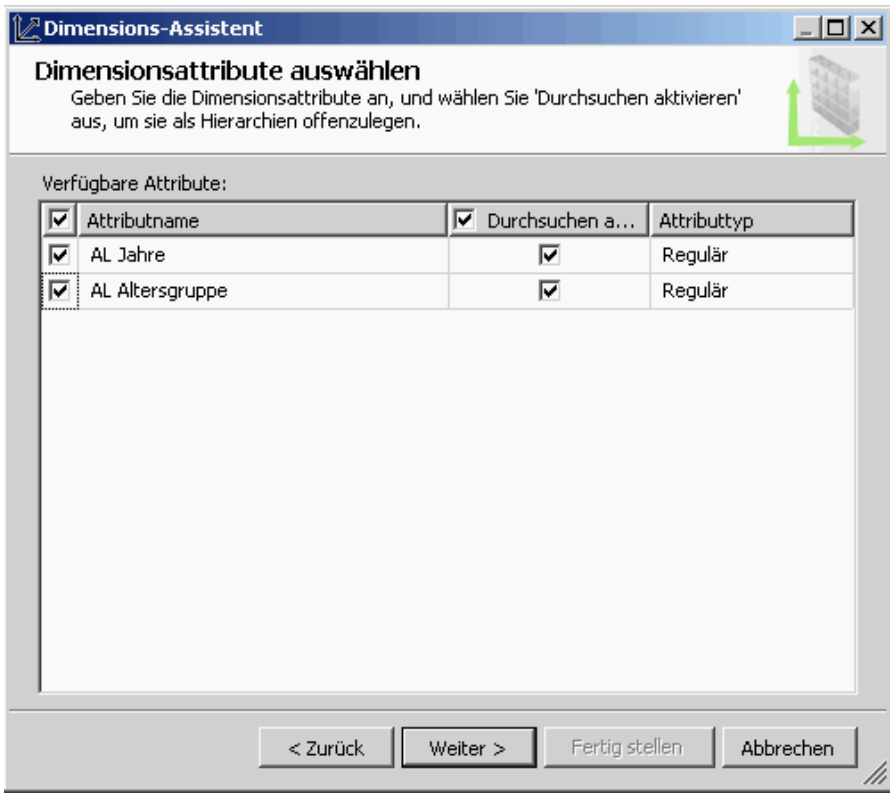

Abbildung 52: Auswahl weiterer Dimensionsattribute im Dimensionsassistenten

Für die Hierarchisierung der Dimensionen gibt es im SSAS die Möglichkeit, so genannte Parent-Child-Beziehungen zu projektieren. In Parent-Child-Beziehungen besitzt jedes Tupel eine Beziehungsspalte mit der Information seiner übergeordneten Hierarchie. Mit dieser Möglichkeit lassen sich Hierarchien sehr einfach auch zur Laufzeit durch Änderung der Daten anpassen.

Zur Veränderung der Hierarchie wird die jeweilige Dimension angewählt. Damit wird der Dimensionsexplorer geöffnet. Dieser teilt sich in drei Bereiche. Im linken Bereich sind die Dimensionsattribute angeordnet. Zentriert sind Hierarchien dargestellt. Rechts wird die Datenquellensicht der Dimensionstabelle angezeigt. Die Dimension `Verkehrszone` hat ein Attribut `VZ_Parent`. Dieses Dimensionsattribut wird im linken Fenster angewählt. Dann wird zu den Eigenschaften (eigenes Fenster rechts unten) gewechselt.

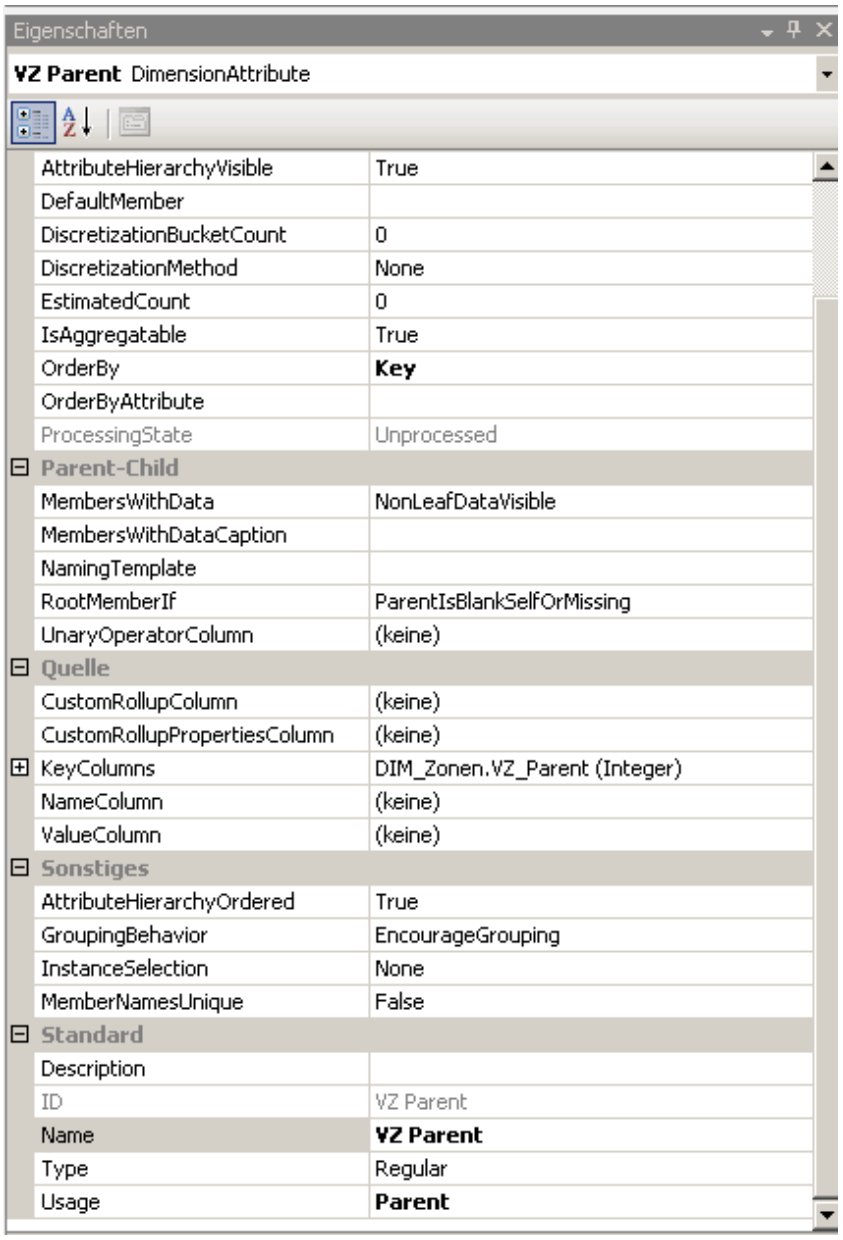

Abbildung 53: Eigenschaften des Dimensionsattributs `VZ_Parent` mit angewählter Parent-Eigenschaft

Die Eigenschaft Usage (ganz unten im Eigenschaftsfenster) in Abbildung 53 hat drei Ausprägungen. Regular beschreibt ein normales Dimensionsattribut. Key wird eingestellt für das Schlüsselattribut. Die Ausprägung Parent schließlich definiert das Attribut als Parent in einer Parent-Child-Beziehung. Das Attribut Child ist fixiert mit dem Schlüsselattribut. Für das Schlüsselattribut `VZ_ZonenNr` muss in Folge die Eigenschaft AttributeHierarchyVisible in Abbildung 54 auf False gestellt werden.

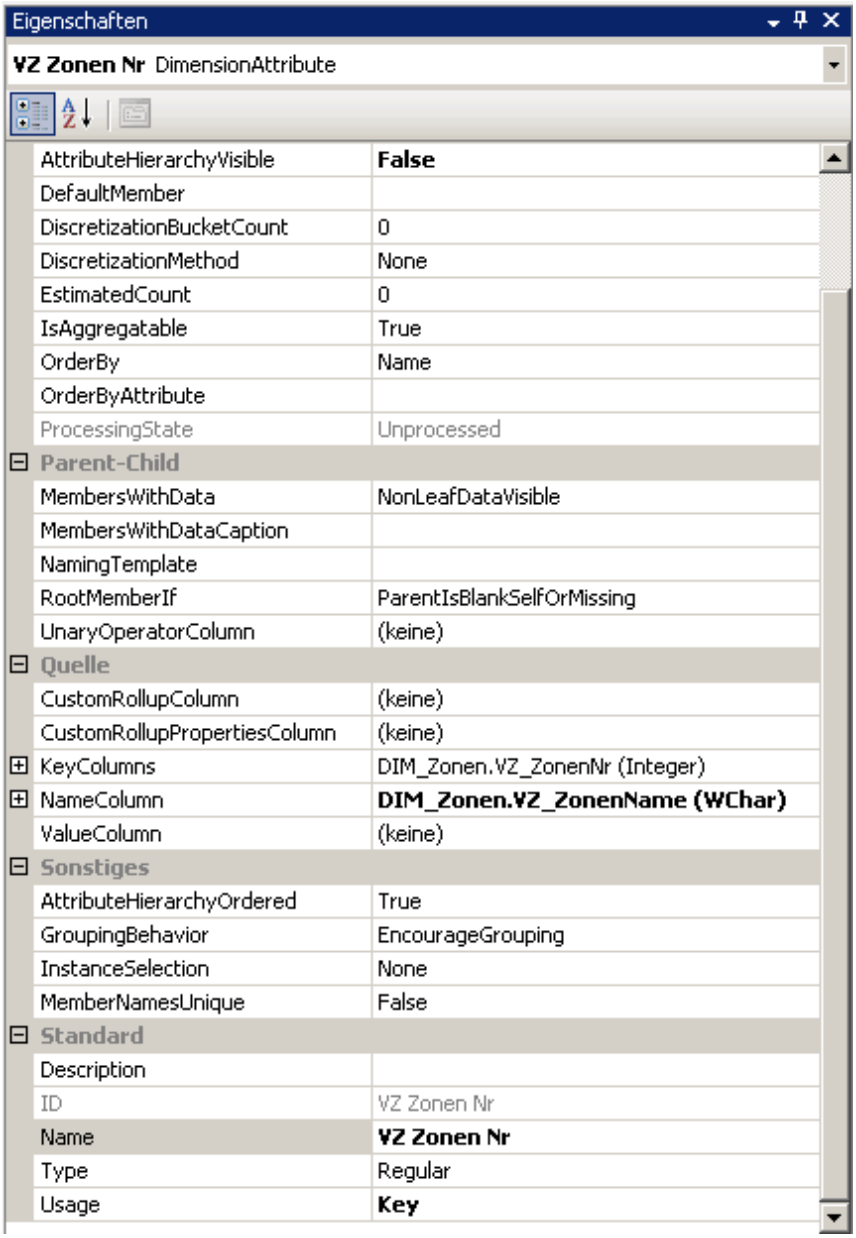

Abbildung 54: Eigenschaften des Dimensionsattribut VZ_ZonenNr

Die Dimensionen `Alter`, `Fahrscheincode`, `Tarifebenen` und `Verkehrsunternehmen` beinhalten ebenfalls Parent-Child- Beziehungen. Eine Einstellung erfolgt jeweils nach der oben vorgestellten Vorgehensweise.

Ein Doppelklick auf den Cube im Projektmappen-Explorer öffnet die Ansicht des Cubes. Links oben befindet sich die Ansicht der Measures, darunter die Ansicht der verwendeten Dimensionen. Zur Bestimmung der Measures können diese aus den Faktentabellen mittels „Drag and Drop“ in den Bereich Measures verschoben werden. Die zugehörigen Tabellen werden automatisch als Faktentabellen deklariert. Die Auswahl der Dimensionen kann links unten im Bereich Dimensionen durch Anwahl rechte Maustaste – „Cubedimension hinzufügen“ getätigt werden.

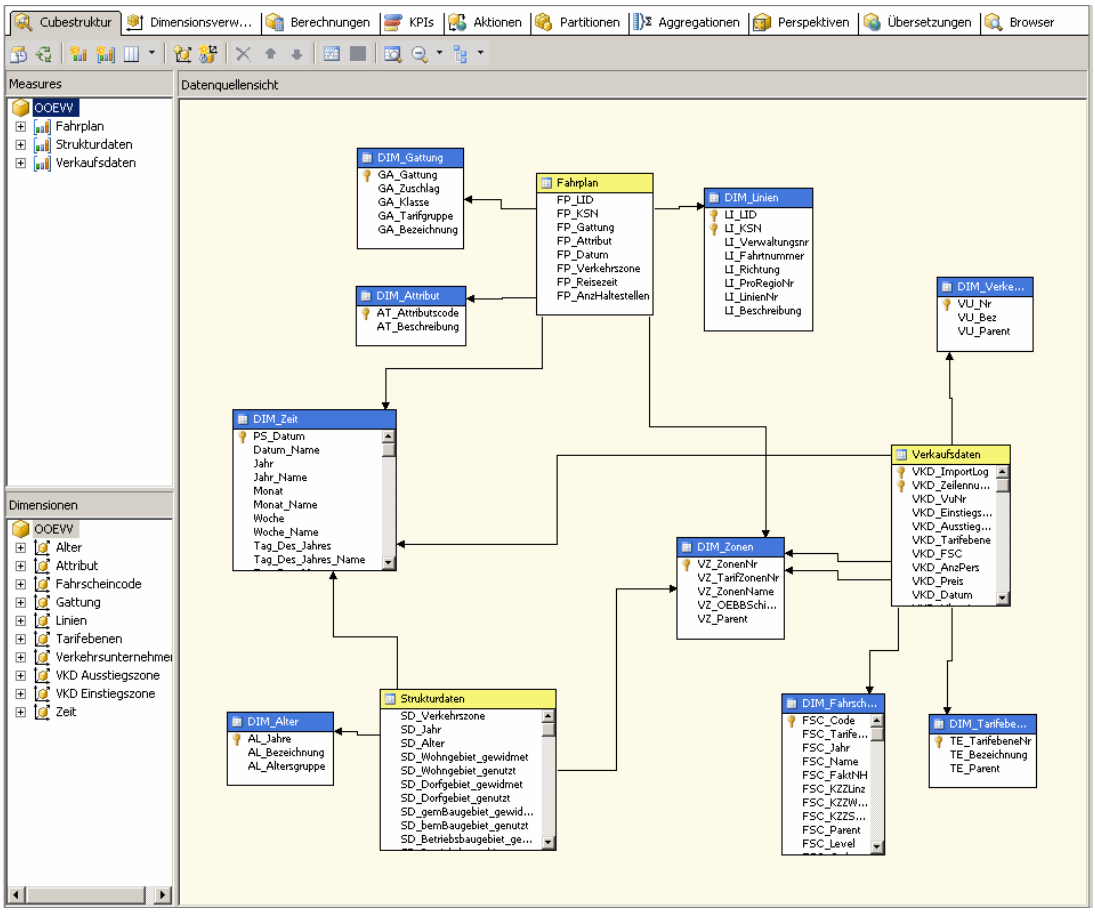

Abbildung 55: Fertige Zuordnung der Measures und Dimensionen in der Cubestruktur

Die fertige Zuordnung ist in Abbildung 55 zu sehen. Links oben werden die drei Faktentabellen mit den jeweiligen Measures abgebildet. Links unten werden die dazu verwendeten Dimensionen dargestellt. Die Datenquellensicht zeigt nun die Tabellen mit eingefärbter Namensbezeichnung. Dimensionstabellen sind blau gefärbt, Faktentabellen sind gelb gefärbt.

Die Dimensionsverwendung kann im Register Dimensionsverwaltung angepasst werden. Es werden die Dimensionen und die Measuregruppen (gleich den Faktentabellen) in einer Matrix gegenübergestellt. In den Zellen kann das Beziehungsattribut zwischen Measuregruppe und Dimension eingestellt werden. Graue Zellen markieren, dass keine Beziehung zwischen Dimension und Measuregruppe besteht.

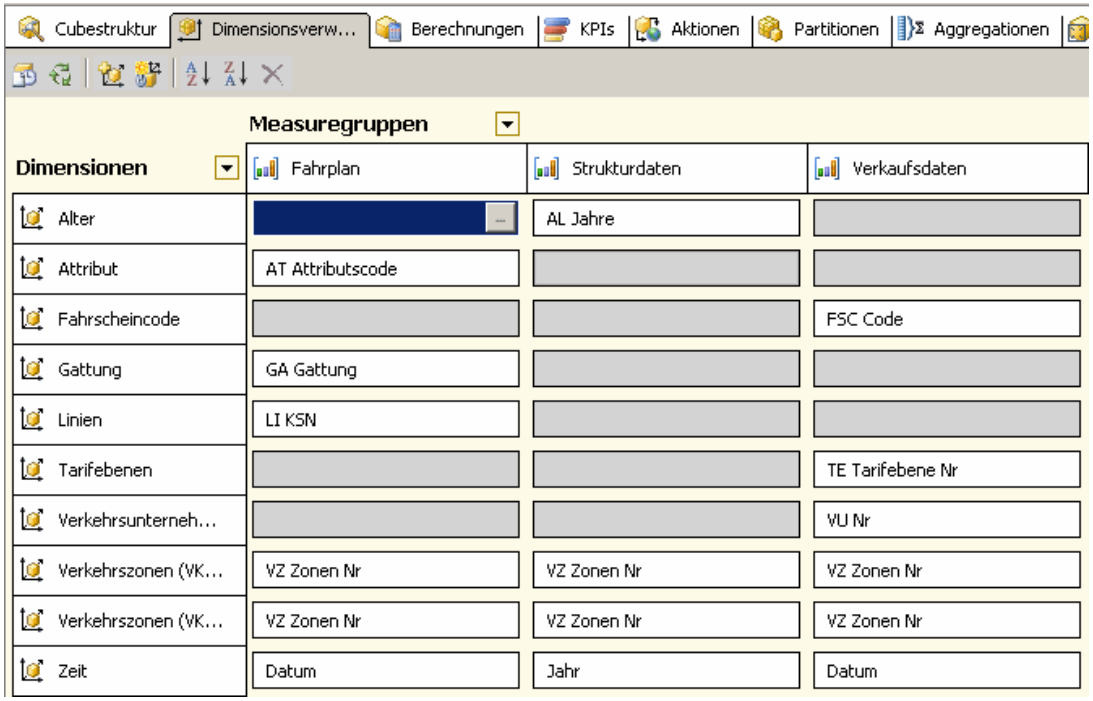

Abbildung 56: Dimensionsverwendung für Measuregruppen im SSAS

Abbildung 56 veranschaulicht die Verknüpfungen zwischen den Dimensionen und den Measuregruppen. Die Dimension `Verkehrszonen` wurde zur Modellierung auf zwei Dimensionen aufgeteilt. Zeilen mit vollständigen Zelleneinträgen ergeben gemeinsame Dimensionen für alle drei Faktentabellen. Wie der Abbildung 56 zu entnehmen ist, sind dies die Dimensionen `Zeit` und `Verkehrszonen`.

Nach dem Übersetzen des Cube im SSAS ist die Phase 5 – Logisches Design abgeschlossen.

7 Importieren der Daten in das Data Warehouse

Zum Importieren der Daten in ein Data Warehouse wurde in Kapitel 2.1 Abbildung 3 die zweite Schicht mit dem Begriff ETL (Extrahieren – Transferieren – Laden) eingeführt. Dieser Begriff ETL wurde in Kapitel 2.2 weiter vertieft.

ETL teilt den Prozess des Datenimports in die drei Stufen Extrahieren, Transformieren und Laden. Diese werden im SQL-Server von Microsoft im Werkzeug SQL-Server-Integration-Services (SSIS) vereint.

7.1 Vorstellung des Werkzeuges SQL-Server-Integration-Services

Der Importprozess für ein Data Warehouse setzt im SQL-Server ein SSIS-Projekt voraus. Diese können mit dem SQL-Server Business-Intelligence-Development-Studio erstellt und bearbeitet werden.

Nach dem Start des SQL-Server Business-Intelligence-Development-Studio erscheint die Startseite. Hier kann ein neues SSIS-Projekt durch Auswahl einer SSIS-Vorlage angelegt werden. Die einzelnen Importprozesse werden in verschiedene SSIS-Pakete im SSIS-Projekt

aufgeteilt. Das Einfügen eines neuen Paketes bringt eine leere offene Paketseite in der Ablaufsteuerungssicht zum Vorschein.

Pakete stellen zwei Hauptfunktionen dar. Die erste Hauptfunktion ist die Ablaufsteuerung. Über Pakete können Prozesse für den SQL Server angestoßen, andere Pakete aufgerufen, E-mails versandt und auch geplante Wartungstasks wie Datenbanksicherungen oder Indexerneuerung durchgeführt werden.

Der Datenfluss wird von (Knight, Veerman, Dickinson, Hinson, & Herbold, 2008) als zweite Hauptfunktion genannt. Dieser beinhaltet den eigentlichen ETL-Prozess. Das Einfügen eines Datenflusstasks in das Paket startet die Implementierung des ETL-Prozesses. Das Öffnen des Datenflusstasks schaltet das Paket in die Ansicht Datenfluss um.

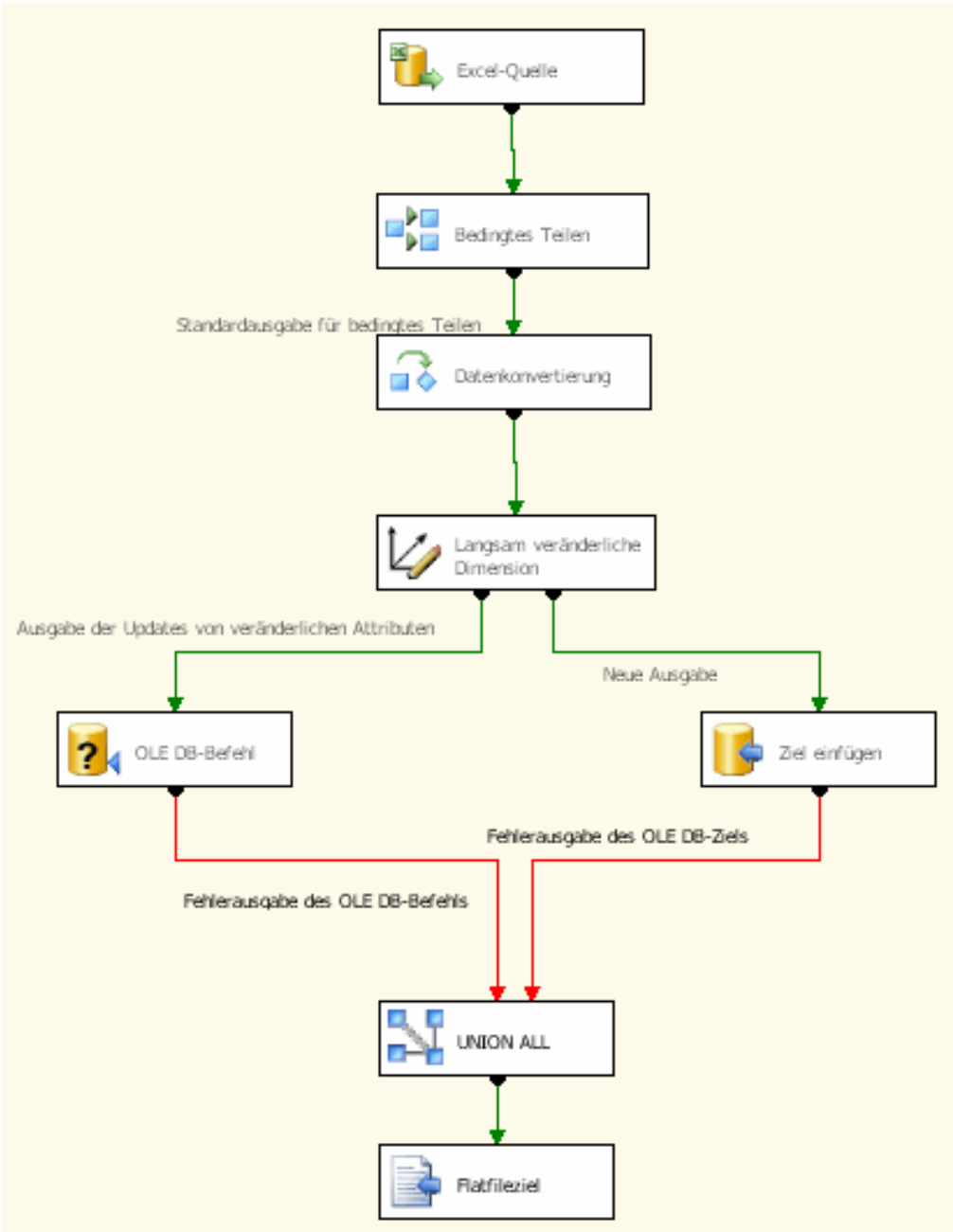

Abbildung 57: Einfacher ETL-Prozess als Datenfluss im SSIS-Paket

Abbildung 57 veranschaulicht ein einfaches Datenflusspaket zum Laden einer Dimension. Die verschiedenen Werkzeuge werden als Blöcke dargestellt. Die Verbindungen erfolgen mit Pfaden. Diese können grün für erfolgreiche Weiterbearbeitung oder rot für Weiterverarbeitung im Fehlerfall sein.

In der Toolbox erscheinen die möglichen Werkzeuge für den Datenflusstask. Diese sind nach dem ETL-Prozess gegliedert.

Sechs verschiedene Datenflussquellen bieten die SSIS an.

- ADO.NET-Quelle
- Excel-Quelle
- Flatfilequelle
- OLE-DB-Quelle
- Rohdatendatei-Quelle
- XML-Quelle

Die ADO.NET-Quelle ist ähnlich einer OLE-DB-Quelle, welche weiter unten vorgestellt wird. Sie wird zur Verbindung von Datenbanken mit den SSIS benötigt. Die Unterscheidung liegt in der Datenquelle. Verlangt die Datenquelle explizit eine ADO.NET-Verbindung, so ist eine ADO.NET-Quelle zu verwenden. Kann eine OLE-DB-Quelle eingesetzt werden, so ist diese zu bevorzugen.

Eine Excel-Quelle kann direkt aus Excel-Tabellen Daten importieren. Die Excel-Tabelle wird dabei wie eine Datenbank verwaltet. Jede Zeile in der Tabelle wird als Tupel gewertet. Die Spalten in der Tabelle ergeben die möglichen importierten Spalten. Ein Assistent prüft während der Konfiguration die Datenquelle und bestimmt die Datentypen. Als Datentypen wird zwischen String und Double unterschieden. Zahlen werden automatisch in Double umgewandelt, alle anderen Felder als Text gewertet. Aus dieser Prüfung wird bestimmt, welche Spalte in SSIS welchen Datentyp zugewiesen bekommt. Ein Benutzereingriff ist nicht möglich. Sind in einer Spalte erst später Werte mit alternativem Format, so kann der Importprozess nur fehlerhaft durchgeführt werden. Diese Felder werden mit NULL befüllt. Abhilfe kann das Anlegen einer Musterzeile oben im Quelldokument bringen. Eine andere Lösung ist der Export der Quelle als Textformat und das Einlesen über die Flatfilequelle.

Die Flatfilequelle bietet das Einlesen von Textquellen an. Für die Spaltenerkennung bietet der Verbindungsmanager die Einstellungen „Mit Trennzeichen“, „Feste Breite“ und „Rechter Flatterrand“ an. Die Einstellung „Mit Trennzeichen“ verlangt ein Trennzeichen, welches von Spalte zu Spalte verschieden sein kann. Die Einstellung „feste Breite“ gibt für jede Spalte eine feste Breite vor. Mit der Einstellung „Rechter Flatterrand“ gilt für die erste bis zur vorletzten Spalte eine feste Breite. Nur die letzte Spalte wird mit Trennzeichen, zum Beispiel Wagenrücklauf und Zeilenvorschub, abgeschlossen. Das Ausgabeformat der einzelnen Spalten kann vorgegeben werden.

Zum Import der Daten aus einer Datenbank wird eine OLE-DB-Quelle verwendet. Diese stellt eine Verbindung zur Datenquelle her und stellt deren Felder mit ihren Eigenschaften zur Verfügung. Die Schnittstelle OLE-DB-Quelle unterstützt alle Datenbanken, die zum OLE-DB Übertragungsstandard kompatibel sind. Als Beispiele nennen (Knight, Veerman, Dickinson, Hinson, & Herbold, 2008) SQL-Server, Oracle oder auch DB2.

Eine Rohdatendatei importiert Rohdatendateien, welche mit SSIS auch exportiert werden können. Der optimierte Zugriff auf diese Quellen erhöht massiv die Geschwindigkeit der Verarbeitung. Diese Quellen werden genutzt, um Daten zwischen verschiedenen Paketen auszutauschen (Knight, Veerman, Dickinson, Hinson, & Herbold, 2008, S. 142).

Für alle XML-basierten Quellen gibt es eine eigene XML-Quelle. Eine Angabe eines XSD-Schemas oder die Abbildung des inneren Datenschemas der XML-Quelle ist möglich.

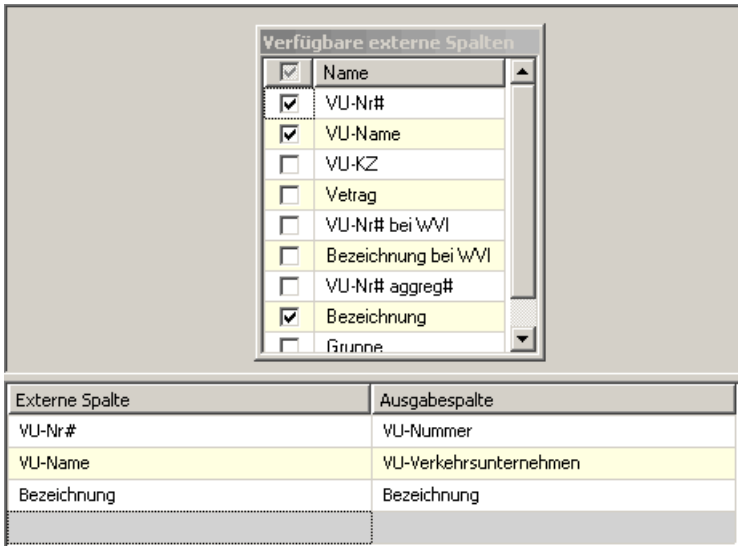

Abbildung 58: Auswahl der Spalten aus einer Quelle in SSIS zur Weiterverarbeitung

Allen Quellen ist es gemein, dass die zu importierenden Spalten ausgewählt werden können (Projektion), welche weiter verarbeitet werden sollen. Wie Abbildung 58 zeigt, können diese für eine verbesserte Lesbarkeit anders benannt werden.

Zum Transformieren der Daten stellen SSIS insgesamt 29 Werkzeuge zur Verfügung. Einen Auszug aus der Liste der Werkzeuge inklusive Kurzbeschreibung stellt die nachstehende Tabelle 12 bereit.

Abgeleitete Spalte	Ersetzt eine Spalte oder erzeugt eine neue Spalte mithilfe eines Ausdrucks
Aggregat	Aggregiert Werte nach bestimmten Spalten
Bedingtes Teilen	Teilt Tupel nach Ausdrücke in verschiedene Pfade. Die Anzahl der Pfade kann variiert werden
Datenkonvertierung	Ändert den Datentyp einer oder mehrerer Spalten
Langsam veränderliche Dimension	Gestaltet den Einfüge- und Updateprozess für Dimensionen
Multicast	Dupliziert die Tupel in mehrere Pfade
OLE-DB-Befehl	Ermöglicht Zugriff auf eine Datenbank mit SQL
Scriptkomponente	Eigene Programmierung von Funktionen in C#
Sortieren	Sortiert Tupel nach bestimmte Spalten, kann doppelte Werte ausfiltern
Union All	Zusammenführen von Tupeln in einen Pfad, keine JOIN-Operation – Anzahl der Tupel steigt
Zusammenführungsjoin	Führt zwei Pfade in einen Pfad zusammen und vereint die Tupelmenge anhand einer oder mehrerer Spalten – Anzahl der Spalten steigt.

Tabelle 12: Auszug aus den Werkzeugen in SSIS zum Transformieren der Daten

Es gibt noch weitere Werkzeuge für die Transformation der Daten. Weiterführende Information zu den vorgestellten und den nicht angeführten Werkzeugen ist in der einschlägigen Literatur verfügbar. Als Beispiel wird (Knight, Veerman, Dickinson, Hinson, & Herbold, 2008) angeführt.

SSIS stellen für den Ladeprozess zwölf unterschiedliche Datenflussziel-Werkzeuge zur Verfügung. Abbildung 59 liefert eine Übersicht der Werkzeuge.

Abbildung 59: Übersicht der möglichen Datenflussziele in SSIS

Das Datenflussziel ADO.NET kann Daten in eine Datenbank laden, welche über eine ADO.Net-Schnittstelle verfügt. Mit einem Recordsetziel kann ein ADO-Datensatz exportiert werden. Wie schon in Kapitel 2.2.1 bei den Datenflussquellen, sollte das später vorgestellte OLE-DB-Ziel bevorzugt werden.

Mit einem Data Mining-Modelltraining-Ziel kann ein Data Mining-Tool trainiert werden. Zur Thematik Data Mining gibt es weiterführende Informationen unter anderen in (Harinath, Zare, Meenakshisundaram, Carroll, & Lee, 2009). Ein DataReader-Ziel bietet die Möglichkeit, das interne Datenformat in SSIS zu exportieren und mit externen Werkzeugen wie .NET diese weiter zu verarbeiten.

Excel-Tabellen können mit dem Excel-Ziel erstellt werden. Die Auswahl der Spalten und Reihenfolge kann einfach mit diesem Werkzeug konfiguriert werden.

Direkt in die Dimension oder Partition des Data Warehouse zu schreiben ermöglicht die Dimensionsverarbeitung oder die Partitionsverarbeitung. Ein Flatfileziel generiert eine Textdatei nach den Angaben der Konfiguration. Trennzeichen oder fixe Spaltenbreiten sind möglich.

Das OLE-DB-Ziel lädt eine Datenbank mit den Daten aus dem Datenfluss. Die Zuordnung der Spalten erfolgt im Konfigurationsmenü. Eine Auswahl an Lademöglichkeiten umfasst die Angabe von SQL-Befehlen oder normales beziehungsweise schnelles Laden für Tabellen oder Sichten. Das Ziel kann fix oder über Variable vorgegeben werden. OLE-DB-Ziele sind die am häufigsten verwendeten Ziele.

Mit dem Rohdatendatei-Ziel schafft sich SSIS die Möglichkeit, Daten in kompakter und performanter Weise auszulagern und mit der in Kapitel 2.2.1 vorgestellten Rohdatendatei-Quelle an anderer Stelle wieder zu importieren.

Die SQL-Server-Ziele sind speziell auf SQL-Server abgestimmt. Diese Ziele sind nur anzuwenden, wenn die Datenbank auf der gleichen Maschine wie das Paket von SSIS läuft (Knight, Veerman, Dickinson, Hinson, & Herbold, 2008). Eine Besonderheit von SQL-Server Compact-Ziel ist die Möglichkeit, Datenbanken auf mobilen Geräten wie Smartphones oder Pocket-PCs zu laden.

7.2 ETL-Prozesse für Dimensionen

Das Data Warehouse für den OÖ Verkehrsverbund aus Abbildung 55 (Kapitel 3.1) umfasst neun verschiedene Dimensionen. Jeder dieser Dimensionen unterliegt eine Dimensionstabelle. Für den Ladeprozess der einzelnen Dimensionstabelle wird ein zugeordneter ETL-Prozess entworfen. Später werden noch für die drei Faktentabellen ETL-Prozesse vorgestellt.

Für Dimensionen, deren Ausprägungen seitens des OÖ Verkehrsverbundes festgelegt werden können, wird eine eigene Excel-Datei „Dimensionen.xls“ generiert, welche die vordefinierten Ausprägungen speichert.

Der allgemeine Aufbau eines ETL-Prozesses für Dimensionen beginnt mit dem Einlesen der Daten. Je nach Quelle werden Excel-Quellen, OLE-DB-Quellen und Flatfilequellen eingesetzt. Nach dem Transformationsprozess werden die Daten als *„Langsam veränderliche Dimension“* in die zugehörige Dimensionstabelle geladen. Nach dem Ladeprozess gibt es noch eine Fehlerauswertung, welche alle fehlerhaften Tupel in eine Textdatei schreibt.

Das Laden der `Verkehrszonen` umfasst drei Datenflüsse. Der erste Datenfluss importiert die Daten für die Mobilitätskreise. Während des Importierens werden mit dem Werkzeug

„bedingtes Teilen“ die leeren Zeilen in der Quelltabelle verworfen. Im Datenkonverter werden die Datentypen der Mobilitätskreisnummern auf INT geändert. Das Werkzeug „Abgeleitete Spalte“ erzeugt eine Spalte Parent für die Hierarchie. Da Mobilitätskreise die oberste Klassifikationsstufe beschreiben, werden die eigenen Mobilitätskreisnummern eingefügt.

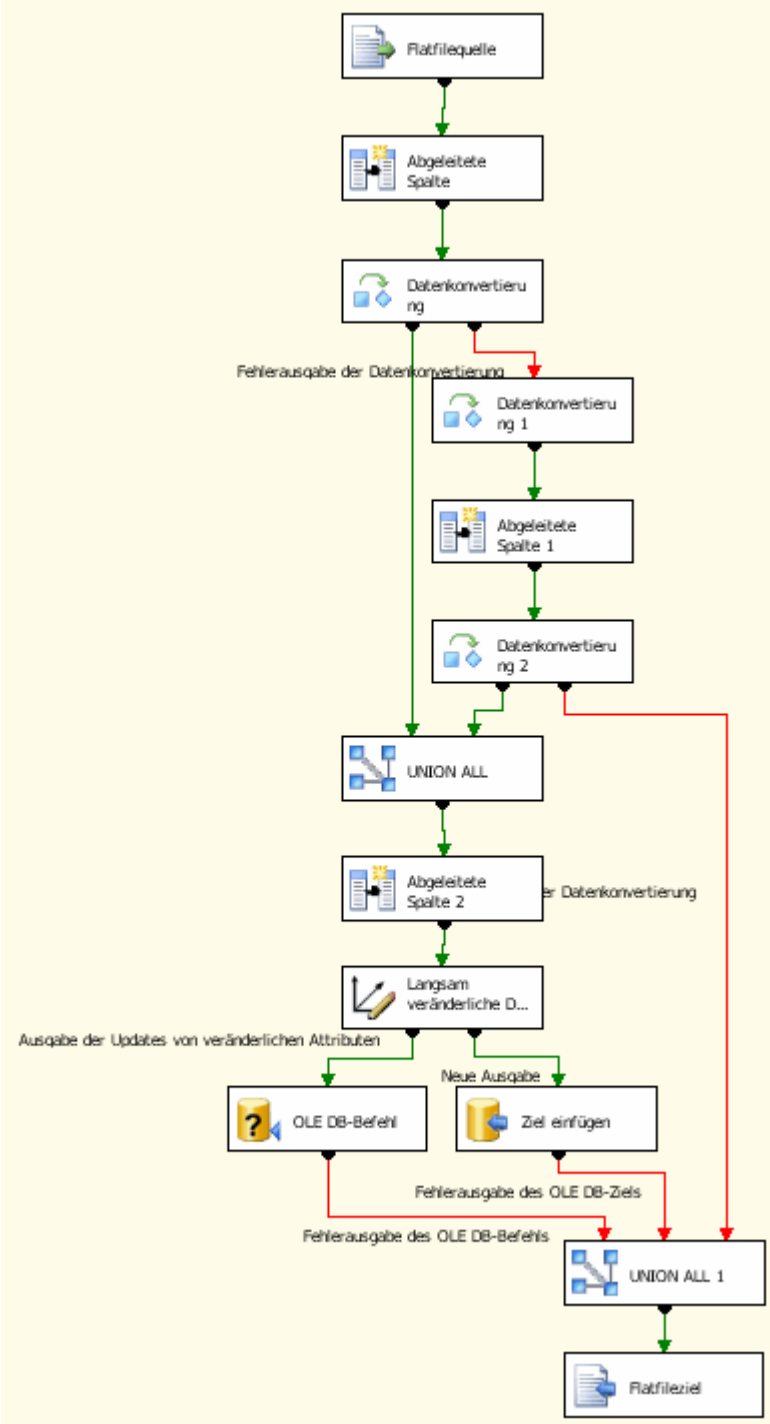

Abbildung 60: SSIS-Paket für die Dimensionstabelle Zonen

Der zweite Datenfluss importiert die Daten für die Konzeptregionen. Die leeren Zeilen werden wieder mittels „bedingtes Teilen“ aussortiert und die Datentypen entsprechend angepasst. Die Spalte Parent wird mit dem Werkzeug „Abgeleitete Spalte“ generiert. Dazu wird die `Konzeptregionsnummer` ausgelesen und diese durch Zehn geteilt. Dies ergibt die `Mo-`

bilitätskreisnummer. Werden fünfstellige Nummern erkannt, so werden diese als Parent eingefügt. Die Kernzonen Linz, Wels und Steyr sind ebenfalls Konzeptregion und Mobilitätskreis.

Im dritten Datenfluss werden erst die Zonen importiert. Wie in Abbildung 60 abgebildet, werden nach dem Import die NULL-Werte in der Spalte „Sonstige Info“ durch den nummerischen Wert 0 ersetzt. Danach werden die Werte in das Zahlenformat INT konvertiert. Werden fehlerhafte Konzeptregionsnummern während der Konvertierung ausgesiebt, so werden diese zuerst als Text konvertiert, die ersten beiden Zeichen herausgehoben und als Zahl formatiert wieder in die Spalte eingetragen. Zusätzliche (überflüssige) Zeichen in der Spalte werden dadurch gelöscht. Treten bei einer neuerlichen Konvertierung nach INT wieder fehlerhafte Tupel auf, so werden diese für eine manuelle Bearbeitung in einer Textdatei gespeichert. Die erfolgreich rekonstruierten Tupel werden mittels Union-All-Werkzeug (Abbildung 60 Mitte) dem normalen Datenfluss wieder zugeführt.

Für die Verkehrsunternehmen werden nur die Spalten Verkehrsunternehmernummer, Bezeichnung und Gruppierung eingelesen. Die Spalten Verkehrsunternehmernummer und Gruppierung werden nach INT konvertiert. Die Spalte Bezeichnung wird auf 50 Zeichen eingeschränkt.

Nach dem Extrahieren der Tarifebenen aus der Excel-Tabelle „Dimensionen.xls“ werden die Leerzeilen mittels „Bedingtes Teilen“ aussortiert. Danach werden die Tarifebenennummer und die Gruppennummer nach INT konvertiert.

Von der Tabelle Fahrscheincode werden die Spalte Jahr, Tarifebene, Fahrscheincode, Kartenname, Fahrscheincode 2 als Gruppierung, Nutzungshäufigkeit und die Kernzonenzuschläge eingelesen. Excel liefert für die Leerspalten einen Bindestrich „-“. Dieser wird ersetzt durch die Ziffer 0. Danach werden die Ziffernspalten konvertiert und die Textspalten auf 50 Zeichen begrenzt.

Die Fahrscheincodes werden versioniert, da sich diese von Jahr zu Jahr ändern. Dazu gibt es eine Spalte Jahr für die Jahreskennung.

Einer der einfachen ETL-Prozesse ist jener für die Dimensionstabelle Alter. Die Daten werden aus der Excel-Tabelle „Dimensionen.xls“ ausgelesen und die Zahlen in INTEGER sowie die Länge der Bezeichnung konvertiert.

Die Dimensionstabelle Linien wird aus zwei Quellen befüllt. Information aus der Quelle „Linien“ wird ergänzt mit Daten aus dem Header der Fahrplandaten.

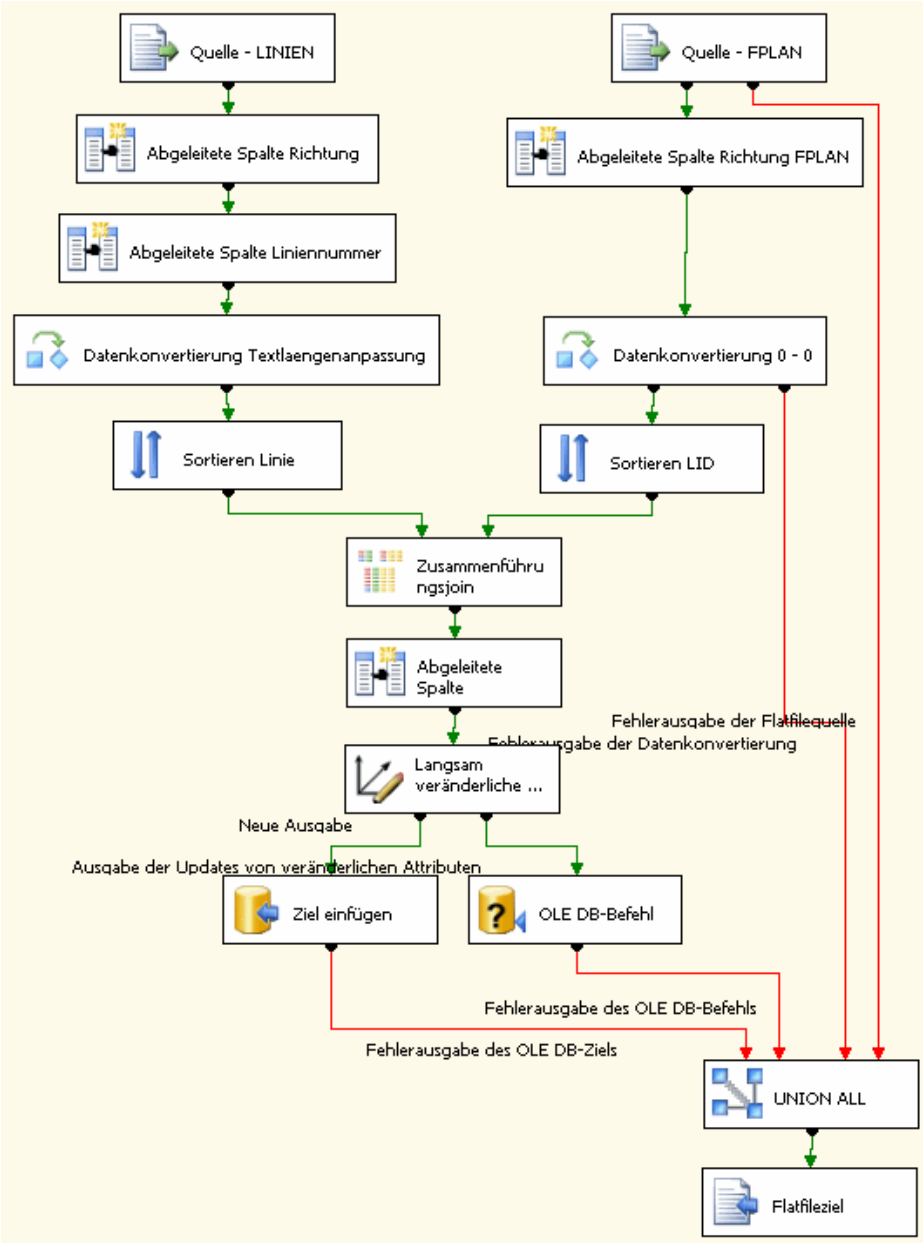

Abbildung 61: SSIS-Paket für die Dimensionstabelle `Linien`

Aus den eingelesenen Daten wird für beide Quellen in Abbildung 61 die Spalte `Richtung` aufbereitet. Die Richtung ist in der Quelle `Linien` mit einem Buchstaben nach der `Liniennummer` gekennzeichnet. Diese Buchstaben werden ausmaskiert und in einer eigenen Spalte für die Richtung ausgewertet. Das zweite Werkzeug „Abgeleitete Spalte“ bereinigt die `Liniennummer` um die Richtungsbuchstaben. Die Längen der Textspalten werden mittels Datenkonvertierung an die Zielgrößen angepasst. Danach werden die Tupel nach `Linien-ID` und `Richtung` sortiert.

Die `Richtung` der zweiten Quelle `FPLAN` wird aus einer Textzeile extrahiert. Weiters wird in diesem Werkzeug noch die `Fahrtnummer` und die `Verwaltungsnummer` bereinigt. In der Quelle gibt es Tupel, welche keine `Verwaltungsnummer` mitbringen. Damit ist die Spalte leer. Die folgende Datenkonvertierung meldet einen Fehler, wenn kein Wert für die Konvertierung bereitgestellt wird. Damit dieser Fehler nicht auftritt, wird die Ziffer 0 für leere

Verwaltungsnummern eingetragen. Eine Datentypänderung erfahren die Spalten `Richtung`, `Fahrtnummer` und `Verwaltungsnummer` mit dem Werkzeug Datenkonvertierung. Für das Zusammenführen der beiden Datenströme werden gleich sortierte Mengen vorausgesetzt. Dies wird durch einen vorgeschalteten Sortieralgorithmus sichergestellt.

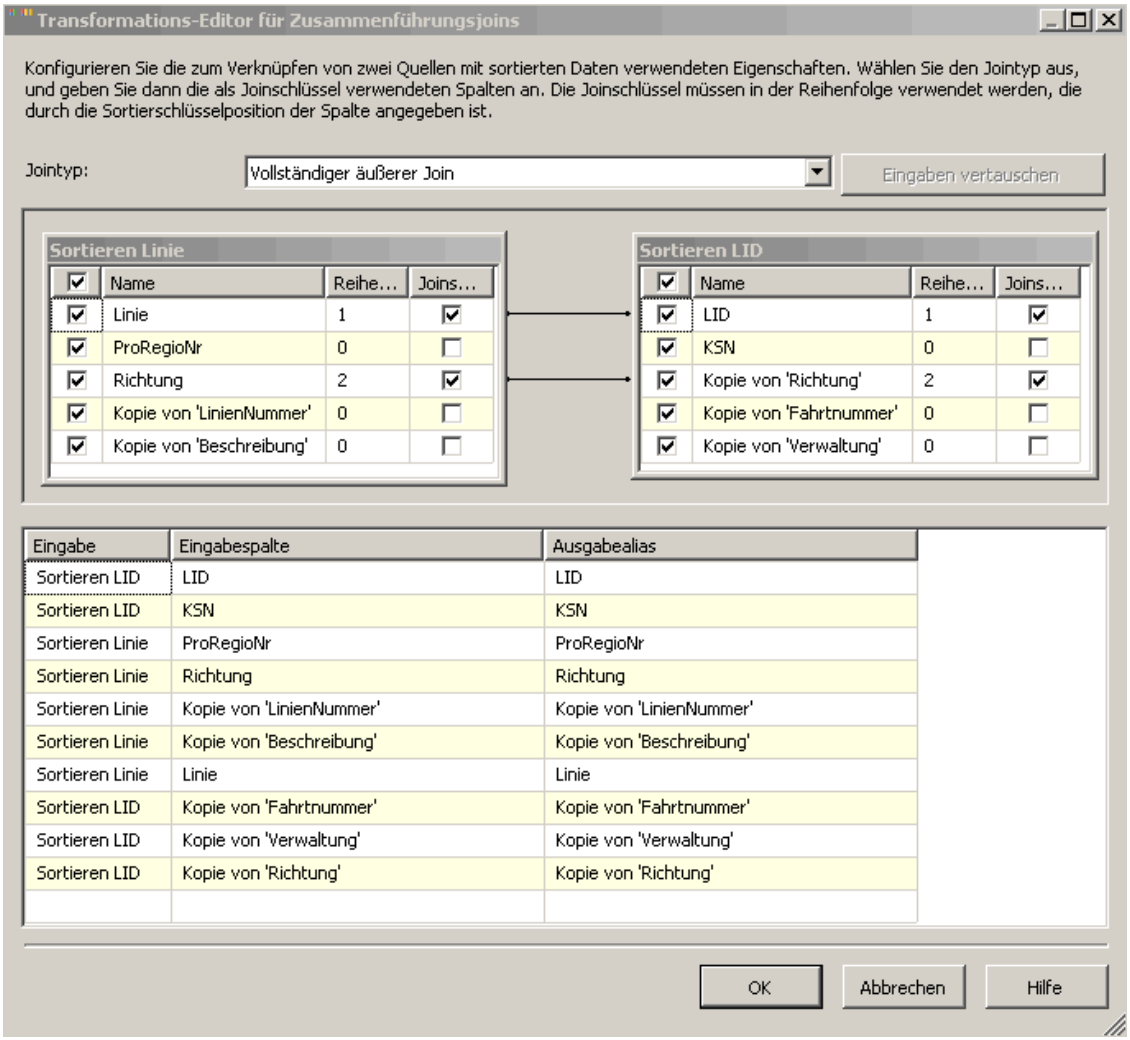

Abbildung 62: Zusammenführungsjoin der Quellen `FPLAN` und `Linien`

Der Zusammenführungsjoin in Abbildung 62 führt beide Datenströme anhand der JOIN-Attribute `Linien-ID` und `Richtung` zusammen. Als JOIN-Typ wird „Vollständiger äußerer JOIN“ eingestellt. Ein vollständiger äußerer Join versucht für die größere Menge möglichst vollständig, die neuen Spalten mit dem Inhalt der kleineren Menge aufzufüllen. Die Tupelanzahl nach dem Join ist genau gleich mit der Anzahl der größeren Menge.

Nach der Zusammenführung der Datenquellen befindet sich in Abbildung 61 ein Werkzeug „Abgeleitete Spalte“ zum Auffüllen jener Spalten, welche keine `Linien-ID` oder `Kurs-nummer` eingetragen bekommen haben. Dieser Zustand tritt auf, wenn eine Linie existiert, zu welcher keine Fahrten registriert sind.

Aus der Excel-Datei „Dimensionen.xls“ werden die Werte der Tabelle `Attribut` exportiert und in die Dimensionstabelle `DIM_Attribut` eingefügt. Ein Update der Daten ist möglich. Historische Werte werden nicht benötigt.

Die Daten werden aus der Excel-Tabelle eingelesen, mittels Datenkonvertierung die Längen der Textspalten `Attributscode` und `Attribut` auf drei Stellen beziehungsweise 50 Stellen angepasst und dann als veränderliche Dimension in die Dimensionstabelle gespeichert.

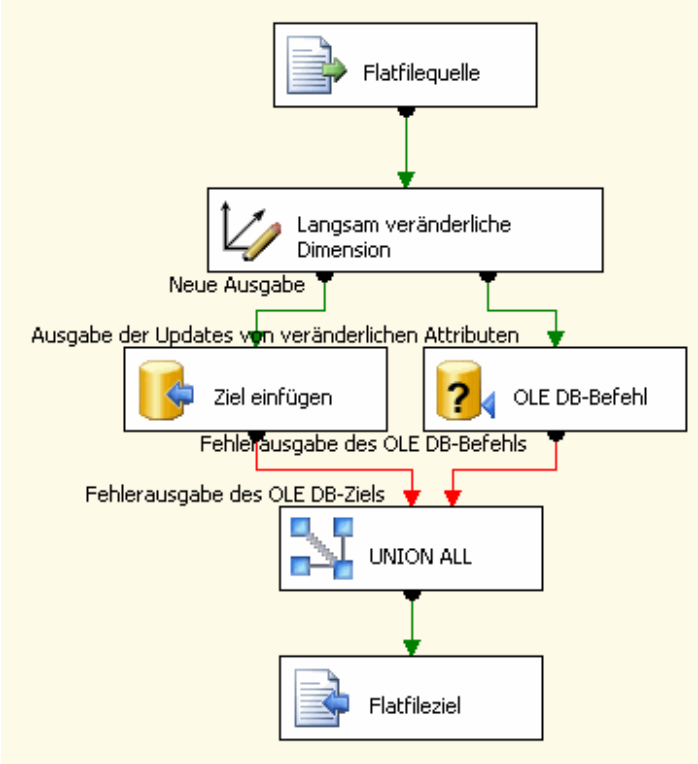

Abbildung 63: SSIS-Paket für die Dimensionstabelle `Gattung`

Wie Abbildung 63 veranschaulicht, lässt sich die Dimensionstabelle `Gattung` am einfachsten laden. Die Information wird aus einer Textdatei extrahiert und noch in der Flatfilequelle formatiert. Der Datenfluss geht sofort weiter zum Datenziel.

7.3 ETL-Prozesse für Faktentabellen

Die Einbindung der Daten in den Cube erfolgt über die Faktentabellen. Fakten bestehen aus Measures und Fremdschlüsselelementen auf die Dimensionstabellen. Die Measures werden für die Verwendung im Cube aufbereitet. Die Fremdschlüssel werden an das Format der Dimensionen angepasst. Vor dem Laden gibt es eine Überprüfung, ob gleichartige Datensätze in der Tabelle bereits existieren. Eine Duplizierung der Werte im Cube ist nicht erwünscht, da dies Auswertungsfehler zur Folge hat.

Für die Historisierung der Measures ist die Dimension `Zeit` vorgesehen. Diese ordnet jedem Measure eine zeitliche Komponente zu. Eine systemeigene Historisierung ist nicht vorgesehen. Je nach Faktentabelle liegt die Granularität der Zeit in Tagen oder Jahren vor. Für `Strukturdaten` reicht eine jährliche Festlegung der Historie aus. Die Änderungshäufigkeit der Quelldaten liegt im Allgemeinen in diesem Zeitraffer.

Für Fahrplandaten oder Verkaufsdaten wird eine tagesgenaue Hierarchisierung festgelegt. Diese Daten erfahren in den Geschäftsprozessen häufige Änderungen. Die Datenmenge ist demzufolge wesentlich höher als jene der Strukturdaten.

Die folgenden Kapitel gehen auf die einzelnen ETL-Prozesse der Faktentabellen Verkaufsdaten, Strukturdaten und Fahrplandaten ein.

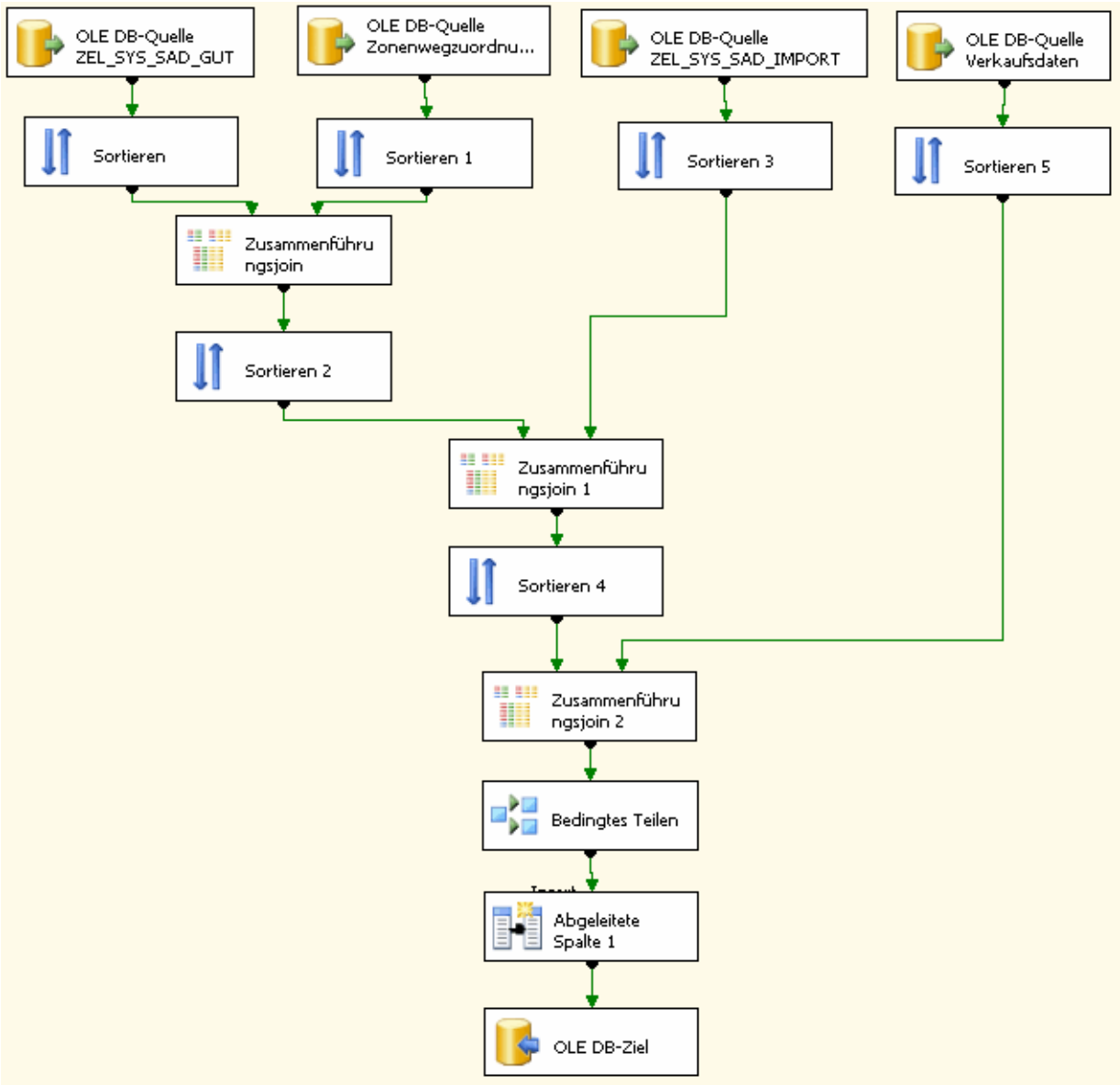

Abbildung 64: ETL-Prozess für die Faktentabelle Verkaufsdaten

Im oberen Bereich der Abbildung 64 befinden sich die Datenquellen. Von links nach rechts betrachtet, beinhaltet die erste Datenquelle die Fahrkartendaten und eine Referenz zur Zonenwegbeschreibung. Die Fahrkartendaten umfassen das Verkaufsdatum, der Preis, die Tarifebene, die Anzahl der Personen, die Fahrernummer, die Fahrscheinnummer, die Kursnummer, die Liniennummer und den Kartentyp. Die zweite Quelle umfasst die Zonenwegbeschreibung. In dieser Quelle werden die Einstiegszone, die Ausstiegszone und der Weg dazwischen abgebildet. Die nächste

Quelle stellt die Verkehrsunternehmensdaten zur Verfügung. Die letzte Quelle in der Reihe stellt das Datenziel dar. Diese wird über die Schlüsselelemente abgefragt, ob bereits ein Tupel mit diesen Daten geladen wurde. Eine Verhinderung doppelter Werte in der Faktentabelle ist das Ziel.

Das Hauptaugenmerk dieses ETL-Prozesses liegt in der Zusammenführung der einzelnen Quellen. Die beiden Quellen links oben in Abbildung 64 werden nach dem Fremdschlüssel der `Zonenwegzuordnung` sortiert. Ein Left-Outer-Join fügt zum Inhalt der Tabelle `Fahrkartendaten` die Daten für die `Einstiegszone`, die `Ausstiegszone` und die `Anzahl` der durchfahrenen Zonen hinzu.

Ein Umsortieren ermöglicht das Verknüpfen der Daten mit der dritten Quelle. Ein Left-Outer-Join verbindet die kumulierten Daten von oben mit den Daten der `Verkehrsunternehmen`. Nach einem weiteren Sortieralgorithmus werden die neuen Quelldaten mit den bereits vorhandenen Daten aus der Faktentabelle verglichen. Dies erfolgt durch eine Join-Operation der neuen Quelldaten mit den Zieldaten. Dabei wird eine Spalte der Zieldaten in die Quelldaten übernommen. Das Werkzeug „Bedingtes Teilen“ sortiert anhand dieser beigefügten Spalte die vorhandenen von den neu einzufügenden Tupel aus. Dazu wird diese Spalte auf den Wert NULL abgefragt. Befindet sich ein anderer Inhalt außer NULL in der Spalte, so wurde ein passendes Tupel in der Faktentabelle gefunden. Das Tupel aus den Quelldaten wird ausgeschieden. Alle verbleibenden Tupel werden in die Faktentabelle eingefügt.

Die Strukturdaten sind auf insgesamt vier verschiedene Quellen aufgeteilt. Eine Excel-Tabelle beinhaltet die Bevölkerungsdaten der letzten 10 Jahre. Diese wurde in Kapitel 4.2 näher erläutert. Die Zellen haben die Anzahl der Einwohner mit dem bestimmten Alter zum Inhalt. Diese Matrix ist vervielfältigt in Männer, Frauen und Gesamt.

Eine weitere Excel-Tabelle hält die Daten zu Gemeindefläche und Umfang. In einer dritten Tabelle finden sich die gewidmeten Flächen, aufgeteilt nach Gemeinden (siehe Kapitel 4.3). Eine Zuordnungstabelle liefert die Zuordnung von Gemeinde zu Verkehrszone.

Ein vorgelagerter Prozess generiert aus den Bevölkerungsdaten einzelne Tupel nach `Gemeinde`, `Alter` und `Jahr` gegliedert. Als Inhalt der Tupel findet sich die `Einwohnerzahl` getrennt nach `Geschlecht`.

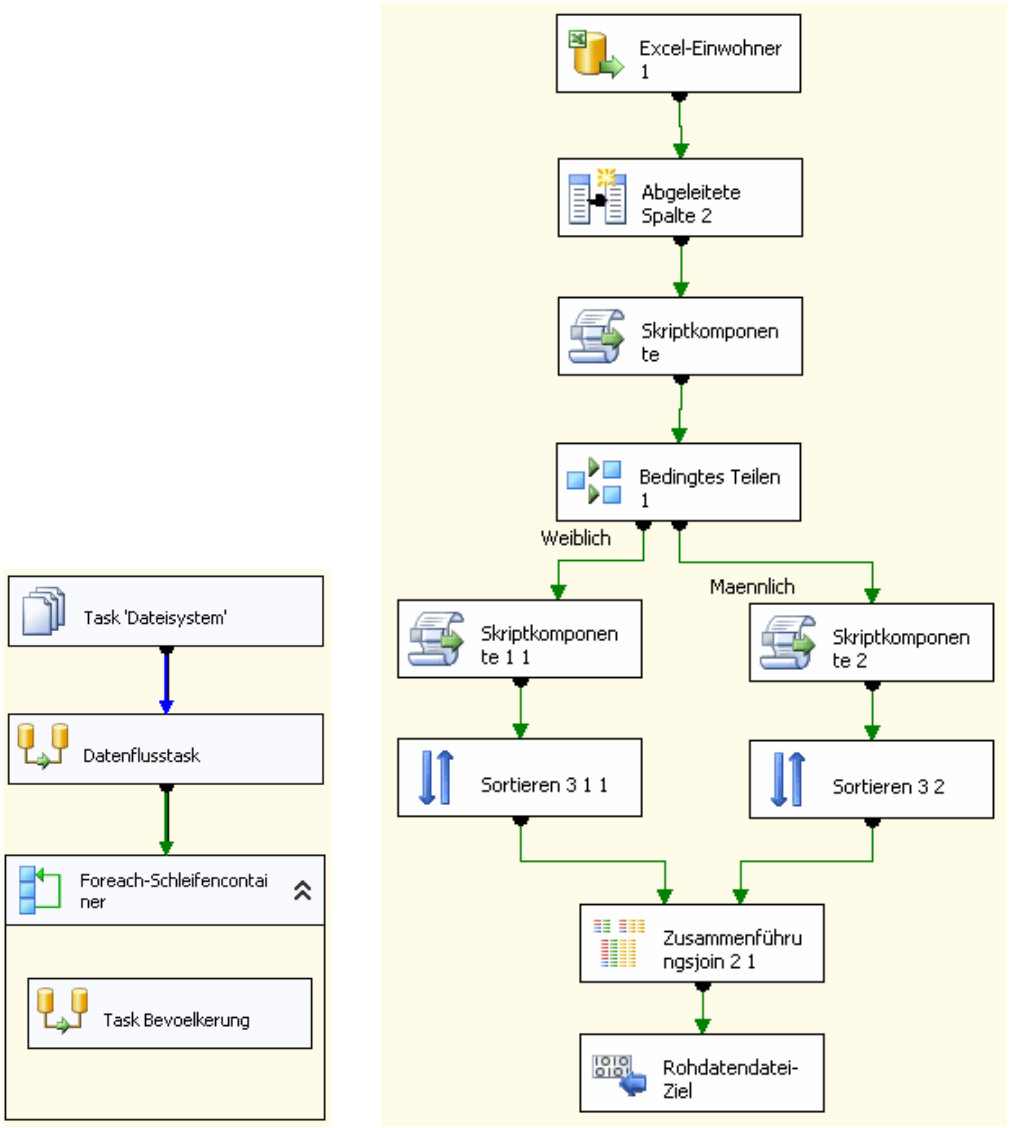

Abbildung 65: Einlesen der verschiedenen Bevölkerungstabellen in einer Schleife

Links in Abbildung 65 befindet sich ein Foreach-Schleifencontainer. Dieser iteriert über alle Excel-Dateien in einem bestimmten Ordner. Der Ordner beinhaltet alle Excel-Dateien mit den Bevölkerungsdaten. Im Container befindet sich der Datenflusstask Bevölkerung, welcher rechts detailliert abgebildet ist.

Dieser Prozess gestaltet sich etwas schwieriger, da es mit den verfügbaren Werkzeugen nicht möglich ist, neue Tupel zu generieren oder tupelübergreifende Aktionen zu gestalten.

```
public override void EingabeO_ProcessInputRow(EingabeOBuffer Row)
{
    if (!Row.Geschlecht_IsNull && !jahrEingelesen)
    {
        int pos = Row.Geschlecht.IndexOf("20");
        if (pos > 0)
        {
            jahrString = Row.Geschlecht.Substring(pos, 4);
            jahrString = jahrString + "-01-01 00:00:00";
              MessageBox.Show(jahrString);
            jahr = Convert.ToDateTime(jahrString);
            jahrEingelesen = true;
        }
    }
    if (jahrEingelesen)
        Row.Jahr = jahr;
}
```

Abbildung 66: Skript zum Einfügen des Jahres für die Bevölkerungsstatistik

Mittels Skriptkomponenten kann dafür Abhilfe geschaffen werden. Die erste Skriptkomponente (Abbildung 66) liest das Jahr der Bevölkerungsstatistik ein, welche in die erste Zelle links (Zelle A1) der Datenquelle eingetragen ist. Ist das Feld nicht leer und das Jahr noch nicht eingelesen, wird aus dem Feld `Geschlecht` (Spalte A in der Quelltabelle) der Wert eingelesen. Aus diesem Wert wird durch Suche von 20 (Suche der ersten beiden Stellen der Jahreszahl 20xx) die Position der Jahreszahl im String erhoben. Gibt es eine Position größer „0“, so wurde ein Wert gefunden. Dieser wird in einen Jahresstring mit vier Zeichen ausgelesen. Der String wird um Formatzeichen für das Datumsformat nach Microsoft ergänzt (Knight, Veerman, Dickinson, Hinson, & Herbold, 2008). Danach wird der String als Datum formatiert. Als letztes wird registriert, dass eine Jahrzahl erhoben wurde. Diese wird in Folge in jedes Tupel eingetragen, so dass mit den verfügbaren Werkzeugen darauf zugegriffen werden kann.

```
public override void EingabeO_ProcessInputRow(EingabeOBuffer Row)
{
    jahr = Row.Jahr;
    gde = Convert.ToInt32(Row.Gde);

    alter = 0;
    einwohner = Convert.ToInt32(Row.F4);
    this.CreateNewOutputRows();
    alter = 1;
    einwohner = Convert.ToInt32(Row.F5);
    this.CreateNewOutputRows();
    alter = 2;
    einwohner = Convert.ToInt32(Row.F6);
    this.CreateNewOutputRows();
    alter = 3;
```

```
        alter = 107;
        einwohner = Convert.ToInt32(Row.F110);
        this.CreateNewOutputRows();
        alter = 108;
        einwohner = Convert.ToInt32(Row.F111);
        this.CreateNewOutputRows();
    }

    public override void CreateNewOutputRows()
    {
        AusgabeOBuffer.AddRow();
        AusgabeOBuffer.Gde = gde;
        AusgabeOBuffer.Jahr = jahr;
        AusgabeOBuffer.Alter = alter;
        AusgabeOBuffer.Einwohner = einwohner;
```

Abbildung 67: Skript zur Generierung von Einwohnertupeln

Im nächsten Schritt werden die Leerzeilen ausgeschieden. Gleichzeitig erfolgt eine Aufteilung nach `Geschlecht` zur Parallelverarbeitung. In einer Skriptkomponente (Abbildung 67) werden die Eingangstupel zerlegt und in einzelne Alterstupel gesplittet. Dazu wird eine Variable mit dem `Alter` angegeben. Dann wird die Anzahl der Einwohner mit dem jeweiligen Alter aus der zugehörigen Spalte ausgelesen. Danach wird die Methode „CreateNewOutputRows()“ aufgerufen. Diese erzeugt ein neues Tupel und beschreibt die Spalten in diesem neuen Tupel mit der Gemeindenummer, dem Herkunftsjahr der Quellstatistik, dem Alter und der Einwohnerzahl zu diesem Alter. Dieser Vorgang wird für jede Altersspalte wiederholt.

Die gesplitteten Tupel werden sortiert und wieder zusammengefügt. Die Parallelverarbeitung schafft die Möglichkeit, männliche und weibliche Einwohnerzahlen in ein gemeinsames Tupel zu bringen. Die gemeinsame Datenmenge wird in eine Rohdatendatei gespeichert, um vom nachstehenden Prozess wieder eingelesen werden zu können.

Dieser Prozess erzeugt aus 455 männlichen und 455 weiblichen Tupeln (je ein Tupel männlich und weiblich für jede Gemeinde in Oberösterreich) und 108 Altersjahrgängen insgesamt 455 x 108 = 49.140 Tupel pro eingelesenem Jahr.

Links oben in der Abbildung 68 zeigt sich die Rohdatendatei mit den Bevölkerungsdaten. Diese wurden im letzten vorgestellten Prozess (Abbildung 65: Einlesen der verschiedenen Bevölkerungstabellen in einer Schleife) bereits vorbereitet. Die Flächenwidmungsdaten werden aus einer Textdatei importiert, da mit der Excel-Quelle Probleme auftreten, wenn in Ziffernspalten Nichtziffernelemente vorkommen. Ein Abspeichern der Quelldatei als CSV-Datei (Comma-Separated-Values) ist im Vorfeld notwendig.

Nach dem Einlesen der `Bevölkerungsdaten` werden alle Leerzeilen mittels Werkzeug „Bedingtes Teilen“ aussortiert. Das Werkzeug „Abgeleitete Spalte“ ersetzt sämtliche Bindestriche durch die Ziffer 0, damit diese im folgenden Werkzeug zu INT konvertiert werden können.

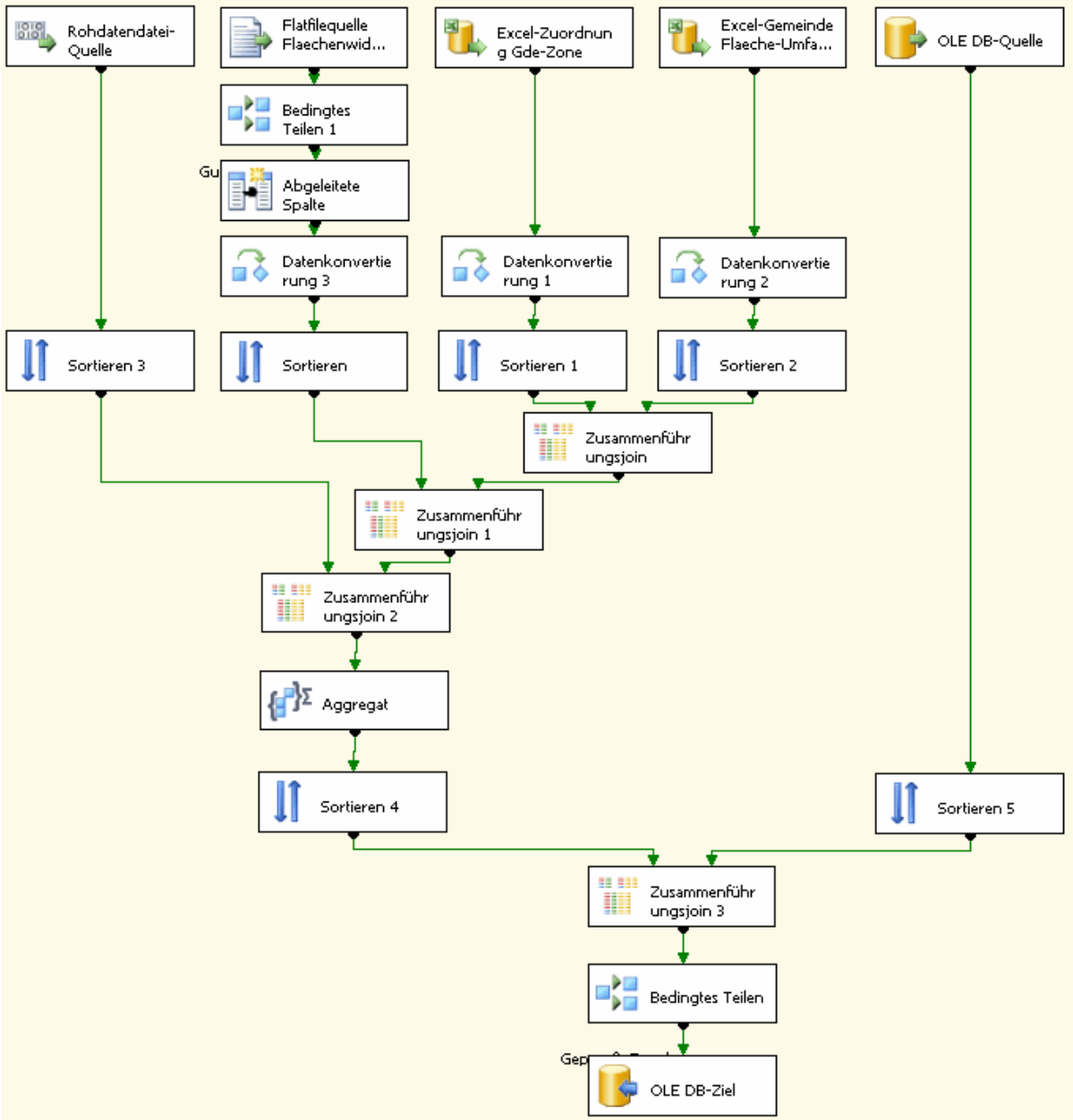

Abbildung 68: ETL-Prozess für die Faktentabelle `Strukturdaten`

Zwei Sortierwerkzeuge bereiten die Daten aus der `Bevölkerungsquelle` und der `Flächenwidmungsquelle` auf eine Zusammenfügungsoperation vor. Die Daten aus der `Zuordnungstabelle` und den `Gemeindedaten` werden ebenfalls durch Konvertieren in den Datentyp INT übergeführt und in Folge sortiert. Durch drei aufeinander folgende Join-Operationen werden alle vier Datenquelle zu einem Datenfluss verdichtet. Die verdichteten Daten werden in weiterer Folge nach `Zonen`, `Zeit` und `Alter` aggregiert.

Ein Vergleich durch Verknüpfung mit den Faktentabellendaten verhindert das mehrfache Einfügen bereits vorhandener Tupel. Diese werden mit dem Werkzeug „Bedingtes Teilen“ entfernt.

Der ETL-Prozess für `Fahrplandaten` wird ebenfalls von vier Quelldatendateien gespeist. In der Datenquelle `FPLAN` befinden sich die Fahrplandaten. Der Aufbau wurde bereits in Ka-

pitel 4.1 ausreichend erläutert. Aus der Quelle `Bitfeld` werden die aktiven Tage eingelesen. Diese Quelle wurde ebenfalls schon in Kapitel 4.1 vorgestellt. Die Quelle `Attribut`, welche in Kapitel 4.1 näher gebracht wurde, stellt die Attributsinformationen zur Verfügung. Eine Excel-Quelle liefert die Zuordnung zwischen `Haltestelle` und `Verkehrszone`.

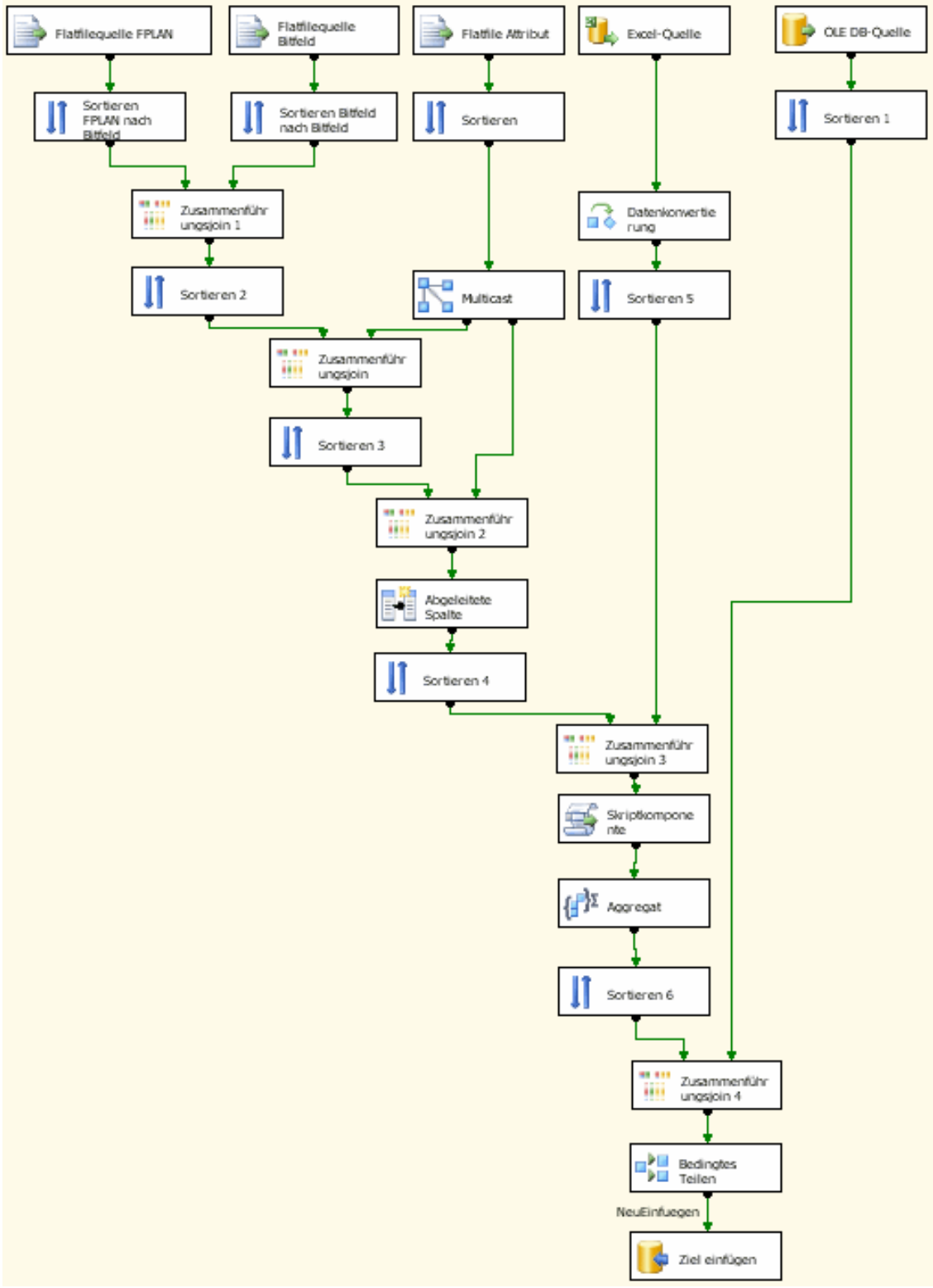

Abbildung 69: ETL-Prozess für die Faktentabelle `Fahrplandaten`

In einem ersten Schritt werden die Bitfelddaten in Abbildung 69 mit den Fahrplandaten verknüpft. Eine weitere Verknüpfung erfolgt für die Tabelle `Attribut`. Die Daten aus der Quelle `Attribut` werden vervielfacht, um für mehrere Verknüpfungsvorgänge zur Verfügung zu stehen. Attribute können bei Busverbindungen mehrfach vorkommen. Nach dem Verknüpfen der Attributswerte werden diese ausgewertet. Dazu wird im Werkzeug „Abgeleitete Spalte“ nach dem Schlüsselwort „Rufbus“ gesucht. Wird das Schlüsselwort gefunden, so wird in eine neue eigene Spalte eine Referenz eingetragen. Mit dem folgenden Sortieralgorithmus werden die Spalten zur Attributsbestimmung aus dem Datenfluss ausgeschieden.

Im nächsten Schritt in Abbildung 69 werden die Zonenzuordnungsnummern eingelesen. Zonenzuordnungsnummern schaffen eine Verbindung zwischen `Haltestellennummer` und `Verkehrszonennummern`.

```
private Int32 minuteCalc(string value)
{
    if (value != null)
        return ((Int32.Parse(value) / 100) * 60) + (Int32.Parse(value) % 100);
    else
        return 0;
}

private void ausgabe(EingabeOBuffer Row, DateTime date)
{
    AusgabeOBuffer.AddRow();
    AusgabeOBuffer.LID = Row.LID;
    AusgabeOBuffer.KSN = Row.KSN;
    AusgabeOBuffer.Zone = Row.KopievonZonenNr;
    AusgabeOBuffer.Gattung = Row.Gattung;
    AusgabeOBuffer.Attribut = Row.Attribut;
    AusgabeOBuffer.Haltestelle = Row.HaltestelleAnkunft;
    AusgabeOBuffer.Verkehrstag = date;
    AusgabeOBuffer.Reisezeit = minuteCalc(Row.AnkunftAnkunft) - minuteCalc(Row.AbfahrtAbfahrt);
 }
```

Abbildung 70: Methoden zur Umrechnung von Zeitangaben in Minuten und zur Ausgabe von Tupeln

Eine Skriptkomponente berechnet aus dem `Bitfeld` und dem Datum `Fahrplanstart` aktuelle Haltestellenzeiten, welche dann als eigene Tupel ausgegeben werden. Abbildung 71 zeigt einen Codeausschnitt der Berechnung. In der ersten Codezeile nach dem Aufruf wird das `Bitfeld` als Character-Array ausgelesen. Diese werden dann in einer For-Schleife abgearbeitet. Dazu wird der ASCII-Wert des Zeichens in eine Zahl ausgelesen, der Wert 48 abgezogen (= 30 HEX – Wert des ASCII Zeichens „0“) und dann verglichen, ob sich die Zahl zwischen 0 und 9 befindet. Ist dies nicht der Fall, so muss das Zeichen aus dem `Bitfeld` zwischen A und F liegen. Daher wird der Zahl noch einmal 7 subtrahiert. 48 und 7 ergeben 55. Der Wert des ASCII-Zeichens „A“ ist 65. Damit ergibt sich eine Differenz von 10. Der ausgelesene Wert für das Zeichen „A“ erhält somit die Zahl 10 beziehungsweise „F“ dann 15.

```
public override void EingabeO_ProcessInputRow(EingabeOBuffer Row)
{
    char[] zeichen = Row.Bitfeld.ToCharArray();
    for (int i = 0; i < zeichen.Length; i++)
    {
        int zahl = zeichen[i];
        zahl -= 48;
        if (zahl > 9)
            zahl -= 7;
          System.Windows.Forms.MessageBox.Show(val.ToString());
        int bit3 = zahl / 8;
        zahl = zahl % 8;
        int bit2 = zahl / 4;
        zahl = zahl % 4;
        int bit1 = zahl / 2;
        int bitO = zahl % 2;
        DateTime datum = Row.StartDatum;

        if (bit3 == 1)
        {
            datum = Row.StartDatum.AddDays(i * 4);
              System.Windows.Forms.MessageBox.Show(datum.ToString());
            this.ausgabe(Row, datum);
        }
        if (bit2 == 1)
        {
            datum = Row.StartDatum.AddDays((i * 4) + 1);
            this.ausgabe(Row, datum);
        }
        if (bit1 == 1)
        {
            datum = Row.StartDatum.AddDays((i * 4) + 2);
            this.ausgabe(Row, datum);
        }
        if (bitO == 1)
        {
            datum = Row.StartDatum.AddDays((i * 4) + 3);
            this.ausgabe(Row, datum);
        }
    }
}
```

Abbildung 71: Codeauszug zur Berechnung des aktuellen Datums aus `Fahrplanstart` und `Bitfeld`

Durch schrittweise Division berechnet die Skriptkomponente in Abbildung 71 anschließend eine duale Darstellung des Bitcodes. Diese Berechnung liefert als Ergebnis das Bitmuster des Zeichens aus dem `Bitfeld` in den Variablen Bit 3, Bit 2, Bit 1 und Bit 0.

Hat nun eines der Bits 0 bis 3 den Wert „1“, so wird dem Datum, welches das Startdatum des Fahrplans beinhaltet, die Anzahl der Tage dazu addiert. Die Anzahl der Tage berechnet sich aus der Anzahl der Schleifendurchläufe, welche gleich ist zur Position des Zeichens im Bitfeld, mal Vier für die Anzahl der Bits pro Zeichen. Für Bit 2 wird noch eine Stelle oder Tag, für Bit 1 zwei Stellen oder Tage und für Bit 3 drei Stellen oder Tage addiert. Nach der Berechnung des Datums wird die Ausgabe aufgerufen.

Die Ausgabe der Tupel wird in Abbildung 70 dargestellt. Die Eingangswerte werden einfach ausgegeben. Für den Verkehrstag wird das berechnete Datum ausgegeben. Die `Reisezeit` errechnet sich aus der Ankunftszeit minus der Abfahrtszeit. Die Zeiten werden mit der Methode „minuteCalc“ in Abbildung 70 oben vom Zeitformat in Minuten umgerechnet.

Eine Ausgabe erfolgt so oft, wie eines der Bits den Wert „1“ hat. Aus einem Bitfeld werden daher genau so viele Tupel generiert, wie der binäre Gegenwert zum Bitfeld den Wert „1“ aufweist.

Die Datenmenge erhöht sich in diesem Transformationszyklus von zirka 430.000 Tupel `Fahrplandaten` auf zirka 64 Millionen Tupel. Nach der Aggregation über die Fremdschlüssel aus der Faktentabelle verbleiben zirka 7,1 Millionen Tupel zum Eintrag in die Faktentabelle. Vor dem Eintrag in die Faktentabelle werden die Tupel anhand der Fremdschlüssel geprüft. Bei gleicher Fremdschlüsselkonfiguration werden die Tupel ausgeschieden.

7.4 Optimierung der ETL-Prozesse

Für ETL-Prozesse ist es wichtig, deren Aufbau performant zu gestalten. Sortieralgorithmen und Aggregationen sind die aufwendigsten Prozesse. Daher wurden für diesen ETL-Prozess zwei Varianten getestet. In der ersten Variante befindet sich die Einbindung der `Zonen` erst nach der Skriptkomponente. Für die Einbindung ist es erforderlich, der Datenfluss zur sortieren. Erst nach Einbindung der Verkehrszonennummern werden die Daten aggregiert.

Für die zweite Variante wurde der Algorithmus so wie in Abbildung 69 gestaltet. Die Einbindung der `Zonennummer` wurde vor die Skriptkomponente gelegt. Dies impliziert zwar mehr Daten über die Skriptkomponente, verlegt aber den Sortieralgorithmus in einen Bereich mit weniger Datenfluss.

Algorithmus	Anzahl Tupel	Laufzeit
Variante 1 ohne Sortierprozess	1.455.768	Abbruch nach 4h 56Minuten
Variante 2 mit Sortierprozess	7.561.173	1h 53Minuten
ETL-Prozess Fahrplan	60.394.668	20h 27Minuten
Sortierprozess für Attribute	438.848	16 Sekunden

Tabelle 13: Gegenüberstellung der Performance von ETL-Prozessen

Wie Tabelle 13 anschaulich zeigt, wird der Prozess wesentlich durch Sortieralgorithmen gebremst. Ist bei einer Datenmenge von zirka 430.000 Tupeln noch keine merkliche Geschwindigkeitseinbuße festzustellen, so steigt die Laufzeit für das Sortieren oder Aggregieren mit der Tupelmenge enorm an.

8 OLAP-Auswertungen im erweiterten Cube

SQL Server Analysis Services (SSAS) bietet einen Browser für Auswertungen am Cube über seine Dimensionen an. Nach Anwählen des Cubes im Projektmappen-Explorer öffnet sich die Ansicht Cubestruktur. Diese Ansicht besitzt oben eine weitere Auswahlleiste. Ganz rechts befindet sich der Auswahlpunkt Browser.

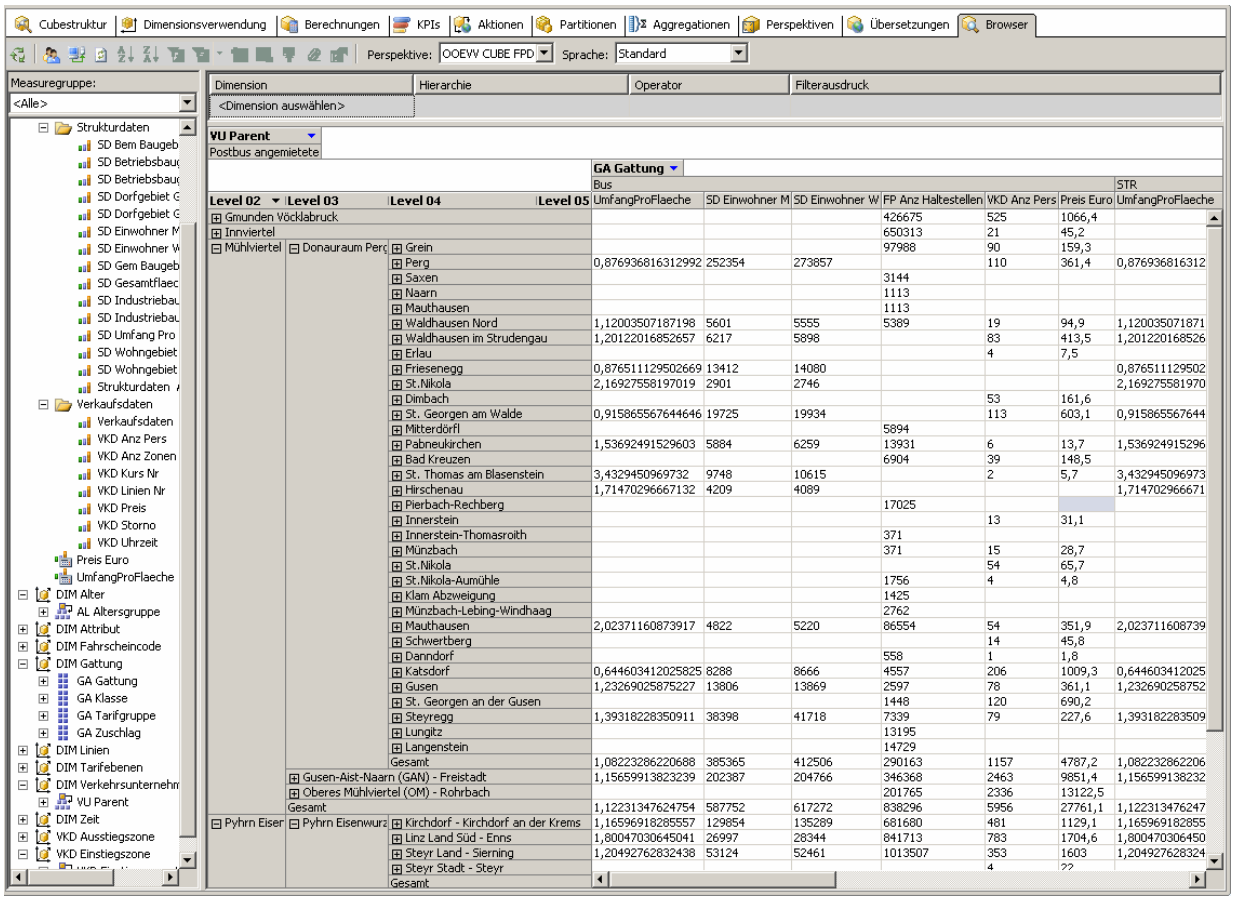

Abbildung 72: Allgemeine Browseransicht im Cube mit Auswertedaten

Links am Bildschirm befindet sich ein Fenster mit den Auswahlmöglichkeiten im Cube (siehe Abbildung 72). Im oberen Bereich finden sich die Measures, gruppiert nach Faktentabellen. Darunter finden sich die Dimensionen.

Mittels Anwahl und Ziehen (Drag and Drop) können die Measures auf die aktive Bildschirmfläche verschoben werden. Die Anordnung und Reihung derselben erfolgt in gleicher Weise.

Aktive Dimensionsbereiche sind links und oben neben der aktiven Auswertungsfläche für die Measures angeordnet. Nach Auswahl einer Dimension kann diese unter Anwahl einer Hierarchiestufe, falls vorhanden, in den aktiven Dimensionsbereich gezogen werden.

In der Werkzeugleiste links oben befinden sich mehrere Tasten zur Aktualisierung der Daten. Sind neue Daten mittels der ETL-Prozesse in die Tabellen geladen worden, so ist es erforderlich, diese in den Cube zu übernehmen. Die Taste links außen (drei grüne Pfeile kreisförmig angeordnet) startet die Generierung des Cubes. Die dritte Taste von links verbindet nach Fertigstellung den Browser neu. Die Daten im Browser werden erst zu diesem Zeitpunkt aktualisiert.

Für den Vergleich der Ergebnisse aus dem Cube werden die Fragestellungen an diese Arbeit aus Kapitel 1.2 beantwortet. Dazu werden die Fragen noch einmal nachstehend aufgelistet.

- Wie verteilen sich die verkauften Fahrscheine auf Regionen/Zonen unter Berücksichtigung der Bevölkerung pro Fläche (spezifischen Bevölkerungsdichte)?
- Wie angemessen ist das Fahrplanangebot bezogen auf die Bevölkerung pro Fläche (spezifischen Bevölkerungsdichte)?

8.1 Auswertung der ersten Frage Aufteilung der verkauften Fahrscheine

Die Cubeansicht in Abbildung 73 ist links in die `Verkehrszonen` mit verschiedenen Gruppierungen untergliedert. Die Hierarchiestufen ergeben sich aus einer Parent-Child-Beziehung der Verkehrszonendaten. Weiters gibt es noch eine Gesamtgliederung nach `Jahr`.

Jahr: (Mehrere Elemente)

Spaltenfelder hierher ziehen

Level 02	Level 03	Level 04	Level 05	Verkaufserfolg	VKD Anz Pers	SD Einwohner W	SD Einwohner M	SD Gesamtfla
⊞ Gmunden Vöcklabruck				1034,45243608979	525	71684	70080	279329,74
⊟ Innviertel	⊞ Schärding - Schärding			1,740838657689	1	16430	16280	56942,832
	⊞ Ried im Innkreis - Ried im Innkreis			32,2683743462715	20	17092	16837	54741,683
	⊞ Braunau - Braunau am Inn					44802	43131	108640,55
	Gesamt			29,933147795621	21	78324	76248	220325,07
⊟ Mühlviertel	⊞ Donauraum Perg (RVDP) - Perg			1703,86242059474	1275	31382	31683	84277,712
	⊟ Gusen-Aist-Naarn (GAN) - F	⊞ Mönchdorf		1,#INF	2			
		⊞ Königswiesen		1,#INF	25			
		⊞ Unterweißenbach		1333,21779054762	266	473	540	5077,2542
		⊞ Kaltenberg		1,#INF	42			
		⊞ Bad Zell		547,627816294923	135	493	697	4827,2377
		⊞ Schönau im Mühlkreis		538,083050537109	144	558	567	4203,7738
		⊞ Tragwein		1,#INF	50			
		⊞ Burbach		1,#INF	8			
		⊞ Weberberg		26,7628672617269	15	1368	1415	4965,4039
		⊞ Pierbach		30,3726827163322	5	168	240	2478,4109
		⊞ Weitersfelden		102,502258352564	20	411	473	4530,5998
		⊞ St. Oswald bei Freistadt		396,807239910652	159	979	935	4776,6607
		⊞ Liebenau		176,195123696849	29	685	685	8323,7006
		⊞ Schöneben		86,2678823590279	14	545	575	6901,4305
		⊞ St. Leonhard bei Freistadt		106,557865267022	36	624	666	3818,3235
		⊞ Gutau Nord		35,1698358374906	16	975	1017	4378,6445
		⊞ Amesreith		1,#INF	28			
		⊞ Kefermarkt		226,405135465923	31	185	290	3469,1109
		⊞ Neustadt-Gutau-Altmühle		147,299693761631	66	1011	1080	4666,7221
		⊞ Harrachstal				435	465	4768,0251
		⊞ Harrachst Waldfeld				466	514	4768,0252
		⊞ Neumarkt Selker		3,4624784396695	2	1041	1078	3668,4959
		⊞ Langfirling				705	751	3818,3235
		⊞ Gutau				1245	1296	4938,9656
		⊞ Freistadt		1299,20385554483	460	295	295	1666,3701
		⊞ Windhaag bei Freistadt		1,#INF	126			
		⊞ Mairspindt		1,#INF	7			
		⊞ Rainbach im Mühlkreis		293,687268972397	122	1080	1152	5373,0326
		⊞ Kerschbaum		23,7018741805714	13	1465	1482	5373,0325
		⊞ Leopoldschlag		1,#INF	71			
		⊞ Summerau		4,2847147900522	2	1224	1284	5373,0323
		⊞ Sandl		1,#INF	74			
		⊞ Oberrauchenödt		8,32751156395948	5	1450	1418	4776,6606
		⊞ Waldburg		85,9027276850024	21	332	356	2814,3369
		⊞ Waldurg-Oberschwandt				594	597	2893,6073
		⊞ Pichling						

Abbildung 73: Browseransicht im Cube mit berechneten Measures für erste Fragestellung

Abbildung 73 zeigt die Measures für die erste Fragestellung. Im Measure `VKD_AnzPers` wird die Anzahl der transportierten Personen nach `Verkehrszonen` gegliedert.

Rechts neben den verkauften Fahrkarten (Anzahl Personen) finden sich die Einwohnerzahlen. In der ersten Spalte rechts werden die Anzahl der weiblichen Bewohner im Feld `SD_EinwohnerW` aufgelistet. Die männlichen Bewohner werden weiter rechts im Feld

SD_EinwohnerM gelistet. Auf den ersten Blick täuschen die Zahlen etwas, wenn man diese mit den Einwohnerzahlen der Orte vergleicht. Die hier angeführten Zahlen sind aber auf Verkehrszonen bezogen. Deshalb kann kein direkter Vergleich mit den Einwohnerdaten der Gemeinden erfolgen.

In der letzten Spalte rechts befindet sich noch die Flächenangabe zur Zone. Auch hier gilt es wieder zu beachten, dass sich diese Angaben auf Verkehrszonen und nicht auf Gemeinden beziehen.

Aus diesen Measures kann das Measure Verkaufserfolg (erste Spalte links in Abbildung 74) berechnet werden. Die erste Frage beschäftigt sich mit der Verkaufsverteilung im Verhältnis zur Bevölkerung pro Fläche. Für eine bessere Auswertung wird die Frage in eine Formel verpackt. Die Werte werden kalkuliert und als berechnetes Measure ausgegeben. Dieser Wert ist einheitenlos und wird als „Verkaufserfolg“ bezeichnet.

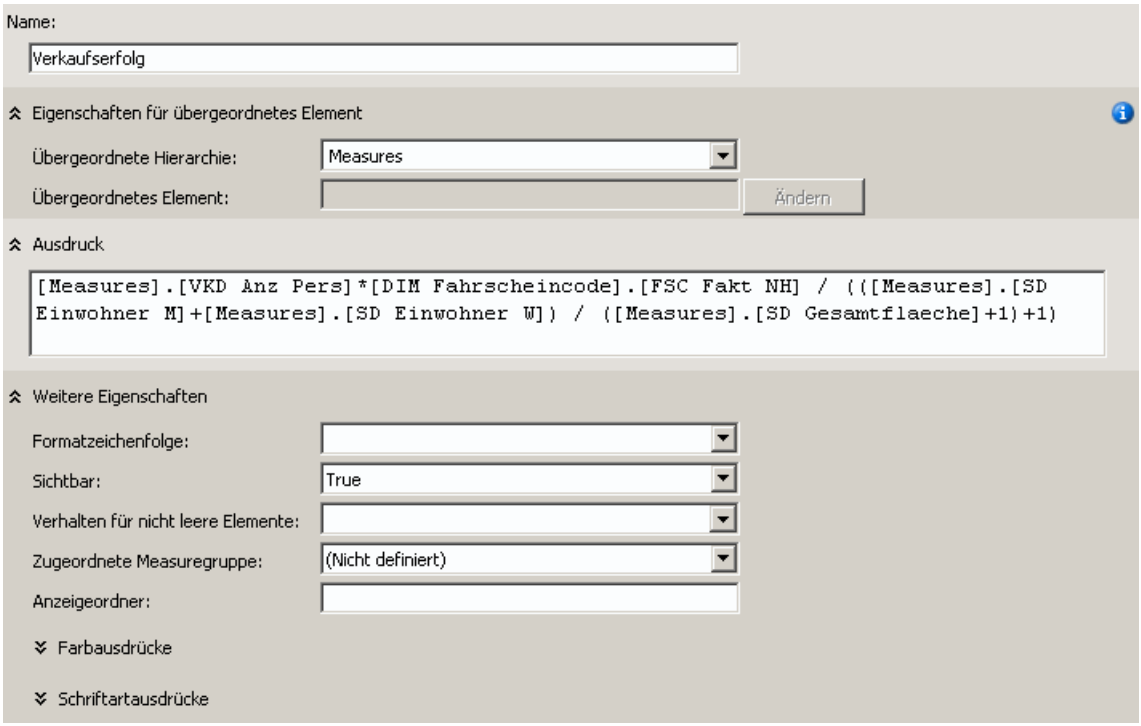

Abbildung 74: Berechnung des Measure Verkaufserfolg für den Cube

Mit steigender Zahl an verkauften Fahrkarten steigt auch der „Verkaufserfolg“. Bremsend wirkt die Anzahl der Bevölkerung pro Fläche. Eine höhere Bevölkerungsdichte lässt den „Verkaufserfolg“ sinken. Abbildung 74 zeigt die Berechnungsformel im Formeleditor des SSAS, welcher unter dem Menüpunkt Berechnung zu finden ist. Der Vorteil dieses berechneten Measures ist, dass die erste Frage mit einer einzigen vergleichbaren Kennzahl zusammengefasst wird. Eine Gegenüberstellung der einzelnen ausgewerteten Kennzahlen ist somit sehr leicht möglich.

Stellt man zum Beispiel die Verkehrszonen St. Oswald bei Freistadt und Liebenau wie in Abbildung 75 gegenüber, so kann man erkennen, dass Liebenau bei einem Fünftel der verkauften Personenanzahl an Karten nur gerade einmal die Hälfte unter dem Verkaufserfolg von St. Oswald liegt.

	Spaltenfelder hierher ziehen				
Level 04 **Level 05**	Verkaufserfolg	WKD Anz Pers	SD Einwohner W	SD Einwohner M	SD Gesamtfla
⊞ St. Oswald bei Freistadt	396,807239910652	159	979	935	4776,6607
⊞ Liebenau	176,195123696849	29	685	685	8323,7006

Abbildung 75: Gegenüberstellung zweier Verkehrszonen zur ersten Frage

Die Gesamtfläche der `Verkehrszone` Liebenau ist doppelt so groß wie jene der `Verkehrszone` St. Oswald, weist jedoch gerade einmal zwei Drittel der `Einwohner` im Vergleich zur `Verkehrszone` St. Oswald auf. Diese beiden Faktoren `Größe` und `Einwohnerzahl` wirken sich dahingehend aus, dass in der `Verkehrszone` St. Oswald ein höherer `Verkaufserfolg` mehr verkaufte Fahrkarten benötigt als in der `Verkehrszone` Liebenau. Ordnet man alle Verkehrszonen absteigend nach dem `Verkaufserfolg`, so weisen besonders niedrige `Verkaufserfolg`-Werte den Analysten auf Verkehrszonen hin, in denen unter Umständen ein Überangebot an öffentlichen Verkehrsmitteln besteht und somit Verbindungen gestrichen werden könnten. Besonders hohe Verkaufserfolge dagegen weisen Zonen mit ausbaufähigem Fahrangebot aus.

8.2 Auswertung der zweiten Frage Fahrplanangebot

Jahr
(Mehrere Elemente)

			Spaltenfelder hierher ziehen				
Level 02	**Level 03**	**Level 04**	Fahrplandichte	FP Anz Haltestellen	SD Einwohner W	SD Einwohner M	SD Gesamtflaeche
⊞ Gmunden Vöcklabruck			760840,751936833	386138	71684	70080	279329,743142538
⊞ Innviertel			232715,970231051	163265	78324	76248	220325,072431654
⊟ Mühlviertel	⊞ Donauraum Perg (RVDP) - Perg		175398,93848273	131251	31382	31683	84277,7125920057
	⊟ Gusen-Aist-Naarn (GAN) - Freist	⊞ Haid bei Königswiesen	1,#INF	460			
		⊞ Mönchdorf	1,#INF	4419			
		⊞ Königswiesen	1,#INF	2435			
		⊞ Unterweißenbach	21141,024964398	4218	473	540	5077,254217386
		⊞ Kaltenberg	1,#INF	992			
		⊞ Bad Zell	35960,8932700332	8865	493	697	4827,2377880811
		⊞ Schönau im Mühlkreis	9169,83198623657	2454	558	567	4203,77383232117
		⊞ Tragwein	1,#INF	4898			
		⊞ Burbach	1,#INF	3321			
		⊞ Weberberg	301,528304482123	169	1368	1415	4965,4039726257
		⊞ Pierbach	22165,9838463793	3649	168	240	2478,4109096527
		⊞ Weitersfelden	7416,03839180804	1447	411	473	4530,5998191833
		⊞ St. Oswald bei Freistadt	11584,7748909764	4642	979	935	4776,6607370376
		⊞ Liebenau	6452,38694365703	1062	685	685	8323,7006711959
		⊞ Schöneben	9581,89693344917	1555	545	575	6901,4305887222
		⊞ St. Leonhard bei Freistadt	4081,7582278673	1379	624	666	3818,3235054016
		⊞ Gutau Nord	2391,54883694936	1088	975	1017	4378,6445617675
		⊞ Amesreith	1,#INF	2082			
		⊞ Kefermarkt	72588,4077869616	9939	185	290	3469,1109466552
		⊞ Trosselsdorf	1,#INF	229			
		⊞ Neustadt-Gutau-Altmühle	5517,04307543564	2472	1011	1080	4666,7221159935
		⊞ Harrachstal	6468,62079112053	1221	435	465	4768,0251531601
		⊞ Harrachst Waldfeld	462,206530026027	95	466	514	4768,0252571106
		⊞ Neumarkt im Mühlkreis	1,#INF	3640			
		⊞ Neumarkt Selker	4199,9863473191	2426	1041	1078	3668,4959068298
		⊞ Langfirling	138,991171556515	53	705	751	3818,3235054016
		⊞ Gutau	3741,64061979814	1925	1245	1296	4938,9656181335
		⊞ Freistadt	70125,9402811363	24829	295	295	1666,3701625466
		⊞ Windhaag bei Freistadt	1,#INF	1005			
		⊞ Mairspindt	1,#INF	490			
		⊞ Rainbach im Mühlkreis	12765,7671095133	5303	1080	1152	5373,0326585769
		⊞ Kerschbaum	1037,4128006727	569	1465	1482	5373,0325546264
		⊞ Leopoldschlag	1,#INF	3531			
		⊞ Summerau	2395,15556763918	1118	1224	1284	5373,0323467254
		⊞ Sandl					

Abbildung 76: Browseransicht im Cube mit berechneten Measures für zweite Fragestellung

Die Cubeansicht in Abbildung 76 ist links wieder in die `Verkehrszonen` mit verschiedenen Gruppierungen untergliedert. Die Hierarchiestufen ergeben sich aus einer Parent-Child-Beziehung der Verkehrszonendaten. Weiters gibt es eine Gesamtgliederung nach `Jahr`.

In Abbildung 76 werden nun die Measures zur Beantwortung der zweiten Fragestellung gelistet. Die Unterteilung der Verkehrszonen zeigt eine Einteilung der Anzahl an Busstopps in der Spalte `FP_Anz_Haltestellen`. Die Spalte `FP_Anz_Haltestellen` beinhaltet sämtliche Busstopps für diese Zone innerhalb eines Jahres.

Rechts neben der Spalte `FP_Anz_Haltestellen` finden sich die Einwohnerzahlen. In der ersten Spalte rechts werden die Anzahl der weiblichen Bewohner im Feld `SD_EinwohnerW` aufgelistet. Die männlichen Bewohner werden weiter rechts im Feld `SD_EinwohnerM` aufgelistet. Auf den ersten Blick täuschen die Zahlen etwas, wenn man diese mit den Einwohnerzahlen der Orte vergleicht. Die hier angeführten Zahlen sind aber auf `Verkehrszonen` bezogen. Deshalb kann kein direkter Vergleich mit den Einwohnerdaten der Gemeinden erfolgen. In der letzten Spalte rechts befindet sich noch die `Flächenangabe` zur Zone. Auch hier gilt es wieder zu beachten, dass sich diese Angaben auf `Verkehrszonen` und nicht auf Gemeinden beziehen.

Für die Fahrplandaten wurde ein ähnliches berechnetes Measure „Fahrplandichte“ definiert (siehe Abbildung 77). Die Berechnung für das Measure `Fahrplandichte` (erste Spalte links in Abbildung 76) ist mit diesen Werten nun möglich. Gegenüber der obigen Berechnung werden die Anzahl der Fahrgäste durch die Anzahl der Busstopps ersetzt. Damit ergibt sich bei einer höheren Anzahl an Busstopps ein höherer Wert. Mit sinkender Bevölkerungsdichte steigt ebenfalls der Wert „`Fahrplandichte`“. Abbildung 76 veranschaulicht die Browsersicht des Cubes mit den eingetragenen berechneten Measures.

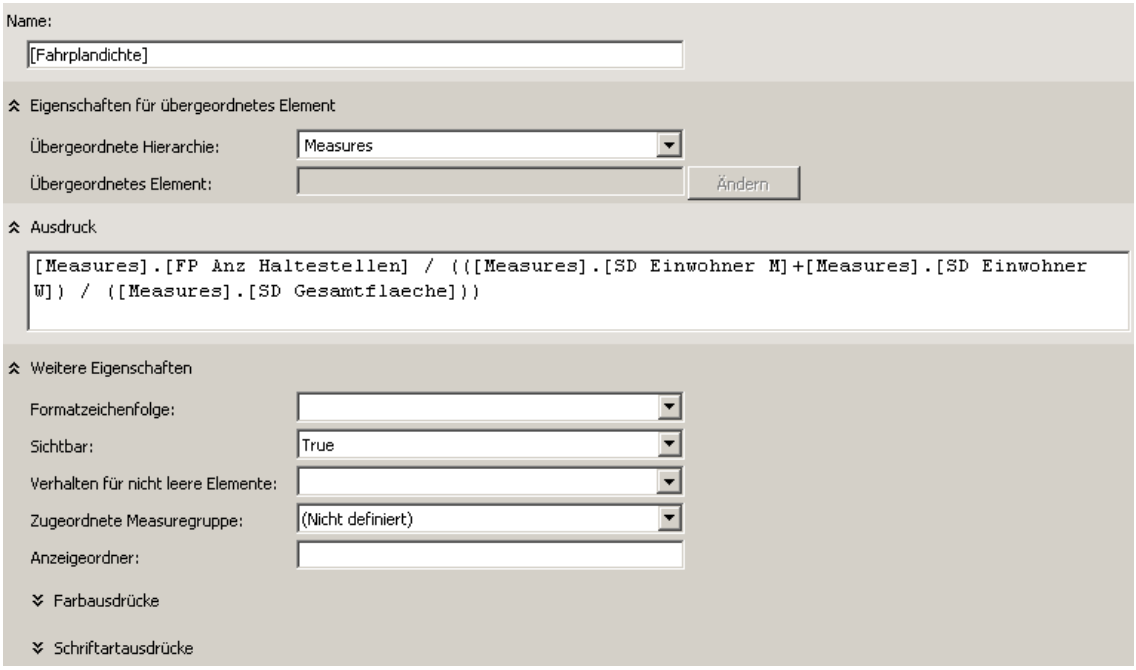

Abbildung 77: Berechnung des Measure Fahrplandichte für den Cube

Stellt man nun wieder die Verkehrszonen St. Oswald bei Freistadt und Liebenau wie in Abbildung 78 gegenüber, so kann man erkennen, dass St. Oswald die doppelte „Fahrplandichte“ gegenüber Liebenau aufweist.

	Spaltenfelder hierher ziehen				
Level 04	Fahrplandichte	FP Anz Haltestellen	SD Einwohner W	SD Einwohner M	SD Gesamtflaeche
⊞ St. Oswald bei Freistadt	11584,7748909764	4642	979	935	4776,66073703766
⊞ Liebenau	6452,38694365703	1062	685	685	8323,70067119598

Abbildung 78: Gegenüberstellung zweier Verkehrszonen zur zweiten Frage

Wie in der Auswertung der ersten Frage schon festgestellt, ist die Verkehrszone Liebenau flächenmäßig etwa doppelt so groß wie die Verkehrszone St. Oswald. Bei den Einwohnerzahlen hingegen weist die Verkehrszone Liebenau gerade einmal zwei Drittel von Verkehrszone St. Oswald auf. Der Unterschied zwischen der Anzahl Busstopps in St. Oswald mit 4.642 und Liebenau mit 1.062 beträgt in etwa das Vierfache. Durch die Berechnung der Fahrplandichte nach der Formel in Abbildung 77 ergibt sich eine Differenz zwischen St. Oswald und Liebenau von weniger als der Hälfte. Weniger Personen auf mehr Fläche rechtfertigen eine geringere Fahrplandichte. Eine gleichmäßige Fahrplandichte über die Verkehrszonen ergäbe sich dann, wenn alle Faktoren der Fahrplandichten gleich groß wären.

Die neu ins Data Warehouse eingebundenen demographischen und geographischen Daten ermöglichen somit, eine besser auf spezifische demographische Gegebenheiten in den lokalen Gemeinden abgestimmte Bewertung der Fahrplandichte zu berechnen. Gerade in ländlichen Gebieten mit vergleichsweise dünner Bevölkerungsdichte stellen öffentliche Verkehrsverbindungen ein wichtiges Fortbewegungsmittel dar. Deshalb wird die Bevölkerungsdichte als Korrekturfaktor in der Kennzahl Fahrplandichte verwendet (Abbildung 77).

Mit Hilfe der hier gezeigten Kennzahlen Verkaufserfolg und Fahrplandichte ist es den Verantwortlichen beim OÖVV möglich, das Angebot an öffentlichen Verkehrsmitteln besser auf die Nachfrage abzustimmen. Dabei werden insbesondere regional spezifische Besonderheiten berücksichtigt, was eine Entscheidungsfindung „mit Fingerspitzengefühl“ ermöglicht (das heißt, Einsparungspotenzial beim Fahrplanangebot zu finden, ohne die Bedürfnisse ländlicher Gemeinden zu sehr zu vernachlässigen).

9 Fazit

In dieser Arbeit wurde aus den Grundlagen ein Data Warehouse für die Verwendung mit Fahrplandaten, Verkaufsdaten und Gemeindedaten entwickelt.

Das in dieser Arbeit entwickelte Datenmodell für ein Data Warehouse kann für verschiedene Personentransportunternehmen eingesetzt werden, welche ein Verbindungsnetz mit Zonenmodell betreuen. Der Cube bietet die Möglichkeit, weitere Werte zu berechnen. Dazu können unter dem Menüpunkt Berechnungen sämtliche Measures und Dimensionsattribute verknüpft, berechnet oder abgefragt werden. Die Grundlagen für weitere Berechnungen sind geschaffen und damit auch eine Einsatzmöglichkeit bei verschiedenen Verkehrsunternehmen.

Adaptierungen ergeben sich an den Datenquellen. Für diese ist es erforderlich, an das bestehende innere Datenschema angepasst zu werden. Um der Wiederverwendbarkeit Genüge zu tun, wurden alle ETL-Prozesse so weit wie möglich mit den in Kapitel 2.2.2 vorgestellten Werkzeugen umgesetzt. Damit sind eine leichte Lesbarkeit und eine übersichtliche Struktur geschaffen.

In der Eingangsphase wurde die Fragestellung erhoben. Ein Theorieblock erläuterte die Grundlagen des Data Warehousing. Dazu wurden Grundbegriffe des Data Warehousing definiert, die Grundlagen und Eigenschaften des ETL-Prozesses erklärt sowie Vorgehensmodelle für die Entwicklung eines Data Warehouse aufgezeigt.

Eine Analyse der Quelldaten leitete zu Beginn des praktischen Teiles in die Beschreibung der Istsituation ein. Aus diesen Quellen heraus wurden nach dem Vorgehensmodell von (Golfarelli & Rizzi, 1998) mehrere Faktenschemata entwickelt. Diese Faktenschemata wurden weiters bereinigt und zusammengeführt. Eine theoretische Ableitung zu SQL-Statements nach (Lorentz, 2005) schloss die Entwicklung der Faktenschemata ab.

Im nächsten Schritt wurde das Data Warehouse auf dem SQL-Server von Microsoft entwickelt und implementiert. Die ETL-Prozesse für dieses Data Warehouse wurden im Anschluss daran entwickelt und optimiert. Den Abschluss bereitete ein Testlauf mit einem Datenausschnitt aus Echtzeitdaten zur Überprüfung der Funktionalität.

Ein Data Warehouse zur Analyse von Fahrverkaufsdaten, Fahrplandaten und Strukturdaten ist geschaffen. Für die weitere Verwendung der Information in Berichten oder Data-Mining-Anwendungen sind in weiterer Folge Strukturen zu generieren.

10 Quellenverzeichnis

Abello, A., & Romero, O. (07 2010). A framework for multidimensional design of data warehouses. *Data & Knowledge Engineering , Volume 69* (Issue 11), S. 1138-1157.

Ali El-Sappagh, S., Ahmed Hendawi, A., & Ali Hamed El Bastawissy, D. (05 2011). A Proposed Model for Data Warehouse ETL-Processes. *Journal of King Saud University - Computer and Information Sciences* , S. 1-23.

Amt der oö. Landesregierung Abteilung Geoinformation und Liegenschaft. (2010). *Digitales oberöstereichisches Rauminformationssystem*. Abgerufen am 21. 05 2011 von DORIS: http://doris.ooe.gv.at/viewer/%28S%28edpnjg55xfbdrd55r24nns55%29%29/init.aspx?ks=alk&karte=adr&sichtbar=%ufffd.+Grundkarte+Speed&unsichtbar=Ortsplan+Speed

Armstrong, R. (2001). Seven steps to optimizing data warehouse performance. *Computer , 34* (12), S. 76-79.

Astor, E., & Bussian, R. (1997). *Die Geographie* (3. Auflage Ausg.). Wien: Dudenverlag.

Azevedo, P., Brosius, G., Dehnert, S., Neumann, B., & Scheerer, B. (2009). *Business Intelligence und Reporting mit Microsoft SQL Server 2008.* Unterschleißheim: Microsoft Press.

Bauer, A., & Günzel, H. (2009). *Data Warehouse Systeme - Architektur Entwicklung und Anwendung* (3. Ausg.). Heidelberg: dpunkt.verlag.

Bergamaschi, S., Guerra, F., Orsini, M., Sartori, C., & Vincini, M. (03 2011). A semantic approach to ETL technologies. *Data & Knowledge Engineering* .

Bimonte, S., Tchounikine, A., & Miquel, M. (2005). Towards a Spatial Multidimensional Model. *DOLAP 05* (S. 39-46). Bremen: ACM.

Bodendorf, F. (2006). Datenmanagement. In F. Bodendorf, *Daten- und Wissensmanagement* (S. 7-67). Berlin Heidelberg: Springer.

Brodie, M., & Schmidt, J. (07 1982). Final report of the ANSI/X3/SPARC DBS-SG relational database task group. *SIGMOD Rec. , Volume 12* (Issue 4), S. 1-62.

Buzydlowski, J., Song, I., & Hassell, L. (1998). A framework for object-oriented on-line analytic processing. *Proceedings of the 1st ACM international workshop on Data warehousing and OLAP* , S. 10-15.

Church, R. (2002). Geographical information systems andlocation science. *Computers & Operations Research* (29), S. 541-562.

Corral, K., Schuff, D., & Louis, R. (10 2006). The impact of alternative diagrams on the accuracy of recall: A comparison of star-schema diagrams and entity-relationship diagrams. *Decision Support Systems , Volume 42* (Issue 1), S. 450-468.

Da Silva, J., De Oliveira, A., Fidalgo, R., Salgado, A., & Times, V. (2007). Modelling and querying geographical data warehouses. *Information Systems , 35* (5), S. 592-614.

Datawarehouse4u.Info. (2009). *Fact constellation schema*. Abgerufen am 08. 05 2011 von http://datawarehouse4u.info/Data-warehouse-schema-architecture-fact-constellation-schema.html

Demirel, H. (07 2004). A dynamic multi-dimensional conceptual data model for transportation applications. (sciencedirect, Hrsg.) *ISPRS Journal of Photogrammetry and Remote Sensing , Volume 58* (Issues 5-6), S. 301-314.

Dori, D., Feldman, R., & Sturm, A. (09 2008). From conceptual models to schemata: An object-process-based data warehouse construction method. *Information Systems , Volume 33* (Issue 6), S. 567-593.

Galindo-Legaria, C., Grabs, T., Gukal, S., Herbert, S., Surna, A., Wang, S., et al. (2008). Optimizing Star Join Queries for Data Warehousing in Microsoft SQL Server. *24th International Conference on Data Engineering* (S. 1190-1199). IEEE.

Goeken, M. (2006). *Entwicklung von Data-Warehouse-Systemen* (1. Ausg.). Wiesbaden: Deutscher Universitätsverlag.

Golfarelli, M., & Rizzi, S. (1998). A methodological framework for data warehouse design. (ACM, Hrsg.) *Proceedings of the 1st ACM international workshop on Data warehousing and OLAP* , S. 3-9.

Guo, F., & Song, H. (2010). Research and application of data-mining technique in timetable scheduling. *2nd International Conference on Computer Engineering and Technology. 1*, S. 409-412. ICCET.

HaCon. (2004). *Dokumentation HAFAS Rohdatenformat - Eingabedateien der Datenaufbereitung.* internes Dokument.

Hake, G., & Grünreich, D. (1994). *Kartographie.* Berlin: Walter de Gruyter.

Harinath, S., Zare, R., Meenakshisundaram, S., Carroll, M., & Lee, D. (2009). *Professional Microsoft SQL Server Analysis Services 2008 with MDX* . Indianapolis: Wiley Publishing.

Herring, J. (2010). *OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option* . Open Geospatial Consortium.

Humm, B., & Wietek, F. (2005). Architektur von Data Wahouse und Business Intelligence Systemen. *Informatik Spektrum* , 3-14.

Huynh, N., Mangisengi, O., & Tjoa, A. (2000). Metadata for object-relational data warehouse. In *Design and Management of Data Warehouses.*

Inmon, W. (11 1996). The data warehouse and data mining. (ACM, Hrsg.) *Commun. ACM , Volume 39* (Issue 11), S. 49-50.

Inmon, W., & Kelley, C. (1993). *Rdb - VMS: Developing a Data Warehouse.* New York: John Wiley & Sons, Inc.

Kemper, H., Mehenna, W., & Unger, C. (2006). *Business Intelligence - Grundlagen und praktische Anwendung.* Wiesbaden: Vieweg & Sohn.

Kimball, R. (1996). *The data warehouse toolkit: practical techniques for building dimensional data warehouses.* New York: John Wiley & Sons, Inc.

Knight, B., Veerman, E., Dickinson, G., Hinson, D., & Herbold, D. (2008). *Professional Microsoft SQL Server 2008 Integration Services.* Indianapolis: Wiley Publishing.

Konopasek, K. (2010). *SQL Server 2008 R2 - Der schnelle Einstieg.* München: Addison-Wesley.

Lechtenbörger, J., & Vossen, G. (2003). Multidimensional normal forms for data warehouse design. (sciencedirect, Hrsg.) *Information Systems , Volume 28* (Issue 5), S. 415-434.

Leitner, P. (2008). *Datenqualitätsanalyse beim Ladevorgang in Data Warehouses am Beispiel der ETL-Prozesse der OÖGKK.* Linz.

Leser, H., Haas, H., Mosimann, T., & Paesler, R. (1997). *Wörterbuch Allgemeine Geographie.* München: Duetscher Taschenbuch Verlag.

Levene, M., & Loizou, G. (05 2003). Why is the snowflake schema a good data warehouse design? (siencedirect, Hrsg.) *Information Systems , Volume 28* (Issue 3), S. 225-240.

List, B., Bruckner, R., Machaczek, K., & Schiefer, J. (2002). A Comparison of Data Warehouse Development Methodologies Case Study of the Process Warehouse. *Database and Expert Systems Applications - 13th International Conference*, (S. 203-215). Aix-en-Provence, France.

Lorentz, D. (2005). *Oracle Database SQL Reference, 10g Release 2 (10.2).*

Marotta, A., & Ruggia, R. (2002). Data warehouse design: a schema-transformation approach. *22nd International Conference of the Chilean* (S. 153-161). Computer Science Society.

Moody, D., & Kortink, M. (2000). From enterprise models to dimensional models: a methodology for data warehouse and data mart design. *2nd International Workshop on Design and Management of Data Warehouses*, (S. 5.1-5.12). Stockholm.

Object Modeling Group. (2005). *Object Constraint Language Specification*. Abgerufen am 19. 05 2011 von OMG - OCL: http://www.omg.org/spec/OCL/

Oracle. (2003). *The Multidimensional Data Model*. Abgerufen am 08. 05 2011 von Diagram of the Logical Multidimensional Model: http://www.stanford.edu/dept/itss/docs/oracle/10g/olap.101/b10333/multimodel.htm

Pendse, N., & Creeth, R. (1995). The OLAP Report. *Business Intelligence* .

Powell, G. (2006). The basics of data warehouse data Modeling. In G. Powell, *Oracle Data Warehouse Tuning for 10g* (S. 3-29). Burlington: Digital Press.

Prat, N., Akoka, J., & Comyn-Wattiau, I. (12 2006). A UML-based data warehouse design method. (sciencedirect, Hrsg.) *Decision Support Systems , Volume 42* (Issue 3), S. 1449-1473.

Ravat, F., & Teste, O. (2000). A Temporal Object-Oriented Data Warehouse Model. *Database and Expert Systems Applications* , S. 583--592.

Ravat, F., Teste, O., Tournier, R., & Zurfluh, G. (2009). Finding an application-appropriate model for XML data warehouses. In F. Ravat, O. Teste, R. Tournier, & G. Zurfluh, *Information Systems* (S. 662-687). Toulouse: Elsevier B.V.

Reuven Bakalash, S., Guy Shaked, B., & Joseph Caspi, H. (07. 05 2002). Relational database management system having integrated non-relational multi-dimensional data store of aggregated data elements. *United States Patent US 6,385,604* .

Rifaie, M., Blas, E., M.A., M., Mok, T., Kianmehr, K., Alhajj, R., et al. (2008). Data Warehouse Architecture for GIS Applications. *Proceedings of iiWAS* (S. 178-185). Linz: ACM.

Rifaie, M., Kianmehr, K., Alhajj, R., & Ridley, M. J. (2008). Data warehouse architecture and design. *IEEE International Conference on Information Reuse and Integration* (S. 58-63). IRI.

Rivest, S., Bedard, Y., Proulx, M., Nadeau, M., Hubert, F., & Pastor, J. (2005). SOLAP technology: Merging business intelligence with geospatial technology for interactive spatio-temporal exploration and analysis of data. *ISPRS Journal of Photogrammetry & Remote Sensing* , S. 17-33.

Srivastava, J., & Chen, P.-Y. (1999). Warehouse creation-a potential roadblock to data warehousing. *IEEE Transactions on Knowledge and Data Engineering , 11* (1), S. 118-126.

Statistikamt, O. (2009). *Flächenwidmung OÖ Gemeinden 2009.* Linz: Statistikabteilung Land Oberösterreich.

Theodoratos, D., & Sellis, T. (11 1999). Designing data warehouses. (sciencedirect, Hrsg.) *Data & Knowledge Engineering , Volume 31* (Issue 3), S. 279-301.

Vassiliadis, P., Quix, C., Vassiliou, Y., & Jarke, M. (05 2001). Data warehouse process management. (sciencedirect, Hrsg.) *Information Systems , Volume 26* (Issue 3), S. 205-236.

Verkehrsverbund, O. (2011). *OOE-Verkehrsverbund - Wir über uns.* Abgerufen am 09. 05 2011 von http://www.ooevg.at/index.php?id=112

Verkehrsverbund, O. (2011). *OOE-Verkehrsverbund - Zonenplan*. Abgerufen am 09. 05 2011 von http://www.ooevg.at/index.php?id=12

Verkehrsverbund, O. (2011). Tarifbestimmungen für den Oberösterreichischen Verkehrsverbund, gültig ab 1.1.2011. In OÖ-Verkehrsverbund, *Fahrpreistafel des OÖVV, gültig ab 1.1.2011* (S. 25-37). Linz.

Vivekanand, G., L., a. Q., & Kamalakar, K. (1999). Star/Snow-Flake Schema Driven Object-Relational Data Warehouse - Design and Query Processing Strategies. *Data Warehousing and Knowledge Discovery* , S. 11-22.

Vonhoegen, H. (2006). *Einstieg in XML* (3. Auflage Ausg.). Bonn: Galileo Press.

W3C, W. W. (2010). *Service Description W3C*. Abgerufen am 16. 01 2011 von http://www.w3.org/standards/webofservices/description

W3C, W. W. (2010). *Web of Service*. Abgerufen am 16. 01 2011 von http://www.w3.org/standards/webofservices/

Wang, J., Pang, M., Zhang, H., & Zhang, D. (2010). Application of Data Warehouse technique in intelligent vehicle monitoring system. *Proceedings of the Ninth International Conference on Machine Learning and Cybernetics*, (S. 1007-1010). Qingdao.

Wikipedia. (01. 04 2011). Abgerufen am 05. 05 2011 von Data Warehouse: http://de.wikipedia.org/wiki/Data-Warehouse#Geschichte_des_Begriffs

Wikipedia. (27. 06 2008). *Snowflake-schema-example*. Abgerufen am 08. 05 2011 von http://en.wikipedia.org/wiki/File:Snowflake-schema-example.png

Wikipedia. (28. 06 2008). *Star-Schema example*. Abgerufen am 08. 05 2011 von http://en.wikipedia.org/wiki/File:Star-schema-example.png

Wu, T. (2010). ETL Function Realization of Data Warehouse System Based on SSIS Platform. *2nd International Workshop on Database Technology and Applications* (S. 1-4). DBTA.

Zeh, T. (08 2003). Data Warehousing als Organisationskonzept des Datenmanagements. *Informatik – Forschung und Entwicklung , Band 18* (Heft 1), S. 32-38.

Zentralmelderegister. (2002-2009). *Bevölkerung in den oö. Gemeinden und Bezirken nach Geschlecht, zusammen und Alter (Einzeljahre und 5-Jahres-Gruppen).* Abteilung Statistik Land Oberösterreich.

Zhu, F., Zhang, Y., & Zhao, L. (09 2008). Ship Scheduling Decision Support System Based on Data Warehouse. *Proceedings of the IEEE International Conference on Automation and Logistics* , S. 2000-2003.