BABENDERERDE INGENIEURE GmbH

FACH HOCHSCHULE LÜBECK
University of Applied Sciences

DIPLOMARBEIT

Frank Stahl

Entwicklung eines Data Warehouse Systems für Vortriebsdaten im Tunnelbau

August 2005

Fachbereich Elektrotechnik
Dekanat
Prüfungsausschuss

FACH
HOCHSCHULE
LÜBECK
University of Applied Sciences

Thema der Diplomarbeit

für

Herrn Frank S t a h l

Entwicklung eines Data Warehouse Systems für Vortriebsdaten im Tunnelbau

Professor Dr.-Ing. Hinrichs

Ausgabedatum: 19. Mai 2005
Abgabedatum: 19. August 2005

(Professor Dr.-Ing. Hoeck)
Vorsitzender des Prüfungsausschusses

Aufgabenstellung:

Während des Tunnelvortriebs wird üblicherweise eine große Anzahl von Messwerten aufgezeichnet. Hier zu nennen sind z. B. Maschinendaten, Setzungsmessung, Stützdruck, Verpressdruck usw. Diese Messgrößen liegen in verschiedenen Datenformaten (z.B. Excel, dBase) vor, welche dann aufbereitet und zu Analysezwecken in einer SQL- Datenbank abgelegt werden müssen. Dieser Prozess soll mit Hilfe eines Softwaresystems automatisiert und benutzerfreundlich gestaltet werden. Musskriterien der Software sind Plattformunabhängigkeit, Sicherung der Datenintegrität, Mehrsprachigkeit sowie die ausschließliche Verwendung kostenfrei nutzbarer Module.

Professor Dr.-Ing. Hinrichs

Fachbereich Elektrotechnik
Dekanat
Prüfungsausschuss

FACH
HOCHSCHULE
LÜBECK
University of Applied Sciences

Erklärung zur Diplomarbeit

Ich versichere, dass ich diese Arbeit selbstständig, ohne fremde Hilfe verfasst habe.

Bei der Abfassung der Arbeit sind nur die angegebenen Quellen benutzt worden. Wörtlich oder dem Sinn nach entnommene Stellen sind als solche gekennzeichnet.

~~Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, insbesondere dass die Arbeit Dritten zur Einsichtnahme vorgelegt oder Kopien der Arbeit zur Weitergabe an Dritte angefertigt werden.~~

Lübeck, den 18. August 2005

Frank Stahl

Fachbereich Elektrotechnik
Dekanat
Prüfungsausschuss

FACH
HOCHSCHULE
LÜBECK
University of Applied Sciences

Zusammenfassung der Diplomarbeit

Fachbereich: Elektrotechnik

Studiengang: KIM-Informatik

Thema: Entwicklung eines Data Warehouse Systems für Vortriebsdaten im Tunnelbau

Zusammenfassung: Design und verfahrenstechnische Entscheidungen im Tunnelbau resultierten meist aus Datenanalysen von anderen Tunnelprojekten sowie aus Erfahrungen, die in den vorangegangenen Vortrieben gesammelt wurden. Die daraus resultierenden großen Datenmengen bergen ein ungenutztes Potential, das Baufirmen, Bauherren und beratenden Ingenieuren wichtige Informationen über Vortriebsprozesse geben kann.

Ein Data-Warehouse-System erlaubt die flexible und effiziente Auswertung aus heterogenen Datenquellen und ist somit prädestiniert für diese Aufgabe.

Die vorliegende Arbeit verfolgt daher das Ziel, die Grundlagen und Eigenschaften eines Data-Warehouse-Systems im Kontext des Tunnelbaus herauszustellen und anhand des Systems „Tunneling Process Control“ kurz TPC den vollständigen Entwurf und dessen Implementierung zu zeigen.

Abschließend kann gesagt werden, dass Data-Warehouse-Systeme sich durchaus auch im technischen Umfeld etablieren können.

Verfasser: Frank Stahl

Betreuender Professor/in: Prof. Dr.-Ing. Holger Hinrichs

WS / SS: SS2005

Danksagung

Die vorliegende Diplomarbeit ist im Rahmen meiner Tätigkeit als Mitarbeiter der Firma Babendererde Ingenieure GmbH entstanden. An dieser Stelle möchte ich bei denjenigen Bedanken, die zum Gelingen dieser Arbeit beigetragen haben.
Ich bedanke mich ausdrücklich bei Herrn Prof. H. Hinrichs für sein Interesse an dieser Arbeit und seine wertvolle Unterstützung während des gesamten Entscheidungsprozesses. Herrn Prof. Schiffer danke ich für die Übernahme der Zweitbegutachtung.

Bei meinem Chef, Herrn Lars Babendererde, bedanke ich mich für die Freiräume zum wissenschaftlichen Arbeiten, bei meinen Arbeitskollegen für den konstruktiven Ideenaustausch und die nette Arbeitsatmosphäre.
Ein weiterer Dank geht an die tapferen Korrekturleser Babara und Ernst Egel und Kristin Scholz.

Schließlich hat auch mein privates Umfeld zum Fortschritt dieser Arbeit beigetragen. Insbesondere möchte ich mich bei meiner Freundin Kristin Scholz, meinen Eltern sowie Großeltern für ihr Verständnis und ihre Unterstützung bedanken.

Vielen Dank

Frank Stahl

Inhaltsverzeichnis

Seite

Teil I Grundlagen

1 Einleitung

In zunehmendem Maße werden im Untertagebau für den Ausbruch von Tunneln und Stollen Tunnelbohrmaschinen (TBM) eingesetzt. Hierbei ist eine Überwachung der Vortriebsarbeiten und somit eine Überwachung der Qualität des endgültigen Bauwerks nur eingeschränkt möglich. So kann während des Vortriebes die Ortsbrust nicht ständig eingesehen werden. Auch hinter der TBM kann die Geologie nicht angesprochen werden, da im Schutz der TBM bereits die Tunnelauskleidung erstellt werden muss. Die für die Ausbauqualität wichtige Ringspaltverfüllung hinter den Tübbingen kann auch nicht direkt inspiziert werden. Daher beruht die Bauüberwachung verfahrensbedingt auf der Sammelung von zur Verfügung stehenden Informationen und auf deren Interpretation. Bei modernen TBM's, die heute zum Einsatz kommen, fallen sehr viele Vortriebsdaten an (ca. alle 2 – 10 Sekunden), wie z. B. die Drehzahl des Schneidrades, der Hydraulikdruck der einzelnen Vortriebszylinder und die Position der TBM. Die Abbildung 1-1 zeigt einen Blick in den Leitstand, in dem alle Informationen zusammenlaufen.

Abbildung 1-1 Blick in den Leitstand einer modernen TBM

Die Erfassung und Auswertung dieser Vortriebsdaten beschränkt sich auf herkömmliche *OLTP-Systeme* (OLTP = Online Transcation Processing), deren einzelnen Komponenten sich aus den Daten des Vermessungssystems, TBM Daten und Messwerte von Oberflächensetzungen beschränken. Diese arbeiten unabhängig voneinander und sind zumeist nicht miteinander kompatibel. Eine bereichsübergreifende Auswertung aller Vortriebsdaten ist somit unmöglich. Weiterhin ist das Bewusstsein der Vorteile einer kompletten und kontinuierlichen Datenerfassung nicht bei allen Beteiligten gleichermaßen vorhanden. Häufig ist auf der Seite der Bauherren die Bereitschaft zu Investitionen in diesem Bereich vor einem Projekt nicht vorhanden. Die großen Vorteile werden nur unzureichend erkannt, z. B. die Verringerung von laufenden Kosten, das Optimieren von gefährlichen Arbeitsabläufen oder die Bearbeitung von Nachträgen. Datenerfassung und -management sind bei der Ausschreibung wie auch im späteren Vertrag meist nicht geregelt, was einen Austausch der vorhandenen Daten zusätzlich erschwert.

1.1 Aufbau der Arbeit

Die Arbeit gliedert sich in vier Teile (plus Anhang). Teil I enthält neben dieser Einleitung die Grundlagen der relevanten Themenbereiche Data-Warehouse und maschineller Tunnelbau. Teil II befasst sich mit der konzeptionellen und logischen Modellierung eines DWS für Vortriebsdaten und deren softwaretechnischen Umsetzung. Kapitel III stellt die Testergebnisse des in Teil III beschriebenen Softwaresystems am Beispiel des Nodo Bologna (Italien) Projekts dar. Eine Zusammenfassung und ein Ausblick runden die Arbeit in Teil IV ab.

1.2 Zielsetzung der Arbeit

Das Ziel der vorliegenden Arbeit ist es, ein Werkzeug für den Bauherrn und seine beratenden Ingenieurbüros zu entwickeln, das eine projektspezifische und interdisziplinäre Datenerfassung und Auswertung ermöglicht. Dazu ist ein Data-Warehouse basierendes System zu entwickeln, das alle bei einem Vortrieb anfallenden Daten in Echtzeit zusammenführt und nach Bedarf auswertet. Die Datenmenge umfasst dabei nicht nur die Positions- und Betriebsdaten der Maschine, sondern auch Messwerte für Oberflächensetzungen oder Extensiometerverformungen und ähnliches. Die Darstellung der Daten hat in leicht verständlichen Diagrammen, basierend auf der Zeit, der Stationierung oder der Ringnummer, zu erfolgen. Die Auswertung kann kontinuierlich, täglich oder wöchentlich erfolgen.

Im Einzelnen sollen mit dieser Arbeit folgende Ziele erreicht werden:

- Herausstellen der Eigenschaften eines Data-Warehouse-System
- Entwicklung eines DWH für Vortriebsdaten
- Entwicklung geeigneter Methoden zur Datenintegration ins Data-Warehouse-System
- Konzipierung und prototypische Implementierung eines Softwaresystem unter dem Namen „Tunneling Process Control“ kurz TPC, unter der Berücksichtigung der zuvor entwickelten Ansätze
- Testen des implementierten Softwaresystems anhand des Bologna Projektes

2 TBM-Vortrieb

Im Laufe der Industrialisierung zu Beginn des 19. Jahrhundert setzte mit dem Ausbau des Eisenbahnnetzes eine rapide Entwicklung im Tunnelbau ein. Der Vortrieb konnte nur im standfesten Gebirge durch Sprengung erfolgen. Der zunehmende Druck, den Tunnel möglichst schnell aufzufahren, zwang die Baufirmen, nach neuen Ideen zu suchen. Die erste einsetzende Mechanisierung erfolgte durch die Bohrung der Sprenglöcher. Parallel dazu wurden Versuche unternommen, bei denen Maschinen komplett das Gestein im Gebirge lösen.

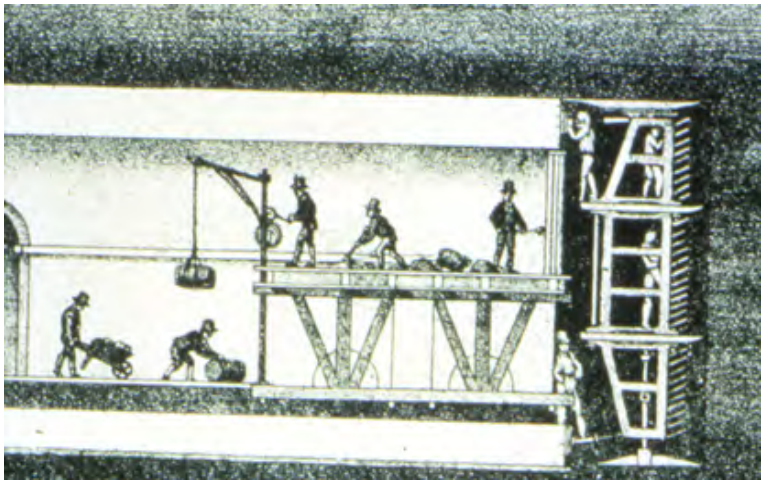


Abbildung 2-1 Tunnelbau im 19. Jahrhundert

Diese scheitern zumeist an den unterschiedlichsten Problemstellungen. Zum einen wurden die materialtechnologischen Grenzen der zur Verfügung stehenden Werkstoffe nicht beachtet und zum anderen war das aufzufahrende Gebirge für eine TBM nicht geeignet. Erfolgreich waren die Einsätze dort, wo das Gebirge die Voraussetzung für einen TBM-Vortrieb bot.

Die erste erfolgreiche Tunnelbohrmaschine war keine TBM im eigentlichen Sinne, da der Abbau nicht von der gesamten Ortsbrust, sondern von Schaufelwerkzeugen erledigt wurde.

Bereits 1851 wurde von Chares Wilson eine Tunnelbohrmaschine entwickelt, die alle Eigenschaften einer modernen TBM vereint.

2.1 Definition

Als eine TBM (Tunnelbohrmaschine) wird allgemein eine Maschine bezeichnet, mit dessen Hilfe ein Tunnel aufgefahren werden kann. Maidl bezeichnet eine Tunnelbohrmaschine (TBM) als eine Vortriebsmaschine für den Tunnel- und Stollenbau die für Fest- oder Losegestein, und die aus einem kreisförmigen Vollschnittbohrkopf, allgemein besetzt mit Rollmeißeln besteht.

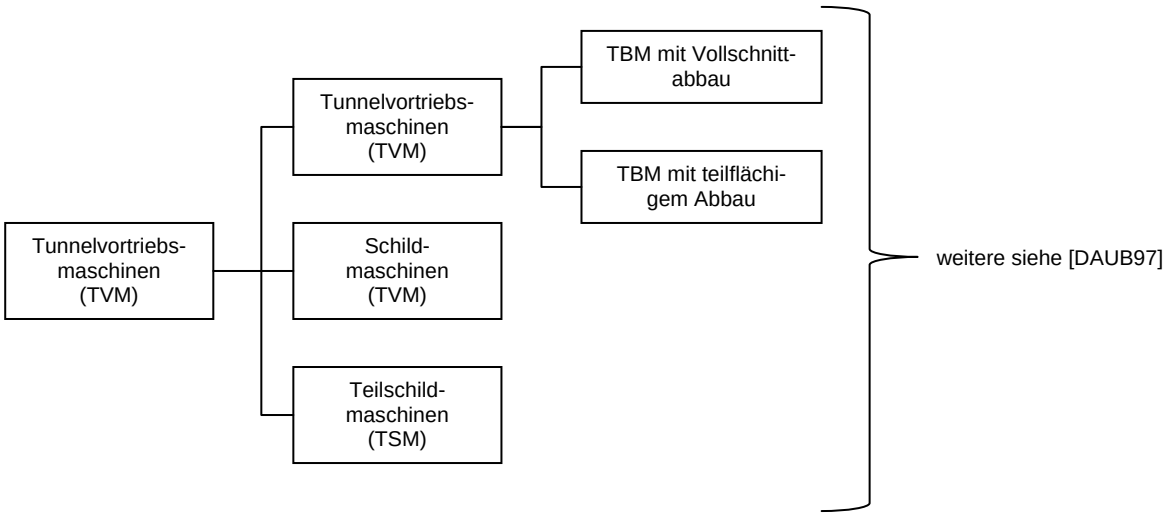


Abbildung 2-2 Auszug Tunnelvortriebsmaschinen nach [DAUB97]

Im Folgenden werden die wesentlichen Vor- und Nachteile eines Vortriebs gegenüber einem konventionellen Vortrieb nach [MSRH01] zusammengefasst:

Vorteile:

- Weit höhere Vortriebsleistungen möglich
- Profilgenauer Ausbruch
- Automatisierter und kontinuierlicher Arbeitsablauf
- Geringerer Personalaufwand
- Höhere Arbeitsbedingungen und Sicherheit
- Mechanisierung und Automatisierung

Nachteile:

- Bessere geologische Aufschlüsse und Informationen als beim Sprengvortrieb notwendig
- Hohe Investitionskosten, deshalb längere Tunnelstrecken erforderlich
- Größere Vorlaufzeit für Konstruktion und Bau der Maschine
- Kreisförmiges Ausbruchprofil

- Einschränkungen bei Kurvenradien und Aufweitungen
- Detaillierte Planung notwendig
- Anpassung an unterschiedlichste Gebirgsarten und hohen Wasserandrang nur begrenzt möglich
- Transport der Maschine mit Nachläufer zum Tunnelportal

2.2 Grundprinzipien und Aufbau einer TBM

Die Abbildung 2-3 zeigt den schematischen Aufbau einer TBM. Im Folgenden werden die einzelnen Elemente vorgestellt und ihre Aufgaben erörtert.

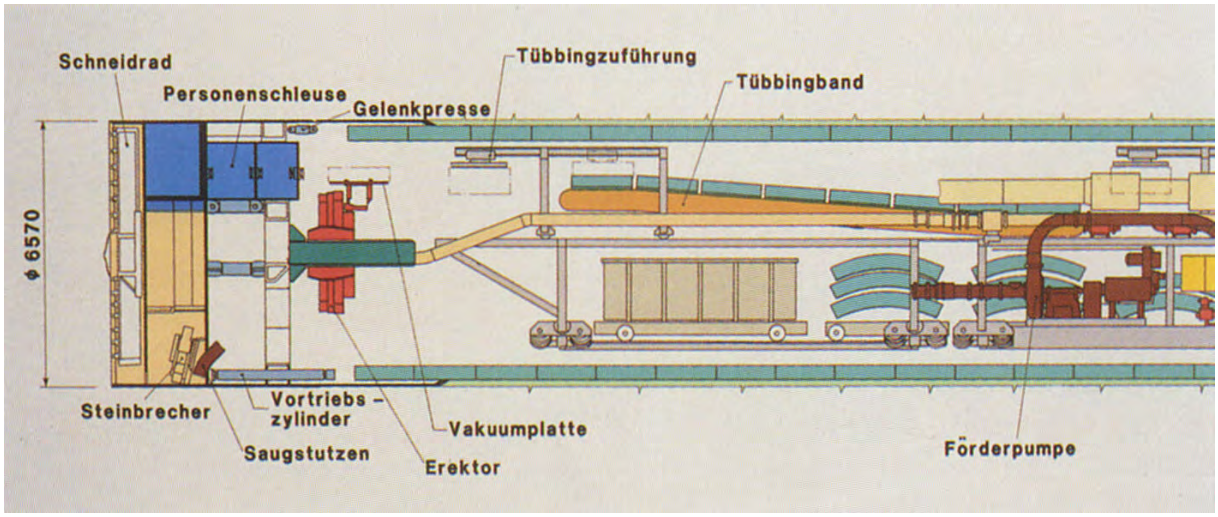


Abbildung 2-3 Schematischer Aufbau einer TBM

Der Schildvortrieb (engl. Shield driving) ist eines der möglichen Verfahren, um Tunnel aufzufahren.

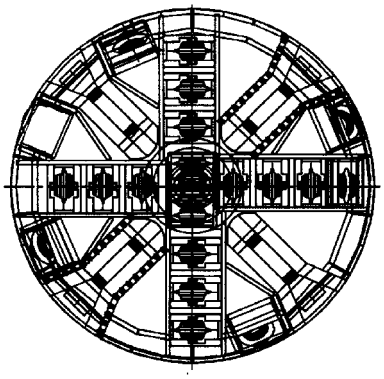


Abbildung 2-4 Schematische Darstellung eines TBM-Schildes

Beim Schild siehe Abbildung 2-4 handelt es sich um rundes Schneidwerkzeug, das meist mit Hartmetall-zähnen (Meißel) ausgestattet ist. Das Schild wird von einer Maschine, rotierend an der Tunnelbrust, in das Bergmaterial vorgeschoben. Über einzelne Aussparungen kann das Abbaumaterial hinter das Schild fallen und wird über eine Förderschnecke zum nächsten Muldengurtförderband abtransportiert. Der Abbau von Böden im Grundwasser oder Böden mit geringer Standfestigkeit kann nur mit geschlossenen Schilden erfolgen.

Abbildung 2-5 Herrenknecht EPB-TBM

Diese Schilde sind mit einem Trennschott zwischen der Abbaukammer und dem Schildschwanz ausgerüstet. Um Einbrüche an der Ortsbrust zu verhindern, muss mit einem ausreichend hohem Stützdruck in der Abbaukammer gefahren werden.

An dieser Stelle wird nur auf EPB Vortriebe eingegangen, so wie sie im Projekt Bologna, Italien vorkommen. *Earth Pressure Balance (EPB)* Vortriebe stützen entweder durch Druckluft oder einem Erdbrei und verhindern somit das Einbrechen der Geländeoberfläche. Im Idealfall ist dabei die Abbaukammer vollständig gefüllt. Bei einer vollständig gefüllten Abbaukammer entsteht eine hohe Reibung und somit muss ein höheres Drehmoment aufgebraucht werden, um das Schneidrad zu dre-

hen. Um diesem Problem entgegenzuwirken, werden durch die Auslässe, die sich im Schneidrad befinden, eine Suspension in das Abbaumaterial indiziert, um damit die Reibung zu minimieren. Dadurch entsteht ein gut konditionierter Erdbrei mit rund 70 % Feststoffanteil. Dieser ermöglicht eine ausreichende Stützung und zugleich wird die Reibung soweit minimiert, dass ein zügiger Vortrieb realisiert werden kann. Da die Tunneldecke nur in den wenigsten Fällen selbststützend ist, wird eine stützende Verkleidung benötigt, welche durch Tübbinge bewerkstelligt wird.

Als Tübbing werden Bauteile bezeichnet, die die Außenhülle eines Tunnels bilden. Ein Tübbing (siehe Abbildung 2-6) ist dabei ein vorgefertigtes Betonsegment. In der gebräuchlichsten Form bilden sechs Segmente plus ein Abschlusssegment einen vollständigen Ring. Der Tunnel setzt sich dann aus einer Vielzahl von Ringen zusammen.

Abbildung 2-6 Lagerung von Tübbingen

Der Vortrieb bezeichnet den eigentlichen Vorgang zum Auffahren eines Tunnels. Hierbei drückt sich die TBM durch 6 Zylinderpressen vom letzten gebauten Ring ca. 1,5m ab. Dies entspricht genau einer Ringbreite. Hat die TBM genau die Breite eines Tübbings erreicht, so wird der Vortrieb gestoppt und im Schutz des Schildschwanzes der Ring gebaut. Da der Durchmesser einer TBM größer ist, als der eines Ringes, muss das fehlende Volumen durch eine nachträgliche Verpressung eines Mörtels aufgefüllt werden. Diese Indizierung des fehlenden Volumens wird durch 6 Auslässe, die sich im Schildschwanz befinden, realisiert.

2.3 Beispielprojekt

Alle in dieser Arbeit angesprochenen Beispiele, bezüglich eines Vortriebs, beziehen sich auf das folgende Projekt.

- *Projektgebiet:* Bologna, Italien
- *Auftraggeber:* Italferr S. p. A.
- *Auftragnehmer:* Arbeitsgemeinschaft San Ruffillo S. c. r. l., bestehend aus Necso Entrecanales Cubiertas S. A., Salini Costruttori S. p. A. und Ghella S. p. A.
- *Bauwerk:* Bau von zwei Eisenbahntunneln mit einem TBM Außendurchmesser von 9,34m und einer Tunnellänge von je 6,2km für die Hochgeschwindigkeitsstrecke Milano – Napoli.
- *Bauzeit:* 2003 bis ca. 2006
- *Geologie:* Die beiden Tunnel durchörtern verschiedene Abschnitte aus Ton, Kiesen und Sanden oberhalb und unterhalb des Grundwassers, wobei auch mit Steinen und Blöcken zu rechnen ist.
- *Ortsbruststützung:* Aktive Stützung durch EPB-TBM. Je nach vorgefundenen Bodenverhältnissen aktive Stützung mit Erdbrei oder passive im offenen Modus.
- *Auftrag BI:* Beratung des Bauherren während des Vortriebs durch regelmäßige Baustellenbesuche und damit einhergehender Beurteilung des Baufortschritts.

  Weitere Unterstützung des Bauherren durch Stützdruckberechnungen, Setzungsabschätzungen, Entwicklung von Methoden zur besseren Volumenkontrolle u. ä. .

Abbildung 2-7 Lageplan mit Trass, Start- und Zielschacht

3 Data-Warehouse-Systeme

Anfangs der 80er Jahre wurde das Konzept eines unternehmensweit einheitlichen Datenbestandes erstmals erwähnt. Ziel der damaligen Projekte war die intelligente und kreative Nutzung von unternehmensweit verfügbaren Datenbeständen. Doch schon Ende der 60er Jahre wurde an der Modellierung und Verarbeitung von Daten in mehrdimensionalen Matrizen gearbeitet.

In den frühen 70er Jahren kam die Firma MDS mit ihrem Produkt Express auf den Markt und läutete damit das Zeitalter der *Management Decision Systems* ein. Express war ein Softwaretool, in dem sowohl Analysefunktionen, als auch Datenmanagement integriert waren. Hauptsächlich wurde Express zur Durchführung von Marktanalysen benutzt.

Die Schlagwörter „Data Supermarket“ und „Super Database“ kamen mit Beginn der 80er Jahre auf. Später stellte IBM ein internes Projekt unter dem Namen „European Business System“ vor, das später in „Information Warehouse Strategie“ umbenannt wurde. Hierbei ging es um die Bewältigung von Heterogenität und Informationsexplosion.

Solche Systeme sollen garantieren, dass die Entscheidungsfunktion innerhalb der Unternehmung effizienter und schneller wahrgenommen werden kann.

3.1 Definition

Die folgenden Begriffe stammen aus dem Informationsmanagement in der Betriebswirtschaft. Derzeit gibt es keine einheitliche Definition, da von der Anwenderseite die betriebswirtschaftlichen Anforderungen, sowie die tägliche Praxis und auf der technischen Seite die Grundlagen der Datenbanksysteme stehen.

»Als ein Data-Warehouse-System (DWS) wird ein Informationssystem bezeichnet, das aus allen für den Data-Warehouse-Prozess notwendigen Komponenten besteht.« [BaGu02]

»Zentrale Komponente eines DWS ist eine physische Datenbank, die das eigentliche Data-Warehouse (DWH) darstellt.« [Hinr02]

Die Definitionen eines *Data-Warehouse (DWH)* unterscheiden sich vor allem im generellen Zweck eines Data-Warehouse, im Umfang und Umgang der Daten. Inmon prägte als einer der Ersten den Begriff des DWH.

»A data-warehouse is a subject-oriented, integrated, time-variant, non-volatile collection of data in support of management's decision-making process. « [Inmo02]

Die folgende Definition von Bauer und Günzel ist weniger begrenzend, ist aber auf einen speziellen Zweck, die Analysefunktion, ausgerichtet:

»Ein Data-Warehouse ist eine physische Datenbank, die eine integrierte Sicht auf beliebige Daten zu Analysezwecken ermöglicht.« [BaGu04]

Das Spektrum der Definitionen endet bei der Definition von Zeh, die ohne Restriktionen an Umfang und Umgang mit den Daten sowie ohne Zweckbestimmung ist:

»Data-Warehouse ist ein physischer Datenbestand, der eine integrierte Sicht auf die zugrunde liegenden Datenquellen ermöglicht.« [Zeh03].

Der *Data-Warehouse-Prozess* (engl. *data warehousing*) beschreibt den vollständigen Vorgang zum Bewirtschaften und Auswerten eines DWH. Im Einzelnen sind das folgende Punkte:

- *Datenbeschaffung*: Extraktion der relevanten Daten aus den Quellsystemen, Transformation und gegebenenfalls Bereinigung der Daten in einem Arbeitsbereich (englisch *staging area*) sowie Laden in das DWH.
- *Datenhaltung*: Langfristige Speicherung der Daten im DWH
- *Datenauswertung*: Analyse der Daten im jeweiligen Data-Mart

3.2 Abgrenzungen von transaktionalen Systemen

Anfragen

Anfragen	transaktional	analytisch
Fokus	Lesen, Schreiben, Modifizieren, Löschen	Lesen, periodisches Hinzufügen
Transaktionsdauer und –typ	kurze Lese-/Schreibtransaktionen	Lange Lesetransaktionen
Anfragestruktur	einfach strukturiert	komplex
Datenvolumen einer Anfrage	wenige Datensätze	viele Datensätze
Datenmodell	anfrageflexibles Datenmodell	Analysebezogenes Datenmodell

Tabelle 3-1 Gegenüberstellung der Anfragecharakteristika von transaktionalen und analytischen Anwendungen nach [BaHo04]

Daten

Daten	Transaktional	analytisch
Datenquellen	meist eine	mehrere
Eigenschaften	nicht abgeleitet, zeitaktuell	Abgeleitet, konsolidiert, historisiert, integriert, stabil
Datenvolumen	Megabyte – Gigabyte	Gigabyte – Terabyte
Zugriffe	Einzeltupelzugriff	Bereichsanfragen

Tabelle 3-2 Gegenüberstellung der Datencharakteristika von transaktionalen und analytischen Anwendungen nach [BaHo04]

Anwender

Anwender	transaktional	analytisch
Anwenderzahl	Ein-/Ausgabe durch Sachbearbeiter	Auswertung durch Manager, Controller, Analysten
Antwortzeit	ms – min	s – min

Tabelle 3-3 Gegenüberstellung der Anwendercharakteristika von transaktionalen und analytischen Anwendungen nach [BaHo04]

3.3 Anforderungen an ein DWS

Aus den aufgezeigten Unterschieden zu herkömmlichen Transaktionssystemen ergeben sich somit als wichtigste Anforderungen an ein DWH [CCS93]:

- *Multidimensionale Datenhaltung:*
  Die Sicht eines Ingenieurs auf die Vortriebsdaten ist multidimensional. Um eine einfache und intuitive Analyse der Vortriebsdaten zu ermöglichen, sollten die Daten ebenfalls in Dimensionen angeordnet sein.
- *Konstant schnelle Antwortzeiten:*
  Alle Analysen sollten mit einer annähernd gleichen, möglichst hohen Geschwindigkeit, vom System durchgeführt werden, um dem Benutzer optimale Bedingungen zu bieten.
- *Verwaltung großer Datenmengen:*
  Da in einem DWH historische Daten, auch aus länger zurückliegenden Zeitabschnitten, in einer speicherintensiven Struktur gespeichert werden, muss das DBMS diese Datenmenge verwalten können.
- *Flexible Abfrage großer Datenmengen:*
  Da bei strategischen Analysen oftmals größere Zeiträume über mehrere Dimensionen hinweg flexibel abgefragt werden, muss das Datenmodell diesbezüglich optimiert sein.

3.4 Referenzarchitektur

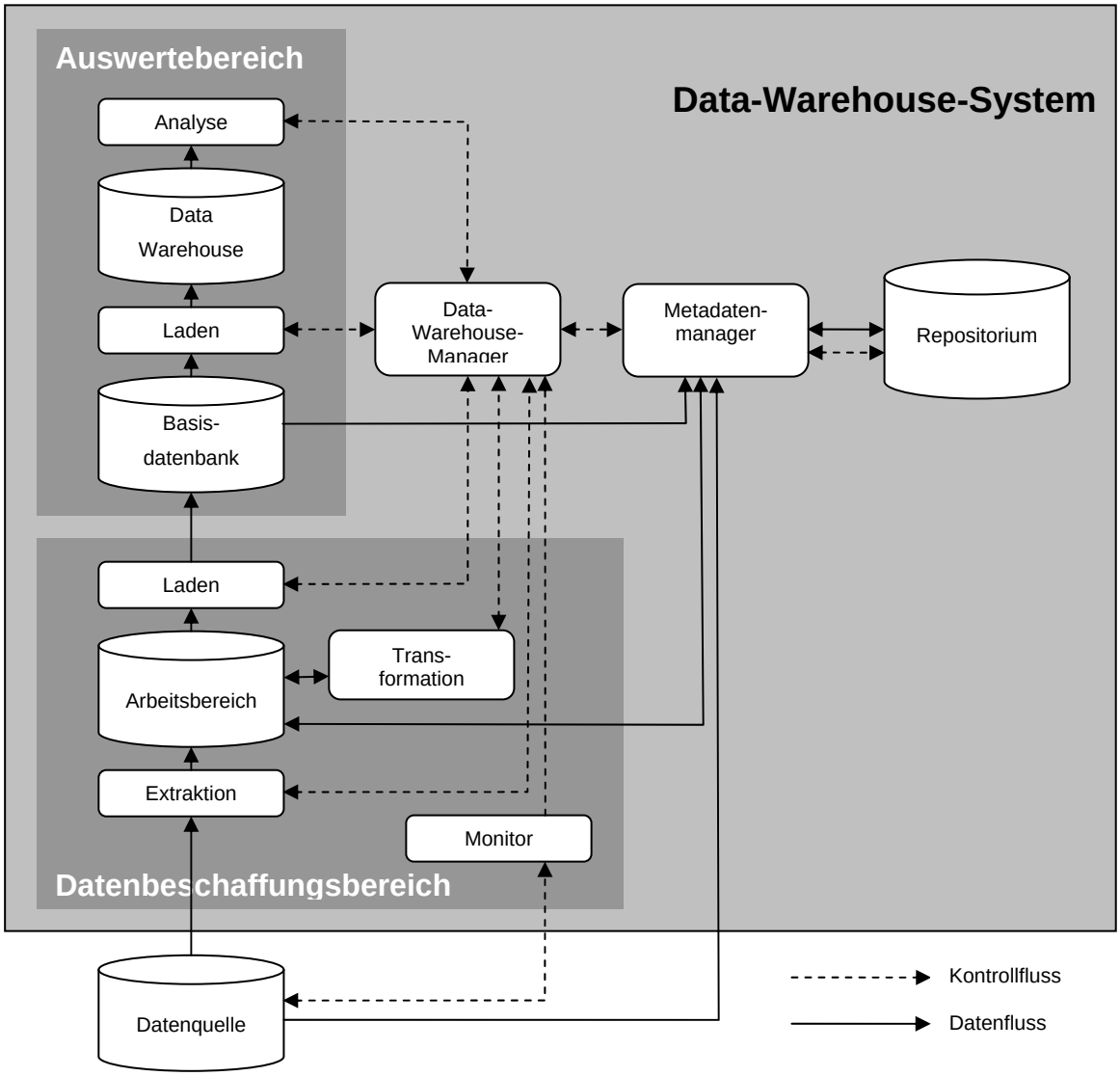


Abbildung 3-1 Referenzmodell für die Architektur von Data-Warehouse-Systemen [BaHo04]

In Abbildung 3-1 wird eine idealisierte DWS-Architektur mit den dazugehörigen Komponenten nach Bauer und Günzel gezeigt. Ergänzend zur Architektur wird in dieser Darstellung das DWS in die Bereiche *Datenakquisition, Datenbank, Datenauswertung und Datenquellen* aufgeteilt.

3.4.1 Datenquellen

Als Datenquellen werden die zu integrierenden, meist heterogenen realen Datenquellen bezeichnet. Die Datenquelle ist kein Bestandteil des DWS, sondern stellt den Ausgangspunkt des Datenflusses beim Data-Warehousing dar [Hinr02].
Die eigentlichen Quelldaten liegen dabei in den unterschiedlichsten Formaten vor:

- Datenbanken, meist Relational

- Semistrukturiert, z. B. XML
- Proprietäre Dateiformate
- Unstrukturierte Informationen, z. B. Text

Die Datenquellen spielen im Data-Warehouse-Prozess eine große Rolle, da sich ihre Beschaffenheit unmittelbar auf die Analyseergebnisse auswirkt. Die Auswahl von geeigneten Datenquellen ist somit von besonderer Bedeutung. Im Einzelnen sollten die folgenden Faktoren nach Bauer und Günzel berückschichtig werden, die zu einer Auswahl der Datenquellen führen:

- Der Zweck des Data-Warehouse
- Die Qualität der Quelldaten
- Die Verfügbarkeit (rechtlich, sozial, organisatorisch, technisch)
- Der Preis für den Erwerb der Quelldaten

Zweck des Data-Warehouse-Systems

Da sich die Auswahl der relevanten Datenquellen direkt aus den Verwendungszweck des DWS ableitet, müssen sich die Datenquellen natürlich für den Verwendungszweck eignen.

Qualität der Quelldaten

An die Beschaffenheit der Quelldaten werden vor allem von den Nutzern und Betreibern, aber auch von den Planern und Erstellern eines DWS bestimmte qualitative Anforderungen gestellt. Hinrichs unterteilt sie im Kontext des DWH in folgende Qualitätsmängel:

- Inkorrekte Daten, verursacht durch Eingabe-, Mess- oder Verarbeitungsfehler
- Logisch widersprüchliche Daten
- Unvollständige, ungenaue bzw. zu grobe Daten
- Duplikate im Datenbestand
- Uneinheitlich repräsentierte Daten
- Veraltete Daten
- Für den Verwendungszweck irrelevante Daten
- Unverständliche Daten, bedingt durch qualitative mangelhafte Metadaten

Die Abbildung 3-2 verdeutlicht an einigen Beispielen die unzureichende Qualität von Quelldaten.

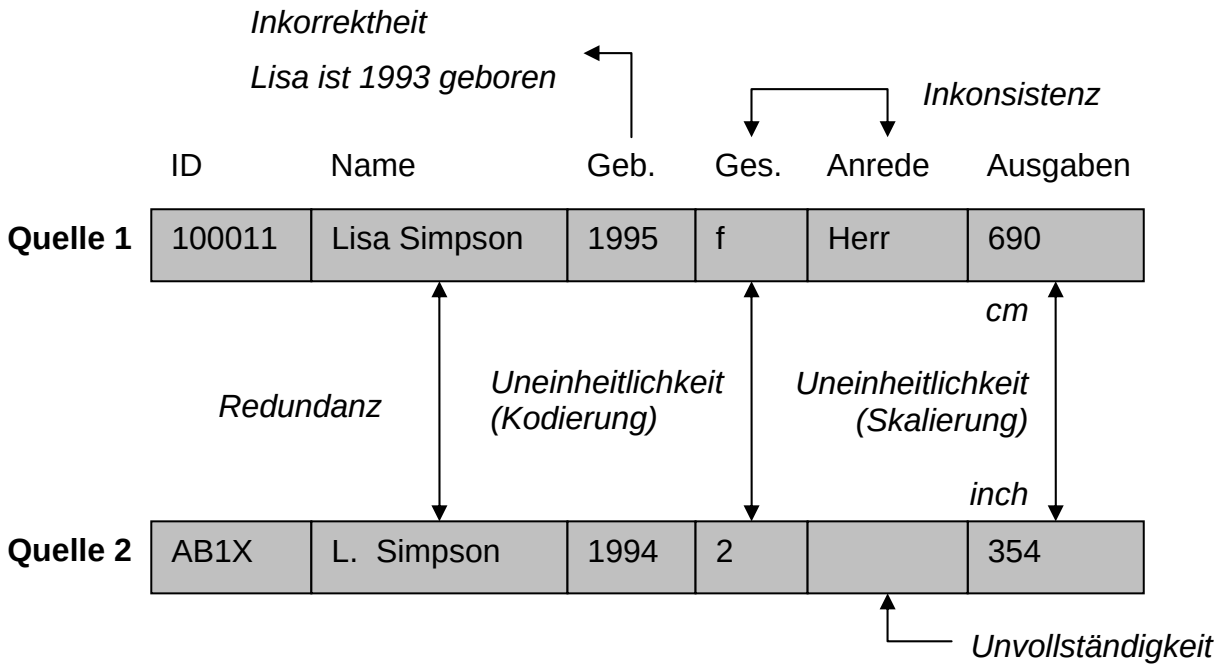


Abbildung 3-2 Beispiele für Datenqualitätsmängel nach [Hinr02]

Laut den Studien von der Meta Group [Meta99] und der Standish Group [Stan99] kann eine mangelhafte Qualität der Quelldaten zu erheblichen Zusatzkosten bis hin zum Scheitern des DWH-Projekts führen.

Die Folgen mangelhafter Datenqualität lassen sich zusammenfassend mit dem Leitsatz *»Garbage in, garbage out«* charakterisieren [Hinr02].

Verfügbarkeit der Daten

Auch wenn alle Datenquellen ausfindig gemacht wurden, heißt das nicht sofort, dass auch die Daten verfügbar sind. Laut [BaGu04] müssen folgende Bedingungen erfüllt sein:

- Organisatorische Voraussetzung
  - Ist es rechtlich zulässig, auf die Datenquelle zuzugreifen (Datenschutz)?
  - Werden Zustimmungen von Betriebsrat und Datenschutzbeauftragten benötigt bzw. sind diese informiert?
  - Sind die Besitzverhältnisse der Daten geklärt?
  - Ist die Vertraulichkeit der Daten gewährleistet?
  - Wie ist die Verfügbarkeit und Aktualität der Daten?
- Technische Vorraussetzung

- Ist der Durchgriff auf die Daten technisch realisierbar (softwaretechnische Zugriffsmöglichkeiten, Rechner- und Netzverfügbarkeit)?
- Ist der Schutz vor unberechtigtem Zugriff bei der Übertragung gewährleistet?
- Können die Daten hinreichend schnell übertragen werden?

3.4.2 Basisdatenbank

In der Referenzarchitektur ist die Basisdatenbank zwischen dem Arbeitsbereich und dem DWH angesiedelt. Sie bietet eine integrierte Datenbasis für verschiedene Analysen. Dabei ist sie unabhängig von konkreten Analysen d.h., dass noch keine Aggregationen vorgenommen wurden. Weitere Aufgabe ist die Versorgung des DWH mit bereinigten Vortriebsdaten u. U. durch Verdichtung.

In der Praxis wird die Basisdatenbank oft weggelassen, da der Aufbau und dessen Pflege sehr aufwendig und teuer werden können.
Nach [Inmo02] ist die Basisdatenbank der Operational Data Store (ODS).

3.4.3 Arbeitsbereich

Der Arbeitsbereich ist eine der drei Datenhaltungskomponenten und dient zur Unterstützung der Datenaktualisierung der Basisdatenbank. Dabei werden die Daten aus den Datenquellen aufgenommen, im Arbeitsbereich transformiert und in die Basisdatenbank geladen [Hinr02, BaGu04].

3.4.4 Data-Warehouse

Ein Data-Warehouse ist die zugrunde liegende Datenbank für alle Analysewerkzeuge. Die Daten werden im Rahmen des Data-Warehousing aus verschiedenen Quellen extrahiert, durch Transformation bereinigt und vereinheitlicht, um danach in das Data-Warehouse geladen zu werden. Das zugrunde liegende Datenmodell wird im Kapitel 3.5 diskutiert [BaGu04].

3.4.5 Repositorium

Hierbei handelt es sich üblicherweise um eine integrierte Datenbank, in der die Metadaten des DWS gespeichert und vom Metadaten-Manager verwaltet werden. Nach [BaGu04] definieren sich Metadaten wie folgt. „Unter dem Begriff Metadaten wird jede Art von Information verstanden, die für den Entwurf, die Konstruktion und die

Benutzung eines Informationssystems benötig wird. Aufgrund der Verwendung im Data-Warehouse werden Metadaten in *fachliche* und *technische* unterschieden.

Fachliche Metadaten dienen hauptsächlich dem Benutzer. Mit deren Hilfe kann er effektive Analyseanwendungen auf den Daten, die im DWH gespeichert sind, vollziehen. Weiterhin gehören dazu u.a. vordefinierte Anfragen, Berichte, Datum und Uhrzeitformate oder auch Maßeinheiten.

Dagegen dienen technische Metadaten hauptsächlich den Administratoren, Anwendungsentwicklern und Softwarewerkzeugen.

3.4.6 Monitor

Die Komponente Monitor unterstützt den Vorgang im Arbeitsbereich durch die Auswahl aller relevanten Daten aus den jeweiligen Datenquellen. Durch ihre Auswahlfunktion bestimmt sie daher maßgeblich die Qualität eines DWS. Nach [BaGu04] können die folgenden Monitor-Strategien unterschieden werden:

- *Triggerbasiert:*
  Jede Änderung an den Quellen wird aktiv vom Datenbanksystem erkannt und darauf das geänderte Tupel in eine Datei oder andere Datenstruktur geschrieben.
- *Replikationsbasiert:*
  Bei diesem Dienst werden die Geänderten Tupel in eine zusätzliche Tabelle geschrieben. Die replizierten Daten können somit direkt benutzt werden.
- *Zeitstempelbasiert:*
  Jeder geänderte Datensatz in den Quellen bekommt ein Zeitstempel. Dieser speichert den Zeitpunkt der Änderung. Sollen die Daten ins DWH geladen werden, muss nur verglichen werden, welche Daten sich seit dem letzten Ladevorgang geändert haben.
- *Log-basiert*:
  Vorgenommende Transaktionen werden in eine Log-Datei geschrieben. Durch Auswertung dieser Datei kann ermittelt werden, welche Daten ins DWH zu laden sind.
- *Snapshot-basiert*:
  In gleichmäßigen Abständen wird eine Snapshot der Datenquelle gemacht. Durch den Vergleich zweier aufeinander folgenden Snapshot können die Änderungen identifiziert werden.

3.4.7 ETL-Prozess

ETL-Prozess ist eines der wichtigsten Elemente im Data-Warehousing, da es die Weise beschreibt, wie die Daten ins Data-Warehouse geladen werden. ETL steht dabei für »Extraktion, Transformation und Laden«. Darunter wird ein Prozess verstanden, der eine Datenintegration von den zu meist heterogenen Datenquellen über das ODS ins DWH vornimmt. Weitere Aufgaben sind das Überwachen der Datenquellen und die Entdeckung von Änderungen durch den Monitor, die Transformation der Daten zwecks Integration sowie das Sichern der Datenqualität [Hinr02, BaGu04].

3.4.7.1 Extraktionskomponente

Die Extraktionskomponente ist ein Teil des Datenbeschaffungsbereichs und ist verantwortlich für das Übertragen von Daten aus den Datenquellen in den Arbeitsbereich. Durch die unterschiedlichen Monitor-Strategien können die konkreten Umsetzungen variieren. Die einzelnen Monitorstrategien wurde in Abschnitt 3.4.6 diskutiert.

3.4.7.2 Transformationskomponente

Die Datentransformation ist die zentrale Aufgabe des ETL-Prozesses, bei dem die Ausgangsdaten an das geforderte Zielschema angepasst werden müssen. Die vorhandene Datenqualität wird analysiert und ggf. mittels automatisch ausgewählter Algorithmen aus dem Bereich des *Data Cleansing* und des *Data Transforming* erhöht. Bei Durchführung der Konsistenzprüfungen und der gegebenenfalls notwendigen Korrektur müssen die heterogenen Ausgangsdaten miteinander in Verbindung gesetzt werden. Dieser Vorgang kann auch bei mittelgroßen Datenmengen bereits sehr zeitaufwendig sein, da zum Beispiel zur Prüfung der referenziellen Integrität sehr viele so genannte Lookups nötig sind. Ein Lookup liegt vor, wenn während einer Transformation eine weitere Abfrage durchgeführt wird, um zusätzliche Informationen zu dem momentan behandelten Datensatz zu erhalten. Zum Beispiel kann nachgesehen werden, ob zu allen in einer Bestellung aufgeführten Artikelnummern auch entsprechende Datensätze in der Artikeltabelle vorhanden sind, falls dies nicht durch eine entsprechende Fremdschlüsselbedingung abgesichert ist [BaGu04].

3.4.7.3 Ladekomponente

Nach der erfolgreichen Datentransformation werden die bereinigten und aufbereiteten (z. B. aggregierten) Daten in die Basisdatenbank bzw. das DWH geladen. Der eigentliche Ladevorgang wird dabei von den mitgelieferten Datenbankmanagementsystemladewerkzeugen realisiert (z. B. SQL*Loader von Oracle). Die Werkzeuge werden auch als Bulk-Loader bezeichnet. Der Ladevorgang kann in online und offline Vorgänge unterteilt werden. Dabei steht beim Online-Ladevorgang das DWH bzw. die Basisdatenbank weiter zu Verfügung. Im Offline-Ladevorgang steht das DWH in der Zeit des Ladevorgangs nicht für Auswertungen zu Verfügung [Hinr04].

3.4.8 Data-Warehouse-Manager

Er ist die zentrale Komponente eines DWS, einerseits für die Ablaufsteuerung und anderseits für die Analyse der Daten im DWS zuständig. Im Data-Warehouse-Prozess initialisiert, steuert und überwacht er die einzelnen Prozesse des Datenbeschaffungsprozesses. Änderungen von den Datenquellen werden vom Monitor an den DWH-Manager weitergeleitet, um die richtige Extraktionskomponente zu laden. Nach dem Beginn des Ladevorgangs überwacht und koordiniert der DWH-Manager die weiteren Vorgänge des Bereinigens, Integrierens, Konsolidierens, Aggregierens der Daten. Dabei lädt der DWH-Manager die Daten über den Arbeitsbereich in die Basisdatenbank bis ins DWH.

Im möglichen Fehlerfall übernimmt der DWH-Manager die Protokollierung und startet geeignete Wiederanlaufprozesse.

Um die einzelnen Aufgaben zu bewältigen, benutzt der DWH-Manager die Informationen, die im Repositorium gespeichert sind. Dazu kommuniziert der DWH-Manager mit dem Metadaten-Manager. Weiterhin stellt der DWH-Manager eine Schnittstelle bereit, die es den Komponenten des DWS erlauben, auf das Repositorium zuzugreifen, um einzelne Werte und Parameter zur Laufzeit zu laden [BaGu04].

3.4.9 Metadatenmanager

Den letzten Bestandteil des DWS stellt der Metadatenmanager dar. Dieser steuert die Metadatenverwaltung des DWS. Dabei handelt es sich um eine Datenbankanwendung mit Konfigurationsmanagement, Zugriffs-, Anfrage- und Navigationsmöglichkeiten für Metadaten. Damit die Komponenten des DWH Metadaten verwenden und auch aktualisieren können, bietet der Metadatenmanager ein

Application Programming Interface (API) für den Zugriff auf das Repositorium an. Letztendlich ergibt sich der Kontrollfluss zwischen dem DWH-Manager und dem Metadatenmanager aus den metadatengesteuerten Prozessen. Hierbei werden die Metadaten über den Metadatenmanager aus dem Repositorium geholt und an den Data-Warehouse Manager zur weiteren Verarbeitung übergeben. Dieser liefert im Idealfall diese Metadaten an metadatenfähige Werkzeuge, die diese interpretieren und ausführen. Der Metadatenmanager verarbeitet technische Metadaten für die Komponenten, die Datenquelle und den Arbeitsbereich. Business-Metadaten werden hingegen mit Komponenten wie Data-Warehouse, Data-Mart und Analyse ausgetauscht. [BaGu04]

3.5 Multidimensionale Datenmodell für Vortriebsdaten

3.5.1 Objekte

Eine multidimensionale Datenstruktur im DWH besteht immer aus drei grundlegenden Objekten, den Fakten, den Dimensionen und den Regeln [GaGl97].

Fakten sind Variablen, die eine Vortriebskennzahl repräsentieren (z. B. Vortrieb, Aushub, Kosten). Fakten können in verschiedene Versionen aufgeteilt werden (z. B. Ist-Level, Soll-Level).

Dimensionen spiegeln die Sichtweise wider, nach der die Fakten aufgeschlüsselt werden (z. B. Projekt, TBM, Raum, Zeit). Diese Dimensionen lassen sich zu Hierarchien verdichten (z. B. Sekunde – Stunde - Tag). Innerhalb einer Dimension wird zwischen Dimensionselementen, nach denen verdichtet wird (z. B. Indikator), und Dimensionsattributen unterschieden. Dimensionsattribute haben dabei lediglich beschreibenden Charakter (z. B. Abmessungen). Regeln sind Vorschriften, die festlegen, wie eine Kennzahl berechnet wird (z. B. Gesamtvolumen=Auslass1+Auslass2).

3.5.2 Beziehungen

Es gibt zwei grundsätzliche Beziehungstypen, die kennzeichnen, wie Fakten und Dimensionen zueinander stehen. Das eine ist das so genannte Star-Schema. Synonym kann der Begriff multidimensionaler Datenwürfel oder Hypercube verwendet werden. Hier sind alle Kennzahlen zusammengefasst und alle Dimensionen sternförmig angeordnet.

Üblicherweise ergibt eine Kennzahl (Fakt) einen Sinn, aber nicht immer über alle Dimensionen. Aus diesem Grund wird das Star-Schema zum Galaxy-Schema erwei-

tert, indem die einzelnen Kennzahlen nur den sinnvollen Dimensionen zugeordnet werden. Synonym hierzu ist der Begriff Multicube, der mehrere multidimensionale Datenwürfel repräsentiert. Diese beiden Beziehungstypen werden später als logische Architekturmodelle noch ausführlicher beschrieben.

3.5.3 Aggregationen

Sinn und Zweck von OLAP-Anwendungen ist es, die Masse aller Daten zunächst auf höchster Hierarchiestufe zu einer aussagekräftigen Kennzahl zu verdichten. Die entstandene Kennzahl kann dann für die Planung und Entscheidungsfindung wieder aufgeschlüsselt werden, um positive oder negative Ursachen beim Vortrieb zu erkennen. Dies geschieht meist über mehrere Hierarchieebenen und auch -pfade. Dieses Aufsplittern bzw. Zusammenfassen einer Kennzahl wird auch als Drill down bzw. Roll up bezeichnet.

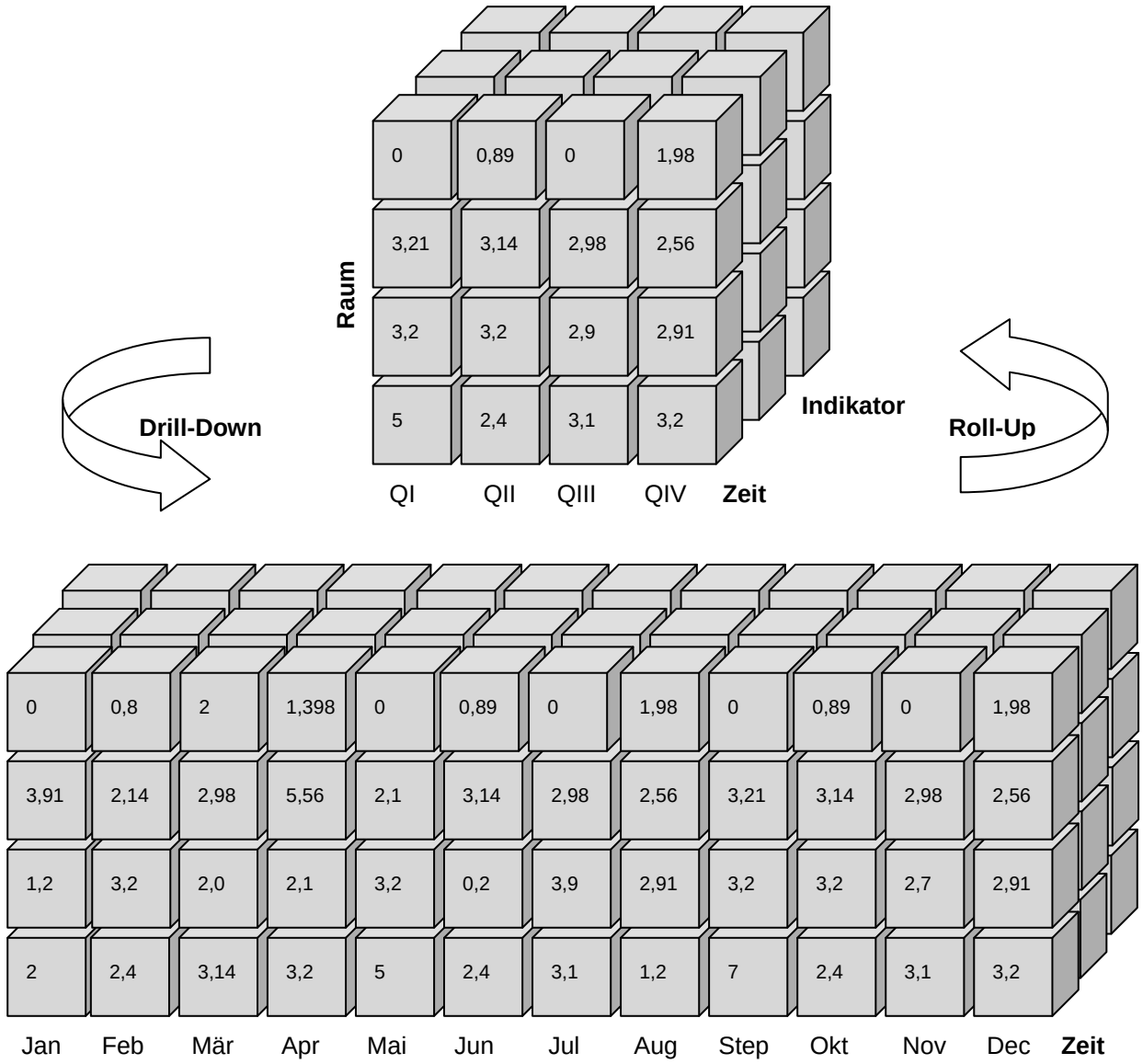


Abbildung 3-3 Driling

Um eine Kennzahl sinnvoll verdichten zu können, sollte sie sich allen beteiligten Dimensionen gegenüber additiv verhalten. Dadurch kann sie auf allen Dimensionshierarchiestufen mittels eines Summen-Operators zusammengefasst werden. Ist dies nicht über alle Dimensionen der Fall, wird die Kennzahl als semi-additiv bezeichnet. Die Lösung ist hier ein Galaxy-Schema, welches die Kennzahl den entsprechenden Dimensionen zuordnet. Lässt sie sich über keine Dimension aufsummieren, ist die Kennzahl nicht-additiv. In diesem Fall lässt sich eine Aggregation teilweise über Operatoren wie Durchschnitt, Minimum oder Maximum erreichen. Die meisten Vortriebsdaten sind nicht-additiv. Manchmal liegt der Grund für die Nicht-Additivität einer

Kennzahl auch in einer zu geringen Granularität einer Dimension. In diesem Fall würde eine Aggregation zu inkonsistenten Daten führen [Kimb96b, GoMR98].

3.5.4 Operationen

Die für das analytische Vorgehen wichtigen Operationen sind Navigieren, Auswählen und Anordnen. Diese Operationen werden zur Navigation sowie zur Datenauswahl und –anordnung innerhalb von Datenwürfeln oder über Datenwürfel hinweg verwendet.

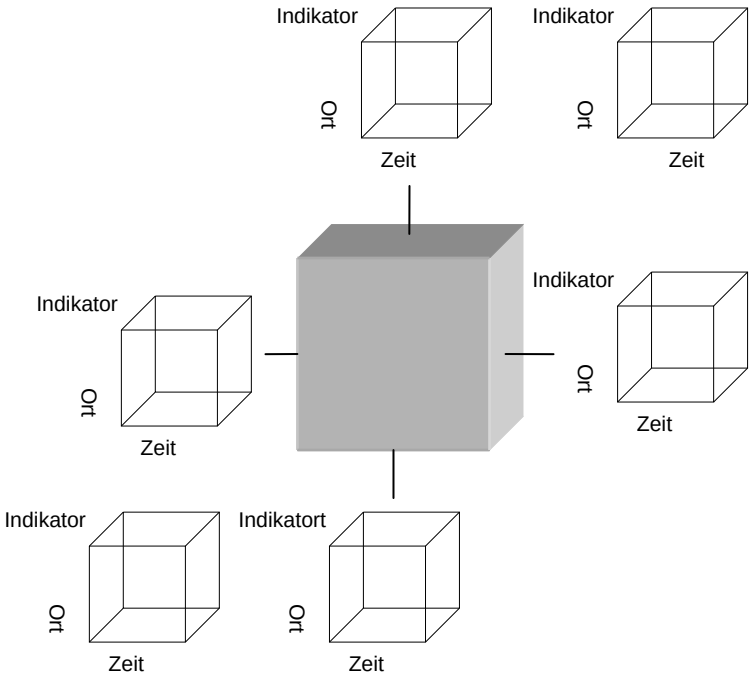


Abbildung 3-4 Privoting / Rotating

Die wichtigsten Navigationsoperationen sind Drill down bzw. Roll up, Drill across und Drill around.

Drill down entspricht dabei dem Sprung auf eine tiefere Hierarchiestufe (z. B. von Jahr nach Quartal), wobei die Aggregation aufgelöst wird. Roll up bewirkt genau das Gegenteil. Es wird eine Hierarchiestufe höher gesprungen, die Daten somit verdichtet (z. B. von Quartal nach Jahr). Drill across ist eine Navigationstechnik, mit der eine Kennzahl über mehrere Würfel, die eine Wertkette bilden, verfolgt werden kann. Dabei ist zu beachten, dass die Dimensionen der einzelnen Würfel die gleiche Granularität aufweisen müssen. Bei Drill around sind die einzelnen Würfel nicht wie oben nacheinander, sondern kreisförmig umeinander angeordnet, d.h., es können so alle an einer Dimension beteiligten Kennzahlen ausgewertet werden (Kennzahlen

aus verschiedenen Fakttabellen). Drill across und Drill around kennzeichnen dabei auch die Beziehungen, die zwischen einzelnen Datenwürfeln möglich sind.

Beim Drill across werden gleiche Kennzahlen miteinander verknüpft, beim Drill around werden gleiche Dimensionen miteinander verbunden [Kimb96b].

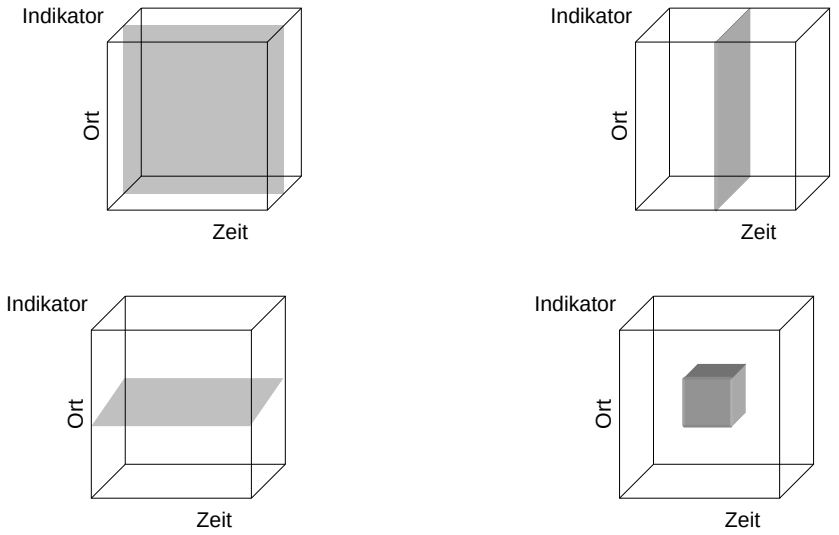


Abbildung 3-5 Slice and Dice

Operationen zum Auswählen von Daten sind Slice und Dice:

- Slice entspricht einem Schnitt durch einen Datenwürfel, der dazu dient, eine bestimmte Datenauswahl innerhalb einer Dimension zu treffen (z. B. alle Setzungen aus dem Projekt „Bologan“).
- Dice verkleinert den Würfel insgesamt über alle Dimensionen und entspricht damit einem Würfelausschnitt. Zum Anordnen von Daten werden die Operationen Rotation und Ranging [MuBe96] angewendet. Bei der Rotation wird die Sichtweise auf die ausgewählten Dimensionen eingestellt (z. B. Verpressung pro Tunnelmeter oder Aushub pro Ring). Dies würde bei einer tabellarischen Darstellung die Vertauschung von Zeilen und Spalten repräsentieren. Ranging beschreibt die Anordnung der Elemente innerhalb einer ausgewählten Dimension, also der Sortierung. Rotation und Ranging wird auch als Pivoting bezeichnet.

3.6 Datenbanken für das Data-Warehouse

Derzeit werden für DWH und OLAP, je nach Hersteller, verschiedene Datenbanktechnologien eingesetzt. Da jede Technologie Vorzüge, aber auch Nachteile hat und das Thema DWH, jedenfalls im Vergleich zu Transaktionsdatenbanken, im Prinzip noch in den Kinderschuhen steckt, hat sich noch keine dieser Technologien so herausragend bewährt, um als Standard zu gelten.

3.6.1 Differenzierung Data-Warehouse - OLAP

Bezüglich DWH und OLAP herrscht in der Literatur eine teilweise synonyme, teilweise differenzierte Verwendung der Begriffe. Um die für DWH und OLAP verwendeten Datenbanken einordnen zu können, lassen sich die Begriffe folgendermaßen differenzieren. Das DWH, wo Daten multidimensional als Star- oder Galaxyschema abgespeichert werden, kann ein relationales DBMS (RDBMS), ein multidimensionales DBMS (MDBMS) oder ein objektorientiertes DBMS (OODBMS) sein.

3.6.2 Relationaler Ansatz

Relationale Datenbanken

Als relationale Datenspeicher für Vortriebsdaten lassen sich alle bekannten RDBMS verwenden, die auch für OLTP-Systeme verwendet werden. Als Vorteile sind hier anzuführen, dass es sich um ausgereifte, gut erforschte Produkte handelt, die auf einem verständlichen relationalen Modell basieren (Tabellen). Alle bekannten Produkte können mit großen Datenmengen umgehen (verteilte und partionierte Datenhaltung). Relationale Datenbanken besitzen eine standardisierte Abfragesprache (SQL), eine ausgereifte Metadatenverwaltung und sind für Mehrbenutzerbetrieb ausgelegt. Nachteile ergeben sich jedoch besonders dadurch, das SQL keine speziellen Konstrukte zur Durchführung multidimensionaler Operationen besitzt. Die Nachbildung dieser Konstrukte mit herkömmlichen SQL-Befehlen ergibt umfangreiche Abfragegebilde, die nur paarweise abgearbeitet werden können (pairwise join) und somit mehrfach durchlaufen werden müssen.

Außerdem ist es nur bedingt möglich, das Abfrageergebnis hinsichtlich seiner Struktur nachträglich zu verändern (z. B. Pivoting). Das Ergebnis ist eine langsame Abfragegeschwindigkeit und eine inflexible Abfrageabarbeitung. Durch das abfragespezifische Leistungsverhalten lassen sich deshalb keine konstanten Ant-

wortzeiten realisieren. Weiterhin gilt, dass die im DWH üblichen, hierarchischen Strukturen nur mangelhaft darstellbar sind und so Aggregationen nicht explizit unterstützt werden [MuBe96, LeTe00].

Relationale OLAP-Server

Relationale OLAP-Server bestehen prinzipiell auch aus einer relationalen Datenbank, die jedoch bezüglich multidimensionaler Speichertechnik, Indexierung und Analysefunktionen optimiert ist, um eine bessere Performance und Auswertung zu erreichen. Mit anderen Indexierungsmethoden lassen sich so mehrere Verknüpfungen in einer Abfrage gleichzeitig durchführen, wobei zugleich auf eine optimale Reihenfolge der Abarbeitung geachtet wird. Ein erweiterter SQL-Befehlssatz ermöglicht verbesserte Analyseoperationen. Spezielle für OLAP ausgelegte Speichertechniken (Caches usw.) helfen, die Forderung nach konstanten Antwortzeiten zu erfüllen und steigern die Abfrageperformance.

3.6.3 Multidimensionaler Ansatz

Multidimensionale Datenbanken sind noch nicht allzu lange auf dem Markt, aber dafür speziell auf die Bedürfnisse von multidimensionaler Datenhaltung und -analyse zugeschnitten.

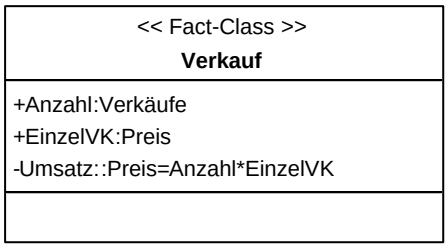


Abbildung 3-6 mUML: Fact-Class Klasse nach [BaGu04]

Die Daten werden in mehrdimensionalen Arrays gespeichert, die miteinander verknüpft werden können. Diese Struktur entspricht im Prinzip dem logischen Abbild des mehrdimensionalen Datenwürfels, wodurch sich ein intuitives Verständnis ergibt. Multidimensionale Datenbanken haben eine auf multidimensionale Analysen zugeschnittene Abfragesprache, wodurch sich die Abfrageperformance gegenüber relationalen Systemen im Allgemeinen deutlich verbessert. Multidimensionale Datenbanken sind jedoch noch in einem proprietären Stadium, das keine Vereinheitlichung zulässt. Jedes System ist anders aufgebaut und strukturiert. Es existiert keine standardisierte Abfragesprache, wodurch auch kein einheitlicher Funktionsumfang

gegeben ist. Erste Erfahrungen mit multidimensionalen Datenbanken haben gezeigt, dass sich keine besonders großen Datenbestände verwalten lassen, weshalb ihr Einsatz momentan auf MOLAP-Server beschränkt ist [MuBe96, LeTe00].

3.7 Datenmodellierung für das Data-Warehouse

Die Auswertung strategischer Informationen im DWH setzt voraus, dass die auszuwertenden Vortriebsdaten in integrierter und konsistenter Form vorliegen. Um diese Datenqualität zu erreichen, müssen die Anforderungen an das DWH sehr genau bestimmt werden. Zudem sollte das DWH eine multidimensionale Globalsicht auf den Tunnelbau bieten. Die Anwendung dieser Methoden zur multidimensionalen Datenorganisation ist die Aufgabe der Datenmodellierung, die bereits auf der höchsten Ebene beginnen muss, um die tunnelbautechnische Sichtweise zu erfassen.

3.7.1 Architektur des Datenmodells

Ein ideales Datenmodell für eine DWH baut auf einem unternehmensweiten konzeptionellen DWH-Schema auf, welches den Anwendungsbereich aus der tunnelbautechnischen Sicht (multidimensional) darstellt.
Das konzeptionelle Schema sollt:

- anwendungsübergreifend, d.h. für zukünftige Ziele und Aufgaben im Unternehmen erweiterbar sein.
- systemunabhängig, d.h. unabhängig von den vorhandenen oder zukünftigen Arbeitsmitteln sein
- sprach- und sachgerecht, d.h., den Aufgabenstellungen und Handlungszielen angemessen sein (fachlich konsistent)

Aus diesem konzeptionellen Schema lässt sich eine 3-Schema-Architektur für Datenbanksysteme entwickeln.
Diese besteht aus:

- einem logischen Schema, welches die Tabellen-, Objekt- oder Array-Struktur festlegt
- einem externen Schema, welches die Sicht der Benutzer und Anwendungen auf das logische Schema beschreibt (Views),
- einem internen Schema, welches die physische Organisation der Tabellen, Objekte oder Arrays auf externen Speichern beschreibt [Balz99].

3.7.2 Relationale Speicherung

Die relationale Speicherung stellt eine besondere Problematik der multidimensionalen Konstrukte im relationalen Datenbankmodell dar. Das in Kapitel 3.5 vorgestellten multidimensionalen Datenbankmodell soll hinsichtlich seiner Vor- und Nachteile bezüglich der Abbildung in ein relationales Datenbankschema diskutiert werden.

3.7.3 Vorgehensweise zur Entwicklung eines Datenmodells

Ein Datenmodell repräsentiert einen Ausschnitt aus der Realwelt. Der Realitätsausschnitt eines DWH für Vortriebsdaten ist die multidimensionale Sichtweise von Ingenieure (Anwendern) auf den Vortrieb. Diese multidimensionale Sicht sollte sich im Datenmodell wieder finden. Um diesen Realitätsausschnitt möglichst ideal auf ein Datenmodell abzubilden, muss versucht werden, alle notwendigen Elemente der Realwelt zu erfassen und zu dokumentieren. Gelegentlich existieren bei der Entwicklung eines DWH bereits logische und konzeptionelle Unternehmensdatenmodelle (für OLTP-Systeme), die die Datengrundlage des DWH bilden. Oft müssen jedoch aus heterogenen Datenquellen Informationen über die zur Verfügung stehenden Daten gewonnen werden, aus denen dann Analysemöglichkeiten herausgearbeitet werden können, die jedoch nicht immer auch dem Analysebedarf entsprechen.

3.7.3.1 Klärung des Analysebedarfs

Um den Analysebedarf der einzelnen Anwender zu klären, müssen zuerst mittels Interviewtechnik oder Fragebogen Aussagen gesammelt werden, in denen die Anwender beschreiben, welche Informationen sie für ihre Analysen benötigen. Dabei sind folgende Fragen relevant [Rad96]:

- Welche Vortriebsprozesse sollen analysiert werden?
- Mit welchen Kennzahlen (Fakten) wird gearbeitet?
- Welchen Detaillierungsgrad (Granularität) müssen die Vortriebsdaten besitzen (täglich, monatlich oder sekündlich)
- Über welche Dimensionen ist eine Auswertung sinnvoll?
- Wie lassen sich diese Dimensionen verdichten?
- Welche Dimensionsattribute werden benötigt?
- Sind die Attribute stabil oder ändern sie sich mit der Zeit?

Das Ergebnis dieser Fragen ist eine Aussagensammlung, die die Sicht einzelner Anwender auf ihren Analysebereich widerspiegelt.

3.7.3.2 Konzeptionelles Datenmodell

Das konzeptionelle Datenmodell für ein DWH sollte zumindest folgende Anforderungen erfüllen [GaGl97]:

- Die zu analysierenden Kennzahlen (Fakten) müssen ersichtlich sein.
- Die Dimensionen müssen erkennbar sein.
- Es muss unterschieden werden können, ob die Dimension hierarchisch oder nicht-hierarchisch ist.
- Die Hierarchiestufen einer Dimension müssen abgebildet werden können.
- Es müssen multiple Konsolidierungspfade einer Hierarchie darstellbar sein.
- Eine Zeitdimension muss eingeführt werden.
- Die Beziehungen zwischen Dimensionen und Fakten müssen ersichtlich sein.
- Es muss zwischen Dimensionselementen und Dimensionsattributen unterschieden werden können.
- Die Herleitung von Kennzahlen muss ersichtlich sein (Regeln).
- Die Granularität der Daten muss erkennbar sein.
- Verschiedene Versionen einer Kennzahl sollen erkennbar sein (Soll-Ist).
- Kennzahlen mit gleichen Dimensionen sollen als Kennzahlendimension zusammengefasst werden können.

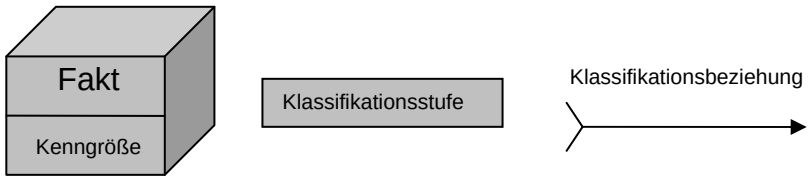


Abbildung 3-7 Die grafische Notation der ME/R-Elemente

Bevor mit der logischen Modellierung begonnen wird, muss die Entscheidung für die zu verwendende Architektur und Technologie getroffen werden. Diese sollte mit der vorhandenen Software und der zukünftigen Planung abgestimmt werden. Anschließend wird das konzeptionelle Datenmodell in das entsprechende logische DWH-Schema umgesetzt.

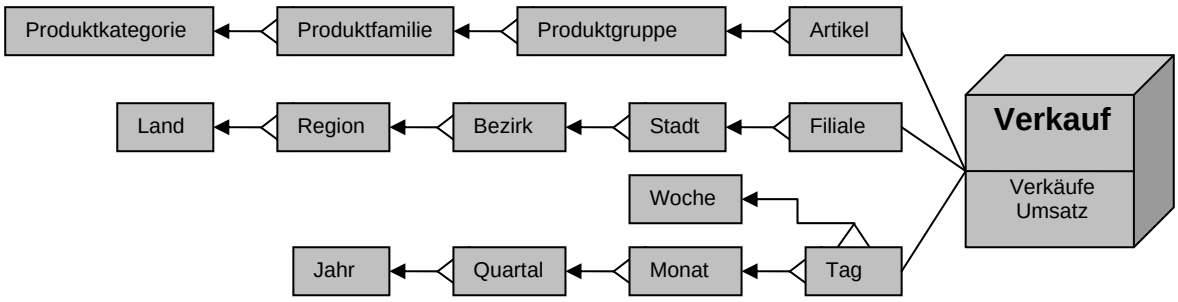


Abbildung 3-8 Kaufhausszenario in ME/R-Notation nach [BaGu04]

3.7.3.3 Vom konzeptionellen zum relationalen logischen Modell

Um vom konzeptionellen Modell zu einer relationalen Tabellenstruktur zu kommen, sollte folgendermaßen vorgegangen werden:

- Kennzahlen mit gleichen Dimensionen werden in einer Relation (Fakt-Tabelle) zusammengefasst, deren Attribute die Kennzahlen sind.
- Jede Dimension wird als Relation definiert, welche die Dimensionselemente und –attribute aller Hierarchiestufen als Attribute enthält (denormalisiert).
- Jede Dimensionsrelation erhält einen künstlichen Primärschlüssel.
- Der Schlüssel der Fakt-Tabelle setzt sich aus den Primärschlüsseln der zugehörigen Dimensionsrelationen zusammen.
- Jede Dimensionsrelation unterhält zur entsprechenden Kennzahlenrelation eine 1:n-Beziehung.

3.7.3.4 Logische Datenarchitektur

Ungeachtet dessen, welche Technologie (relational, objektorientiert oder multidimensional) verwendet wird, muss, auch abhängig vom verwendeten Softwareprodukt, eine Entscheidung zwischen den beiden Modell-Architekturen Star- und Galaxyschema und eventuell auch Snowflakeschema getroffen werden. Diese Architekturen unterscheiden sich hinsichtlich ihres Speicher- und Zeit- und Konsistenzverhaltens, was bei der Entscheidung für ein Modell berücksichtigt werden muss.

3.7.3.5 Physische Aspekte

Um ein DWH-Modell zu optimieren, müssen einige Gedanken auf physische Aspekte gerichtet werden, die insbesondere bei sehr großem Datenumfang ins Gewicht fallen.

Dazu gehören die Steigerung der Abfrageperformance und die Aufteilung des Datenbestands. Hier gibt es mehrere Möglichkeiten, das Problem zu geringer Performance zu lösen oder zumindest die Situation zu verbessern [BaGu04].

- *Vorverdichtung:*
  Um die Abfrage-Performance zu erhöhen, werden Aggregationen vorverdichtet und entweder in der Relation, dem Würfel oder der Klasse selbst abgespeichert oder als separates verdichtetes Modell integriert (materialized views). Die Entscheidung, ob alle oder nur die wichtigsten Aggregationen vorverdichtet werden, muss durch Benchmarking herausgefunden werden. Im Kontext des Tunnelbau ist eine Vorverdichtung nicht möglich, da die Vortriebsdaten immer in der kleinsten Hierarchiestufe benötigt werden.
- *Indexierung:*
  Durch spezielle Indexierungstechniken kann ebenfalls versucht werden, die Abfrageperformance zu verbessern. Damit ist es beispielsweise möglich, mehrere Verknüpfungen einer Abfrage gleichzeitig auszuführen.
- *Abfrageoptimierung:*
  Mit speziellen für DWH ausgelegten Abfrageoptimierungen wird versucht, die Reihenfolge der Abfrageabarbeitung zu optimieren.
- *Parallelisierung:*
  Durch Multiprozessorsysteme können Abfragen aufgeteilt und parallel bearbeitet werden (Hardwarelösung).

Um große Datenbestände auf begrenzten physikalischen Medien ablegen zu können, müssen Verteilungs- und Partionisierungsstrategien ausgearbeitet werden.

- *Verteilung:*
  Große Datenbestände können Über Client/Server-Systeme auf verschiedenen Rechnern abgelegt werden.
- *Partionierung:*
  Sehr große Dimensions- oder Fakttabellen können horizontal (Datengruppen) oder vertikal (Attributgruppen) partioniert werden [BaGu04].

3.7.3.6 Schema-Integration

Beim Aufbau eines DWH wird im Allgemeinen so vorgegangen, dass zunächst Data Marts für einzelne Vortriebsbereiche entwickelt werden, die nach und nach zu einem DWH-Modell für das gesamte Unternehmen zusammengefügt werden. Damit dieses Gesamtmodell realisiert werden kann, müssen jedoch bereits beim Aufbau der Data Marts Festlegungen getroffen werden, die ein späteres Zusammenfügen dieser Data Marts zu einem so genannten Supermart (dem Gesamtmodell) möglich machen. Nur so kann ein erfolgreiches Architekturkonzept für eine DWH entstehen [Kimb96a]. Besonders wichtig ist dabei die Definition von einheitlichen Stamm-Dimensionen (z. B. Kunde, Produkt, Region), auf die das gesamte Unternehmen zugreift. Diese setzten sich in der Regel aus einer Kombination von Attributen verschiedenster Quellen zusammen. Dabei sollte die Granularität der Daten so klein wie möglich gehalten werden, damit die Flexibilität gewahrt bleibt. Wichtig ist auch, dass nicht der Schlüssel der Unternehmensstammdaten verwendet, sondern, dass ein eigener anonymer DWH-Schlüssel definiert wird. Für die Definition der Fakten gilt, dass diese ebenfalls unternehmensweit definiert werde sollten. Es muss ein Glossar angelegt werden, das eine standardisierte Terminologie für die Unternehmenskennzahlen vorgibt. Jede Kennzahl wird nur unter dieser einen Definition angesprochen. Dadurch ist auch die einheitliche Herleitung der Kennzahl gewährleistet. Geschäftsbereichsspezifische Attribute sollten dagegen in eigenen Dimensionen zusammengefasst werden, auf die über das zugehörige Data Mart zugegriffen werden kann und die auch dort gewartet werden. Ansonsten würde sich ein unnötig hoher Verwaltungsaufwand für das gesamte DWH ergeben. Die korrekte Schema-Integration ist eine Aufgabe für das Metamodell, dessen Wichtigkeit im folgenden Abschnitt ersichtlich wird.

3.7.3.7 Metadatenmodellierung

Die Aufbereitung von Metadaten spielt im DWS eine gewichtige Rolle, da sie zum einen der Wahrung der Übersicht in einem solch komplexen System dient, zum anderen lassen sich im DWH viele Vorgänge mittels Metadaten steuern. Dazu gehören die Datenextraktion und -transformation aus den OLTP-Systemen, der automatisierte Ladevorgang zu bestimmten Zeitpunkten, die Verteilung der Datenbestände auf Data Marts, sowie der Zugriff auf die Analyse- und Reportingtools. Metadaten für DWH lassen sich in drei Kategorien unterteilen [Sach98]:

- *Metadaten für DWH-Benutzer*:
  Hier findet der Benutzer Informationen darüber, welche Daten im DWH enthalten sind, wo sie zu finden sind, wie und ob er darauf zugreifen kann, wie lange der Zugriff dauert und welche Datenqualität er erwarten kann.
- *Metadaten für den Data-Warehouse-Administrator*:
  Der DWH-Adminstrator findet hier Informationen, die ihn bei der Wartung und Pflege des DWH unterstützen. Das sind insbesondere Informationen über die Herkunft der Daten, über die Ladeprogramme und die Tabellen und Abfragen des Data-Warehouse. Auch Informationen über die zu verwaltenden Zugriffsrechte sind hier zu finden.
- *Metadaten für den Data-Warehouse-Entwickler* Der DWH-Entwickler hat Zugriff auf Informationen, die ihm bei der Erweiterung und Äderung des DWH hilfreich sind. Das sind beispielsweise Informationen über die Quelldaten und -systeme, über die Transformations- und Ladeprogramme sowie über die DWH-Tabellen und -Abfragen.

3.8 Analyseansätze

Den Begriff Analyse bezeichnet [BaGu04] als einen Oberbegriff, der alle Operationen umfasst, mit denen Daten des DWH durchgeführt werden können. Dabei zählen zur Analyse neben der Anfrage auch die Darstellung und vor allem die Gewinnung neuer Informationen. Die Gewinnung dieser neuen Informationen kann durch einfache arithmetische Operationen oder durch komplexe Untersuchen erfolgen diese werde in Kapitel 3.8 genauer diskutiert. Die so gewonnen Ergebnisse müssen der Basisdatenbank oder dem DWH zurückgeführt werden können, um so eine Rückkopplung zu erreichen. Durch die Rückkopplung kann eine erhöhte Qualität der Datenbasis und damit der Analysen erreicht werden. Die in der Praxis eingesetzten Analysewerkzeuge werden im Sprachgebrauch als *Business Intelligence Tool (BIT)* bezeichnet. Diese geben den Endanwendern die Möglichkeit, die gesammelten Daten interaktiv zu präsentieren. BIT bilden somit eine Schnittstelle vom DWH zum Endanwender [BaGu04].

Die Präsentation der Daten und der Analyseergebnisse werden die typischen Informationsdarstellungstypen, wie Tabellen und Grafiken genutzt.

- *Tabellen:*
  In diesem Element werden die zu meist numerische Daten dargestellt. Diese Kreuztabellen werden als Privot-Tabellen bezeichnet und erlauben für die Analyse das Vertauschen der Tabellenspalten und –zeilen.
- *Grafiken:*
  Diese Form der Darstellung ist neben den Tabellen die wichtigste Darstellungsart von multidimensionalen Daten.
- *Text und multimediale Elemente:*
  Durch die Einbindung von weiteren Informationssystemen wird eine Integration von internen und externen Datenquellen nötig, die nicht als Kennzahlen in der Datenbank verfügbar sind.

In den Kapiteln 3.8.1 und 3.8.2 werden die Analyseansätze die für diese Arbeit in Frage kommen vorgestellt.

3.8.1 DMX

Multidimensional Expressions (MDX) ist eine komplex und mächtig von Microsoft entwickelte OLAP-Anfragesprache der *XML for Analysis-Spezifikation*. MDX bildet keinen Standard im OLAP Bereich, wird aber von vielen gängigen OLAP-Systemen unterstützt. Mondrian [Mond05] stellt dabei eine freie Implementierung dar.

Die grundlegenden Bestandteile von MDX sind *Measures* und *Dimensions*, die den Fakten und Dimensionen eines DWH entsprechen. Die Dimensionen bestehen aus einer Menge von *Members* (Klassifikationsknoten), die in verschiedenen Levels (Klassifikationsstufen) über M*ultiple hierarchies* (Klassifikationspfade) miteinander verbunden sind, über die aggregiert werden kann. Die Members müssen jeweils eindeutig bezeichnet sein.

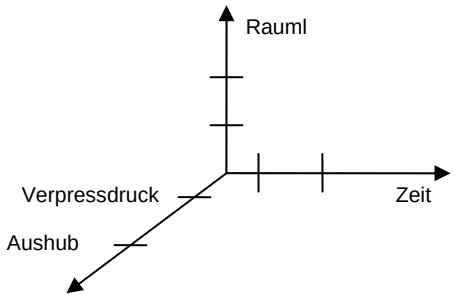


Abbildung 3-9 Measures als Dimension in Anlehnung an [Hinr04]

Dabei werden in der Regel aus einem *Cube* (FROM) eine Menge von Dimensionen und zu ihnen gehörenden Klassifikationsknoten ausgewählt (SELECT) und auf verschiedene Achsen der Ergebnistabelle (COLUMNS, ROWS, PAGES) abgebildet. Mit einem slice (WHERE) kann eine Auswahl innerhalb der Fakten getroffen werden. Mit einfachen eckigen Klammern werden Zeichenketten als Namen gekennzeichnet. Geschweifte Klammern dienen der Definition von Mengen.

```
SELECT {[Measures]. [Unit Sales],[Measures]. [Store Cost]} ON COLUMNS,
order(except([Promotion Media]. [Media Type]. members,
{[Promotion Media]. [Media Type]. [No Media]}),
Measures]. [Unit Sales],DESC) ON ROWS
FROM Sales
```

Abbildung 3-10 Beispiel für eine MDX-Anfrage

3.8.2 JOLAP

Java Based OLAP (JOLAP) ist eine API der Java-Plattform, die eine einheitliche, plattform- und herstellerunabhängige Schnittstelle für OLAP Systemen bietet. Die Spezifikation dieser API wurde durch die JSR-69 (JOLAP) Expert Group im Jahr 2000 begonnen und bis heute nicht beendet.

JOLAP nutzt sowohl das *Common Warehouse Metamodel (CWM)* als auch das Java *Metadata Interface (JMI)* als Grundlage für eine operative API, die in der *Java 2 Enterprise Edition Plattform (J2EE)* bereitgestellt werden. JOLAP liefert damit einen offenen und Standard-basierten Ansatz. JOLAP kann somit als OLAP-Gegenstück der *Java Database Connectivity (JDBC)* betrachtet werden, die für relationale Datenbanken zum Einsatz kommen [Hype05, Wiki05a].

3.9 Zusammenfassung

In diesem Kapitel wurden die einzelnen Komponenten des DWS diskutiert. Der Schwerpunkt lag dabei auf der Modellierung eines multidimensionalen Datenmodells für Vortriebsdaten.

Teil II

Konzept und Realisierung

4 Konzept

Dieses Kapitel beschäftigt sich mit dem Entwurf eines DWS für Vortriebsdaten das im Folgenden *Tunneling Process Control,* kurz *TPC,* genannt wird. Abbildung 4-1 verdeutlicht, dass das TPC in einer 3 Schichten-Architektur (engl. 3 tier architecture) entwickelt wird. Die einzelnen Schichten und deren Komponenten, die in der Architektur von TPC vorkommen, werden im laufenden Kapitel ausführlich beschrieben. Im ersten Schritt werden die wichtigsten Anforderungen an das TPC herausgestellt. Anhand der Anforderungen wird im Abschnitt 4.2 die Modellierung der Metadaten diskutiert. Darauf aufbauend wird in Abschnitt 4.3 ein Konzept für das DWH vorgestellt und dessen zugrunde liegende multidimensionale Datenmodell erörtert. Abgeschlossen wird das Kapitel durch das Vorstellen des softwaretechnischen Entwurfs.

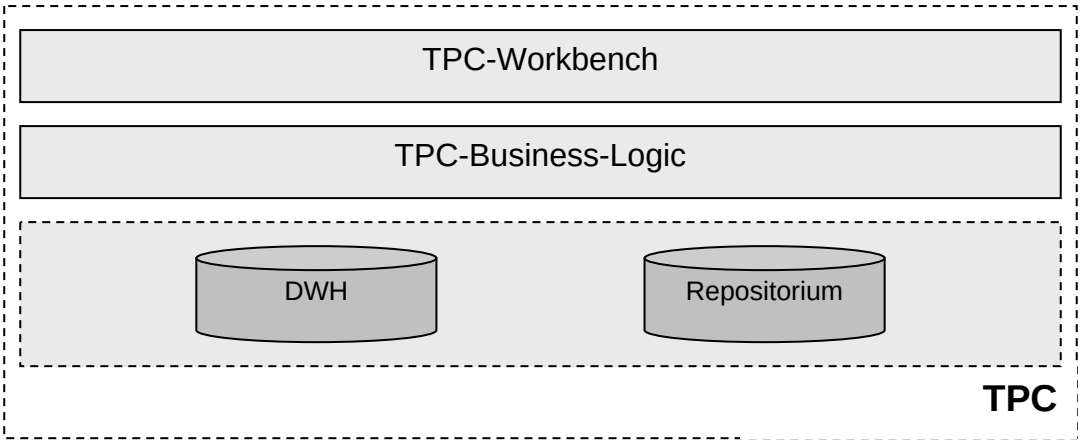


Abbildung 4-1 Grobarchitektur von TPC

4.1 Anforderungen an das TPC

In diesem Abschnitt werden alle Anforderungen, die zum Betrieb des fertigen TPC-notwendig sind, aufgelistet. Das betrifft die anwendungspezifischen, technischen und Realisierungsanforderungen. Abbildung 4-2 zeigt dabei den gewünschten Informationsfluss der Vortriebsdaten von den Sensoren bis hin zu den Konsumenten.

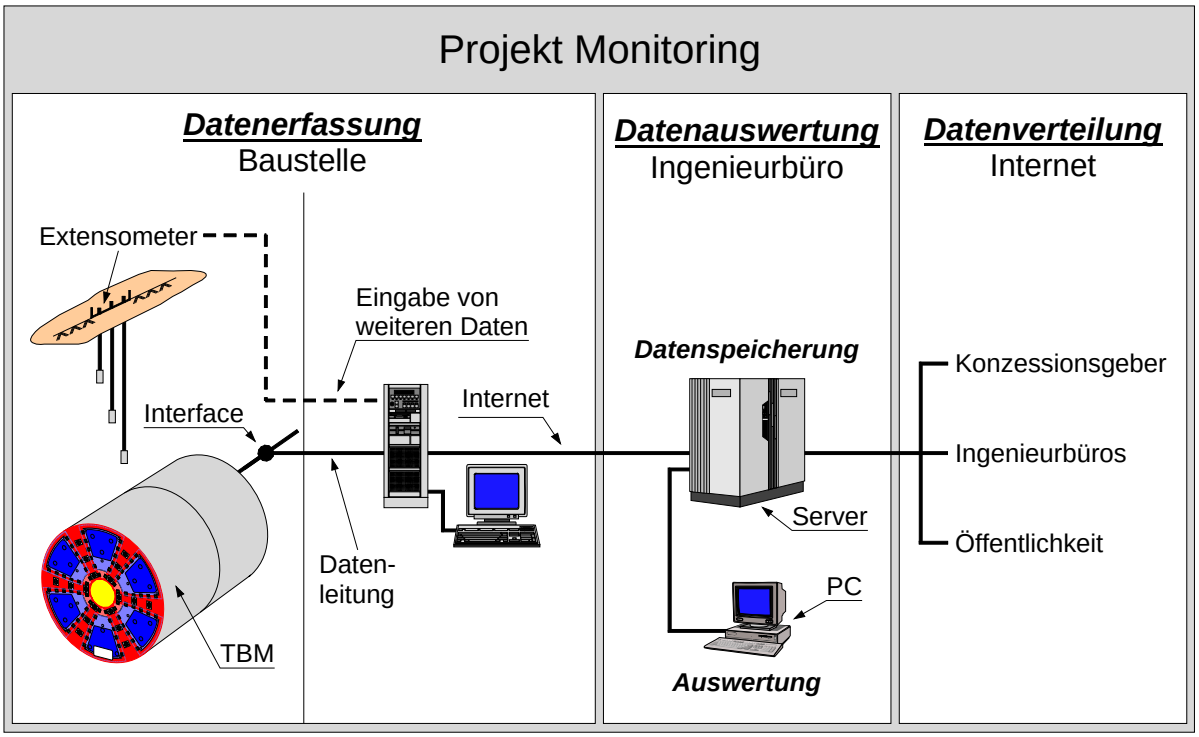


Abbildung 4-2 Projekt Monitoring

4.1.1 Anwendungsspezifische Anforderungen

Die anwendungsspezifischen Anforderungen beschreiben Anforderungen, die für den operativen Betrieb notwendig sind sowie aus den Wünschen der Endbenutzer stammen.

- *Multilingualität:*
  Änderung der Ausgabesprache zur Laufzeit durch Auswahl einer anderen Sprache. Als Standardsprache ist Englisch in das System zu hinterlegen.
- *Analyse und Berichtserstellung:*
  Für die Erstellung eines *Berichtes* (Wochen-, Stations- oder Spezialbericht) müssen vordefinierte Templates für den Inhalt und das Aussehen bereitstehen oder von vom Benutzer erzeugt werden können. Falls bei der Erstellung eines Templates oder Bericht Fehler auftreten, so hat das System geeignet darauf zu reagieren und den Benutzer ggf. zu informieren.
- *Datenintegration:*
  Die Datenintegration soll einen abgeschlossen Vorgang darstellen, in dem der Benutzer die Quelle angibt und den Vorgang anstößt. Bei der Datenintegration soll zwischen den Datenquellen des Vortriebes und den Datenquellen der Setzungsmessungen unterschieden werden.

4.1.2 Technische Anforderungen

Das TPC soll aus Gründen der Plattformunabhänigkeit in Java implementiert werden. Des Weiteren soll das TPC alle gängigen relationalen DBMS über die JDBC Schnittstelle unterstützen. Es darf nur auf Open Source bzw. kostenlose Tools oder Frameworks zurückgegriffen werden.

Eine klare Trennung von den auszuwertenden und Metadaten soll durch die Trennung der Vortriebsdaten und Setzungsdaten erreicht werden. Im Einzelnen werden die Vortriebsdaten im DWH und die Metadaten im Repositorium. Die Metadaten sind unternehmensweit zentral zu speichern, um allen weiteren zu implementierenden Applikationen diese Informationen zur Verfügung stellen zu können.

4.1.3 Realisierungsanforderungen

Den Implementierungsarbeiten liegt ein ingenieurmäßiges Softwaretechnik-Konzept zugrunde. Als Modellierungsnotation für die softwaretechnische Realisierung dient die *Unified Modeling Language* (UML) [Horn05, Balz99]. Einzelne Komponenten werden in einzelnen *Packages* zusammengefasst.

4.2 Metadaten-Schemata

Die Metadaten spielen eine entscheidende Rolle beim Entwickeln und Betreiben eines Data-Warehouse. Dieser Abschnitt stellt das zugrunde liegende Metadatenschemata in Abbildung 4-3 vor.

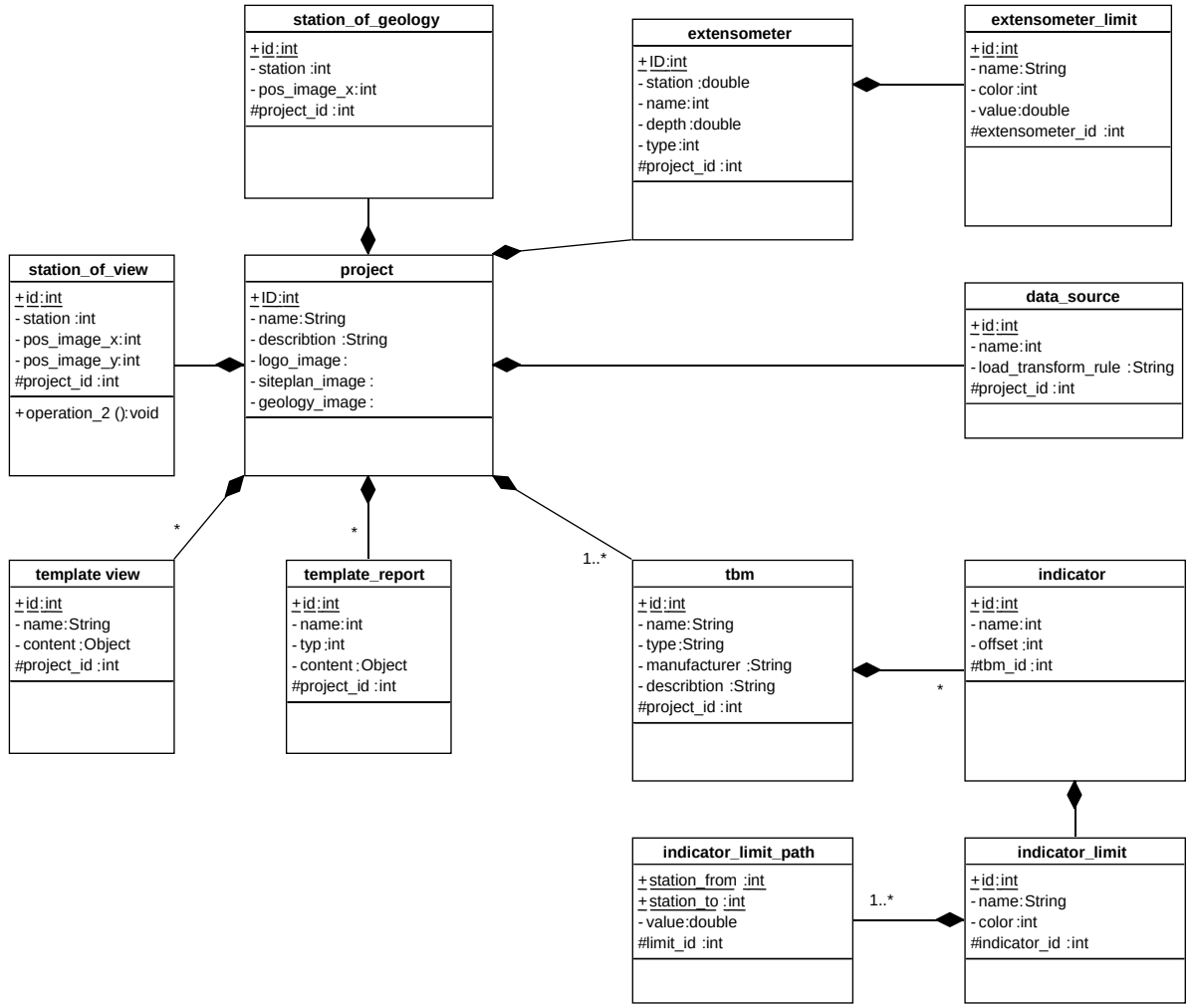


Abbildung 4-3 E/R-Diagramm der Metadaten

Projekt

Den Mittelpunkt bildet die Relation *project.* Es beschreibt die grundlegenden Eigenschaften eines Projekts. Dazu gehören neben dem Namen, eine Kurzbeschreibung, das Projektlogo, ein Übersichtsplan sowie die dazugehörige Geologie.

TBM

Zu jedem Projekt gehört mindestens eine TBM, welche durch die Relation *tbm* beschrieben wird. Die Attributliste einer TBM umfasst den Namen, den Type (Slurry, EPB oder Rohrvortrieb), den Hersteller und eine Kurzbeschreibung.

Messgrößen

Jede TBM liefert eine Reihe von Messgrößen (engl. indicator) dazu existiert die Relation *indicator*. Dazu gehört unter anderem ein Timestamp, der die Projektzeit der Baustelle definiert. Alle weiteren Größen beziehen sich auf diesen Timestamp. Die Position der TBM wird in Tunnelmetern oder als Ringnummer angegeben. Durch die räumliche und zeitliche Zuordnung der weiteren Größen z. B. Stützdruck, Verpressung oder Aushubvolumen kann eine Auswertung über die Zeit oder die Tunneltrasse erfolgen. Je nach Projekt können mehr als 100 Messgrößen pro TBM geliefert werden. Um möglichst eine flexible und schnelle Arthoc –Auswertung zu machen besitzt jeder Indicator eine Reihe von Attributen. Die Wichtigsten sind folgend:

- *name:*
  Die Beschriftung des Sensors in einem Report.
- *offset:*
  Beschreibt die räumliche Differenz zwischen der Ortsbrust und dem Sensor.
- *title:*
  Standardbeschriftung des Diagramms.
- *unit:*
  Physikalische Einheit dieses Sensors.
- *lower_bound:*
  Untere Grenze des Anzeigeintervall auf der Ordinatenachse
- *upper_bound:*
  Obere Grenze des Anzeigeintervall auf der Ordinatenachse

Grenzwerte

Für jede Messgröße können mehrere Trigger-Level existieren. Die Relation *indicator_limit* und *indicator_limit_path* beschreiben diesen Sachverhalt. Dabei enthält die Relation *indicator_limit* die grundlegenden Eigenschaften wie Name und Farbe des Trigger-Level und die Relation *indicator_limit_path* den dazugehörigen Path mit den Trigger-Level-Werten.

Extensometer

Da nicht nur TBM-Daten ausgewertet werden, sondern auch Extensometer bzw. Mehrfachextensometer, wird eine Beschreibung für diese Messgrößen benötigt. Die Relation *extensometer* enthält die entschiedenen Attribute, um ein Extensometer oder Mehrfachextensometer zu beschreiben. Dabei handelt es sich um:

- *station:*
  Position der Extensometer angegeben in Tunnelmetern
- *name:*
  Name des Sensors z. B. ST 64-3 oder INT 3-1
- *depth:*
  Gibt die Tiefe der Messung an. Bei einem Oberflächenextensometer ist das Attribut 0.
- *type:*
  Gibt an, ob es sich um eine Einfach- oder Mehrfachextensometer handelt. Dies ist wichtig, um eine korrekte grafische Aufbereitung zu gewährleisten.

Templates

Die TPC- Templates beschreiben die unterschiedlichsten Reporte. Dabei wird in Report und Ansicht unterschieden. Das Reporttemplates enthält die Informationen, um die Anzahl der Seiten, der Diagramme, die Messgrößen und der Skalierungen zu definieren. Das Ansichttemplates dagegen speichert die grafische Aufbereitung. Hier sind Information über das Seitenformat, die Logos oder die Größe der Diagramme hinterlegt. Dazu existieren im Metadatenschemata die Relation *template_view* und *template_report.*

Geologie und Lageplan

Zu jedem Projekt gehören eine Geologie und ein Lageplan der Tunneltrasse. Diese liegen in digitaler Form vor und können somit leicht in das Projekt integriert werden. Die Schwierigkeit besteht darin, die Position der TBM in den Lageplan einzuzeichnen oder den richtigen Ausschnitt aus der Geologie auszuschneiden. Um dies zu vereinfachen, werden Metainformation über die relative Position innerhalb des Bildes benötigt. Die Relationen *station_of_geology* und *station_of_seitplan* stellen diese Metainformation zur Verfügung.

Extraktion Transformation und Laden

Für die Integration von TBM-Daten, Setzungsmesswerten oder anderen Messgrößen werden Metainformation benötigt. Diese Informationen bildet das Entitie *data_source* ab. Die Einzelnen Attribute haben dabei die folgenden Eigenschaften:

- *name*:
  Name der Datenquelle.

- *load_rule*:
  Regel, die einen Ladevorgang beschreibt, hinterlegt in einem XML- Format.
- *transform_rule:*
  Regeln, die einzelne Transformationen beschreiben, hinterlegt in einem XML- Format

4.3 Multidimensionales Datenmodell für Vortriebsdaten

Im Vergleich zum Datenmodell für ein klassisches Datenbanksystem hat das Datenmodell eines DWS sich weniger an den Funktionen operativer Anwendungssysteme, sondern an der Analyse operativer Daten zu orientieren. Um somit ein multidimensionales Datenmodell für Vortriebsdaten zu entwickeln, muss die Analyse der Vortriebsdaten im Mittelpunkt stehen.

4.3.1 Konzeptuelle und Logische Modellierung

Um die konzeptionelle Modellierung vorzunehmen, muss wurde sich mit den Analysewünschen der Endbenutzer auseinandersetzen. Laut dem Pflichtenheft werden Auswertungen nach der Zeit, der Station und der Ringnummer benötigt. Alle gemessenen Größen sollen dabei über die Zeit, die Station und die Ringnummer aufgetragen werden. Dabei soll sich die Auswertung speziell auf ein Projekt und genau auf eine TBM beziehen. Auswertungen über mehre Projekte und TBM sollen an dieser Stelle nicht diskutiert werden. Das konzeptionelle Modell soll diesen Sachverhalt aber abbilden können.

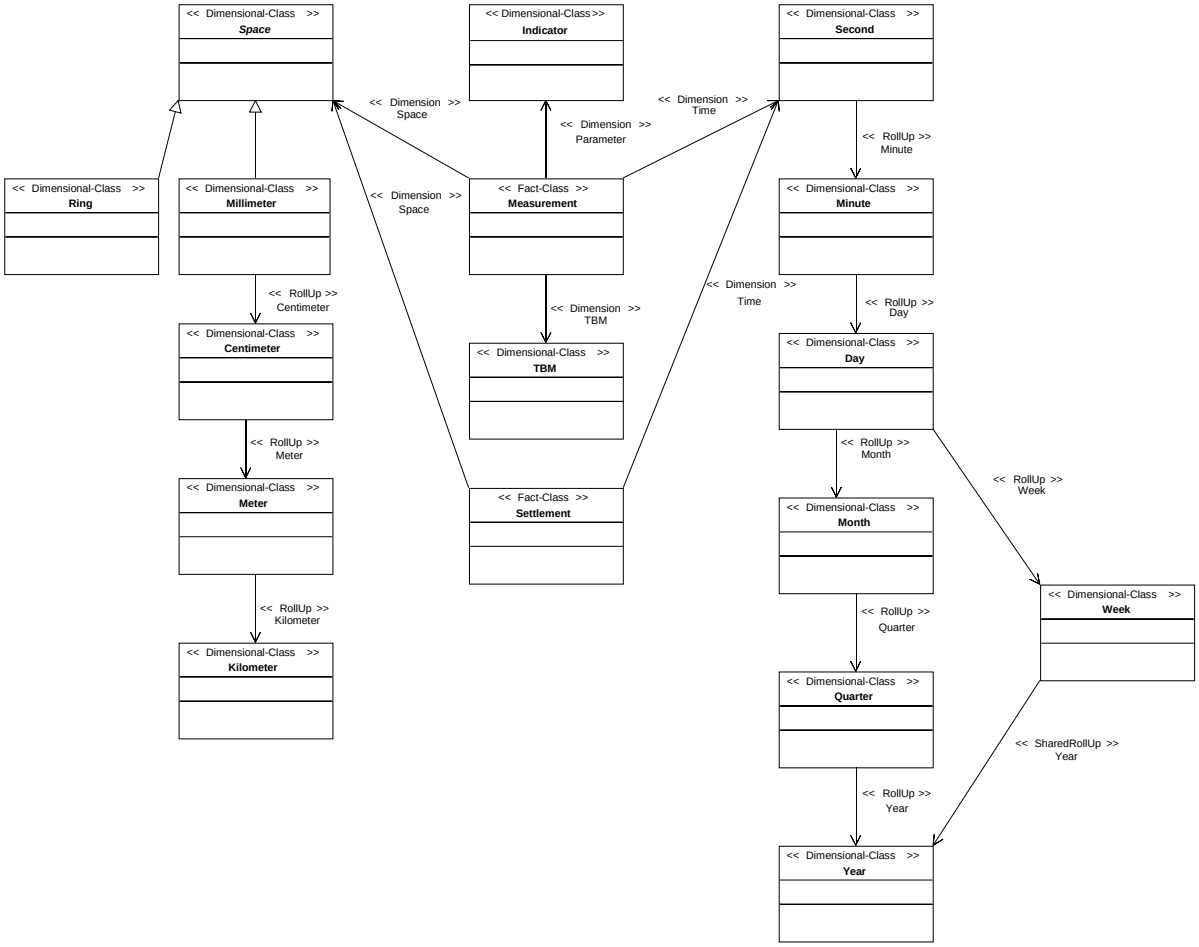


Abbildung 4-4 mUML-Schema der Vortriebsdaten

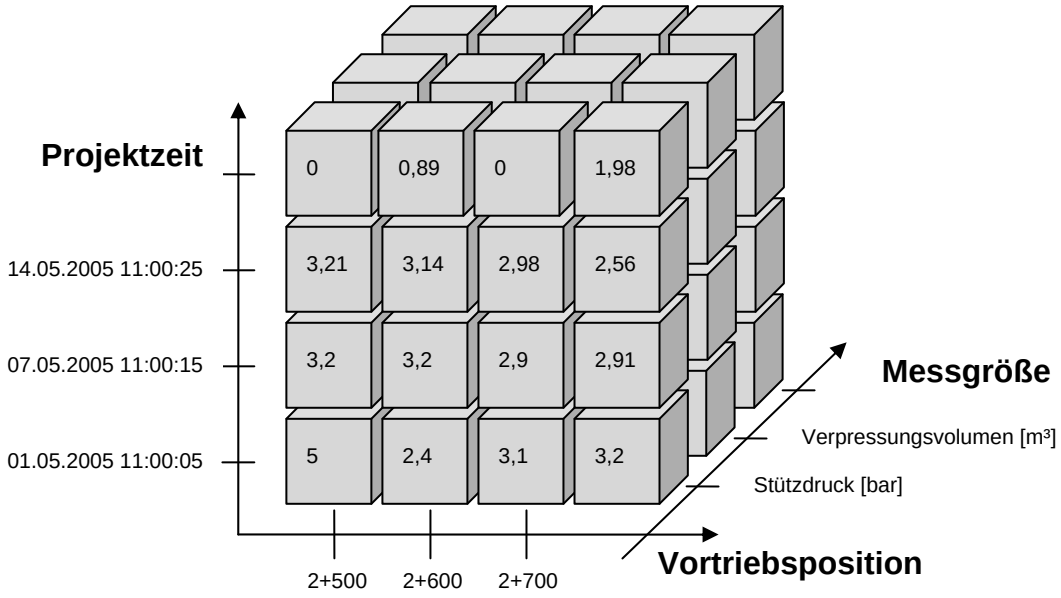
Die Abbildung 4-4 zeigt ein multidimensionales UML-Schema am Beispiel der Vortriebsdaten. Die einzelnen Dimensions-Klassen und Fakt-Klassen sollen anhand von Beispielen vorgestellt werden.

- *Dimension Zeit:*
  Repräsentiert die zeitlichen Zusammenhänge der Vortriebsdaten in den Klassifikationsstufen Sekunde, Minute, Tag, Woche, Quartal und Jahr.
- *Dimension Raum:*
  Die räumliche Beschreibung im Tunnelbau bezieht sich auf die Tunnelmeter, die als Station bezeichnet wird. Die kleinste gemessene Einheit bezogen auf den Raum, ist der Millimeter. Eine andere Beschreibung des Tunnels kann über die Ringnummer erfolgen. Ein Ring hat durchschnittlich eine Breite von 1,5 Meter. Dadurch entstehen die Klassifikationsstufen Millimeter, Zentimeter, Meter, Ring und Kilometer.
- *Dimension Indikator:*
  Beschreibt alle auszuwertende Größen wie Vortrieb, Stützdrück usw.

- *Dimension Projekt*
  Beschreibt ein Projekt, welches mit anderen Projekten auswertbar sein soll.
- *Dimension TBM*
  Beschreibt eine TBM, die mit anderen TBMs auswertbar sein soll.
- *Fakt-Messungen:*
  Enthält alle Kennzahlen, die beim Vortrieb gemessen wurden.
- *Fakt-Setzungen:*
  Enthält alle Kennzahlen, die durch die Extensometer gemessen wurden.

4.3.2 Umsetzung des multidimensionales Datenmodells

Die in Kapitel 4.3 vorgestellte multidimensionale Sicht auf die Vortriebsdaten im Data-Warehouse bedarf einer Transformation, um eine Abbildung auf Relationen zu relasieren. Dazu müssen die Dimensionen und Fakten in Relationen umgewandelt werden.



Projektzeit	Vortriebsposition	Messgröße	Messwert
01.05.2005 11:00:05	2+500	Stützdruck	3,5
07.05.2005 11:00:15	2+600	Stützdruck	3,7
07.05.2005 11:00:15	2+600	Verpressvolumen	7

Abbildung 4-5 Dualismus von Würfel und Tabelle in Anlehnung an [BaGu04]

Die Abbildung 4-5 zeigt den Zusammenhang zwischen der Abbildung von einem *Measurement-Fakt* auf eine Relation. Zum besseren Verständnis wurden in der

Abbildung 4-5 die Dimensionen Projekt und TBM weggelassen. Des Weiteren wurden die Fremdschlüssel durch die eigentlichen Werte ersetzt.

Die Dimensionen Projekt und TBM sind nur zwingend notwendig, wenn eine Auswertung zwischen zwei unterschiedlichen TBM und/oder Projekten erfolgen soll. Diese Form der Auswertung wird in der Regel selten benötigt, da die TBM und die Projekte meist autonom sind. Um eine saubere Modellierung zu gewährleisten und um auf zukünftige Benutzerwünsche einzugehen, wurde dieser Sachverhalt jedoch mitmodelliert.

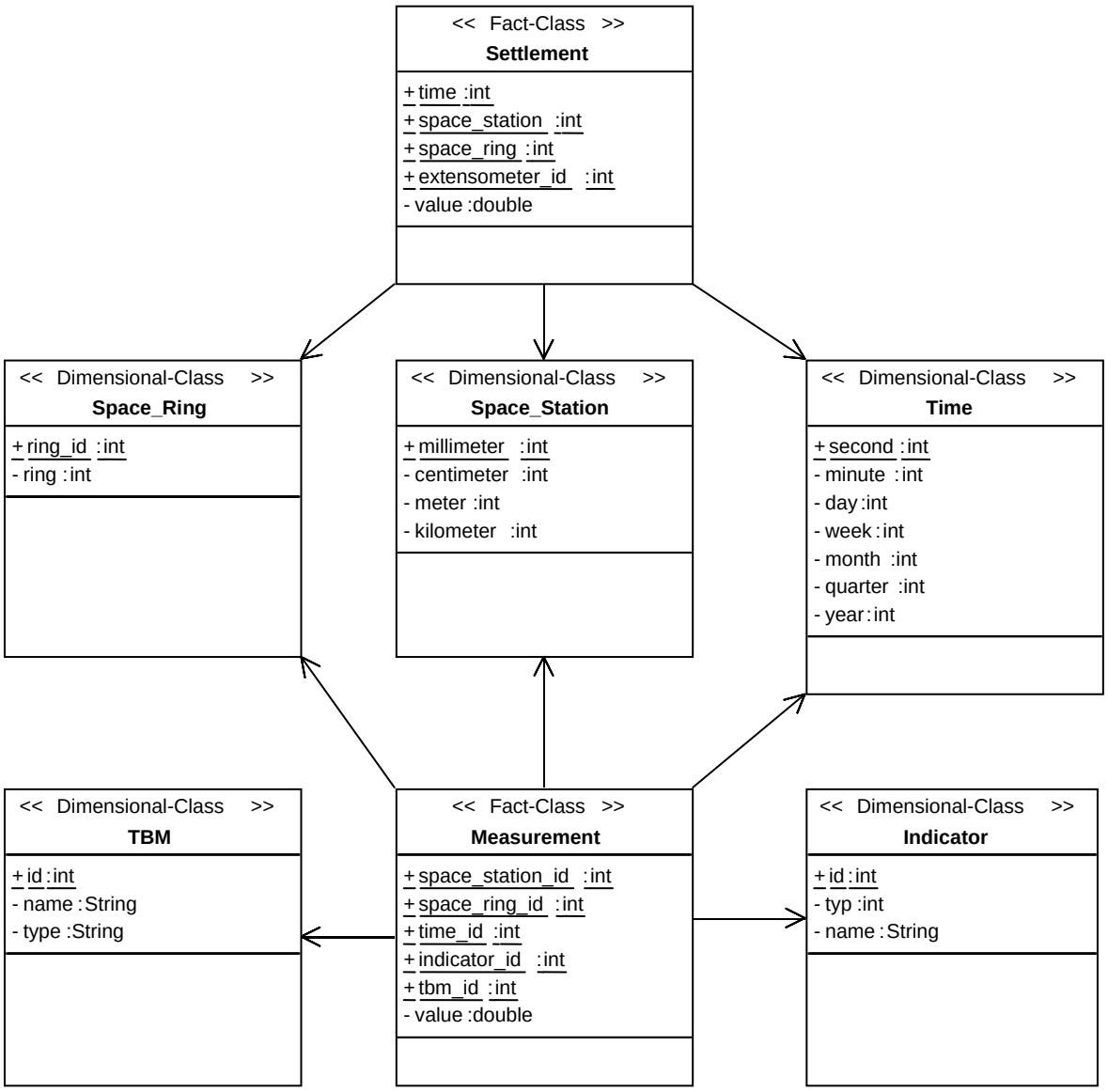


Abbildung 4-6 Galaxy-Schema für Vortriebsdaten

4.4 Integration der Vortriebsdaten

In den folgenden Abschnitten werden die ermittelten Datenquellen hinsichtlich ihrer Architektur, Verfügbarkeit und Qualität beschrieben. Anhand der TBM-Daten werden die dazugehörigen Datenintegrationsmodule und deren Architektur vorgestellt

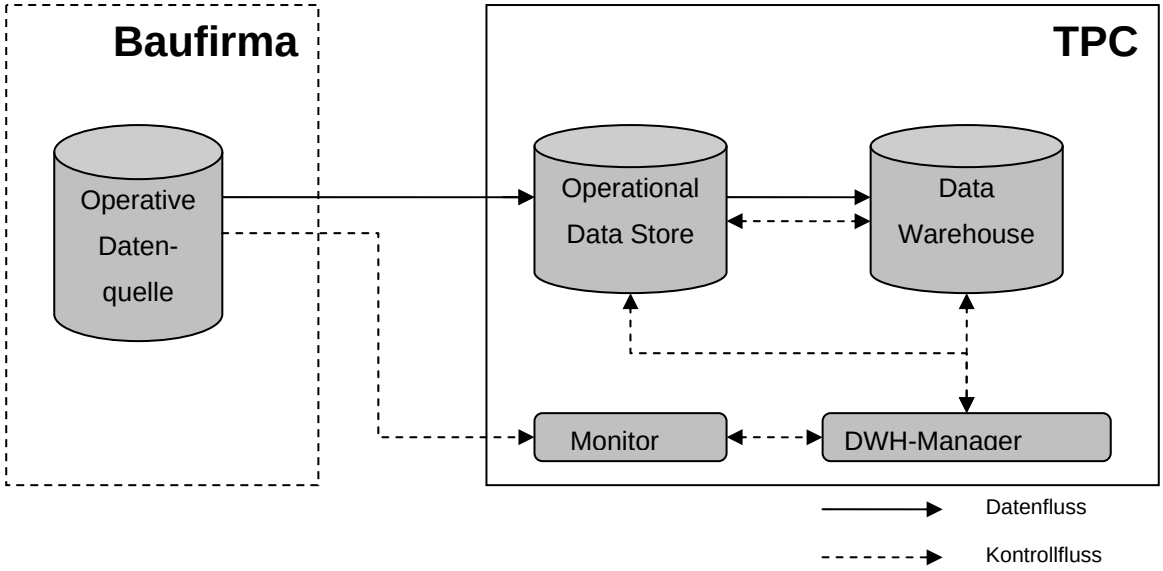


Abbildung 4-7 Datenflussarchitektur

Die Datenquelle aus Abbildung 4-7 steht stellvertretend für alle externen operativen Datenquellen, z. B. Datenbank des Vermessungssystems oder Setzungsmesswert-datenbank.

4.4.1 Datenquellen

Beim derzeitigen Stand des Entwicklungsprozess von TPC werden zwei externe Datenquellen genutzt. Im Laufe der Entwicklung können weitere hinzukommen. Ein Beispiel dafür würde eine Geo-Datenbank sein, die Information in Form von Lageplänen zur Verfügung stellt. Konkret werden TBM-Daten und Setzungsdaten integriert.

TBM-Daten

Unter TBM-Daten werden alle Daten verstanden, die von der Vortriebsmaschine geliefert werden. Dabei handelt es unter anderem um die Daten des Vermessungssystems (Geostationäre Position der TBM und Projektzeit) oder Daten, die den Druckverlauf in einer Abbaukammer beschreiben.

Die TBM-Daten werden alle 10 Sekunden von den Sensoren aufgenommen, über eine SPS verarbeitet und in einer Datenbank der Baufirma zwischengespeichert. In regelmäßigen Abständen, zumeist jeden Montag, werden die Daten per eMail im DBase (DBF) Dateiformat an das Büro Babendererde Ingenieure GmbH gesendet, um dort eine Auswertung der relevanten Vortriebsdaten vorzunehmen. Eine vollautomatische Extraktion der relevanten Vortriebsdaten direkt aus den Datenquellen ist ausgeschlossen, weil die Baufirma dies verweigert. Um dennoch eine möglichst optimale Integration der Vortriebsdaten zu realisieren, wird jede DBF-Datenbankdatei als eine Datenquelle aufgefasst, sie es heißt zu extrahieren, tansformieren und in das DWH zu laden.

Tabellenvoransicht

Date	Time	0	1	2	3	4	5	6	7
21.07.2005	06:00:05	0.0	1.41	1.42	1.78	1.71	0.0	0.0	2.38
21.07.2005	06:00:06	0.0	1.41	1.42	1.78	1.71	0.0	0.0	2.38
21.07.2005	06:00:16	0.0	1.41	1.42	1.78	1.71	0.0	0.0	2.38
21.07.2005	06:00:26	0.0	1.41	1.41	1.77	1.7	0.0	0.0	2.37
21.07.2005	06:00:37	0.0	1.41	1.41	1.77	1.7	0.0	0.0	2.37
21.07.2005	06:00:47	0.0	1.41	1.41	1.77	1.7	0.0	0.0	2.37
21.07.2005	06:00:57	0.0	1.4	1.41	1.77	1.7	0.0	0.0	2.37
21.07.2005	06:01:07	0.0	1.4	1.41	1.77	1.7	0.0	0.0	2.37
21.07.2005	06:01:18	0.0	1.4	1.4	1.76	1.69	0.0	0.0	2.37
21.07.2005	06:01:28	0.0	1.4	1.4	1.76	1.69	0.0	0.0	2.37
21.07.2005	06:01:38	0.0	1.4	1.4	1.76	1.69	0.0	0.0	2.36

Abbildung 4-8 Datenquelle in tabellarischer Darstellung (TBM-Daten)

In dieser Arbeit wird davon ausgegangen, dass sich die Datenquellen über die ganze Laufzeit des Projektes nicht in ihrem Schema ändern. Die Integration von Datenquellen wird immer von Benutzer angestoßen, da dieser die zu integrierenden Datenquellen auswählen muss. Die Erweiterung hin zur vollständig automatisierten der zu integrieren Datenquellen, ist durch die Erweiterung der Monitor-Komponente jederzeit möglich, macht aber in diesem Projekt nur bedingt Sinn.
Folgende einzelnen Schritte umfassen eine vollständige Datenintegration:

- Auswahl aller zu integrierenden Datenquellen durch Benutzer.
- Laden und verarbeiten der benötigten Metadaten.
- Laden der Quelldaten ins ODS.
- Einheitlichen Timestamp aus den Spalten Date und Time erzeugen.
- Laden der bereinigten Daten ins DWH

Setzungsdaten

Ein Setzungsdatensatz beschreibt die absolute Setzung eines geostationären Punktes, bezogen auf die Geländeoberfläche, die meist in Millimetern angegeben wird.

Als Setzungsmessgeräte dienen Extensometer oder Mehrfachextensometer. Extensometer unterscheiden sich dahingehend, dass moderne vollautomatisch eine Setzungsmessung vornehmen und die ermittelte Setzung in einer Datenbank hinterlegen. Ältere hingegen werden manuell abgelesen und der ermittelte Wert wird nachträglich in die Datenbank eingespeist.

Im Bologna-Projekt kommen manuelle und voll automatisierte Extensometer zum Einsatz. Diese werden dem Endbenutzer von einem Subunternehmer via einem Webportal bereitgestellt. Wie auch schon bei den TBM-Daten ist kein direkter Zugriff auf die Datenquellen möglich.

Abbildung 4-9 GIDIE Setzungsdaten Export

Abbildung 4-9 zeigt ein Screenshot des GIDIE Webportals, wo durch Auswahl eines Sensors und Angabe des Zeitraumes eine Abrufung der Setzungsdaten erfolgt. Die Ausgabe erfolgt in einer Exceltabelle. Tabelle 4-1 zeigt den Auszug aus solch einer Exceltabelle.

Timestamp	Setzung [mm]
2005-07-21 12:05:00	12,3
2005-07-21 13:05:00	13,0
2005-07-21 14:05:00	13,51

Tabelle 4-1 Auszug einer Setzungsmessung

4.4.2 Komponenten der Datenintegration

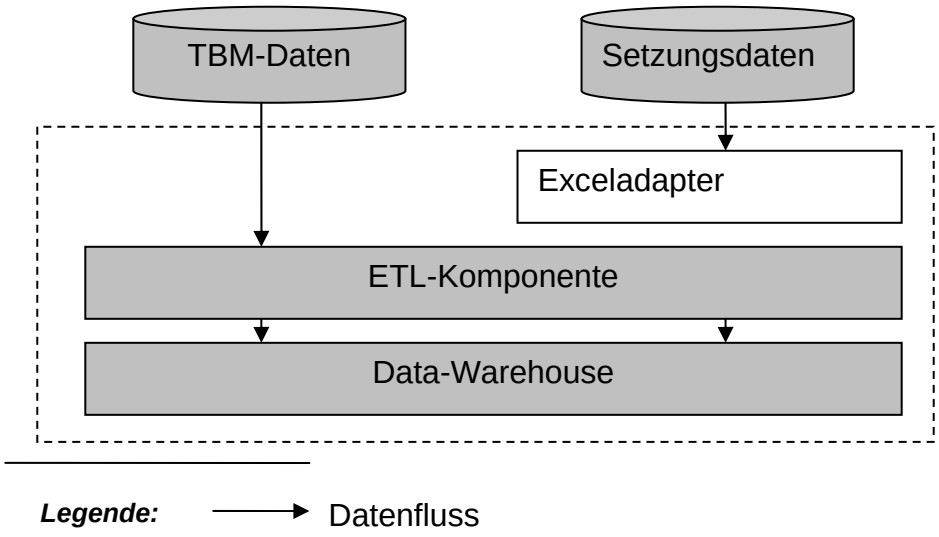


Abbildung 4-10 Grobarchitektur der Datenintegrationskomponente

4.5 Softwaretechnischer Entwurf von TPC

4.5.1 Multilingualität

Durch den internationalen Einsatz des TPC-Systems wird eine Mulilingualität der Benutzeroberfläche unablässig. Jede GUI, dir zur Laufzeit gezeichnet werden soll und die Multilingualität unterstützen will, hat sich dazu an dem Language Objekt über die Methode *addLanguageChangeListene(LanguageChangeListener listener)* anzumelden. Durch die Registrierung am *LanguageChangeListener* ist das Language-Objekt in der Lage, das *LanguageChangeEvent* auszulösen und an jede GUI zu senden. Abbildung 4-11 zeigt das Klassendiagramm in Anlehnung an ein Observerpattern und dessen Zusammenspiel mit einer Konkreten GUI. Die Konkrete GUI steht stellvertretend für alle GUI's, die sich am Language-Objekt anmelden.

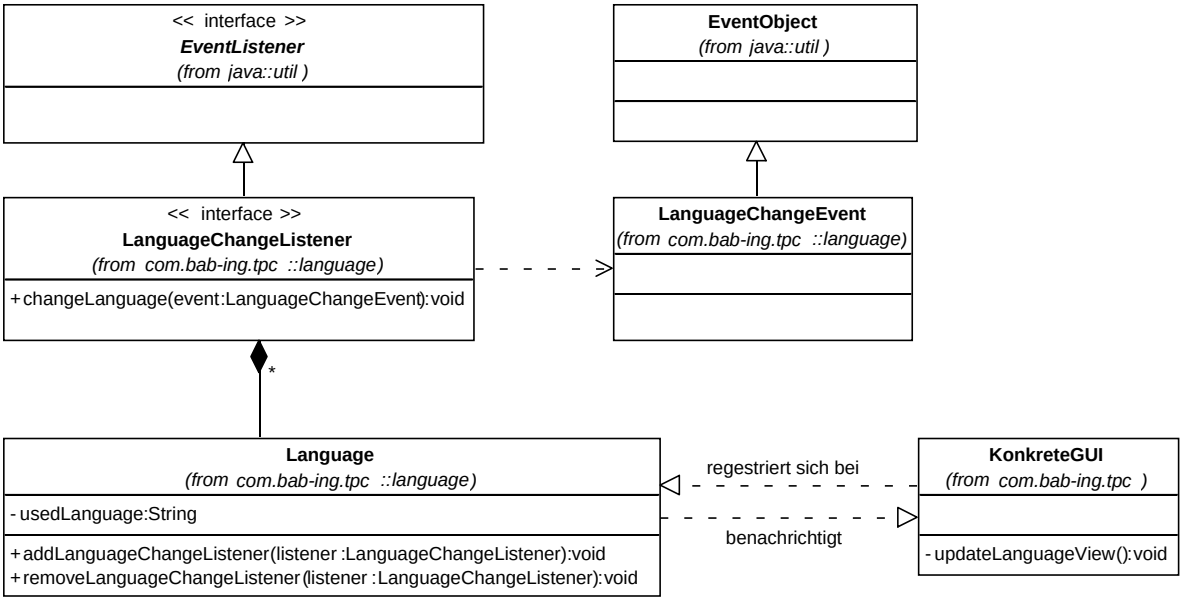


Abbildung 4-11 Klassendiagramm eines Observers für die Sprachumschaltung

Die Behandlung der geworfenen Events bleibt jeder GUI selber überlassen. Generell werden alle Label innerhalb der GUI neu gesetzt.
Die Sprachressourcen liegen dafür in den Properties-Dateien:

language	Standardsprachressource (Englisch)
language_en_GB. properties	Englische Sprachressourcen
language_de_DE. properties	Deutsche Sprachressourcen
language_it_IT. properties	Italienische Sprachressourcen

Durch die Forderung, dass das TPC leicht um Sprachen erweitert werden soll, ist es notwendig alle Sprachressourcen möglichst in einer Tabelle zu halten. Da dies ein Widerspruch zu dem vorgestellten Properties –Ansatz ist, wird ein zusätzlicher Mechanismus benötigt. Die Idee besteht darin, dass eine Tabelle generiert wird, in der alle Label aller Sprachen nebeneinander stehen. Durch ein Anwendertool könnten so aus der Exceltabelle die entsprechenden Properties-Dateien erzeugt werden.

Language Label	de_DE	en_GB
TITLE_TPC	Tunneling Process Control	Tunneling Process Control
LABEL_ADD	Hinzufügen	Add

Tabelle 4-2 Auszug aus der LanguageBundel Datei

4.5.2 DWH-Handler

Der Data-Warhouse-Handler soll die gewünschten Funktionen zum Integrieren und Analysieren bereitstellen.

4.5.3 MetaData-Handler

4.6 Zusammenfassung

In diesem Kapitel wurde das zugrunde liegende Konzept für ein DWS für Vortriebsdaten diskutiert. Der Schwerpunkt lag dabei auf der Erfassung der Anforderung an das TPC und die Integration der Vortriebsdaten in DWH.

5 Realisierung

Die besonderen Aspekte der Realisierung sowie genauer Details der Implementierung werden in diesem Kapitel erörtert. Dabei werden insbesondere spezielle Techniken und aufgetretene Probleme hervorgehoben. Ein Blick auf die Entwicklungsumgebung zeigt, mit welchen Tool die Arbeit umgesetzt wurde. Der Abschnitt 5.2 geht speziell auf die einzelnen Schichten des realisierten Softwaresystems ein und zeigt den Umfang der Implementierungsarbeiten.

5.1 Entwicklungsumgebung

- *Betriebssystem:*
  Linux Suse 9.3 Distribution
- *IDE:*
  Bei Netbeans 4.2 handelt es sich um eine sehr mächtige integrierte Entwicklungsumgebung für Java, die durch Plugins erweitert werden kann.
- *Versionskontrolle:*
  Eine Versionskontrolle wie zb. CSV wurde in dieser Arbeit nicht benutzt.
- *Unit-Tests:*
  JUnit ist ein Tool für den automatischen Unit-Test in Java. Damit lassen sich Fehler im Programmcode finden, die sich in die Implementierung oder in spätere Änderungen eingeschlichen haben.
- *Ant:*
  Ant ist das Standard Building-Tools für Java, vergleichbar mit dem Unix / Linux-Tool "make" jedoch nicht so mächtig.

5.2 Schichtenarchitektur von TPC

Abbildung 5-1 zeigt das TPC in einer 4 Schichtenarchitektur im Vergleich zur Abbildung 5-2 in der das TPC aus der Sicht eines DWS dargestellt wird. In der Abbildung 5-1 wurde bewusst die *Schicht 2* hinzugefügt um im Gegensatz zu der Abbildung 4-1 den Zugriff auf die Datenbank zu kapseln. Die einzelnen Schichten werden in den nächsten Kapiteln genauer, hinsichtlich ihrer Realisierung, untersucht. *Schicht 1* entspricht der Datenbankschicht. Die Schichten zwei und drei repräsentieren die Business-Logik. *Die Schicht 4* entspricht der TPC-Workbench-Schicht mit allen grafischen Benutzeroberflächen. Die einzelnen Schichten enthalten nicht nur

selbst entwickelte Komponenten. Die wichtigsten verwendeten Fremdkomponenten werden im Abschnitt 5.3 vorgestellt.

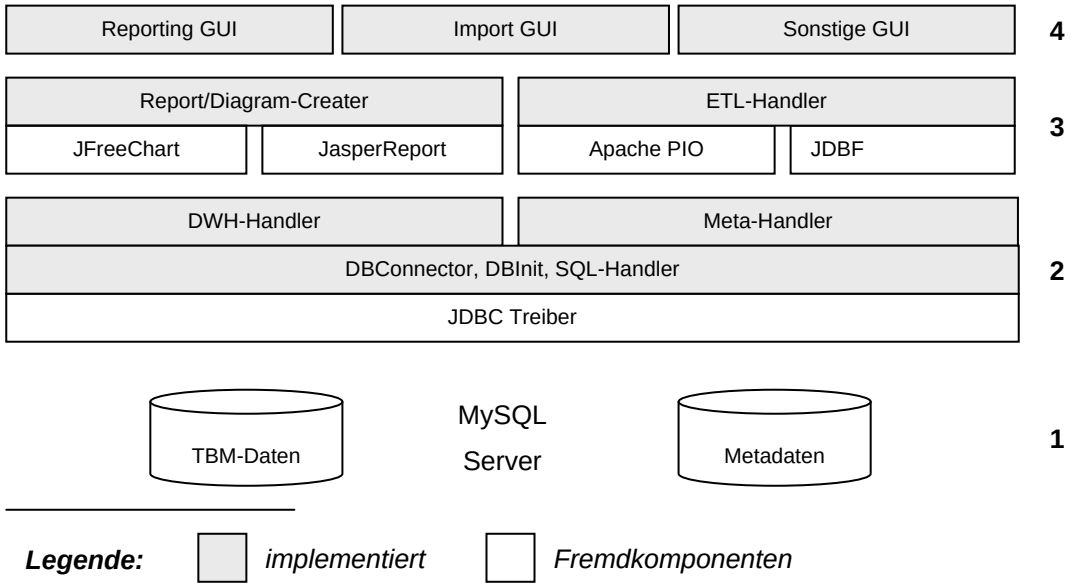


Abbildung 5-1 Schichtenarchitektur von TPC

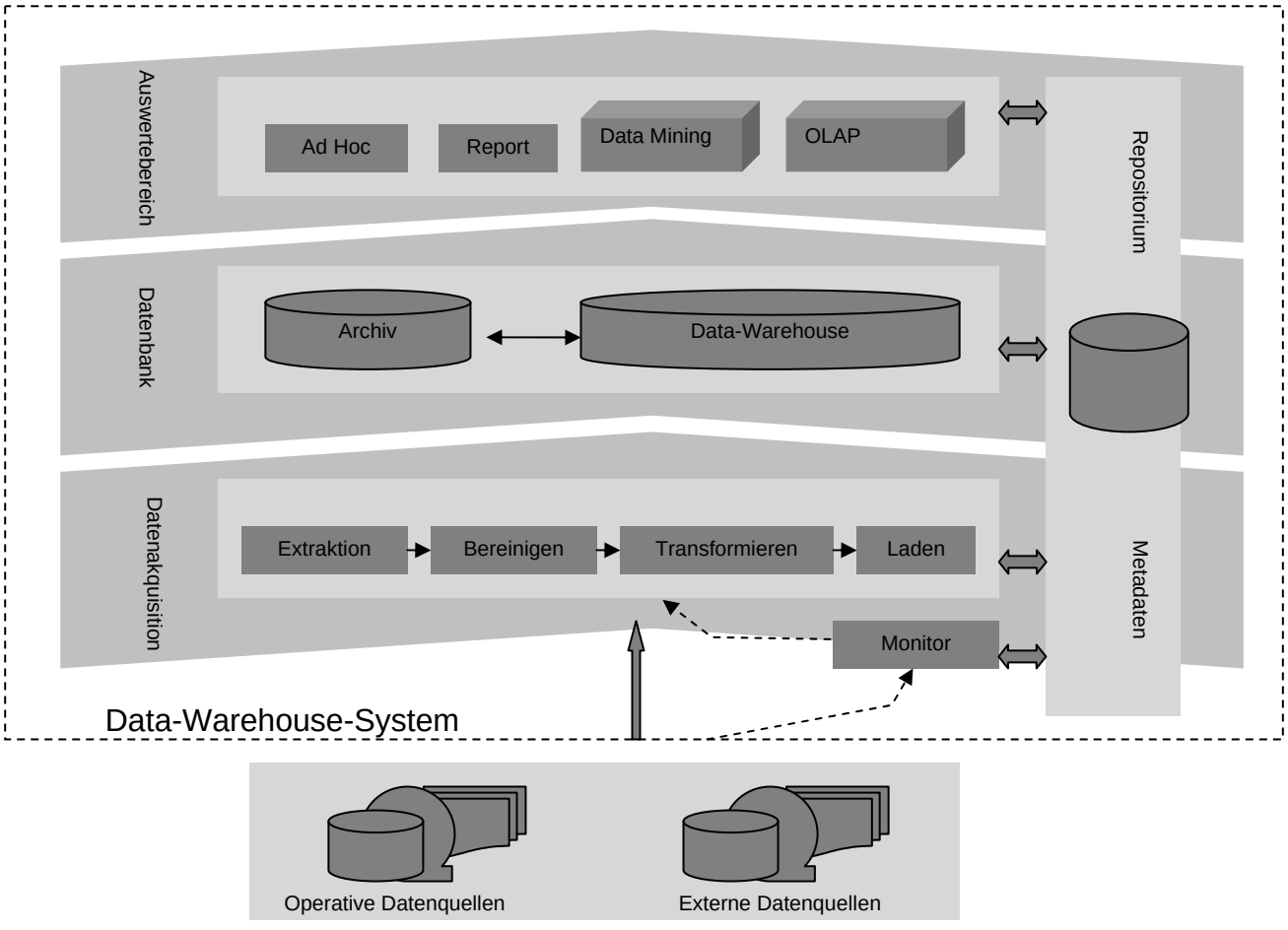


Abbildung 5-2 Architektur von TPC aus der Sicht eines Data-Warehouse-Systems

5.2.1 Datenbank

Das TPC wurde mit zwei Datenbanken in der Schicht 1 ausgestattet. Eine für das Repositorium und eine für das Data-Warehouse. Für beide Datenbank-Server gilt, dass sie robust, flexibel, schnell sein sowie mit großen Datenmengen umgehen müssen. Da in dieser Arbeit nur Open Source Datenbanken genutzt werden dürfen, fiel die Wahl auf MySQL. Dabei handelt es sich um einen sehr schnellen und robusten SQL (Structured Query Language) Datenbank-Server. Er wurde entwickelt, um viele große Datenbanken zu verwalten. Für Entwicklungszwecke eignet er sich besonders gut, da er für fast jedes Zielsystem vorhanden ist und über Standardschnittstellen abgefragt werden kann. MySQL erlaubt das Verwalten von mehreren verschiedenen Datenbanken nebeneinander. Damit muss nur eine DBMS aufgesetzt werden. Weitere Vorzüge sind Geschwindigkeit, Flexibilität und seine einfache Benutzbarkeit.

Für das TPC wurden zwei Datenbanken erstellt, „*TPC-DWH*“ für das Data-Warehouse und „*TPC-Repository*“ für Repositorium. Das in Abschnitt 4.2 vorgestellte Metadaten-Schemata wurde anschließend in die TPC-Repository-Datenbank integriert.

Das in Abschnitt 4.3.2 vorgestellte Galaxy-Schema wurde vor der Integration in die Datenbank dahingehend verändert, dass die Dimensionen Raum und Zeit als eigenständige Dimensionen gelöscht wurden. Sie enthalten keine weiteren Informationen die zur Analysezwecken benutzt werden können. Die Fakt-Tupel tragen selber die Information über Zeit und Raum in der feinsten Granularität. Somit werden diese Dimensionen nicht benötigt.

Um den unbefugten Zugriff auf die Datenbasis zu verhindern, werden ausschließlich Intranet-Anfragen bearbeitet. Programme oder die Personen, die direkt mit der TPC-MySQL-Datenbasis kommunizieren wollen, müssen sich im gleichen Subnetz befinden.

5.2.2 Datenbankverbindungsschicht

Eine der wichtigsten Anforderungen bestand darin, dass das TPC auf prinzipiell jedem Datenbanksystem aufgesetzt werden kann, um bestehende Datenbanken der zukünftigen Kunden benutzen zu können. Um diese Anforderung zu erfüllen, wurden die aktiven Funktionalitäten, z.B. Verbindungserstellung und Datenbankinitialisierung in einer weiteren Schicht zusammengefasst, organisiert und implementiert. Die

Kommunikation mit dem darunter liegenden Datenbanksystem erfolgt über die Schicht des JDBC.
Zur direkten Verarbeitung der Meta- und Vortriebsdaten existieren in dem zu implementierenden System die Klassen:

- *DWH-Handler:*
  Schnittstelle, um auf die Datenbasis des DWH schreibend und lesend zuzugreifen. Die Funktionalität wurde ich der Klasse *DWH-Handler* zusammengefasst.
- *Metadaten-Handler:*
  Schnittstelle, die einen schreibenden und lesenden Zugriff auf alle Metadaten erlaubt. Die Funktionalität wurde ich der Klasse *Metadaten-Handler* zusammengefasst.

Alle Funktionen der DWH und Metadaten-Handler sind *static* deklariert, damit für jede Instanz des aufgerufenen TPC-Systems nur ein Objekt, der Handler, existiert.

5.2.3 Business-Logik-Schicht

Business-Logik-Schicht ist die Schnittstelle zwischen der Datenbank- und Workbenchschicht. Folgende Funktionalitäten werden durch die Schicht der Business-Logik bereitgestellt:

- Vermittler zwischen GUI und der Datenbankschicht
- Regelt die internen Abläufe zum Integrieren von externen Datenquellen
- Regelt die internen Abläufe zum Erstellen beliebiger Reporte
- Auswertung der Vortriebsdaten durch die Komponenten *DiagramCreater* und *ReportCreater.* Eine genau Betrachtung der Analysekomponenten erfolgt in Kapitel 5.5
- Bereitstellen von internen Modellen für die grafischen Komponenten

5.2.4 Workbench-Schicht

Die Workbench-Schicht stellt dem Benutzer eine grafische Benutzeroberfläche zur Verfügung, die als Desktop für weitere interne Fenster dient. Um Oberflächen zu erstellen, bietet Java verschiedene Möglichkeiten. Die einfachste bieten das AWT (Abstract Window Toolkit) bzw. Swing. Während die AWT-Elemente so genannte Heavyweightkomponenten sind, die ihr eigenes Aussehen mitbringen, sind Swingkomponenten von der jeweiligen Plattform abhängig, lassen sich aber vielfältiger an-

passen. Swing ist das neuere und ausgereiftere Konzept. Die Entscheidung fiel daher zu Gunsten von Swing.

Die einzelnen Aufgaben der Workbenchschicht sind:

- Bereitstellen von Schnittstellen zwischen Benutzer und Business-Logik
- Container zum Andocken weiter Benutzerschnittstellen
- Laden von Offline-Reporten ohne Datenbankzugriff
- Umschalten der Sprachen zur Laufzeit
- Verwalten aller internen Fester
- Ausgabe von Status- und Fehlermeldungen

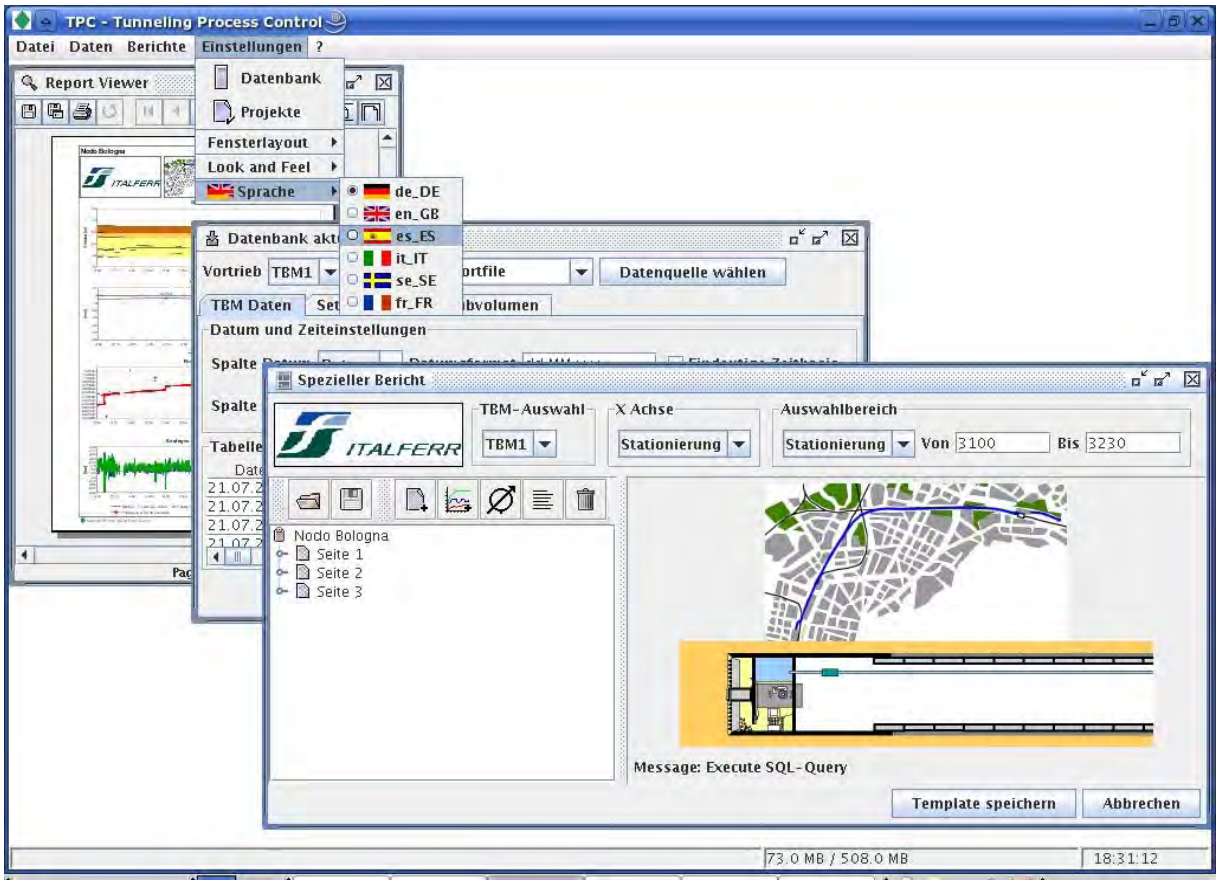


Abbildung 5-3 Screenshot von TPC-Workbench unter Suse 9.3

Alle einzelnen grafischen Komponenten, die innerhalb der Workbench-Schicht entwickelt wurden, leiten sich von einem MVC-Pattern ab.

Am Beispiel der Visualisierung eines Content-Templates wird die Implementierung gezeigt:

- *Model*

  ReportModel hält alle Informationen über ein Template zusammen und beschreibt damit die Struktur eines Reports.
- *View:*

  Zum Darstellen wird die JTree Komponente von Swing genutzt. Das eigentliche Rendern übernimmt die Klasse ReportModelRenderer.
- *Controller:*

  Implementierung mehrer Eventlistener innerhalb der Klasse SpecialReport, die Ereignisse auslösen, wenn z.B. auf die Button „Seite hinzufügen“ oder „Series löschen“ gedrückt wird.

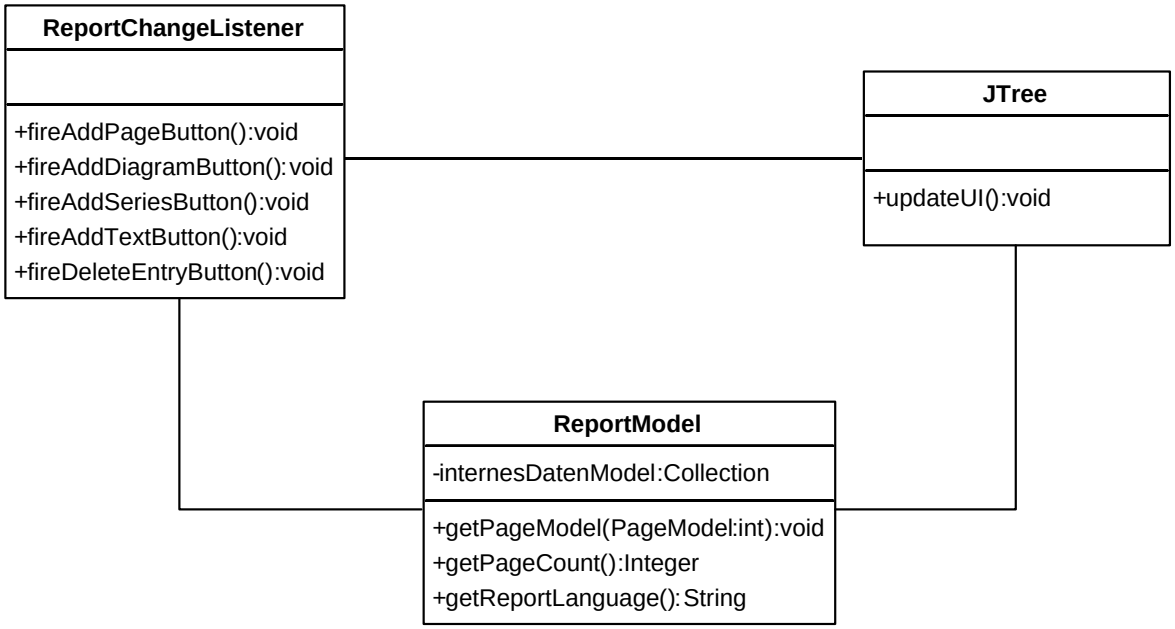


Abbildung 5-4 UML MVC-Pattern für die Visualisierung eines Reporttemplates

5.3 Fremdkomponenten von TPC

Die hohe Funktionalität des zu entwickelnden Softwaresystems konnte nur durch die Benutzung von Fremdkomponenten erreicht werden. Dieser Abschnitt stellt einige ausgewählte Fremdkomponenten vor und zeigt ihre Nutzung anhand von Implementierungsbeispielen.

5.3.1 JFreeChart

Bei JFreeChart handelt es sich um eine Open Source Java API zum Erstellen von Diagrammen, die offscreen oder auf ein Panel gezeichnet werden [JFCh05]. Das folgende Codebeispiel zeigt exemplarisch das Anlegen einer *XYSerie,* die die Datenbasis repräsentiert. *XYSeriesCollection* fasst diese XYSerien zusammen und stellt sie dem Diagramm zum Rändern zur Verfügung. Durch die Erzeugung einer Instanz vom Typ ChartPanel kann dieses Diagramm in jede JavaSwing-Oberfläche integriert werden.

```
XYSeries xyseries = new XYSeries("Stützdruck");
xyseries.add(1.0D, 500.19999D);
xyseries.add(5D, 694.1002D);
XYSeriesCollection xydataset =new XYSeriesCollection(xyseries));
JFreeChart jfreechart = ChartFactory.createXYLineChart("Support Preas-
ure", "[bar]", "Station [m]", xydataset, PlotOrientation.VERTICAL, true,
true, false);
ChartPanel chartpanel = new ChartPanel(jfreechart);
```

Quellcode 5-1 Beispielcode zum Erstellen eines JFreeDiagram

5.3.2 JasperReport

JasperReport ist eine in Java geschriebenes Open-Source Reporting Tool. Es ist in der Lage dynamische generierte grafische Erzeugnisse auf Monitor, Drucker oder in Dateiformate wie PDF, HTML, XML, XLS, CSV auszugeben. Es eignet sich ideal in Verbindung mit JfreeChart. Dabei kann es in Desktop Java Applikationen, Servlet- oder EJB-Container laufen. JasperReport benutzt für die das Layout eine eigene XML-Beschreibungssyntax. Die Daten bezieht es aus den verschiedensten Datenquellen wie relationale Datenbanken, Arrays oder Collections mit beliebigen Java Objekten.

```
<textField>
<reportElement positionType="Float" x="144" y="24" width="97"
height="24"/>
<textElement> <font size="12"/> </textElement>
<textFieldExpression class="java.lang.String">
<![CDATA[$F{coil_id}]]>
</textFieldExpression>
</textField>
```

```
<field name="coil id" class="java.lang.String"> </field>
```

```
<queryString>
<![CDATA[SELECT coil_id, …
FROM tabelle
WHERE coil_id = $P{ID}]]>
</queryString>
```

```
<staticText>
<reportElement positionType="Float" x="48" y="24" width="83"
height="24"/>
<textElement isStyledText="true">
<font fontName="SansSerif" size="12" isBold="true" isItalic="false"
pdfFontName="Helvetica-Bold" pdfEncoding="Cp1250" isPdfEmbedded="true"/>
</textElement>
<text><![CDATA[Statischer Text]]></text>
</staticText>
```

```
<parameter name="TBM-ID" class="java.lang.String">
<defaultValueExpression>
<![CDATA["TBM-1"]]>
</defaultValueExpression>
</parameter>
```

Quellcode 5-2 Auszug aus einem Reporttemplates

Die beiden unteren Code-Bruchstücken zeigen die Definition eines statischen und eines dynamischen Textfeldes. Beide Definitionen enthalten Angaben zur Position des Textfeldes im Report (innerhalb des <reportElement>), die Breite („width“) und Höhe („height“) und Angaben zur Textausgabe (innerhalb von <textElement> bzw. <font>), beispielsweise die Schriftgröße. Der auszugebende Text befindet sich in den Elementen <text> und <textFieldExpression> im Abschnitt „CDATA“: Als statischer Text wird „Statischer Text“ ausgegeben, als dynamischer Text der augenblickliche Wert des Felds $F{coil_id}, was in dem Beispielfall „TBM-2“ ist. Die Erstellung der beschriebenen XML-Datei kann mit einem einfachen Text-Editor erfolgen. Das wird in der Praxis in der Regel nicht getan. Ideal ist die Benutzung eines Text-Editors für das „Feintuning“, also beispielsweise für das Verschieben von Feldern und das Ändern von Fonts.

In dieser Arbeit wurde der iReport [IRep05] zum Erzeugen von JasperReport-Datei verwendet. Die fertige XML-Datei wird mit einem (zu JasperReports gehören-

den) Werkzeug in eine zur Laufzeit benötigte Berichtsdatei übersetzt. Diese Berichtsdatei wird zur Laufzeit erzeugt und mit Daten gefüllt. Das folgende Beispiel zeigt den Code, um Report mit Daten zu füllen und ihn als PDF-Datei auszugeben.

```
Map parameters = new HashMap();
parameters.put("TBM-ID","TBM-2");
Connection conn
  =DriverManager.getConnection("jdbc:mysql://BAB/TPC-DWH","usr","pass");
filledReportName =
JasperFillManager.fillReportToFile(compiledReportName,parameters,conn);
pdfReportName=JasperPrintManager.printReportToPdfFile(filledReportName);
```

Quellcode 5-3 Beispielcode zum Erstellen eines Reports

Zunächst wird eine Hashmap „parameters“ angelegt und dort der Parameter „ID“ eingetragen. Danach wird eine JDBC-Verbindung zur Datenbank aufgebaut, welche die Berichtsdaten enthält und zusammen mit dem Parameter an die Methode „fillReportToFile“ übergeben, welche die Daten in den Bericht einträgt und das Ergebnis in einer Datei im JasperReports- eigenen Format ablegt. Berichte in diesem Format können mit dem (zu JasperReports gehörenden) JasperViewer angeschaut werden. Abschließend wird der Bericht (mit der Methode „printReportToPdfFile“) in das PDF-Format gewandelt.

5.3.3 JavaDBF

JavaDBF ist eine Java API zum Lesen und Schreiben von XBase-Dateien. In der Version 0.4 wird JavaDBF in das TPC eingesetzt. Ein großer Nachteil ist, dass die Bibliothek kein JDBC-API anbietet, um die Daten im Sinne einer Datenquelle leicht anzusprechen. Ein weiterer gravierender Nachteil von JavaDBF ist, dass diese Bibliothek nicht „Thread sicher“ ist.

Nichtsdestoweniger musste diese Bibliothek eingesetzt werden, weil es keine Alternativen dazu gibt, außer selber solch eine Komponente zu entwickeln. Auf Grund von Zeitmangel wurde sich dagegen entschieden. Das folgende Codebeispiel kann genutzt werden, um alle Felder einer DBF-Datei auszulesen und in die Konsole zu schreiben.

```
import java.io.*;
import com.linuxense.javadbf.*;
public class JavaDBFReaderTest {
  public static void main( String args[]) {
    try{
      InputStream inputStream  = new FileInputStream( args[ 0]);
      DBFReader reader = new DBFReader( inputStream);
      int numberOfFields = reader.getFieldCount();
      for( int i=0; i<numberOfFields; i++) {
        DBFField field = reader.getField( i);
        System.out.println( field.getName());
      }
      Object []rowObjects;
      while( (rowObjects = reader.nextRecord()) != null) {
        for( int i=0; i<rowObjects.length; i++) {
          System.out.println( rowObjects[i]);
        }
      }
      inputStream.close();
    }
    catch( DBFException e) {
      System.out.println( e.getMessage());
    }
    catch( IOException e) {
      System.out.println( e.getMessage());
    }
  }
}
```

Quellcode 5-4 Beispiel zum Benutzen der JavaDBF-API

5.4 Templates

Im Folgenden werden die zwei Arten von Templates vorgestellt, die die Grundlage zur Erstellung von Reporten bilden. Dabei wird auf die Besonderheiten der Implementierung und Speicherung eingegangen. In der Abbildung 4-3 E/R-Diagramm der Metadaten wurden zwei Relationen vorgestellt, die zum Ablegen der Templates in die Datenbank genutzt werden.

5.4.1 View-Template

Zum Präsentieren von Reporten wurde ein möglichst einfaches und flexibles Reportingtool benötigt. Aus der Fülle der Open Source Reportingtool traf die Wahl auf JasperReport. Dieser ist zu 100 % Kompatibel zum JFreeChat-Framework. Eine ausführliche Beschreibung von JasperReport wurde in Abschnitt 5.3.2 gegeben.

Die Beschreibung des Templates erfolgt in einem XML-Format, das zur Laufzeit in ein JaperReport-Objekt kompiliert wird (siehe Quellcode 5-5).

```
JasperReport jr = JasperCompileManager.compileReport(report.jxml);
```

Quellcode 5-5 Kompilieren einer JasperReport Datei

Dabei wurde im Entwurf angedacht, dass der Inhalt des JasperReport-Datei in Textform im Repositorium gespeichert wird. Aufgrund der Kompilierung in ein JasperReport wird ein installiertes J2SE Development Kit (JDK) benötigt. Da dieses nur bedingt zu Verfügung steht, wird das kompilierte JasperReport-Objekt serialisiert und als Blob (Binaray Long Object) im Repositorium abgespeichert.

```
Meta-Handler.safeViewTemplate(us.getProject(),templates_name,jr);
```

Quellcode 5-6 Speicherung eines Reportobjekt im Repositorium

Ein weiterer wesentlicher Vorteil, das fertige Objekt im Repositorium zu speichern, ist der Geschwindigkeitsvorteil. Der Kompilierungsvorgang kann je nach Zielsystem bis zu einigen Sekunden dauern. Aufgrund der wiederholten Kompilierung kann dies keinem Benutzer zugemutet werden.

```
JasperReport jr =
Meta-Handler.getViewTemplate(us.getProject(),templates_name);
```

Quellcode 5-7 Auslesen eines Reportobjektes aus dem Repositorium

Das Auslesen des JasperReport übernimmt der Meta-Handler. Dazu existiert eine statische Methode, die das Objekt aus der Datenbank liest und in ein JasperReport-Objekt deserialisiert.

5.4.2 Content-Template

Für das Content-Template existieren zwei Ausprägungen. Die Idee bestand darin, zum einen ein Template zu schaffen, was die Struktur eines Reports repräsentiert und zum anderen auch die Information des fertigen Reports speichert. In der Abbildung 5-5 wird das UML-Klassendiagramm der softwaretechnischen Realisierung gezeigt. Den zentralen Punkt bietet die Klasse *ReportModel*. Diese Entity-Klasse beschreibt die globalen Eigenschaften eines Reports und bietet die Funktionalitäten zum Hinzufügen und Löschen von Seiten, Diagrammen, Textfeldern und Serien von Vortriebsdaten.

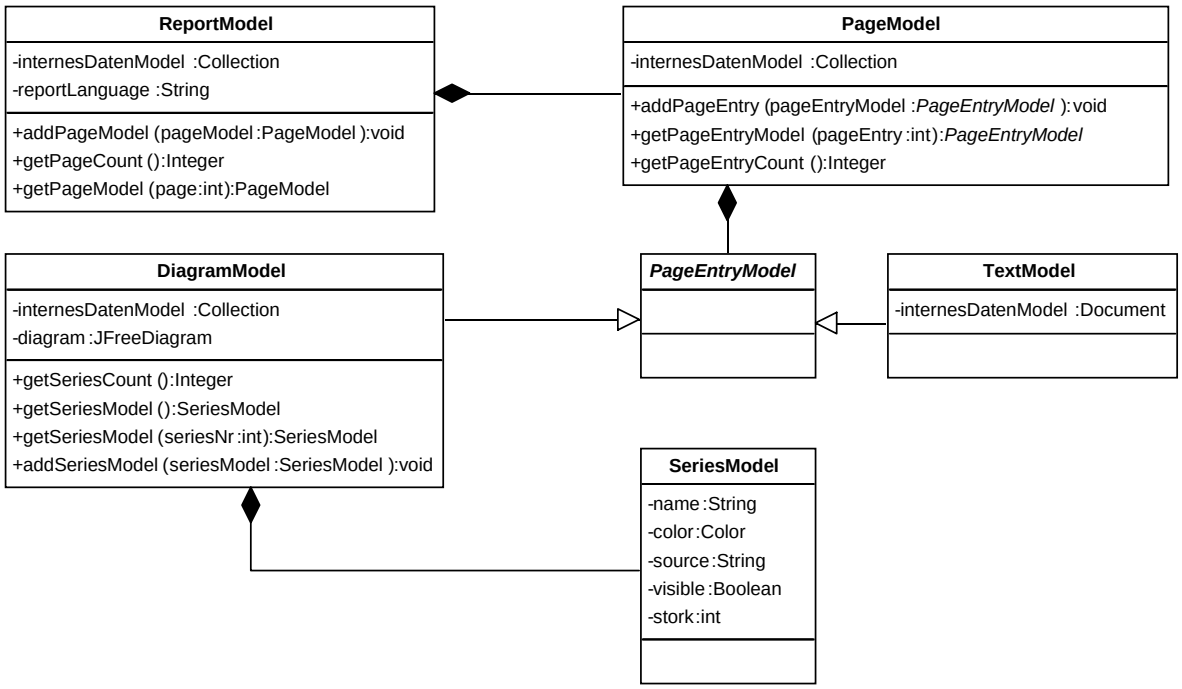


Abbildung 5-5 UML-Klassendiagramm Content-Templetes

Das Besondere an der Klasse *DiagramModel* ist, dass es eine Instanz der Klasse *JFreeChart* aufnehmen kann. Exsistieren in allen DiagramModel-Objekten keine Instanzen der Klasse JFreeChart, so fungiert das *ReportModel* als Template. Ansonsten repräsentiert das *ReportModel* einen fertigen Bericht, der alle Seiten, Diagramme mit deren Serien und Textfelder enthält. Um das Template bzw. den Bericht über die Laufzeit zu erhalten, werden diese serialisiert. Der folgende Quellcode verdeutlicht es am Beispiel der Berichtsabspeicherung.

```
GZIPOutputStream zipout=null;
zipout= new GZIPOutputStream( new FileOutputStream  (filename) );
ObjectOutputStream os = new ObjectOutputStream( zipout );
os.writeObject(reportModel);
os.close();
```

Quellcode 5-8 Serialisieren und komprimieren des ReportModel

5.5 Analyse und Berichterstellung

Für die Analyse der Vortriebsdaten gibt es die unterschiedlichsten Möglichkeiten, die in Kapitel 3.8 diskutiert wurden.

Beim Erstellen des Konzepts wurde sich auf das Nutzen des Open Source OLAP-Server *Mondrian* entschieden, da dieser derzeit der einzige Open Source OLAP-

Server ist. Wie in Abschnitt 5.2.1 schon diskutiert, werden die Dimensionen Zeit und Raum nicht benötigt, da die zugehörigen Werte höherer Hierarchielevel in diesem Fall direkt aus dem Fakt-Tupel ableitet werden können (z. B. Teilen des dort gespeicherten Fremdschlüssels „Millimeter“ der feinsten Ebene durch 1000, um den Rollup auf die Ebene "Meter" zu heben). Mondrian [Mond05] unterstützt in der derzeitigen Version ein solches "funktional" berechnetes Rollup nicht sondern, fordert eine Dimensionstabelle. Die Realisierung scheiterte jedoch kläglich an den feingranularen Dimensionen Raum und Zeit. Die feinste Ebene für die Dimension Raum ist Millimeter und für die Dimension Zeit Millisekunden. Es lässt sich unschwer erkennen, dass die Dimensionstabellen mit unnötig vielen Tupeln gefüllt werden müssen. Bei einer durchschnittlichen Tunnellänge von 1 Km werden 1 Million Tupel nur für die Dimension Raum gebraucht. Viel gravierender ist das Problem bei der Dimension Zeit, da Tunnelprojekte meist über mehre Jahre laufen. Für ein Jahr werden somit ≈30*10^9 Einträge in der Dimensionstabelle Zeit benötigt. Aus diesem Grund wurde sich für eine Hybridlösung entschieden. Konkret heißt es, dass das zu Grunde liegende Datenmodell weiterhin multidimensional ist. Zu einem späteren Zeitpunkt wird auf ein OLAP-Server zurückgegriffen, der die Möglichkeit der „funktional" berechnetes Rollup bietet. Die vollständige Implementierung der Analysefunktion wird durch eigene entwickelte Komponenten abgedeckt.

Bei den Analysekomponenten muss vorab unterschieden werden, in welcher Schicht sie sich befinden. Im Folgenden werden die Benutzeroberflächen und die dazugehörige Business-Logik vorgestellt. Die GUI-Klassen gehören der Workbenchschicht an. Alle weiteren Klassen sind Teil der Business-Logikschicht.

Um Analysen zu erstellen und diese in der Form von Reporten zu publizieren, wurde das Packaget *Report* implementiert.

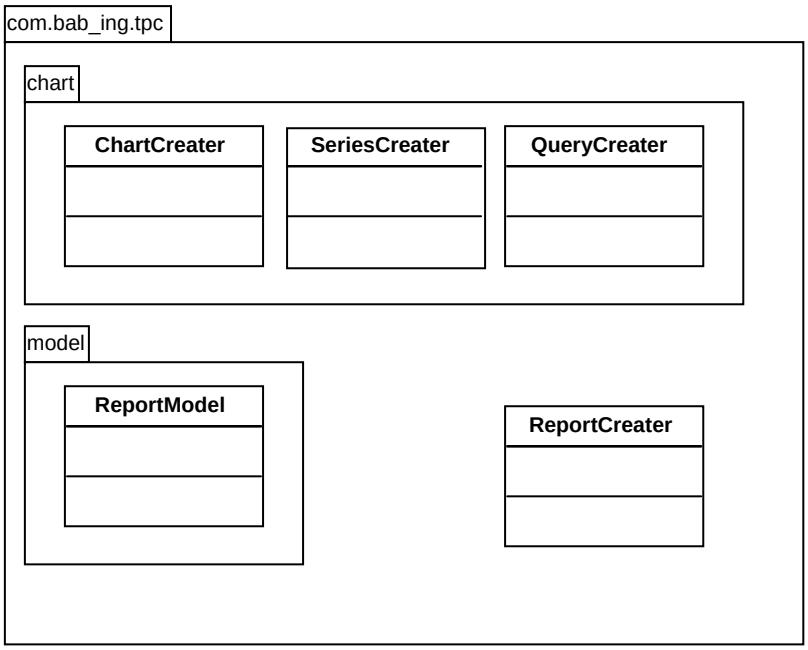


Abbildung 5-6 Report-Paketdiagramm

Die Abbildung 5-6 zeigt das Paket „Report“ dessen Unterpackete und seinen Klassen. Um die Übersichtlichkeit zu wahren, wurden nur Klassen dargestellt, die zum Verständnis der Funktionalität beitragen. Die Klassen der Benutzeroberflächen wurden ebenfalls weggelassen.

Für die Analysetätigkeit stellt das TPC-System Berichtserstellungskomponenten zur Verfügung, die die Funktion abbilden, die im Pflichtenheft beschrieben sind. Die Interaktion mit dem Benutzer erfolgt mit einer der vier GUI Klassen.

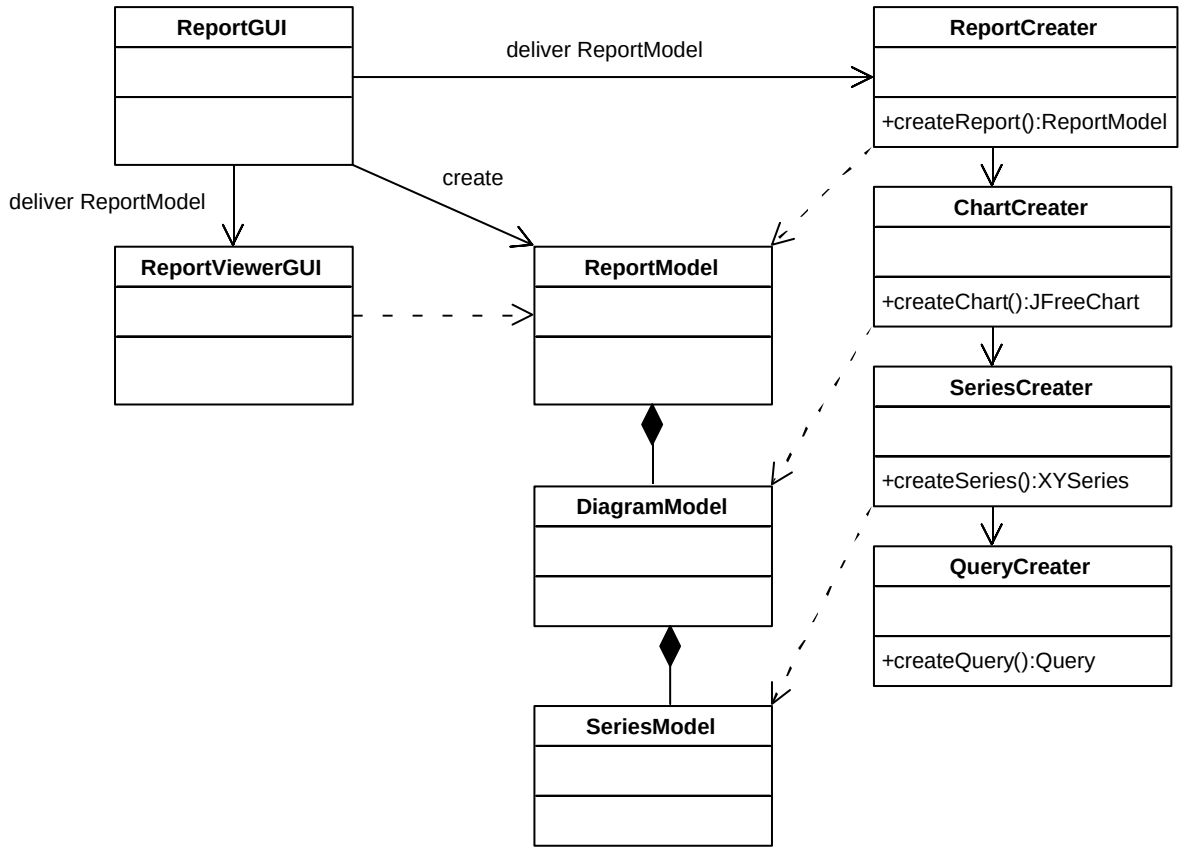


Abbildung 5-7 Klassendiagramm Reporterstellung

ReportGUI

Die Klasse *ReportGUI* in der Abbildung 5-7 steht stellvertretend für alle GUI-Klassen, die dem Benutzer Funktionalitäten zur Berichtserstellung bieten. Diese unterscheiden sich dahingehend, dass entweder Analysen über die Zeit *(WeeklyReportGUI)* oder über die Station *(SectionReport)* erzeugt werden. Bei *SpecialReportGUI* hat der Benutzer alle Freiheitsgrade, um ein Analyse durchzuführen. Die *SpecialReportGUI* kann des Weiteren auch zur Erstellung und zum Laden von Content-Templates genutzt werden.

ReportViewerGUI

Die Klasse stellt dem Benutzer eine ausgereifte Oberfläche zur Verfügung, um den Report zu drucken, zu speichern oder ihn speziell auf seine Bedürfnisse zu verändern. Diese Klasse ReportViewerGUI wird auch benutzt, um Offline-Reporte dazustellen. Wie ein Report erstellt wird, soll das folgende Codesegment verdeutlichen.

```
//Template holen
reportModel=ProjectHandler.getReportTemplate(projName,"WEEKLY_REPORT");
//ReporterCreater Instanze erzeugen
ReportCreater reportCreater = ReportCreater(reportModel);
//Erzeugung eines Reportes einleiten
reportCreater.createrReport();
//Fertigen Report zum Anzeigen aufbereiten
ReportViewerGUI rvGUI= RepotViewerGUI(reportCreater);
//Erzeugtes InternalFrame in das Hauptfenster hinzufügen
MainFrame.getInstance().addChild(rvGUI);
```

Quellcode 5-9 Berichtserstellung

Resultate

Die Abbildung 5-8 zeigt ein 100 Meter Abschnittsbericht der aus den Vortriebsdaten des Nodo Bologna Projekt erzeugt wurde. Die Abschnittsberichte die für das Nodo Bologna angefertigt wurden, unterscheiden sich jedoch in der Anzahl der Seiten und der Anordnung der Diagramme.

Im oberen Bereich sind das Projektlogo, der Lageplan und Information über die aktuelle Vortriebsposition hinterlegt. Das erste Diagramm enthält passend zum Abschnitt die Geologie. In Diagramm wird die Zeit über den Weg aufgetragen. Diagramm zeigt den Stützdruckverlauf von Sensor 1 und 2 in der Fiste. Zusätzlich ist im dritten Diagramm ein Triggerlevel integriert, der im Normalfall nicht unterschritten werden darf. Im letzten Viertel sind die Kommentare zu lesen, die durch einen Mitarbeiter geschrieben wurden.

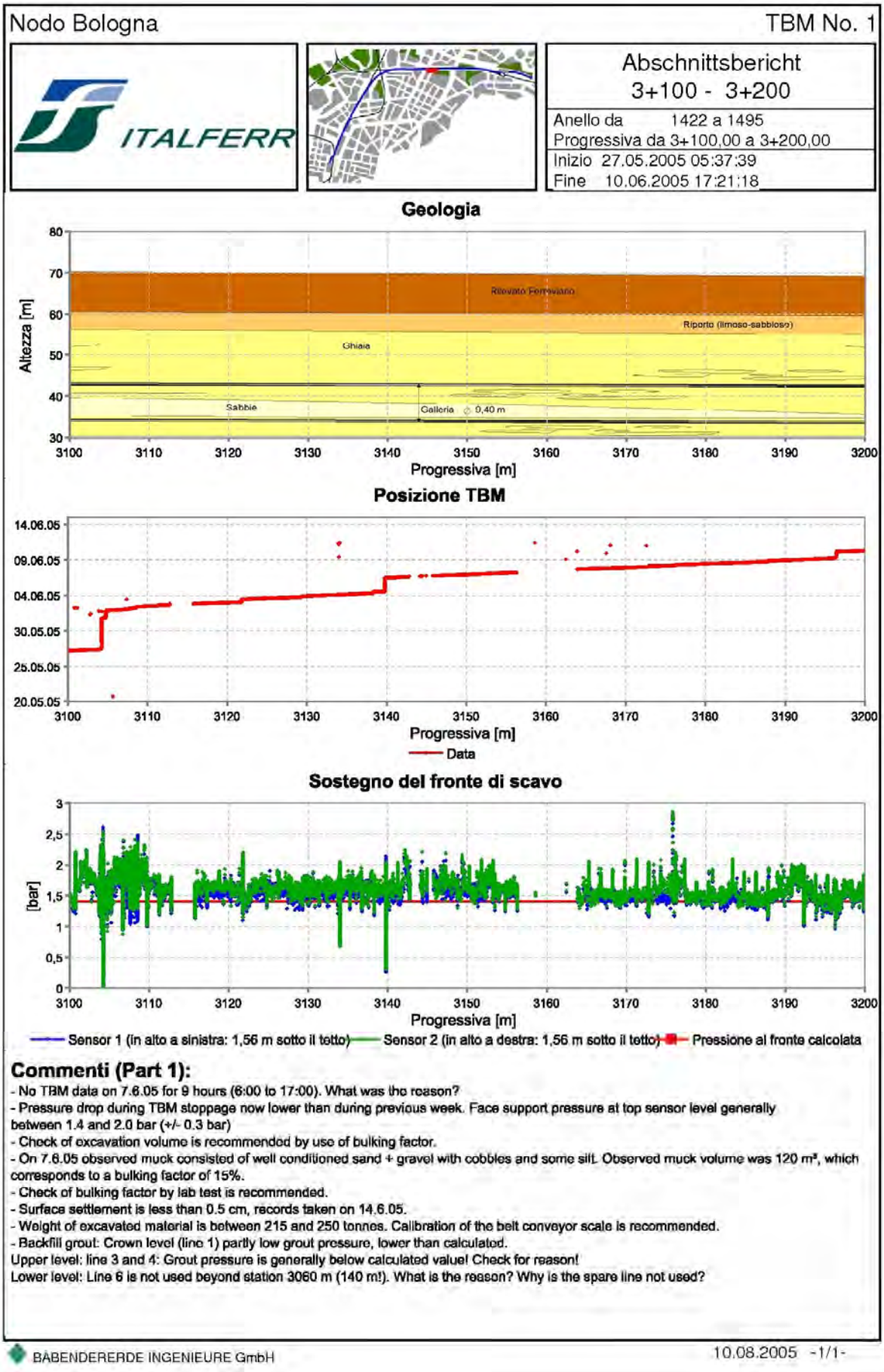


Abbildung 5-8 100 Meter Abschnittsbericht Nodo Bologna

5.6 Optimierung

Dieses Kapitel beschreibt einige Optimierungen, die an TPC vorgenommen wurden. Die Testfälle, die zu einer Optimierung führten, werden in Kapitel 6.2 näher betrachtet. Abschnitt 5.6.1 befasst sich speziell mit Problemen, die sich durch erhöhten Speicherbedarf auszeichnen. Abschnitt 5.6.2 geht dagegen speziell auf Optimierungen ein, die aufgrund von der Laufzeitgeschwindigkeit gemacht werden mussten.

5.6.1 Speicherbedarf

Der Performancetest ergab, dass bei der Datenauswertung der Speicherbedarf proportional zum ausgewählten Intervall und zur Anzahl der zu berücksichtigten Serien anstieg. Dieses war zwar zu erwarten, jedoch nicht in dem Umfang. Durch *Profiling* des betreffenden Abschnittes konnte das Problem erkannt werden. Der Quellcode 5-10 soll das Problem verdeutlichen.

```
public void add(Number x, Number y, boolean notify) {
     final XYDataItem item = new XYDataItem(x, y);
     add(item, notify);
}
public void add(double x, double y){
     add(new Double(x), new Double(y), true);
}
public void add(XYDataItem item, boolean notify) {
     this.data.add(item);
}
```

Quellcode 5-10 Auszug aus der Klasse org.jfree.data.xy.XYSeries

Bei der genauen Betrachtung lässt sich erkennen, dass pro Datenpaar 3 Objekte erzeugt werden müssen: Jeweils ein Objekt für den X und Y Wert des Datenpaares und ein XYDataItem, was in die XYSeriesCollection eingefügt wird. Eine Optimierung könnte dadurch erreicht werden, dass die Entitie-Klasse des XYDataItem dahingehend verändert wird, dass die Information in primitiven Datentypen gespeichert wird.

```
public class XYDataItem implements Cloneable, Comparable, Serializable {
    private double x; /** The x-value. */
    private double y; /** The y-value. */
    public XYDataItem(double x, double y) {
        this.x = x;
        this.y = y;
    }
    …
}
```

Quellcode 5-11 Ausschnitt der optimierten org.jfree.data.xy.XYDataItem Klasse

Der Quellcode 5-11 zeigt den Ausschnitt der optimierenden XYDataItem Klasse.

5.6.2 Geschwindigkeit

Das Laufzeitverhalten des TPC-Systems ist ein entscheidendes Verkaufsargument und wird daher genauer untersucht. Speziell der Bereich der Datenanalyse stellt an das TPC hohe Performanceansprüche

Das folgende Szenario zeigt eine Analyse, die auf der Grundlage eines *Section-Report-Template* basiert. Dargestellt werden 8 Indiktor die jeweils alle 10 Sekunden ein Messwert liefern. Bei einem 100m Abschnitt mit einer Vortriebsgeschwindigkeit von 100m pro Woche ergeben sich somit:

- 8.640 Datensätze pro Tag und Indikator
- 60.480 Datensätze pro Woche und Indikator
- 483.840 Datensätze pro Analyse

Die Berechnung zeigt, dass bei einem idealisierten Abschnittsbericht schon rund eine halbe Million Tupel aus der Datenbank selektiert werden müssen. In der Praxis sind Vortriebsgeschwindigkeit bis zu 100m pro Woche jedoch nicht die Regel. Aus diesem Sachverhalt lassen sich zwei Extremfälle für einen Abschnittsbericht ableiten.

Best Case

Die Vortriebsgeschwindigkeit entspricht der maximalen Vortriebsgeschwindigkeit. Da die Datenerfassung auf die Zeit basiert minimieren sich somit die Anzahl der Datensätze. Dieser Fall tritt nahezu nie ein, da der Vortrieb durch die unterschiedlichsten Faktoren unterbrochen wird z.B. Ringbau, Wartung und Hindernisbeseitigung.

Worst Case

Die Vortriebsgeschwindigkeit muss dazu gegen null gehen. Dieser Fall tritt immer dann ein, wenn unerwartete Hindernisse den Vortrieb behindern oder die Baustelle gesperrt ist.

Aufgrund von Nachverhandlungen stand der Vortrieb im Nodo Bologna Projekt über 10 Monate still. Da solche schweren Fälle auch analysiert werden müssen, wurde das TPC dahingehend optimiert. Die Folgenden Ansätze wurden dazu umgesetzt::

- Pro Indikator eine SQL Anfrage stellen
- Datensätze gruppieren um die doppelten Tupel zu eliminieren
- Aufwendige Joins vermeiden

Das Folgende Codesegment zeigt eine SQL Anfrage, um eine Auswertung des Stützdruckes über 100m zu erstellen.

```
SELECT space_station,value
FROM measurement
WHERE measurement.projekt = project_id AND
      measurement.tbm     = tbm_id
      BETWEEN measurement.space_station ‘2000’ AND ‘2100’
GROUP BY space_station,value;
```

Quellcode 5-12 SQL-Anfrage zur Vortriebsdatenauswertung

5.7 Zusammenfassung

Die Realisierung und prototypischen Implementierung des TPC-System wurde anhand von unterschiedlichsten Beispielen in diesem Kapitel gezeigt. Dabei wurde die einzelnen Schichten und ihre Komponenten vorgestellt.

Teil III Tests

6 Tests

Um eine abschließende Beurteilung ds TPC hinsichtlich der Funktionalität und Performance geben zu können, wurde das TPC einigen Tests unterzogen. Kapitel 6.1 enthält die Überprüft der allgemeinen Funktionalität und Robustheit gegenüber falschen Benutzereingaben. Kapitel 6.2 beschäftigt sich mit der Performance von TPC. Abschließend werden in Abschnitt 6.3 die Ergebnisse aller Tests diskutiert.

6.1 Test der allgemeinen Funktion

Um die allgemeine Funktion von TPC zu testen, wurden alle im Pflichtenheft beschriebenen Funktionen hinsichtlich ihrer Funktionalität und ihrer Fehlertoleranz untersucht.

Um die korrekte Integration der externen Datenquellen zu überprüfen, wurden stichprobenartige Überprüfungen in der Datenbank vorgenommen.

Anhand des Bologna Projekt wurde die Datenauswertung mit den Lösungen des Anwenderprogramms Origin [Orig05] verglichen.

Um das TPC nicht nur Anhand von gelieferten Ergebnissen zu überprüfen, verfügen einzelne Klassen über die Möglichkeit, in einen Debugmodus versetzt zu werden. In diesem Zustand liefern diese abhängig vom Debuglevel unterschiedlich ausführliche Statusinformationen, anhand derer die korrekte Arbeitsweise leicht überprüft werden kann.

6.2 Performancetest

Zusätzlich zu den Funktionstests wurde das TPC noch einem Performancetest unterzogen. Zu diesem Test gehörten eine Untersuchung des benötigten Speichers sowie eine Überprüfung der Arbeitsgeschwindigkeit. Speziell wurden hier die Integrations- und Analysefunktionalität untersucht.

6.3 Testergebnis

Alle durchgeführten Tests, die alle Funktionalitäten des TPC mit einbezogen hatten, zeigten, dass das Data-Warehouse-System entsprechend der Aufgabenstellung funktioniert. Das TPC entspricht allen gestellten Anforderungen und ist somit bereit für den direkten Einsatz zur Unterstützung von vortriebsbegleitenden Analysen im Tunnelbau.

Alle während der Tests auftretenden Fehler hatten nur geringe Ausmaße und konnten ohne größere Probleme behoben werden.
Die intensive Benutzung des TPC im Büro Babendererde GmbH zeigte, dass die Fehlermeldungen in einigen Situationen noch präziser sein könnten. In einigen Fällen konnten Fehler nicht reproduziert werden. Dadurch war es nicht möglich diese Fehler zu beheben. Situationen in denen dies jedoch im Einzelnen vorkommt, werden sich erst im Laufe der Benutzung herausstellen. Deshalb kann dieses Problem zu diesem Zeitpunkt noch nicht optimiert werden.

Teil IV Fazit

7 Zusammenfassung und Ausblick

Die Ergebnisse dieser Arbeit werden im Abschnitt 7.1 noch einmal zusammengefasst und Abschnitt 7.2 zeigt einen Ausblick über zukünftige Versionen von TPC.

7.1 Zusammenfassung

Das in dieser Diplomarbeit entworfene und realisierte TPC-System ermöglicht es, Vortriebsdaten aus den unterschiedlichsten Quellen zu integrieren, analysieren und die Ergebnisse in Form von Reporten zu präsentieren.

Der Einsatz eines Data-Warehouse-Systems eröffnet dabei eine Fülle von interessanten Möglichkeiten, bezogen auf die Integration, Analyse und Reporting von Vortriebsdaten. Die Idee, Vortriebsdaten multidimensional abzuspeichern, stellt sich als eine elegante Möglichkeit dar, um eine langfristige Speicherung und effektive Auswertung sicherzustellen. Um die steigenden Anforderungen an das TPC-System erfüllen zu können, wurde das TPC in mehreren unabhängigen Schichten implementiert. Das Austauschen einzelner Schichten erlaubt es eine projektspezifische Implementierung anzubieten. Gleichzeitig wird durch den Einsatz eines Repositoriums die Erweiterung von TPC um zusätzliche Integrations- und Analysetools erleichtert.

Der Einsatz von Templates, für das Erstellen und Publizieren von Reporten, wurde im TPC-System sehr flexibel gehalten. Benutzer haben damit die Möglichkeit, die Gestaltung der Reporte selber in die Hand zu nehmen und auf ihre Projektanforderungen anzupassen.

Abschließend kann gesagt werden, dass Data-Warehouse-Systeme sich durchaus auch im technischen Umfeld etablieren können. Speziell für das TPC gilt, dass nicht alle Möglichkeiten die ein DWS ausmachen, umgesetzt worden sind. Ein Hauptgrund lag darin, dass zum einen nur „freie“ Implementierungen eingesetzt werden durften und zum anderen, dass einige Feature den Rahmen dieser Diplomarbeit gesprengt hätten.

7.2 Ausblick

Das TPC setzt neue Maßstäbe im Hinblick auf die Auswertung von Vortriebsdaten. Durch eine konsequente Weiterentwicklung kann das TPC in der nahen Zukunft noch wesentlich an Bedeutung für den Tunnelbau gewinnen. Allerdings besitzt das TPC auch spezifische Eigenschaften, die auf der Entwicklungsseite neue Herausforderungen aufwerfen.

Es ist zu überlegen ob, in zukünftigen Versionen der Einsatz von professionellen OLAP-Servern erlaubt werden sollte, um eine Steigerung der Auswertegeschwindigkeit und die Analysefunktionalität zu erreichen.

Des Weiteren könnte eine noch striktere Trennung zwischen der Workbench- und Business-Logik Schicht erfolgen. In Form einer *3-Tier-Architecture* würde sich die Möglichkeit ergeben, die Business-Logik vollkommen auf einen *Application-Server* zu verlagern. Speziell die Java 2 Platform, Enterprise Edition (J2EE) 5.0 bieten neue Möglichkeiten für die Entwicklung eines Data-Warehouse. So wird unter anderem eine vollständige JOLAP API im J2EE 5.0 bereitgestellt.

Auch auf der Seite der Metadaten kann das TPC noch optimiert werden z.B. durch die Nutzung des Java Metadaten Interface (JMI).

Anhang

A Glossar

Das Glossar enthält die wichtigsten Begriffe dieser Arbeit, vor allem aus den Themenbereichen Data-Warehouse-Systeme, Datenbanken, Objektorentierung und maschineller Tunnelbau. Die Begriffe sind in alphabetischer Reihenfolge aufgeführt und haben folgende Notation:
→Verweis auf einen anderen Begriff im Glossar.

Analye
Prozess der interaktiven Untersuchung und Präsentation von →Vortriebsdaten.

Auswertebereich
Teil des →Data-Warehouse-System. Er besteht aus der →Basisdatenbank, dem →DWH und den Komponenten zur →Analyse.

Arbeitsbereich (engl. Staging area)
Meist temporäre Datenhaltungskomponente zur Unterstützung der Datenaktualisierung der →Basisdatenbank. Extrakte (→Extraktion) werden aus den →Datenqellen aufgenommen, im Arbeitsbereich transformiert (→Transformation) und in die →Basisdatenbank geladen [BaHo04].

Basisdatenbank
Physische Datenbank, die eine integrierte Sicht auf (beliebige) Daten darstellt. Die Basisdatenbank dient nicht ausnahmslos einem speziellen Analyseanspruch und unterliegt deshalb nicht einem spezifischen Modellierungsansatz. Eine Aktualisierung und Modifikation des Datenbestands ist möglich. Sie weist Ähnlichkeiten mit einer replizierten, →föderierten Datenbank auf. In Abhängigkeit vom Verwendungszweck kann eine Historisierung stattfinden. In der Literatur wird die Basisdatenbank gelegentlich als →DWH bezeichnet [BaHo04].

Data Mart
Ein Data Mart bietet eiene externe (Teil-)Sicht auf das →DWH, die i. d. R. durch Kopieren und somit durch die Einführung von Redundanzen erreicht wird. Der Data Mart ist anwendungsbereichspezifisch und weist häufig eine höhere Verdichtung auf als das →DWH. Der Data Mart ist optional und kann unter dem Begriff DWH subsumiert werden [BaHo04].

Data Mining

Suche nach unbekannten Mustern oder Beziehungen oder in Daten (Hypothesengenerierung).

Data-Warehouse-Manager

Verwaltungskomponente des →Data-Warehouse-Systems. Der Data-Warehouse-Manager steuert den →Data-Warehouse-Prozess.

Data-Warehouse-Prozess

Der Data-Warehouse-Prozess umfasst alle Schritte vom →Datenbeschaffungsprozess, die Speicherung bis hin zur →Analyse der Daten

Data-Warehouse

Physische Datenbank, die eine integrierte Sicht auf (beliebige) Daten darstellt. Im Unterschied zur →Basisdatenbank, steht der Auswertungsaspekt (analyseorientiertes Schema) im Mittelpunkt, der sich oft in einem →multidimensionalen Schema widerspiegelt. Häufig, aber nicht notwendigerweise, findet eine Historisierung der Daten statt, indem in periodischen Abständen Daten hinzugeladen, aber nicht modifiziert werden [BaHo04].

Data-Warehouse-System

Informationssystem, bestehend aus allen für den →Data-Warehouse-Prozess notwendigen Komponenten. Diese Komponenten sind der →Datenbeschaffungsbereich und die →Analyse, der →Metadatenmanager, der →Data-Warehouse-Manager und die Datenbank →Basisdatenbank, das →DWH und das →Repositorium. Anmerkung: Die Datenquellen gehören nicht zum Data-Warhouse-System [BaHo04].

Data-Warehouse-Zusatzinformationen

Den →Data-Warehouse-Prozess beschreibende Daten wie Regeln zur Transformation, Daten zur Sicherheit, zum Ablaufgeschehen usw. Die Data-Warehouse-Zusatzinformationen sind eine Teil der →Metadaten [BaHo04].

Data Warehousing

Synonym zu →Data-Warehouse-Prozess

Datenbeschaffungsbereich

Ist Teil des →Data-Warehouse-Systems. Er besteht aus dem →Arbeitsbereich, den Komponenten zum →Extrahieren, →Transformieren und →Laden sowie einem Monitor pro Quelle.

Datenbeschaffungsprozess

Der Vorgang, Daten aus Quellsystemen durch den →Datenbeschaffungsbereich in die →Basisdatenbank zu bringen, der sich in die Teilschritte →Extrahieren, →Transformieren und →Laden untergliedert.

Datenquellen

Daten- und →Metadatenlieferanten des →Data-Warehouse-Systems, die sowohl organisationsintern als auch extern sein können.

Detaildaten

Daten mit der feinsten verfügbaren →Granularität.

Dimension

Eine Dimension ist innerhalb des →multidimensionalen Datenmodells eine ausgewählte Entität, mit der eine Analysesicht eines Anwenderbereichs definiert wird. Dimensionen dienen der eindeutigen, orthogonalen Strukturierung des Datenraums.

Dimension des Würfels

Anzahl der →Dimensionen, die einen →Würfel aufspannen.

Dimensionselemente

Basisgranulare →Klassifikationsknoten in der →Klassifikationshierachie.

Extraktion

Selektion eines Teils der Daten aus den →Datenquellen, um sie der nachfolgenden →Transformation zur Verfügung zu stellen.

Geologie

Lehre von Aufbau und der Entwicklungsgeschichte der Erde.

Granularität

Stufe des Verdichtungsgrades der Daten im →Würfel. Dabei haben →Detaildaten den niedriegsten Verdichtungsgrad und zusammengesetzte Daten (z. B. bei Aggregationen) einen höheren Verdichtungsgrad.

Kenngrößen

Kenngrößen sind die Inhalte von →Würfeln. Diese sind im Rahmen der →Analyse oft quantitativ, da häufig aggregiert wird.

Klassifikationshierarchie

Vollständige Zerlegung der nichtleeren Menge in disjunkte Teilmengen nach Auswertungssichten. Sie bilden mittels einer Baumstruktur eine Abstraktionshierarchie über die →Dimensionselemente.

Klassifikationsknoten

Eine Verdichtungsstufe innerhalb einer →Klassifikationshierarchie, d. h. ein Knoten innerhalb der Baumstruktur.

Klassifikationsschema

Schema zur Abstraktion von einer oder mehreren →Klassifikationshierarchien, die einer →Dimension zugeordnet sind.

Klassifikationstuffe

Eine Verdichtungsstufe innerhalb des →Klassifikationsschemas.

Koordinaten im Würfel

Eindeutige Adresse innerhalb eines Würfels, ausgedrückt durch Tupel aus →Klassifikationsknoten. Es wird damit eine →Würfelzelle adressiert.

Laden

Letzter Schritt des →Datenbeschaffungsprozess, bei dem die Daten aus dem →Arbeitsbereich in die →Basisdatenbank oder in das →Data-Warehouse geladen werden. Häufig ist zur Optimierung damit auch eine Aggregation der Daten verbunden.

Metadaten

Gesamtheit aller Schemadaten und →Data-Warehouse-Zusatzinformationen.

Metadatenmanager

Synonym zu →Repositoriummanager

Monitor

Programm zur Beobachtung einer →Datenquelle, um die zu extrahierenden Daten zu bestimmen.

Multidimensionales Datenmodell

System von Strukturen und Operatoren zur Modellierung von →Dimensionen und →Klassifikationshierarchien innerhalb eines Analysekontextes.

Multidimensionales Schema

Beschreibung der Datenstruktur durch →Klassifikationsschemata und →Würfelschemata.

ODS (Operational Data Store)

Physische Datenbank, die eine partielle integrierte Sicht auf (beliebige) Daten darstellt. Das ODS grenzt sich von der →Basisdatenbank und dem →DWH durch die kurzfristige Datenhaltung und die oftmals nicht oder nur partiell durchgeführte →Transformation ab.

OLAP (Online Analytical Processing)

OLAP ist die explorative, interaktive →Analyse auf Grundlage des konzeptionellen →multidimensionalen Datenmodells.

Repositorium

Datenbank zum Speichern von →Metadaten

Repositorummanager

Verwaltungskomponente des →Repositoriums, die als Schnittstelle zum Repositorium fungiert.

Transformation

Anpassung der Daten an vorgegebene Qualitäts- und Schemaanforderungen.

Würfel

Mehrdimensionale Matrix , deren Zellen ein oder mehrere →Kenngrößenwerte enthalten (z. B. Umsatz, Erlös). Der Würfel wird durch die →Dimensionen (z. B. Pro-

dukt, Kunde, Zeit) als Achsen mit ihren jeweiligen Ausprägungen (eigentlich Datenquader) aufgespannt. Ein Würfel ist die Ausprägung eines →Würfelschemas.

Würfelschema

Das Schema eines Würfels wird durch →Dimensionen und →Kenngrößen bestimmt. Ein Würfel entsteht durch Instanziierungen eines →multidimensionalen Schemas.

Würfelzelle

Kleinster Teil eines Würfels, der durch die →Dimensionselemente adressiert werden kann.

Vortriebsdaten

Daten, die von den unterschiedlichsten Sensoren einer →TBM bei einem →Vortrieb aufgezeichnet wurden.

Setzungsdaten

Zusammenfassung aller Oberflächen und Tiefen→ Extensometermessungen.

Ortsbrust

Bezeichnet beim Untertagebau den Bereich, der sich am Ende eines →Bergbauortes bzw. Hohlraums befindet und an dem zurzeit ein →Ausbruch stattfindet.

Untertagebau

Erstellung unterirdischer Hohlräume (Tunnel, Stollen, Schachte, Kavernen usw.) im anstehenden Felsgestein in geschlossener Bauweise.

Tunnel

Eine künstliche Passage, die durch einen Berg, unter einem Gewässer oder einem anderen Hindernis hindurchführt.

Tübbing

Als Tübbing werden Bauteile einer Außenhülle eines Tunnels bezeichnet. Ein Tübbing ist ein vorgefertigtes Betonsegment.

Ring

In der gebräuchlichsten Form bilden sieben →Tübbingsegmente einen vollständigen Ring. Der Tunnel setzt sich dann aus einer Vielzahl von Ringen zusammen.

B Abkürzungsverzeichnis

3NF	Dritte Normalform
AFSS	Automatically Face Support System
API	Application Programming Interface
BIT	Business Intelligence Tool
DBMS	Datenbankmanagementsystem
DTD	Document Type Definition
DWH	Data-Warehouse
DWS	Data-Warehouse-System
E/R	Entity/Relationship
EJB	Enterprise Jave Beans
EPB	Earth Pressure Balance
ETL	Extraktion, Transformation, Laden
GUI	Graphic User Interface
HOLAP	Hybrid Online Analytical Processing
IDL	Interface Definition Language
IT	Informationstechnologie
J2EE	Java 2 Enterprise Edition
JDBC	Java Database Connectivity
JDK	Java Developer Kit
JRE	Java Runtime Enviroment
JSP	Java Server Pages
MDBMS	Multidimensionales Datenbankmanagementsystem
MDX	Multidimensional Expressions
ME/R	Multidimensionales Entity/ Relationship
MML	Multidimensional Modeling Language
MOLAP	Multidimensional Online Analytical Processing
MUML	Multidimensionale Unified Modeling Language
N/A	Not available
ODBC	Open Database Connectivity

ODS	Operational Data Store
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
OLTP	Online Transactional Processing
RDBMS	Relationales Datenbankmanagementsystem
ROLAP	Relational Online Analytical Processing
TBM	Tunnelbohrmaschiene
TPC	Tunneling Process Control
UML	Unified Modeling Language
XML	Extensible Markup Language

C Tabellenverzeichnis

D Abbildungsverzeichnis

E Literatur und Web-Referenzen

[AlCr02] Alur, D. , Crupi, J.: J2EE Patterns, Markt+Technik Verl. , München 2002.

[BaGu04] Bauer, A., Günzel, H.: Data Warehouse Systeme 2004.

[Balz99] Balzert, H. , Lehrbuch der Objektmodellierung, Spektrum Akademischer Verlag, 1999,

[CCS93] Codd, E. F., Codd, S., Salley, C.: Providing OLAP to User-Analysts: An IT Mandate, Codd & Associates, Ann Arbor/Michigan 1993

[DAUB97] Deutscher Ausschuss für unterirdisches Bauen (DAUB): Empfehlung zur Auswahl und Bewertung von Schildmaschinen.

[Engl01] Tunnelbau: http://www.biw.fh-deggendorf.de/alumni/2001/englmaier/tunnelbau/schild.htm

[Esse01] Esser, F. Java 2: Designmuster und Zertifizierungswissen, 2001.

[GaGl97] Gabriel, R., Gluchowski, P.: Semantische Modellierungstechniken für multidimensionale Datenstrukturen, HMD Heft 195

[GoMR98] Golfarelli, M., Maio, D., Rizzi, S.: Conceptual Design of Data Warehouses from E/R Schemes, Proceedings of the Hawaii International Conference On System Science, January 1998, http://www.csr.unibo.it/~golfare/

[Hinr02] Hinrichs, H.: Datenqualitätsmanagement in Data Warehouse-Systemen, Dissertation, Oldenburg 2002.

[Hinr04] Hinrichs, H.: Vorlesung Informationsintegration, Fachhochschule Lübeck SS2004

[Holt02] Holthuis, J.: Der Aufbau von Data Warehouse-Systemen, 2002.

[Horn05] Horn, T.: UML Unified Modelling Language, http://www. torsten-horn. de/techdocs/uml. htm, 2005.

[Hyper05] Hyperion News: JOLAP-Schnittstellen-Spezifikation vor finalem Entwurf http://www. hyperion. com/news_events/news_releases/press_release_1158.cfm

[ITS03] IT-Sicherheit auf dem Prüfstand: http://www.sigs.de/publications/os/2003/05/polster_OS_05_03.pdf,2003

[Inmo02] Inmon, W. H.: Bulding the Data Warehouse. Third Edition, New York: John Wiley & Sons, 2003

[IRep] IReport: Visual report builder for JasperReports
http://ireport.sourceforge.net

[Jdbf05] JavaDBF-API: Bibliothek zum lesen und schreiben von XBase-Dateien,
http://sarovar.org/projects/javadbf/

[JFCh05] Java Free Chart: Java API zum Zeichen von Diagrammen
http://www.jfree.org/jfreechart/index.php

[Kimb96a] Kimball, R.: The Data Warehouse Toolkit, John Wiley&Sons, 1996

[Kimb96b] Kimball, R.: Drilling Down, Up and Accross, DBMS Online 1996,
http://www.dbmsmag.com/9603d05.html

[MuRe01] Müller, G. , Reichbach, M.: Sicherheitskonzepte für das Internet, 2001

[MySQ04] MySQL Referenzhandbuch
http://dev. mysql. com/doc/mysql/de/index. html

[Meta99] N.N.: Data Warehouse Scorecard. Meta Group

[Mond05] Mondrian Open Source OLAP-Server
http://mondrian.sourceforge.net/

[MSRH01] Maidl, B., Schmid, L., Ritz, W., Herrenknecht, M.,:
Tunnelbohrmaschinen im Hartgestein, Bochum 2001

[Muell03] Müller, R. Fachhochschule Lübeck, Vorlesung SWT 2 WS03

[MuBe00] Mucksch, H., Behme, W.: Das Data Warehouse-Konzept: Architektur - Datenmodelle - Anwendungen; mit Erfahrungsberichten. 4. Auflage, Gabler, Wiesbaden 2000

[Lehn02] Lehner, W. : Datenbanktechnologie für Data-Warehouse-Systeme, Konzepte und Methoden, DPunkt, 2002

[LeTe96] Lehner W., Teschke M.: Modellierungsalternativen im "Scientific Computing", GI-Workshop "Multidimensionale Datenbanken", 4.März 1996, Ulm

[Lust02] Lusti, M.: Data Warehousing and Data Mining, 2002

[Oest98] Oestereich, B.: Objektorientierte Softwareentwicklung. Analyse und Design mit der Unfified Modeling Language

[Orig05] Grafik und Datenanalyse Software von Originlab,
http://www.originlab.com/

[Sach98] Sachdeva, S: Meta Data Architecture for Data Warehousing,
DMReview Magazine, April 98,
http://www.dmreview.com/article_sub.cfm?articleId=664

[SaSa04] Saake, G., Sattler, K. U.: Algorithmen und Datenstrukturen, 2004

[Stan99] N.N.: Migrates Headaches. Standish Group, 1999

[Stey01] Steyer, R.: Java 2 Kompendium, 2001

[Sun04] Sun Microsystems, Inc.
http://java. sun. com/j2se/1. 5. 0/docs/api/index. html, 2004

[Ulle04] Ullenboom, C.: Java ist auch eine Insel 4. Auflage, 2004

[VaGD99] Vavouras, A., Gatziu, S., Dittrich, K. R., SIRIUS (Supporting incremental refreshment of information warehouses) 8. Fachtagung BTW (Datenbanksysteme in Büro, Technik und Wissenschaft), März 1999

[Wiki05a] Wikipedia, Die freien Enzyklopädie, Multidimensional Expressions
http://de. wikipedia. org/wiki/MDX

[Wiki05b] Wikipedia, Die freien Enzyklopädie, JOLAP
http://de. wikipedia. org/wiki/JOLAP

[Wiki05c] Wikipedia, Die freien Enzyklopädie, Data-Warehouse
http://de. wikipedia. org/wiki/DWH

[Zeh03] Zeh, T. : Data Warehousing als Organisationskonzept des Datenmanagements. Eine kritische Betrachtung der Data-Warehouse-Definition von Inmon. In: Informatik, Forschung und Entwicklung, Band 18, Heft 1, Aug. 2003

F Quellcodeverzeichnis

G Begleit-CD

Auf der Begleit-CD befindet sich eine lauffähige Version des TPC-Systems mit einer Installationsanleitung. Zusätzlich ist der vollständige Quellcode der Klassen, die Dokumentation der API und dieser Arbeit als PDF Datei vorhanden. Des Weiteren befindet sich auf der CD die aktuellste Version der verwendeten Frameworks und der Dokumentation. Ein Java Runtime Enviroment und die aktuellste MySQL-Datenbank befinden sich ebenfalls auf der CD.