Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Jan Busch

Ein spatio-temporales Data Warehouse für multiskalige, heterogene ökologische Daten

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Jan Busch

Ein spatio-temporales Data Warehouse für multiskalige, heterogene ökologische Daten

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Master of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Thomas Thiel-Clemen
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Eingereicht am: 23. März 2015

Jan Busch

Thema der Arbeit
Ein spatio-temporales Data Warehouse für multiskalige, heterogene ökologische Daten

Stichworte
Data Warehouse, Distributed Facts, Datacube, Ökologische Daten, Multiagentensysteme, Agent based modelling, Georeferenzierung, Geoinformationssysteme, MARS

Kurzzusammenfassung
Die Datenmengen, welche mit der heutigen Technik erfasst und gespeichert werden können, sind eine große Hilfe im Bereich der ökologischen Forschung. Gleichzeitig wird das Management dieser Datenflut mehr und mehr zu einem Problem. Die Heterogenität von Daten spielt hier eine große Rolle, denn die Daten liegen nach Erfassung oft in verschiedenen Formaten, Skalierungen und Strukturen vor. Dies erschwert die Nutzung und Auswertung maßgeblich. Das in dieser Arbeit entwickelte Data Warehouse, bietet einen Ansatz die bestehenden Probleme zu lösen.

Jan Busch

Title of the paper
A spatio-temporal data warehouse for multiscale, hetergenous ecological data

Keywords
Data Warehouse, Distributed Facts, Datacube, Ecological Data, Multiagentensystems, Agent based modeling, Geocoding, Geographic information systems, MARS

Abstract
The massive amount of data, which can be recorded and stored nowadays, are a great support in the field of ecological research. At the same time the management of this data overflow becomes a problem to a greater extent. The problem of data heterogeneity plays a key role because the data is usually stored in different formats, scales and structures. This increases the complexity of using and analyzing the data for ecological research. This paper offers an approach to solve the existing problems using a data warehouse.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Quelltexte

1. Einführung

Die Datenmengen, welche mit der heutigen Technik erfasst und gespeichert werden können, sind eine große Hilfe im Bereich der ökologischen Forschung. Gleichzeitig wird das Management dieser Datenflut mehr und mehr zu einem Problem, für das bisher keine adäquate Lösung existiert (Thiel-Clemen, 2013) (Hochachka u. a., 2012). Viele der erfassten Daten kommen heute von Fernerkundungssensoren, Sensornetzwerken und einzelnen Feldmessungen. Hinzu kommen Satelliten, mit deren Hilfe verschiedenste Informationen erfasst werden können. Fasst man all diese Datenquellen zusammen, ergeben sich Datenmengen, deren Speicherung und Auswertung vor wenigen Jahren nicht möglich gewesen wäre. Heute ist ungenügender Speicherplatz kein Hindernis mehr, die Auswertung hingegen bedarf weiterer Forschung.

Die Bedeutung von Organisation und Planung bei der Erfassung von Daten wird meist unterschätzt. Dieser Mangel an Planung im Vorhinein führt oft dazu, dass innerhalb eines Projektes keine Struktur in der Erfassung eingehalten wird und die Daten in den unterschiedlichsten Formaten aufgezeichnet werden. Die Wege der Erfassung reichen hierbei von der manuellen Notation auf Papier oder der Eingabe in Tabellenkalkulationsprogramme bis hin zur automatischen Erfassung durch Sensornetzwerke, welche die Daten in einer Datenbank ablegen. (Michener und Jones, 2012)

Vor diesem Hintergrund entsteht das Problem der Heterogenität der Daten, denn die Daten liegen nach Erfassung oft in verschiedenen Formaten, Skalierungen und Strukturen vor. Dies erschwert die Nutzung und Auswertung beachtlich und ist neben dem Problem der Datenmenge das Hauptproblem beim Datenmanagement für Forschung in der Ökologie. Die Analogie zum Gebiet „Big Data“ aus dem Bereich der Business Intelligence (BI) ist naheliegend und lässt somit den logischen Schluss zu, Lösungsansätze aus diesem Bereich zu übernehmen und ggf. zu erweitern (McGuire u. a., 2008).

1.1. Motivation

Die Forschungsgruppe Multi-Agent Research & Simulation (MARS[1]) der HAW Hamburg[2] beschäftigt sich mit der Entwicklung eines Frameworks zur Multiagentensimulation aus interdisziplinären Projekten (Hüning u. a., 2014). Die vorliegende Arbeit ist Teil dieser Forschung und interagiert mit einer Reihe von Komponenten des Frameworks.

Im Rahmen der Entwicklung und Erprobung von MARS entstand die Dissertation von Hodabalo Pereki (Hodabalo Pereki, 2013), in welcher die Waldbiomasse im Abdolaye Wildlife Resort im Togo (AWR) untersucht wurde. Das AWR gehört zum so genannten Dahomey Gap (Vollmert u. a., 2003). Neben den klimatischen Veränderungen unserer Zeit wurden auch die Einflüsse durch den Menschen auf die Entwicklung der Waldbiomasse im AWR in die Untersuchungen miteinbezogen. Hierzu wurden Feldmessungen vor Ort durchgeführt, welche den Baumbestand, die Vegetation und weitere Informationen zur korrekten Ermittlung der Waldbiomasse einschließen. (Pereki u. a., 2013). Klimadaten, Höhenkarten und weitere Informationen wurden aus unterschiedlichen freien Quellen bezogen. (Baldowski, 2014) (Baldowski u. a., 2014)

Neben dem Projekt des AWR ist das interdisziplinäre Forschungsprojekt Adaptive Resilience of Southern African Ecosystems (ARS AfricaE[3]) der erste größere Anwendungsfall für das MARS Framework und dient ebenfalls als Anwendungsfall für diese Arbeit. Das Projekt ARS AfricaE beschäftigt sich mit der Landnutzung, dem Klimawandel und den daraus resultierenden Folgen für das südafrikanische semiaride Ökosystem. Die Betrachtung der Resilienz des Ökosystems im Untersuchungsgebiet steht im Vordergrund der Forschung.

Diese Arbeit nutzt die Dissertation von Hodabalo Pereki und das Projekt ARS AfricaE als Anwendungsfälle für das Konzept zur Verwaltung und Auswertung multiskaliger, hetergoener ökologischer Daten.

Aus den Anforderungen der Projekte AWR und ARS AfricaE an das Datenmanagement innerhalb des MARS Frameworks können direkte Anforderungen an ein System zur Verwaltung multiskaliger, heterogener ökologischer Daten abgeleitet werden. Durch die Anforderungen des AWR Projektes ist bereits ein effizientes Geoinformationssystem (GIS) im Rahmen der Masterarbeit von Mariusz Baldowski (Baldowski, 2014) entstanden. Dieses System wird im Kontext von MARS als Ground bezeichnet. In diesem System werden alle anfallenden Geodaten abgelegt, jedoch eignet sich dieses

[1] http://www.mars-group.org/
[2] http://www.haw-hamburg.de/
[3] http://www.ars-africae.org/

System nur in geringem Maße zur Speicherung von tabellarischen Daten. Hierfür ist ein Data Warehouse (DWH) vorgesehen, welches das Ergebnis dieser Arbeit ist. Die Verknüpfung der genannten Systeme bietet einen Weg das Problem der Speicherung sowohl für tabellarische als auch für Geodaten zu lösen. So werden Geodaten im GIS und tabellarische Daten im DWH abgelegt. Es muss ein Weg gefunden werden wie sich Daten aus beiden Systemen abrufen und aggregieren lassen. Dies schlägt sich auch in den Zielen dieser Arbeit nieder. Ferner ist zu evaluieren, ob sich die eingangs genannten Probleme der Datenheterogenität und Menge der Daten mit einem DWH lösen lassen.

1.2. Zielsetzung der Arbeit

In dieser Arbeit soll beleuchtet werden, wie sich ein DWH innerhalb des MARS-Frameworks einsetzen lässt, um die anfallenden ökologischen Daten der genannten und zukünftigen Projekte einbinden zu können. Es soll eine Möglichkeit geschaffen werden das bestehende GIS und das DWH zu verknüpfen. Eine Realisierung des genannten DWH sowie Experimente samt Durchführung des Anwendungsfalls ARS AfricaE gehören zu den Zielen dieser Arbeit. Es ist von besonderer Bedeutung, der Heterogenität und Multiskaligkeit der Daten eine Lösung entgegen zu setzen, welche aktuell vorhandene Probleme in der Analyse von Daten aus der ökologischen Forschung löst.

Die Basis für das DWH und die Integration in das MARS-Framework bilden die folgenden Hypothesen, welche im Verlauf dieser Arbeit geprüft werden sollen:

Hypothese 1
Die Probleme der Heterogenität und Multiskaligkeit von ökologische Daten lassen sich mit Hilfe eines DWH lösen.

Hypothese 2
Die Verknüpfung eines GIS mit einem DWH erleichtert den Umgang mit GIS-Daten und tabellarischen Daten innerhalb der ökologischen Forschung.

Hypothese 3
Das aus Hypothese 2 entstehende Problem der Referenzierung der Daten aus den unterschiedlichen Systemen lässt sich mit einem zu erforschenden Referenizerungssystem effizient lösen.

Hypothese 4
Mit Hilfe eines geeigneten Datenbankmanagementsystems lässt sich ein DWH skalieren, um den wachsenden Anforderungen durch höhere Datenmengen und parallelen Zugriffen gerecht zu werden.

1.3. Abgrenzung

Obwohl die Verwendung eines DWH zur Speicherung und Abfrage multiskaliger, heterogener ökologischer Daten der Kern dieser Arbeit ist, ist die Integration und Exploration der Daten nicht Bestandteil dieser Arbeit. Das Thema Integration/Exploration von Daten aus unterschiedlichen Datenquellen ist hinreichend komplex, um damit gesonderte Forschungen zu betreiben. Es wird im Rahmen des Projektes MARS bereits an solchen Werkzeugen geforscht. Die Projekte MARS Phobos zur Datenintegration und MARS Deimos zur Datenexploration/-extraktion sollen in Zukunft diesen Problembereich abdecken.

Parallel zu dieser Arbeit werden diverse andere Systeme im MARS-Kontext entwickelt. Das bereits genannte GIS als effizientes Werkzeug zum Geodatenmanagement sowie MARS LIFE zur verteilten Ausführung von Multi-Agenten-Simulationen sind nur einige Beispiele. Die verschiedenen Komponenten werden von anderen Mitgliedern der Forschungsgruppe beleuchtet und in Abstimmung mit der gesamten Gruppe entwickelt. Diese sind jedoch von dieser Arbeit durch den jeweiligen Anwendungsbereich abzugrenzen.

Die Bezeichnung des spatio-temporalen DWH für multiskalige, heterogene ökologische Daten innerhalb des Frameworks lautet MARS Rock. Es gliedert sich in die Entwicklungsumgebung ein, welche der Forschung mit MARS zur Verfügung gestellt wird. Von den zu MARS gehörenden Werkzeugen für Datenintegration/-exploration über die Datenhaltung bis hin zur Erstellung von komplexen Szenarien und Durchführung von verteilten Multi-Agenten-Simulationen, ist diese Arbeit der Datenhaltung zuzuordnen.

1.4. Aufbau der Arbeit

Beginnend mit Kapitel 2 (S. 6) werden grundlegende Konzepte und Fachbegriffe aus dem Bereich Datenbanken, Ökologie und Multi-Agenten Systemen erläutert. In Abschnitt 2.1 (S. 6) dieses Kapitels wird der Bereich des Ecological Modelling (EM) erläutert und dazu passend im folgenden Abschnitt 2.1.1 (S. 7) der Begriff der ökologischen Daten definiert. Darauf folgen weitere Erläuterungen zu agentenbasierten Modellen in 2.1.2 (S. 7) und zum Thema Multiagentensysteme in 2.2 (S. 8). Der Bereich der GIS in 2.3 (S. 9) wird auf Grund des Zusammenspiels dieser Arbeit mit der Masterthesis von Baldowski (2014) einführend behandelt. Der Abschnitt 2.5 (S. 12) erklärt die Grundlagen eines DWH, welche im Kontext dieser Arbeit besonders hervorzuheben sind.

In Kapitel 3 (S. 18) wird der aktuelle Stand der Forschung im Bereich der Verwaltung ökologischer Daten dargestellt. Das Kapitel gibt einen Einblick in den Ablauf und die Probleme, welche bei der Datenerfassung und Integration entstehen. Die Analyse und Verschneidung der Daten unterschiedlicher Forschungsgruppen sollen hier dargestellt werden. Ansätze und Probleme, die sich aus der Erfassung und Weiterverarbeitung ergeben, werden aufgezeigt und erläutert.

In Kapitel 4 (S. 22) werden Beispielszenarien für den Einsatz eines DWH für multiskalige, heterogene ökologische Daten beschrieben. Die Szenarien werden analysiert und Anforderungen auf deren Basis definiert. Diese Anforderungen spiegeln sich in der Architektur wider.

Kapitel 5 (S. 28) beschreibt die Komponente MARS Rock, die Teil des MARS-Frameworks ist und aus dieser Arbeit hervorgeht. Das Ergebnis der im Kapitel vorgestellten Architektur bildet die Basis für die Realisierung von MARS Rock in Kapitel 6 (S. 49). Innerhalb dieses Kapitels werden neben Rock weitere wichtige Komponenten des Frameworks erläutert. Des Weiteren werden die verwendeten Werkzeuge aufgeführt und erläutert.

In Kapitel 6 (S. 49) wird die Realisierung von MARS Rock und der zugehörigen API Drill beschrieben, wie sie sich aus den Anforderungen aus Kapitel 4 (S. 22) und der Architektur aus Kapitel 5 (S. 28) ergeben. Die Infrastruktur, auf dessen Basis die verteilte Datenbank realisiert ist, wird näher erläutert. Die API Drill wird vorgestellt und anhand von Quellcodeauszügen eingehend erläutert.

Kapitel 7 (S. 62) führt die Ergebnisse diverser Leistungsmessungen verschiedener Komponenten auf, welche in Kapitel 5 (S. 28) und Kapitel 6 (S. 49) beschrieben sind. Weiterführend wird anhand des Anwendungsfalls von ARS AfricaE ein praktisches Beispiel für die Verwendung von MARS Rock gegeben.

Abschließend diskutiert Kapitel 8 (S. 72) die Ergebnisse dieser Arbeit vor dem Hintergrund der in Kapitel 1 (S. 1) aufgestellten Hypothesen. Es werden verschiedene Probleme der vorliegenden Lösung erläutert und ein Ausblick für mögliche Weiterentwicklungen des Systems gegeben.

2. Grundlagen

Im folgenden Kapitel werden grundlegende Konzepte und Fachbegriffe aus dem Bereich Datenbanken, Ökologie und Multi-Agenten Systemen erläutert. In Abschnitt 2.1 (S. 6) dieses Kapitels wird der Bereich des Ecological Modelling (EM) erläutert und dazu passend im folgenden Abschnitt 2.1.1 (S. 7) der Begriff der ökologischen Daten definiert. Darauf folgen weitere Erläuterungen zu agentenbasierten Modellen in 2.1.2 (S. 7) und zum Thema Multiagentensysteme in 2.2 (S. 8). Der Bereich der GIS in 2.3 (S. 9) wird auf Grund des Zusammenspiels dieser Arbeit mit der Masterthesis von Baldowski (2014) einführend behandelt. Der Abschnitt 2.5 (S. 12) erklärt die Grundlagen eines DWH, welche im Kontext dieser Arbeit besonders hervorzuheben sind.

2.1. Ecological Modelling

Der Bereich Ecological Modelling (EM) liefert eine abstrakte Repräsentation eines ökologischen Systems, auf dessen Basis Analysen und Simulationen erstellt werden können. Mit ihrer Hilfe können Erkenntnisse über ein reales ökologisches System gewonnen werden. Die Granularität der Abstraktion reicht hierbei vom einzelnen Individuum über eine Gruppe verschiedener Spezies bis hin zu globalen Zusammenhängen. Modelle eines Ökosystems werden auf Basis von ökologischen Daten (Sonneneinstrahlung im Untersuchungsgebiet) in Kombination mit bekannten Zusammenhängen (Abhängigkeit der Photosyntheserate von der zur Verfügung stehenden Lichtmenge) erstellt. Diese Modelle werden genutzt, um neue Zusammenhänge aufzudecken oder große Experimente durchzuführen, welche sich in einem realen Ökosystem nicht durchführen lassen. Es ist des Weiteren möglich, ökologische Prozesse, welche sich normalerweise über große Zeiträume abspielen, in deutlich kürzerer Zeit zu simulieren. (Hall, Charles A. S und Day, 1977)

Im EM existieren verschiedene Modelltypen, welche für unterschiedliche Arten von Problemen verwendet werden. So unterscheidet man hierbei zwischen analytischen Modellen und Simulations- bzw. Berechnungsmodellen. Analytische Modelle beschreiben meist lineare Systeme und lassen sich mit Hilfe mathematischer Formeln exakt beschreiben, da das Systemverhalten bekannt ist. Die Gruppe der Simulationsmodelle nähert sich der Lösung mit numerischer Mathematik, da das Systemverhalten nicht

genau bekannt ist und somit nicht mathematisch exakt beschrieben werden kann. Hier ist die Komplexität oft deutlich höher als bei einem analytischen Modell. (Grant und Swannack, 2008)

Die Gruppe der Simulations-/Berechnungsmodelle sind im Kontext dieser Arbeit hervorzuheben, da die Simulationskomponente von MARS auf solchen basiert.

2.1.1. Ökologische Daten

Zur sinnvollen Analyse und Simulation eines ökologischen Modells werden nicht nur Kenntnisse über das Systemverhalten benötigt, sondern auch Daten über Individuen und die Umwelt innerhalb des zu simulierenden Systems. Sie werden in diesem Zusammenhang als ökologische Daten (ÖD) bezeichnet. Qualität und Quantität solcher Daten haben entscheidenden Einfluss auf die Resultate einer Simulation und deren Aussagekraft. Zur Gruppe der ÖD gehören beispielsweise Wetter- und Klimadaten, Daten aus Feldmessungen oder auch Bevölkerungsstatistiken.

Die Erfassung und Speicherung dieser Daten ist nicht vereinheitlicht und entspricht häufig den jeweiligen Anforderungen einzelner Forscher und deren Projekte. Es existiert jedoch speziell zum Zweck der Wiederverwertbarkeit von Daten die Ecological Metadata Language (EML). Mit ihrer Hilfe ist es möglich, die gesammelten ökologischen Daten zu beschreiben und somit Metadaten zu generieren, welche eine zeitliche und räumliche Verortung der Daten ermöglichen. (Fegraus u. a., 2005) In den meisten Fällen wird jedoch eine eigene Struktur innerhalb der Daten verwendet, was zu Problemen bei der Auswertung führen kann.

2.1.2. Agentenbasierte Modelle

Die Klasse der agentenbasierten Modelle (ABM) gehört zu der Gruppe der Simulationsmodelle, welche in Abschnitt 2.1 (S. 6) beschrieben sind. Im Bereich der Ökologie wird auch häufig von individuenbasierten Modellen (IBM) gesprochen. Mit Hilfe von ABM/IBM lassen sich die Interaktion autonomer Agenten und deren Einfluss auf ein System simulieren und analysieren. Sie werden häufig im Bereich der Biologie, Ökologie und den Sozialwissenschaften eingesetzt, da diese Systeme eine hohe Komplexität aufweisen. Ereignisse und Gesetzmäßigkeiten in solchen Systemen sind meist nicht hinreichend bekannt, weshalb der Einsatz eines Simulationsmodells hier bevorzugt wird. (Niazi und Hussain, 2011)

2.2. Multiagentensysteme und Agenten

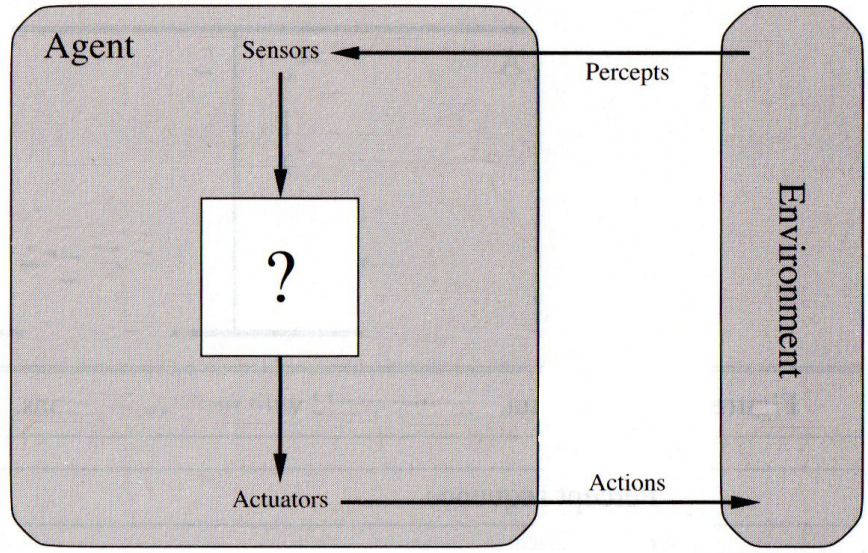

Abbildung 2.1.: Agents interact with environment through sensors and actuators (Russell u. a., 2010)

Der Begriff der Multiagentensysteme (MAS) umschreibt ein computerbasiertes System, welches aus einer beliebigen Anzahl interagierender Einheiten besteht. Diese Einheiten werden als Agenten bezeichnet. Sie können sowohl miteinander als auch mit der Umwelt interagieren. Die Arbeit von Jennings (2000) definiert einen Agenten als gekapseltes Computersystem, welches in einer beliebigen Umwelt existiert und in der Lage ist, flexibel und autonom mit dieser zu interagieren. Jedwede Interaktion mit der Umwelt dient dem Erreichen der Ziele, zu dessen Zweck der Agent geschaffen wurde. Russell u. a. (2010) beschreiben einen Agenten allgemeiner als etwas, was seine Umwelt durch Sensoren wahrnimmt und mit ihr durch Aktoren interagiert. Diese Sicht auf Agenten ist in Abbildung 2.1 (S. 8) dargestellt. Ihre Definition eines Agenten ist nicht auf Software allein beschränkt. Auch Roboter und Menschen passen in das Schema des Agenten, da sie ebenfalls Sensoren (Augen/Kameras, Ohren/Mikrophone) und Aktoren (Stimme/Lautsprecher, Beine/Motoren) besitzen.

Ein Modell aus dem EM in Kombination mit der Definition der Agenten lässt sich als konkrete Ausprägung eines ABM verstehen, jedoch ist ein MAS nicht grundsätzlich mit einem ABM gleichzusetzen. (Niazi und Hussain, 2011)

2.3. Geoinformationssysteme

Das Feld der Geoinformationssysteme (GIS) umschreibt verschiedene Programme und Werkzeuge zur Erstellung, Visualisierung und Analyse von Daten mit spatialem Bezug. Diese Daten werden als Geodaten bezeichnet. Sie beschreiben die geografische Position eines geometrischen Objektes und dessen Eigenschaften. Die Formen geometrischer Objekte reichen hierbei von Punkten über Linien bis hin zu Flächen, welche durch Polygone bzw. Multipolygone beschrieben werden. Das typische Anwendungsgebiet ist die Erstellung von Landkarten, auf denen sich räumliche Analysen durchführen lassen. Hierzu zählen die Berechnung der Fläche eines Areals, Distanzmessungen zwischen geografischen Punkten oder die Mengenberechnung von Objekten in einem Areal. (Baldowski, 2014)

2.3.1. Datenmodelle

Zur Speicherung der Daten werden in GIS das Vektor- oder Rastermodell verwendet. Während die Daten im Rastermodell als eine Matrix von quadratischen Zellen gleicher Größe mit jeweils einem Wert pro Zelle dargestellt werden, liegen die Daten beim Vektormodell in Form von Objekten vor, die eine geografische Position und zugehörige Informationen beinhalten. Beide Verfahren haben Vor- und Nachteile, weshalb ihr Einsatz meist stark vom Anwendungsgebiet abhängt. Bilddaten können beispielsweise nur im Rastermodell dargestellt werden. (Baldowski, 2014)

2.3.2. Projektionen

Es existieren unterschiedliche Projektionen von Geodaten, welche je nach Anwendungsfall Vorteile bieten. Da die Erde eine Kugel im dreidimensionalen Raum ist, benötigt man für eine zweidimensionale Darstellung der Erdoberfläche eine auf Abbildungsvorschriften basierende Projektion.

Die Projektion mit der größten Verbreitung ist WGS84 (EPSG:4326). Diese unterteilt die Erdoberfläche in Längen- und Breitengrade und wird beim Global Positioning System (GPS) verwendet. Eine weitere wichtige Projektion ist die Mercatorprojektion (EPSG:3395). Sie besitzt keine Winkelverzerrung und bietet somit den Vorteil, dass Richtungen und Distanzen korrekt dargestellt werden. Die Mercatorprojektion ist die typische Darstellung der Erdoberfläche, wie man sie aus Anwendungen wie Google Maps oder Microsoft Bing Maps kennt. (Butler u. a., 2014)

2.4. Heterogenität von Daten

Unter der Heterogenität von Daten versteht man die Unterschiede, welche bei Daten aus unterschiedlichen Quellen auftreten können. Hierbei unterscheidet man verschiedene Arten der Heterogenität:

Technische Heterogenität
Die Schnittstelle für den Zugriff auf die Daten unterscheidet sich. Ein Teil der Daten ist z.B. über eine Datenbank verfügbar. Ein anderer Teil liegt in Form von CSV-Dateien vor.

Syntaktische Heterogenität
Die Darstellung identischer Sachverhalte ist verschieden. Als Beispiel liegen Zeitangaben innerhalb der Daten in verschiedenen Formaten vor sodass in Quelle *A* eine alphanumerischer Form („10/31/2013“) vorliegt. In Quelle *B* wird das Datum jedoch mit getrennten Spalten für Jahr, Monat und Tag dargestellt.

Datenmodellheterogenität
Daten liegen in verschiedenen Datenmodellen vor, wie z.B. relationale oder objektorientierte Modelle. Oft liegen hier auch verschiedene Anfragesprachen vor.

Strukturelle Heterogenität
Die Schemata der vorliegenden Daten, unterscheiden sich. Beispielsweise wurden in Schema *A* Geburtsort zusammen mit Postleitzahl für jede Person erfasst. In Schema *B* ist diese Ortsinformation in einer separaten Tabelle abgelegt, welche von Personen referenziert wird.

Semantische Heterogenität
Daten aus unterschiedlichen Quellen haben verschiedene Bedeutungen und Interpretationen. Beispielsweise ist nicht klar definiert was ein Datum bedeutet und wofür Abkürzungen stehen.

Alle Formen der Heterogenität von Daten erschweren deren Integration und Auswertung. Hierfür gilt es Daten zu bereinigen und zu transformieren, um eine homogene Zieldatenbank zu erreichen. (Leser und Naumann, 2007)

2.4.1. Heterogene Skalierung von Daten

Neben der allgemeinen Heterogenität von Daten ist die Heterogenität der Skalierung von Daten ein wichtiger Punkt bei der Auswertung. Der Begriff Skala wird hier im Sinne des spatialen und temporalen Detaillierungsgrad der Daten verstanden. So werden

in der ökologischen Forschung beispielsweise Bewegungsdaten mittels GPS ermittelt. Im Gegensatz dazu werden Temperaturdaten von ein oder mehreren Messtürmen erfasst. Die Temperaturdaten gelten spatial für ein größeres Areal mit dem Messturm als Zentrum. Die Bewegungsdaten sind als Punktdaten zu interpretieren. Während die Bewegungsdaten stündlich erfasst werden, werden die Temperaturdaten jedoch nur alle vier Stunden erfasst. So ergeben sich verschiedene Detaillierungsgrade bzw. Skalierungen in den vorliegenden Daten. Diese verschiedenen spatialen und temporalen Detaillierungsgrade werden in Abbildung 2.2 deutlich. Im gezeigten Beispiel wird angenommen, dass ein Gepard im südafrikanischen Krüger Nationalpark mit einem GPS-Halsband ausgestattet wurde und dieses Halsband minütlich die Position des Geparden speichert. Der Niederschlag wird hingegen über einen Messturm als stündlich summiert erfasst.

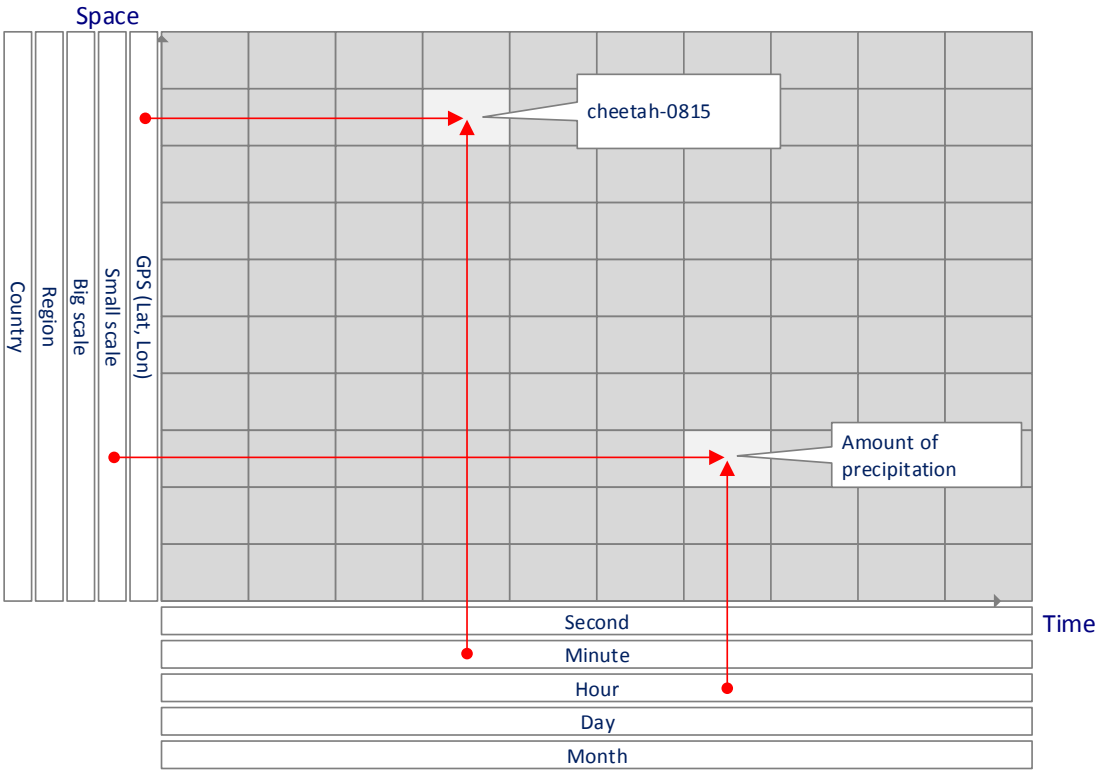

Abbildung 2.2.: Systemzustand mit multiskaligen spatialen und temporalen Referenzen (Thiel-Clemen, 2013)

2.5. Data Warehouses

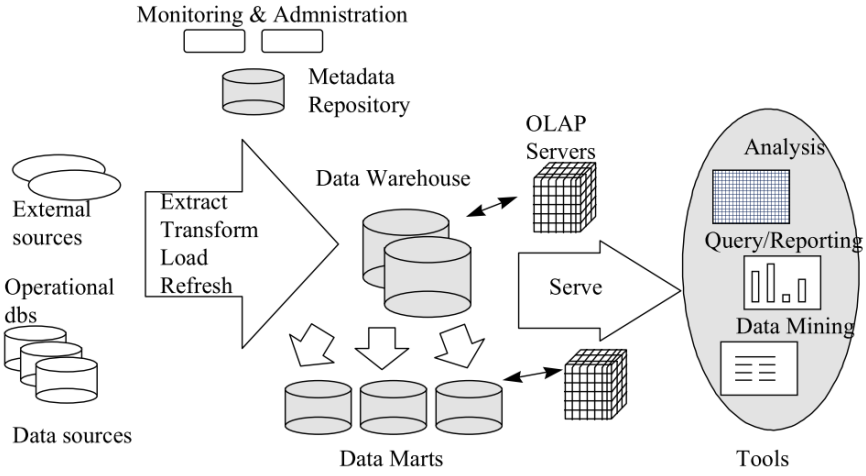

Abbildung 2.3.: Data Warehousing Architecture (Chaudhuri und Dayal, 1997)

Datawarehouses (DWH) werden typischerweise im Bereich der Business Intelligence (BI) eingesetzt. Geschichtlich dienen sie der Unterstützung von Entscheidungsträgern bei der Unternehmensführung. Durch Analyse von Daten des Unternehmens und dem Erstellen von Betriebsberichten (Reports) lassen sich Kennzahlen über den Zustand des Unternehmens gewinnen und auswerten.

Innerhalb eines DWH werden die Daten aus mehreren Datenquellen zusammengefügt und bilden somit eine zentrale Anlaufstelle für Informationen aus unterschiedlichen Bereichen eines Unternehmens. Über einen Extract-Transform-Load-Prozess (ETL) werden die Daten aus den Quellen extrahiert, bereinigt und vereinheitlicht. Dieser Prozess wird meist zyklisch durchgeführt und gewährleistet somit, dass im DWH sowohl aktuelle als auch historische Daten vorliegen. Somit können nicht nur Aussagen über den aktuellen Zustand getroffen werden, sondern auch der Fortschritt des Unternehmens über den Verlauf von Wochen, Monaten und Jahren analysiert werden.

Die Architektur, welche einem DWH zu Grunde liegt, ist in Abbildung 2.3 (S. 12) dargestellt. Es ist zu erkennen, dass die Daten aus den operationalen Datenbanken eines Unternehmens zusammen mit weiteren externe Quellen über einen ETL-Prozess verschmolzen und gemeinsam im DWH abgelegt werden. Oft kommen neben dem DWH mehrere Data Marts (DM) für unterschiedliche Abteilungen eines Unternehmens zum Einsatz. So lassen sich die Daten gezielt für den Informationsbedarf einer bestimmten Abteilung transformieren oder filtern, sodass Analysen im DM mit hö-

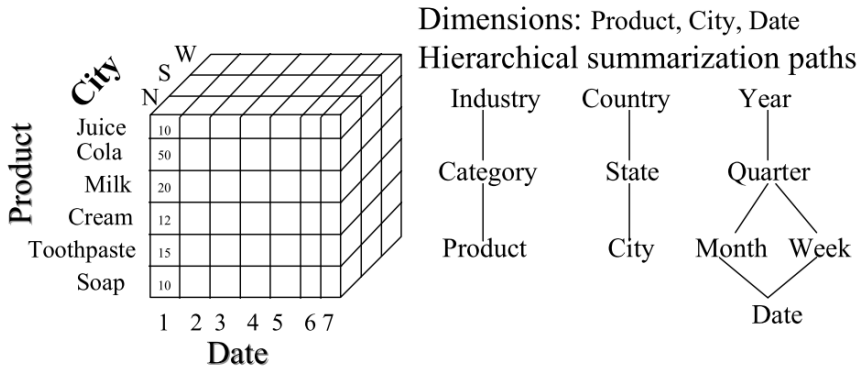

Abbildung 2.4.: Multidimensional data (Chaudhuri und Dayal, 1997)

herer Geschwindigkeit auf einer Teilmenge der Gesamtinformationen durchgeführt werden können. Durch diverse Werkzeuge lassen sich Reports erstellen und gezielt Kennzahlen aus den Daten gewinnen und darstellen. (Chaudhuri und Dayal, 1997).

2.5.1. Datenmodell

Das Datenmodell mit der größten Popularität im Bereich von DWH ist die multidimensionale Sicht der Daten. In einem solchen Modell existieren so genannte Kennzahlen. Als Kennzahlen werden vorberechnete numerische Werte für die Analyse bezeichnet. In der BI sind solche Kennzahlen typischerweise der Return on Investement (ROI), die Anzahl bzw. der Wert der Verkäufe und der Gewinn. Damit eine Kennzahl, wie beispielsweise der Gewinn, einen Kontext erhält, haben Kennzahlen eine Abhängigkeit zu mehreren Dimensionen. Beim Gewinn können Ort der Unternehmensfiliale, das Produkt und der Zeitpunkt die abhängigen Dimensionen darstellen. Somit wird sichtbar wie, wann und wo der Gewinn zustande gekommen ist. Die Dimensionen bestehen aus mehreren Attributen und meist stehen diese in einer Hierarchie zueinander. So hat beispielsweise der Ort oft Attribute, wie Land, Bundesland, Stadt und eine Hierarchie, wie sie in Abbildung 2.4 (S. 13) dargestellt ist. (Chaudhuri und Dayal, 1997)

2.5.2. Entwurfsmuster

Dieser Abschnitt soll einen Überblick über die verwendeten Datenbankschemata geben, die in DWH verwendet werden.

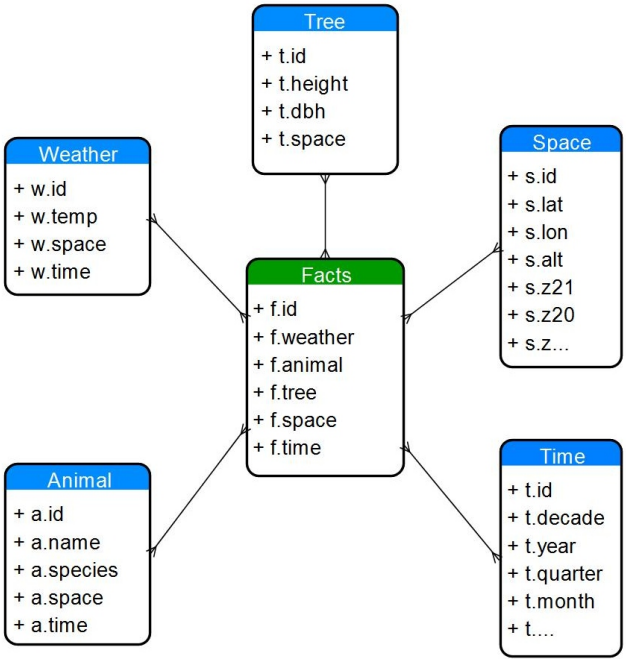

Abbildung 2.5.: Star Schema Beispiel

2.5.2.1. Star Schema

Das Star Schema ist das für DWH typische Schema, mit welchem das multidimensionale Datenmodell umgesetzt wird. Es existiert genau eine Faktentabelle und je eine Tabelle pro Dimension. Eine Normalisierung der Daten findet nicht statt. Die hierdurch entstehende Redundanz und eventuelle Inkonsistenzen innerhalb der Daten werden zur Erhöhung der Geschwindigkeit in Kauf genommen. Seinen Namen hat das Star Schema durch die Anordnung der Tabellen und ihrer Beziehung untereinander. Abbildung 2.5 (S. 14) zeigt, dass die Dimensionen um eine Faktentabelle im Zentrum angeordnet werden und diese über Fremdschlüssel aus der Faktentabelle heraus referenziert werden. Jeder Eintrag in der Faktentabelle repräsentiert einen Fakt, d.h. eine konkrete Zusammenstellung von Informationen aus den verschiedenen Dimensionen. (Chaudhuri und Dayal, 1997) Datenbankabfragen aus einem solchen Schema sind syntaktisch simple aufgebaut und liefern auch bei größeren Datenmengen Ergebnisse ohne große Berechnungsdauer. Quelltext 2.1 (S. 15) gibt ein Beispiel für eine einfache Star-Join-Operation für die Struktur aus Abbildung 2.5 (S. 14).

```
1 SELECT *
2 FROM facts f
3 LEFT JOIN d_time t ON (f.f_time = t.id)
4 LEFT JOIN d_space s ON (f.f_space = s.id)
5 LEFT JOIN d_animal a ON (f.f_animal = a.id)
```

Quelltext 2.1: Star-Join-Operation

Diese Abfrage liefert alle gespeicherten Daten aus den beteiligten Dimensionen. Mit verschiedenen SQL-Befehlen zur Aggregation, wie dem Gruppieren oder Mengenoperationen lässt sich die Abfrage weiter verfeinern. Ein Beispiel gibt Quelltext 2.2 (S. 15). Das Ergebnis dieser Abfrage ist die Anzahl der gesichteten Spezies innerhalb eines Jahres, deren Namen und das Jahr der Sichtung.

```
1 SELECT a.a_species, t.t_year, COUNT(*) AS no_of_animals
2 FROM facts f
3 LEFT JOIN d_time t ON (f.f_time = t.id)
4 LEFT JOIN d_space s ON (f.f_space = s.id)
5 LEFT JOIN d_animal a ON (f.f_animal = a.id)
6 GROUP BY a.a_species, t.t_year
```

Quelltext 2.2: Erweiterte Star-Join-Operation

2.5.2.2. Snowflake Schema

Das Snowflake Schema ist eine Erweiterung des Star Schemas. Die Dimensionstabellen liegen in normalisierter Form vor, was zu Vorteilen bei der Wartung der Dimensionstabellen führt. Das Problem von Redundanzen wird in diesem Schema durch die Verfeinerung der Dimensionstabellen verringert und die Integrität der Daten gesichert. Die höhere Komplexität des Schemas hat jedoch Nachteile bei der Abfragegeschwindigkeit, da ggf. weitere Join-Operationen durchgeführt werden müssen.

2.5.2.3. Fact Constellation Schema

Ein weiteres Schema aus dem Bereich der DWHs ist das Fact Constellation Schema. Bei diesem Schema existieren mehrere Faktentabellen, welche zum Teil auf gemeinsamen Dimensionstabellen basieren. Teilt sich Faktentabelle A ein oder mehrere Dimensionen mit Faktentabelle B, so nennt man dies eine Fact Constellation. Diese Idee der mehrfachen Verwendung von Dimensionen findet sich auch im innerhalb von Abschnitt 5.3.1 (S. 34) beschriebenen Schema wieder, welches im Kontext dieser Arbeit entwickelt wurde.

2.5.2.4. Datenintegration

Die Datenintegration, welche zu Beginn des Abschnitts 2.5 (S. 12) mit dem ETL-Prozess bereits angeschnitten wurde, ist ein wichtiger Punkt bei der Verwendung von DWHs. Für die Extraktion der Daten aus verschiedenen operationalen Datenbanken oder externen Quellen existieren verschiedene Werkzeuge. Eines dieser Werkzeuge wird in 5.1.6 (S. 30) näher beschrieben. Neben der Extraktion aus unterschiedlichen Quellen bieten solche Werkzeuge die Möglichkeit, Daten zu bereinigen, zu transformieren und das Ergebnis der vorangegangenen Operation in das DWH zu laden. Auch das zyklische Laden aktueller Daten wird meist unterstützt.

2.5.3. Management von Data Warehouses und Metadaten

Neben den reinen Fakten und Informationen im DWH müssen Metadaten erfasst und verwaltet werden. Diese Metadaten sind teilweise administrativer Natur, wie z.B. das verwendete Datenbankschema und die physikalischen Grundlagen des DWH. Daneben existieren Metadaten über Herkunft der Daten und Zugriffsrechte auf einzelne Bereiche. Für das Speichern dieser Metadaten kommt oft eine separate Datenbank zum Einsatz, welche mit dem DWH verknüpft ist. Dies ermöglicht den Zugriff auf diese Daten aus unterschiedlichen Prozessen für das Aufsetzen, Benutzen und Administrieren des DWH. Das Management eines DWH ist eine anspruchsvolle Aufgabe und es existieren im BI-Bereich viele verschiedene kommerzielle Produkte, wie z.B. IBM InfoSphere Data Management Hub, HP Power Advisor oder Microsoft Management Data Warehouse.

2.5.4. Online Analytical Processing

Online Analytical Processing (OLAP) ist eine Analysemethodik für multidimensionale Daten. Ein Analyst definiert eine Hypothese und erstellt eine Datenabfrage, deren Ergebnis die gestellte Hypothese bestätigt oder widerlegt. Bei OLAP Systemen steht die Durchführung komplexer Analysevorhaben im Vordergrund, welche ein sehr hohes Datenaufkommen verursachen. Ein OLAP-Cube, wie in Abbildung 2.6 (S. 17) dargestellt, beschreibt die multidimensionale Repräsentation der Daten im DWH und ist die Grundlage für Analysen. (McGuire u. a., 2008)

Typische Operationen beim OLAP werden im Folgenden erläutert:

Roll-Up
: Diese Operation erlaubt es die Daten auf einer höheren Hierarchie- bzw Aggregationsstufe zu betrachten. So lassen sich entlang der Zeitachse z.B. der Gewinn nach Jahren statt Monaten darstellen.

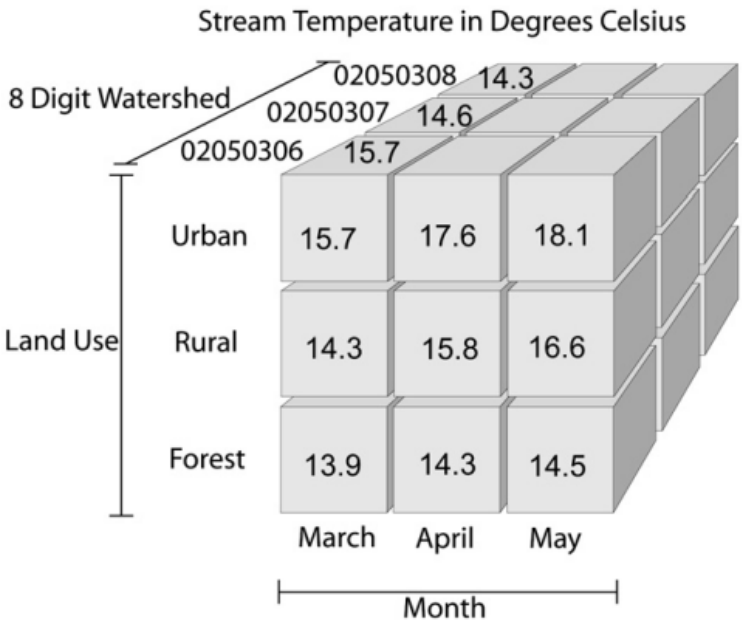

Abbildung 2.6.: Illustration of a data cube with measures of stream temperature and dimensions of land use, watershed, and time. (McGuire u. a., 2008)

Drill-Down

Bei dieser Operation führt man im Gegenteil zum Roll-Up eine Verfeinerung der Aggregationsstufe durch, sodass z.B. Verkaufszahlen nicht nach Ländern sondern Städten analysiert werden können.

Slicing

Beim Slicing lassen sich die Kennzahlen entlang einer Dimension des OLAP-Würfels z.B. für einen einzelnen Monat genauer analysieren.

Dicing

Das Dicing erzeugt einen kleineren Würfel, der aus einer Teilmenge der Daten des Eingangswürfels besteht. So können die gespeicherten Informationen auf verschiedene Arten eingeschränkt werden. Es ist dadurch möglich, Daten nur für einen gewissen Zeitraum und Ort herauszuschneiden. Die Dicing-Operation lässt sich im Prinzip mit der Kombination mehrerer Slicing-Operationen vergleichen.

3. Stand der Forschung

Dieses Kapitel fasst den aktuellen Stand der Forschung im Bereich der Verwaltung ökologischer Daten zusammen und gibt einen Einblick in den Ablauf und die Probleme bei der Datenerfassung und Integration. Die Analyse und Verschneidung der Daten unterschiedlicher Forschungsgruppen soll hier dargestellt werden. Ansätze und Probleme, die sich aus der Erfassung und Weiterverarbeitung ergeben, werden aufgezeigt und erläutert.

3.1. Datenmanagement in der ökologischen Forschung

Der Großteil der ökologischen Forschung basiert auf Messungen, Beobachtungen und Untersuchungen der realen Welt. Auch wenn die Forschung meist auf Feldmessungen basiert, welche von Hand durchgeführt werden, wächst der Anteil der Daten, die durch Fernerkundungssensoren und Satelliten erfasst werden ständig. In der Veröffentlichung von Cushing u. a. (2002) ist beschrieben, dass der fundamentale Aspekt der Verortung der Daten auch mit den neuen Mitteln der Erfassung problematisch bleibt. Das Erfassen der genauen Position eines Objektes in der realen Welt sowie der relativen Position zu anderen Objekten ist nicht nur für den Autor einer Studie, sondern auch für andere Forscher von großer Bedeutung. Obwohl es auf der Hand liegt die Position eines Objektes mit Hilfe eines Werkzeugs wie GPS zu erfassen, geschieht dies in der ökologischen Forschung oft nicht. Viel mehr erfassen viele Projekte Positionsinformationen relativ zu lokalen Bezugspunkten. Unterschiedliche Koordinatensysteme und Einheiten selbst innerhalb eines Projektes führen zu Problemen bei der Auswertung und Verschneidung der Daten.

Für Forscher ist die Planung der Datenerfassung zwar notwendig, jedoch eher sekundär. Primär sollen Daten gesammelt werden, welche zur Beantwortung der Fragen benötigt werden, die im Fokus der Forschungsarbeit liegen. Die Wiederverwertbarkeit durch andere Forscher und Reproduzierbarkeit der Ergebnisse finden oft weniger Beachtung. Expertengespräche, welche von Cushing u. a. (2002) durchgeführt wurden, haben bestätigt, dass die Vorverarbeitung der Felddaten und deren Umwandlung für die Analyse sehr zeitintensiv und fehleranfällig ist. Es wird eine Infrastruktur benötigt,

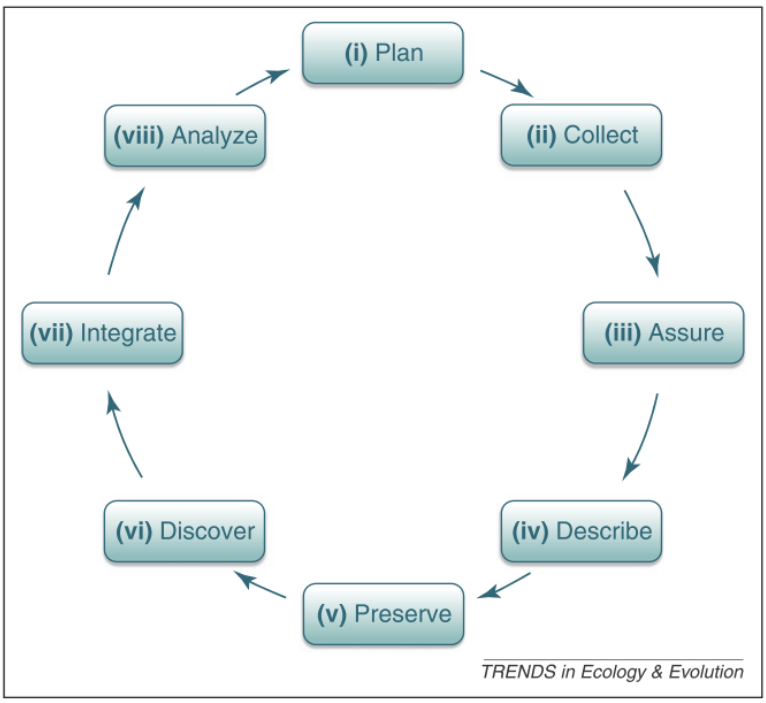

Abbildung 3.1.: The data life cycle (Michener und Jones, 2012)

welche eine Lösung für solche Probleme bietet oder diesen Prozess zumindest vereinfacht und beschleunigt.

Michener und Jones (2012) stellen fest, dass die Erfassung, das Management und die Analyse großer Datenmengen, die von Forschern weltweit zusammengetragen werden, eine große Herausforderung an bestehende und zukünftige Werkzeuge stellt. Gerade die hohe Variabilität der Skalierung ökologischer Daten werden hier genannt. Die Interaktion verschiedener Forschungsgruppen und der Austausch von Informationen zwischen diesen Gruppen macht es erforderlich, die Datenhaltung zu vereinheitlichen und verschieden skalierte Daten zu verschneiden. Eine Lösung für dieses Problem ist bisher nicht vorhanden, jedoch wird dies dringend benötigt.

3.2. Lebenszyklus ökologischer Daten

Michener und Jones (2012) skizzieren einen Zyklus für Daten aus der Ökologie, der sich in unterschiedliche Stufen unterteilt. Dieser Lebenszyklus von Daten ist in Abbildung 3.1 (S. 19) dargestellt. In der Planungsphase (i) wird das Projekt analysiert. Anforde-

rungen werden anhand der Forschungsziele des Projektes erarbeitet. Die folgenden Schritte bauen auf den definierten Anforderungen auf. Bei Forschungsprojekten, die von Regierungsorganisationen oder Unternehmen vergeben werden, ist es inzwischen obligatorisch, einen Plan zum Datenmanagement vorzulegen, um die Verwertbarkeit der Daten und Ergebnisse auf Seiten der Auftraggeber zu sichern. Auf die Planung (i) folgen Datenerfassung (ii) und Qualitätssicherung (iii). Zusätzlich zu den erfassten Daten gilt es Metainformationen (iv) zu erfassen, welche die Daten beschreiben. Die Kombination aus gesammelten Daten und Metadaten wird im darauf folgenden Schritt gespeichert (v). Um einen Überblick darüber zu gewinnen, inwiefern die vorliegenden Daten zur Beantwortung konkreter Fragestellungen genutzt werden können, werden die Daten exploriert (vi). Eventuell werden weitere Informationen aus anderen Quellen integriert (vii). Im letzten Schritt können die Daten statistisch analysiert und visualisiert werden (viii). Dieser Lebenszyklus zeigt die große Bedeutung des Datenmanagement in der Ökologie und beschreibt Aufgaben, welche bei der Entwicklung neuer Werkzeuge zum Datenmanagement berücksichtigt werden müssen.

3.3. Data Warehouses für spatio-temporale Daten

In der ökologischen Forschung ist die Verwendung eines DWH für die Datenexploration bereits erfolgreich eingesetzt worden, wie die Arbeit von McGuire u. a. (2008) beschreibt. Sie sehen OLAP-Systeme als hervorragend geeignet an, da Ökologen typischerweise multidimensionale Analysen durchführen und OLAP-Technologien genau für diesen Anwendungsfall eingesetzt werden können.

Ein wichtiges Forschungsgebiet ist die Modellierung spatialer Dimensionen im DWH, wie die Arbeiten von Pestana u. a. (2005) und Jensen u. a. (2004) zeigen. Sie beschäftigen sich mit der Repräsentation der spatialen Daten im multidimensionalen Datenmodell eines DWH. Die Definition einer spatialen Dimension mit geeigneten Hierarchien, um spatiale Abfragen zu ermöglichen, liegt im Fokus der Arbeiten.

Pestana u. a. (2005) beschäftigen sich mit der Frage, wie sich spatiale Daten, welche sich über die Zeit verändern, in einem spatialen DWH gespeichert und abgefragt werden können. Ihr Ansatz soll es ermöglichen die Evolution einer Geometrie in einem vorgegebenen Intervall und einer vorgegebenen Region zu analysieren. Hierfür erweitern Sie das traditionelle Sternschema um spatiale Datentypen und spatiale Operationen, die Areale auf mögliche Überschneidung, Überdeckung o.ä prüfen können. Zu den spatialen Informationen speichern sie Metadaten. In diesen Daten ist beispielsweise die verwendete Projektion der Geodaten enthalten. Jensen u. a. (2004) beschäftigen sich mit standortbezogenen Diensten. In diesem Szenario zeigen sie die Anforderungen,

die solche Dienste an ein multidimensionales Datenmodell stellen. Auch hier werden die Erweiterung der traditionellen Schemata mit spatialen Datentypen und Operationen propagiert. Weiterführend wird ein Weg aufgezeigt, wie mit der teilweisen Überdeckung innerhalb spatialer Daten umgegangen und eine Aggregation dieser Daten realisiert werden kann.

Die bestehenden Ansätze zur Verwendung eines DWH für spatio-temporale Daten mit heterogener Skalierung zeigen, dass deren Einsatz verschieden Vorteile bietet. Der konkrete Einsatz für Simulationen, wie er im MARS Framework angedacht ist, wirft jedoch weitere Fragen zur Organisation eines großen DWH auf, welches Daten aus verschiedenen Projekten und Quellen vereint. Die dynamische Verknüpfung von Daten aus unterschiedlichen Quellen, ist im klassischen DWH nicht ohne Weiteres möglich. Zwar lassen sich aus einem Datenwürfel mit der Dicing-Operation kleinere Datenwürfel herauslösen, jedoch wird hierfür ein einzelner Würfel benötigt, der alle vorhandenen Informationen verknüpft. Diese Anforderung ist für Datenmengen, wie sie in MARS Rock abgelegt werden sollen nicht praktikabel.

4. Analyse

Das folgende Kapitel beschreibt Beispielszenarien für den Einsatz eines DWH für multiskalige, heterogene ökologische Daten. Die Szenarien werden analysiert und Anforderungen auf deren Basis definiert. Diese Anforderungen sollen durch die Komponente MARS Rock umgesetzt werden.

4.1. Beispielszenarien

4.1.1. Abdoulaye Wildlife Reserve (AWR)

Als ersten konkreten Anwendungsfall für das MARS-Framework diente die Doktorarbeit von Hodabalo Pereki (Pereki u. a., 2013). Im Kontext dieser Doktorarbeit wurde die Waldbiomasse Westafrikas am Beispiel des Abdolaye Wildlife Resort im Togo untersucht. Die Bedrohung der Wälder von Afrika geht hauptsächlich vom Klimawandel oder der Abholzung durch den Menschen aus. Das AWR gehört zum Guineischen Wald Westafrikas und ist als Biodiversitätshotspot die Heimat vieler Pflanzen- und Tierarten. Durch die Rodung und Umwandlung von Wäldern in Ackerflächen ist diese Artenvielfalt unmittelbar bedroht. Vor diesem Hintergrund wurden von Pereki u. a. (2013) der Baumbestand und der Einfluss von Projekten zum Schutz der Wälder auf den selbigen analysiert. Mit den gewonnenen Daten lassen sich Rückschlüsse auf die zukünftige Entwicklung des AWR und ähnlicher Gebiete ziehen. Zusätzlich dienen die Ergebnisse der Entscheidungsunterstützung für Regierungen der Region mit Hinblick auf den Schutz der Artenvielfalt.

4.1.1.1. Datenquellen

Das Gebiet, welches untersucht wurde, liegt zwischen 08°34’ und 08°46’ nördlichem Breitengrad und 01°13’ und 01°25’ östlichem Längengrad. In diesem Areal wurden Stichproben erhoben, welche nach dem „line transect“-Verfahren durchgeführt wurden. Diese weitverbreitete Methode zur Bestimmung der relativen Dichte von tropischen Wäldern basiert auf der Arbeit von Marshall u. a. (2008). Die Stichprobe enthält die Daten von 30 quadratische Arealen mit einer Abmessung von 30x30 Metern. Diese Areale werden als Patches bezeichnet. Die Patches liegen mindestens 500 Meter voneinander entfernt. Pro Patch werden zwischen 20 und 40 Bäume vermessen. Diese Messungen

beinhalten Stammdurchmesser, Kronendurchmesser, Höhe, Spezies, Familie und Position des Patches. Durch die Feldmessungen konnten aus den 30 erfassten Patches 912 Baumdatensätze generiert werden.

Da die erfassten Bäume nur einen sehr kleinen Teil des AWR abbilden, kommen verschiedene Näherungen zum Einsatz. Ein Teil dieser Näherungen basieren auf dem „Normalized Difference Vegetation Index“ (NDVI). Der NDVI berechnet sich aus Reflexionswerten, die von Satellitenaufnahmen im Infrarotbereich bzw. sichtbaren Bereich (ca. 620-700nm) stammen. Je nach relativer Biomasse im Aufnahmegebiet lassen sich Rückschlüsse auf die Vegetation und vorhandene Biomasse ziehen. (Baldowski, 2014)

4.1.1.2. Auswertung

Im Kontext dieser Forschung kommt das MARS-Framework zum Einsatz, um die Entwicklung der Baumbiomasse im AWR in den nächsten Jahren zu simulieren. Hierfür ist es erforderlich die gesammelten Daten in MARS zu importieren. Die Daten aus den Feldmessungen wurden in Microsoft Excel tabellarisch erfasst, wie in Tabelle 4.1 (S. 23) dargestellt ist. Der hier dargestellte Auszug enthält zur Veranschaulichung weniger Attribute als der originale Datensatz. Jede Zeile der Tabelle spiegelt einen Baum des untersuchten Gebietes wider.

Patch	Spezies	Familie	H (m)	Ø (cm)	X (D,dd)	Y (D,dd)	Z (m)
R26	Acacia erhenbergiana	Mimosaceae	25	45,5	8,59321874	1,262318562	254
R26	Acacia erhenbergiana	Mimosaceae	7	31,2	8,59321622	1,262314158	254
R19	Aganope stuhlmannii	Fabaceae	13	105,2	8,60935712	1,388849446	357
R30	Alstonia boonei	Apocynaceae	7,5	41,6	8,59202184	1,264260488	261

Tabelle 4.1.: Auszug aus den gesammelten Stichproben (Hodabalo Pereki, 2013)

4.1.2. ARS AfricaE

Das Projekt ARS AfricaE befindet sich zum Zeitpunkt dieser Arbeit noch in der Aufbauphase. Das Projekt ist ein Verbund aus vier deutschen und sechs südafrikanischen Partnerinstitutionen, welche durch den Deutschen Akademischen Austauschdienst (DAAD) unterstützt und durch das Bundesministerium für Bildung und Forschung (BMBF) finanziert werden. Das Forschungsgebiet umfasst den Krüger National Park (KNP) in Südafrika und das Kataba Forest Reserve in Zambia. An der Forschung sind folgende Institutionen beteiligt:

- Johann Heinrich von Thünen-Institut (Braunschweig, Deutschland)

- Friedrich-Schiller-Universität (Jena, Deutschland)
- Johann Wolfgang Goethe Universität (Frankfurt am Main, Deutschland)
- Hochschule für Angewandte Wissenschaften (Hamburg, Deutschland)
- Zambian Meteorological Department (Lusaka, Sambia)
- Grootfontein AD Institute (Grootfontein, Südafrika)
- University of Witwatersrand (Johannesburg, Südafrika)
- Council for Scientific and Industrial Research (CSIR) (Pretoria, Südafrika)
- Rhodes University (Grahamstown, Südafrika)
- Forest Sense (Kapstadt, Südafrika)

Die Zielsetzung ist es, ein Netzwerk von Forschungsclustern aufzubauen, um die Effekte von Störung und Landnutzungswechsel auf Wasser- und Kohlenstoffkreislauf und ihrer Interaktion entlang eines Aridităts-Gradienten zu erfassen. Es soll versucht werden, mit Hilfe von Felduntersuchungen und Modellen die Funktion und Struktur des Ökosystems zu verknüpfen. Weiterführend soll eine Charakterisierung von Ökosystemstörungen anhand von funktionellen Parametern wie Bruttoprimärproduktion, Wassernutzung und anderer Effizienzparameter durchgeführt werden. Der Aufbau eines individuenbasierten Modellsystems zur Vorhersage von Ökosystemdynamiken unter verschiedenen Störungen und Landnutzungen soll u.A. mit Hilfe des MARS Frameworks realisiert werden. Die Entwicklung von nachhaltigen Langzeitstrategien zum Ökosystem-Management sind die wichtigsten Erkenntnisse, welche aus dem Projekt gewonnen werden sollen.

4.1.2.1. Datenquellen

Das Untersuchungsgebiet, welches als Teilprojekt von ARS AfricaE mit Hilfe des MARS Frameworks behandelt wird, liegt in der näheren Umgebung von Skukuza, dem Hauptcamp des KNP. Das Gebiet liegt zwischen 24°59'24.5"S 31°28'38.5"E und 25°01'37.5"S 31°31'03.5"E. Innerhalb des Gebietes liegt ein Messturm, welcher im Zuge des SAFARI-2000 Experimentes von Scholes u. a. (2001) errichtet wurde. Die Position dieses Messturms ist 25°01'10.8"S 31°29'48.6"E. Informationen zum Niederschlag in der Region stehen durch mehrere Messstationen in der Umgebung von Skukuza zur Verfügung. Im Verlauf des Projektes werden weitere Datenquellen hinzukommen, zum jetzigen Zeitpunkt sind jedoch keine weiteren verfügbar.

4.1.2.2. Auswertung

Die Daten liegen in verschiedenen Dateien vor. Zumeist handelt es sich um CSV-Dateien mit heterogenem Aufbau. Daten gleichen Typs, im konkreten Fall Niederschlagsdaten, liegen für verschiedene Zeitraume über mehrere Dateien verteilt vor, welche jedoch verschieden strukturiert sind. Die Temperaturdaten liegen in einer einzelnen CSV-Datei vor, jedoch ist der Zeitraum für den diese Daten vorliegen deutlich kürzer als bei den Niederschlagsdaten. Während die vorliegenden Niederschlagsdaten den Zeitraum von September 1912 bis 2013 abdecken, liegen die Temperaturdaten nur von 2000 bis 2013 vor. Bei der zeitlichen Skalierung der Daten ist anzumerken, dass die Temperaturdaten nur in einem achttägigen Zyklus erfasst wurden. Bei den Niederschlagsdaten ist kein Muster zu erkennen. Hier liegen zeitweise Daten für einen oder zwei aufeinanderfolgende Tage vor und zeitweise existieren über längere Zeiträume gar keine Daten. Fehlende Werte sind in den Daten unterschiedlich repräsentiert. Während vereinzelt ein Nullwert für fehlende Messdaten verwendet wurde, werden oft negative Werte, wie -777.70 oder -999.90 verwendet. Dies führt ohne Bereinigung der Daten zu Problemen bei der Analyse und Verwendung innerhalb einer Simulation.

4.2. Anforderungsanalyse

Die Anforderungen, welche im Folgenden definiert werden, ergeben sich aus dem Stand der Forschung aus Kapitel 3 (S. 18) und aus den Beispielszenarien aus Kapitel 4.1 (S. 22). Das Zielsystem muss demnach in der Lage sein, große Datenmengen mit spatio-temporalem Bezug speichern zu können und dabei für DWH typische Funktionalität bieten. Hierzu zählen die in Kapitel 2.5.4 (S. 16) genannten Operationen, aber auch konkrete Abfragen anhand von Geometrien. Da weitere Daten, wie Höhenkarten o.ä. in GIS-Formaten anfallen und diese in MARS Ground gespeichert werden, muss ein einheitliches Referenzsystem verwendet werden. So kann gewährleistet werden, dass die in die beiden Systeme Rock und Ground eingepflegten Daten auch korrekt abgefragt werden können. Das System muss skalierbar sein, sodass durch das Bereitstellen weiterer Hardware Geschwindigkeitsverluste durch höhere Datenmengen und Abfragekomplexität ausgeglichen werden.

Die Probleme mit heterogenen, multiskaligen Daten, wie sie auch in den Beispielszenarien aus 4.1 (S. 22) beschrieben sind, fließen ebenfalls in die Anforderungen ein. Das Zielsystem muss die Daten unabhängig von der vorliegenden Heterogenität analysieren können. Verschiedene Skalierungen innerhalb der Daten müssen bei der Auswertung Berücksichtigung finden. Gegebenenfalls müssen die Daten durch weitere Werkzeuge transformiert und bereinigt werden.

4.2.1. Funktionale Anforderungen

[F-AN-01] Multiskaligkeit und Heterogenität

Durch den Einsatz eines DWH und der Verwendung der Aggregationsfunktionalitäten des selbigen wird es ermöglicht, multiskalige, heterogene Daten zu analysieren und auszuwerten. Durch das Multidimensionale Datenmodell eines DWH und einem neu entwickelten Schema wird es ermöglicht, Daten innerhalb des DWH dynamisch zu kombinieren.

[F-AN-02] Schnittstelle zur Datenaggregation

Für spezielle Geodaten wie Höhenkarten o.ä., welche in GIS-Formaten vorliegen, wird im MARS-Framework die Komponente Ground zur Speicherung verwendet. Tabellarische Daten werden hingegen in Rock abgelegt. Diese Form der Datenhaltung erfordert es, eine Schnittstelle zwischen beiden Systemen zu schaffen, welche die Vereinigung der Daten aus Ground und Rock ermöglicht.

[F-AN-03] API zum Datenimport/-export

Es wird eine API bereit gestellt, die das Einfügen, Bearbeiten und Abfragen der Daten ermöglicht ohne die manuelle Eingabe von SQL-Befehlen. Diese Abstraktion sichert die Integrität der Daten und vereinfacht die Verwaltung des Datenspeichers. Weiterhin bietet die API Möglichkeiten, die interne Struktur des DWH zu explorieren.

4.2.2. Nichtfunktionale Anforderungen

[NF-AN-01] Verwaltung großer Datenmengen ohne Geschwindigkeitsverlust

Das System setzt ein geeignetes Datenbankmanagementsystem (DBMS) ein, um große Datenmengen analysieren zu können. Hierfür kommen Konzepte zur Verteilung und Skalierung zum Einsatz, welche es auch bei stetig wachsender Datenbasis ermöglichen, eine gute Performanz zu gewährleisten.

[NF-AN-02] Programmiersprache C#

Im gesamten Framework wird C# und das .NET-Framework (bzw. Mono) zur Entwicklung eingesetzt. Dies gilt auch für die Entwicklung der API Drill.

[NF-AN-03] Integration in das MARS-Framework

Das System ist vollständig in das MARS-Framework und den dazugehörigen Workflow integriert. Andere Komponenten innerhalb des Frameworks nutzen hierfür die aus AN-04 resultierende API.

5. Architektur

Das folgende Kapitel beschreibt die Komponente MARS Rock, welche Teil des MARS-Frameworks ist und aus dieser Arbeit hervorgeht. Das Ergebnis der im Folgenden vorgestellten Architektur bildet die Basis für die Realisierung von MARS Rock in Kapitel 6 (S. 49). Innerhalb dieses Kapitels werden auch weitere Komponenten aus dem Framework beleuchtet, um den Kontext der Architektur von Rock zu erläutern. Des Weiteren werden die verwendeten Werkzeuge aufgeführt und erläutert.

5.1. Werkzeuge

Im folgenden Abschnitt werden verschiedene Werkzeuge und Hilfsmittel erläutert, welche für die Umsetzung von MARS Rock verwendet werden. Die verwendete Software beschränkt sich ausschließlich auf kostenfreie bzw. quelloffene Software. Somit bleibt MARS zu Gunsten der Forschung und Kooperation mit anderen Forschungseinrichtungen offen. Diese Einschränkung führt jedoch dazu, dass eventuell kommerzielle Lösungen mit höherer Performanz außen vor bleiben.

5.1.1. MariaDB

Das Datenbankmanagementsystem MariaDB ist ein von Ulf Michael Widenius initiiertes quelloffenes Projekt. Als Abspaltung von MySQL, welches sich inzwischen als Marke im Besitz von Oracle befindet, wird MariaDB offen unter der GNU General Public Licencse (GPL) entwickelt. Als früherer Hauptentwickler von MySQL hat Widenius die Basisfunktionalitäten der Datenbank entwickelt und das DBMS nach seiner Tochter My benannt. MariaDB trägt den Namen seiner zweiten Tochter Maria.

MariaDB ist abwärtskompatibel zu MySQL und dient als vollständiger Ersatz. Hinzu kommen diverse Erweiterungen. So unterstützt MariaDB mehr Storage Engines, wie z.B. XtraDB als Ersatz für InnoDB, Cassandra als NoSQL Engine und viele Weitere. Die Optimierungen und Erweiterungen machen MariaDB zu einem DBMS mit diversen Einsatzmöglichkeiten und einer zunehmenden Verbreitung. (MariaDB Corporation, 2014b)

5.1.1.1. Galera Cluster

Galera Cluster ist ein Cluster mit synchroner Replikation für MariaDB. Es bietet die Replikation innerhalb einer Multi-Master-Topologie und ermöglicht somit sowohl Lese- als auch Schreiboperationen zu jedem Knoten im Cluster. Des Weiteren ist es möglich zur Laufzeit neue Knoten hinzuzufügen, die sich automatisch in das Cluster eingliedern. Somit lässt sich das DBMS über die Bereitstellung weiterer Knoten skalieren und das sowohl für Lese- als auch für Schreiboperationen. Voraussetzung ist ein Linux System und die Verwendung der Storage Engines XtraDB oder InnoDB. Für den sinnvollen und sicheren Einsatz werden mindestens drei Knoten im Cluster benötigt. Bei Fehlern während einer Transaktion auf einem Knoten (z.B. durch Netzwerkprobleme) ist es den anderen Knoten möglich mit der Transaktion fortzufahren und den fehlerhaften Knoten wieder zu synchronisieren. Existieren lediglich zwei Knoten im Cluster, lässt sich nicht mehr bestimmen welche Datenbasis korrekt ist. (MariaDB Corporation, 2014b)

5.1.1.2. Galera Load Balancer

Der Galera Load Balancer (GLB) ist ein Lastverteiler (engl. Load Balancer) und basiert auf *pen*, einem generischen TCP Lastverteiler. GLB setzt, wie auch Galera Cluster, das Betriebssystem Linux voraus. Es werden verschiedene Verteilungsstrategien unterstützt. So lassen sich einzelnen Knoten unterschiedliche Gewichtungen zuteilen. Je höher das Gewicht eines Knotens ist, desto mehr Last wird auf diesen Knoten verteilt. Der GLB Prozess nimmt Anfragen entgegen und leitet diese an die vorhandenen Knoten auf Basis der gewählten Strategie weiter. Für den anfragenden Nutzer bleibt dieser Vorgang transparent. (MariaDB Corporation, 2014b)

5.1.1.3. MariaDB Manager

Als Managementoberfläche für ein MariaDB Galera Cluster bietet sich die Verwendung des MariaDB Manager an. Dieser bietet eine webbasierte Oberfläche, mit deren Hilfe sich einzelne Knoten installieren und verwalten lassen. Es lassen sich Statusinformationen auslesen und Auslastung des Clusters visualisieren. Des Weiteren ist es möglich einzelne Knoten zu isolieren, Datensicherungen zu erstellen, Knoten zu stoppen oder zu starten. (MariaDB Corporation, 2014a)

5.1.2. Linux Kernel-based Virtual Machine

Kernel-based Virtual Machine (KVM) ist eine Virtualisierungslösung auf Basis von Linux. Sie unterstützt x86-Hardware und die Virtualisierungserweiterungen IntelVT und AMD-V. Die Virtualisierung ist in den Linux Kernel integriert und ermöglicht es

mehrere virtuelle Maschinen mit unterschiedlichen Gastsystemen zu betreiben. Es werden sowohl Windows, als auch Linux/Unix Systeme in 32/64-bit als Gast unterstützt. Virtuelle Maschinen können zur Laufzeit zwischen unterschiedlichen Knoten verschoben werden. (Red Hat, 2014)

5.1.3. OpenNebula

OpenNebula ist ein quelloffener Industriestandard für die Virtualisierung von Rechenzentren. Es bietet einfache Lösungen zum Erstellen und Verwalten von Clouds oder virtualisierten Rechenzentren und stellt viele verschiedene Schnittstellen bereit, um physikalische und virtuelle Ressourcen zu verwalten.

5.1.4. NetTopologySuite

Die NetTopologySuite ist eine C#-Reimplementierung der in Java geschriebenen JTS Topology Suite. Sie ist eine API zur Modellierung zweidimensionaler Geometrien. Sie bietet diverse spatiale Funktionen, mit deren Hilfe sich Geometrien analysieren und manipulieren lassen. So können z.B. Schnittpunkte zweier Polygone oder Flächenberechnungen durchgeführt werden. Bei der Implementierung wurde darauf geachtet, dass die API der Spezifikation „Simple Feature Access“ des Open Geospatial Consortium[1] entspricht. Die Spezifikation umschreibt gültige Datentypen, wie Punkte, Linien, Polygone, etc. und Methoden auf diesen Geometrieobjekten.

5.1.5. Sysbench OLTP Benchmark

SysBench ist ein modularer, plattformübergreifender Benchmark für die Evaluierung von Systemparametern, welche großen Einfluss auf die Performanz einer Datenbank unter hoher Last haben. Der OLTP Testmodus wurde speziell für die Leistungsmessung von DWHs entwickelt. (Alexey Kopytov, 2008)

5.1.6. Pentaho Data Integration

Mit dem Werkzeug Pentaho Data Integration (Kettle)[2] ist es möglich, verschiedene Datenquellen einzubinden, die Daten zu bereinigen, zu transformieren und zu laden. Hierfür bietet Kettle eine grafische Oberfläche, welche es dem Benutzer ermöglicht über eine intuitive Drag & Drop Funktion, verschiedene Operatoren zu einer Art Ablaufdiagramm zusammenzufügen und Schritt für Schritt zu definieren, wie die Daten

[1] http://www.opengeospatial.org/
[2] http://community.pentaho.com/projects/data-integration/

zu behandeln sind. Eine große Anzahl an verfügbaren Datenquellen- und Datensenkenoperatoren machen das Werkzeug universell einsetzbar. So ist es beispielsweise möglich, heterogene Datenquellen, wie mehrere CSV-Dateien, einzulesen und so zu transformieren und zu bereinigen, dass sie sich die Daten problemlos in eine Zieldatenbank speichern lassen. Alternativ ist die Ausgabe als CSV-Datei, welche die Daten der Eingangsdateien kombiniert, ebenfalls möglich.

5.2. Multi-Agent Research and Simulation (MARS)

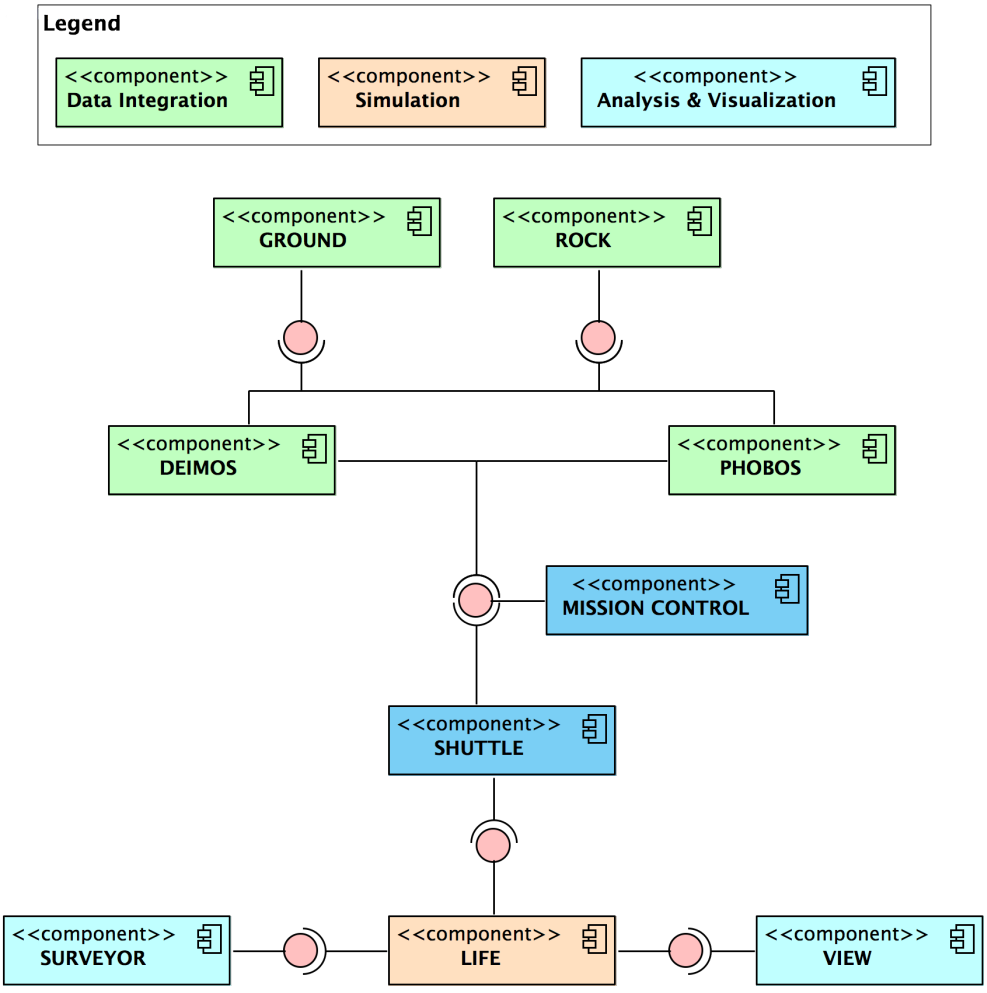

Abbildung 5.1.: Architekturdiagramm des MARS-Frameworks

Das System MARS ist ein Multi-Agenten Framework für Forschungs- und Bildungszwecke. Hinter dem System MARS steht eine Forschungsgruppe der HAW Hamburg, die sich mit der Simulation auf Basis von Mulit-Agenten Systemen (MAS) in interdisziplinären Projekten beschäftigt.

MARS bietet die Möglichkeit verteilte und hoch skalierbare Simulationen durchzuführen und stellt ein Gesamtpaket an Werkzeugen bereit, die für die Vor- und Nachbereitung notwendig sind. MARS besteht aus mehreren Komponenten, die verschiedene Teilaspekte abdecken.

- Integration von Daten via Phobos
- Erstellen von Szenarien via Deimos
- Simulation von Modellen via Life
- Auswertung von Simulationsergebnissen (sowohl grafisch als auch auf Basis konkreter Ergebnisse) via View

Hierbei sind alle Teile des Systems so modularisiert, dass sich verschiedenste Teile der Szenarien und auch die Daten wiederverwenden lassen. Ein großer Augenmerk liegt bei MARS auf der Verteilbarkeit und somit hohen Skalierbarkeit. MARS ermöglicht die Simulation einer hohen Anzahl von Agenten, welche über die Obergrenzen gängiger Simulationsframeworks hinausgeht. Durch das Bereitstellen zusätzlicher Hardware lässt sich dies noch weiter steigern. Ein Diagramm der Architektur von MARS ist in Abbildung 5.1 (S. 31) dargestellt. Die Komponenten des Frameworks werden im Folgenden näher erläutert. (Hüning u. a., 2014)

Es ist zu beachten, dass die im Nachfolgenden genannten Komponenten sich zum Teil noch in der Planung oder Entwicklung befinden. Die Komponenten mit der größten Relevanz im Kontext dieser Arbeit sind Ground, welches in der Arbeit von Baldowski (2014) beschrieben wird, sowie Deimos und Phobos.

5.2.1. Mission Control

MARS Mission Control (MMC) ist die zentrale Weboberfläche zur Interaktion mit dem MARS-System. Die MMC ermöglicht den Zugriff auf alle relevanten Teile der Infrastruktur und bietet eine Schnittstelle zu allen Komponenten von der Datenintegration/-exploration und Erstellung von Szenarien bis hin zur Simulation.

5.2.2. Phobos

Das Ziel der Komponente Phobos ist es, eine optimierte und möglichst einfache Datenerfassung zu ermöglichen. Daten können in einer Vielzahl von möglichen Formaten vorliegen. Aus diesem Grund unterstützt auch Phobos eine große Anzahl gängiger Formate beim Import. Durch eine komfortable Oberfläche und durchdachte Werkzeuge wird der Anwender durch den Importvorgang geführt. Somit ist es möglich, Daten zu importieren, Metadaten zu erfassen, Daten zu bereinigen und zu transformieren. Phobos ist webbasiert und in MMC integriert.

5.2.3. Ground

Ground ist die GIS-Komponente von MARS. Sie unterstützt das Framework mit Geodaten und typischen GIS-Operationen, wie sie in Abschnitt 2.3 (S. 9) beschrieben sind. Zur Manipulation und Interaktion mit Geodaten bietet Ground diverse Werkzeuge. Die Problematik unterschiedlicher Projektionen, Formate und Ausdehnungen werden mit Hilfe der in Ground enthaltenen Werkzeugen gelöst. Zur Verwaltung großer räumlich indizierter Daten nutzt Ground ein Werkzeug zum Geodatenmanagement namens GeoServer. Dieses eignet sich als spatiale Datenbank zur Speicherung von Geodaten. Hier werden primär Rasterdaten und polygonale Vektordaten verwaltet. (Baldowski, 2014)

5.2.4. Deimos

Mit Hilfe von Deimos können Daten aus den Systemen Rock und Ground extrahiert und für eine Simulation zusammengestellt werden. Diese Informationen können als Grundlage für die Initialisierung von Agenten und der Umwelt einer Simulation genutzt werden. Anhand von Metainformationen, welche beim Import mit Phobos erfasst wurden, können in Deimos Daten exploriert werden. Zusätzlich zum Filtern der Informationen anhand von Metadaten, ist es möglich, durch Einschränkung von Raum und Zeit die Daten weiter einzugrenzen.

5.2.5. Life

Life ist die Simulationsumgebung von MARS. Ein konkretes Szenarium wird als Modell mit Agenten umgesetzt, welche auf so genannten Layern existieren und interagieren. In Life setzen sich Layer und Agenten aus in sich geschlossenen Einheiten zusammen. Diese Einheiten werden mit Hilfe des Life Development Kit (LDK) realisiert. Hierzu stellt das LDK eine API zur Verfügung. Für die Entwicklung von Agenten und Layern

mit dem LDK werden fortgeschrittene Kenntnisse im Bereich der Softwareentwicklung benötigt. Life ist in der Lage eine große Menge von Agenten auf einer Vielzahl von Layern zu simulieren, was bei bisherigen Simulationswerkzeugen nicht möglich war. Dies wird durch die Verwendung verschiedener Strategien zur Verteilung und Skalierung der Simulation bewerkstelligt.

5.2.6. View

Als Analysewerkzeug kommt View zum Einsatz. Um eine Darstellung unabhängig von einer speziellen Visualisierungslösung zu erreichen, definiert Vis ein eigenes Protokoll zur Visualisierung. Dieses Protokoll definiert verschiedene Klassen zur Beschreibung von Objekten und Umwelt. Durch die Implementierung des Protokolls durch einen Adapter für beispielsweise die 3D-Engine Unity, kann das Simulationsergebnis in 3D analysiert werden. Zusätzlich bietet MARS die Möglichkeit, die Simulationsergebnisse für statistische Auswertungen in Textform auszugeben.

5.3. MARS Rock

Die Komponente Rock bezeichnet ein DWH zur Speicherung von multiskaligen, heterogenen ökologischen Daten für Simulationszwecke innerhalb des MARS-Frameworks (siehe NF-AN-03). Auch die dazugehörige API namens Drill ist Teil von Rock (siehe F-AN-03). Rock dient der Speicherung jedweder Daten mit spatio-temporalem Bezug, welche in tabellarischer Form vorliegen. Daten, die in GIS-Formaten vorliegen, werden gesondert in Ground abgelegt (siehe F-AN-02). Rock nutzt ein verteilbares und somit skalierendes DBMS zur Gewährleistung kontinuierlicher Performanz trotz steigender Datenmengen (siehe NF-AN-01).

5.3.1. Shared Dimensions Schema

Eine grundlegende Designentscheidung bei Rock ist es, nicht auf die in DWH typischen Schemata, wie sie in 2.5 (S. 12) beschrieben sind, zu setzen, sondern auf ein neues Schema, welches auf den Anforderungen aus Kapitel 4 (S. 22) basiert. Dieses Schema trägt den Namen Shared Dimensions Schema (SDS) und ist im Rahmen dieser Arbeit entwickelt worden. Hierbei werden im Gegensatz zu den üblichen Schemata nicht die Daten über eine große Faktentabelle verknüpft, sondern über eine Vielzahl so genannter Dynamic Fact Tables (DFT). Diese DFTs werden nicht beim Import der Daten erzeugt, sondern nach Bedarf. Das bedeutet, dass eine DFT lediglich die Daten enthält, welche für ein konkretes Simulationsszenarium benötigt werden und auch nur

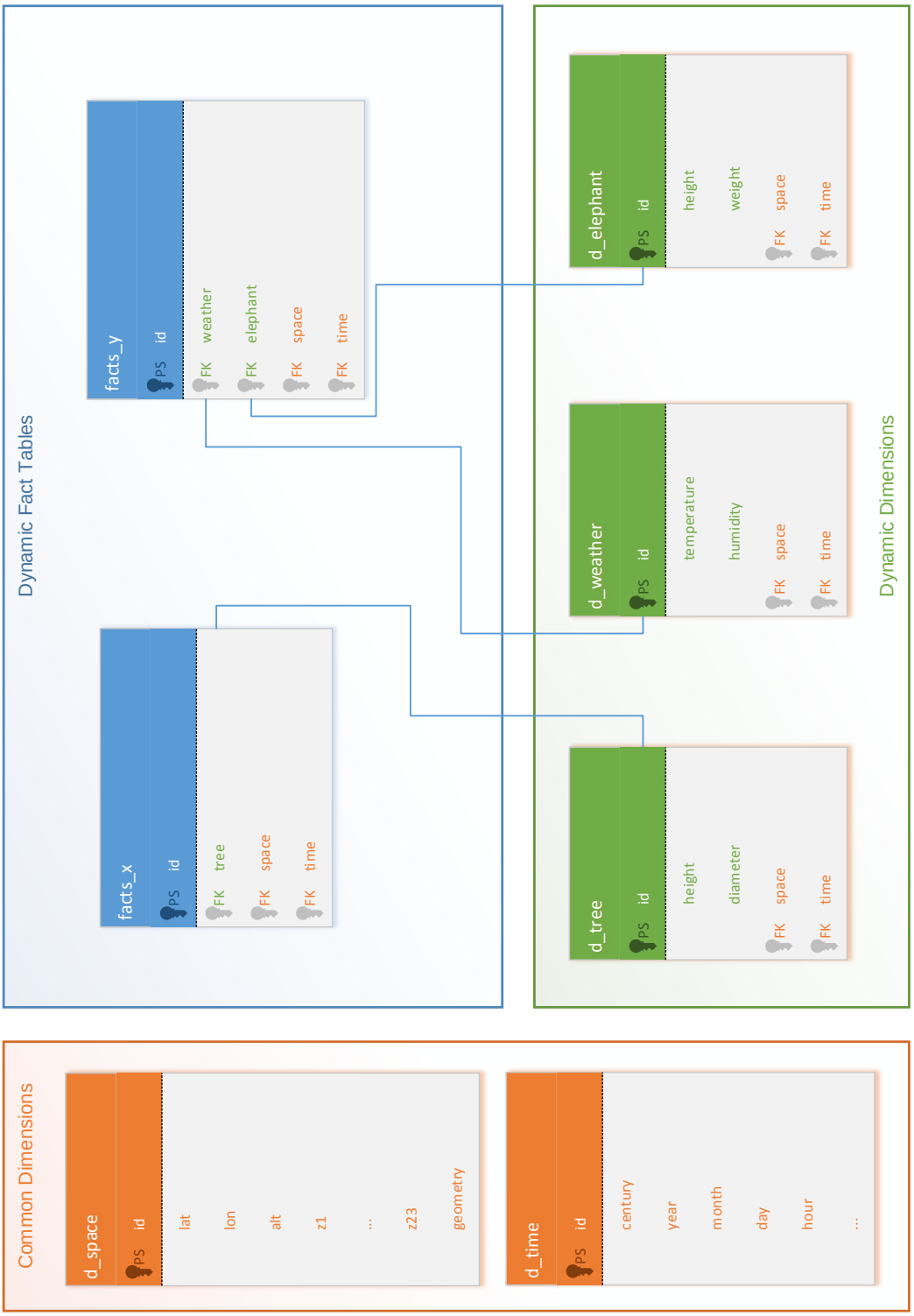

Abbildung 5.2.: Shared Dimensions Schema

die Dimensionen referenziert, welche die benötigten Daten enthalten. Diese Einschränkung ermöglicht es kleinere Faktentabellen zu generieren und höhere Performanz bei der Abfrage von Daten zu gewährleisten. Die dynamische Rekombination von Dimensionen bietet die Möglichkeit beliebige Daten miteinander zu verknüpfen und diese für Analysen und Simulationen zu nutzen. Ein Beispiel für dieses Schema ist in Abbildung 5.2 (S. 35) dargestellt. Es wurden absichtlich nicht alle relationalen Verbindungen in die Zeichnung eingefügt, um die Übersichtlichkeit zu verbessern. So sind Referenzen durch farbliche Informationen zu erkennen. Die farbliche Markierung der Dimensionen deutet zusätzlich die Gruppierung der Dimensionen in unterschiedliche Dimensionstypen an, welche durch das SDS definiert werden. Die verschiedenen Dimensionstypen werden in den folgenden Abschnitten erläutert.

5.3.1.1. Shared Dimensions

```
1 SELECT
2   tree.id f_tree, t.id f_time, s.id f_space
3 FROM
4   d_tree tree
5 LEFT JOIN d_time t ON (tree.d_time = t.id)
6 LEFT JOIN d_space s ON (tree.d_space = s.id)
```

Quelltext 5.1: Indirekte DFT

Das SDS definiert verschiedene Dimensionstypen. Es existieren Common Dimensions (ComDim), welche in Abbildung 5.2 (S. 35) orange dargestellt sind und Dynamic Dimensions (DynDim), welche grün dargestellt sind. Zur Gruppe der ComDim zählen exakt zwei vordefinierte Dimensionen für Zeit und Raum. Die Gruppe der DynDim kann abgesehen von Zeit und Raum, beliebige Dimensionen umfassen. In Abbildung 5.2 (S. 35) werden als Beispiel die DynDims *d_tree*, *d_weather* und *d_elephant* skizziert. Die Unterteilung der Dimensionen in unterschiedliche Gruppen ist die Basis für die Generierung von DFTs, welche im Abschnitt 5.3.1.2 näher beleuchtet werden.

Common Dimension Time

Die temporale Dimension *d_time* gehört zu den Common Dimensions. Sie hat eine Reihe von Attributen, welche in einer Hierarchie zueinander stehen. Diese Hierarchie ist in Abbildung 5.3 (S. 37) dargestellt. Die verschiedenen Attribute stehen mit ihrem Nachfolger in $1 : n$ Beziehungen zueinander und mit ihrem Vorgänger in einer $1 : 1$ Beziehung. Am Beispiel *year* bedeutet das, dass das Jahr 2014 im 21. Jahrhundert liegt, es aber mehrere Quartale, Monate, etc. im Jahr 2014 gibt. Diese Hierarchien dienen als Aggregationslevel im DWH.

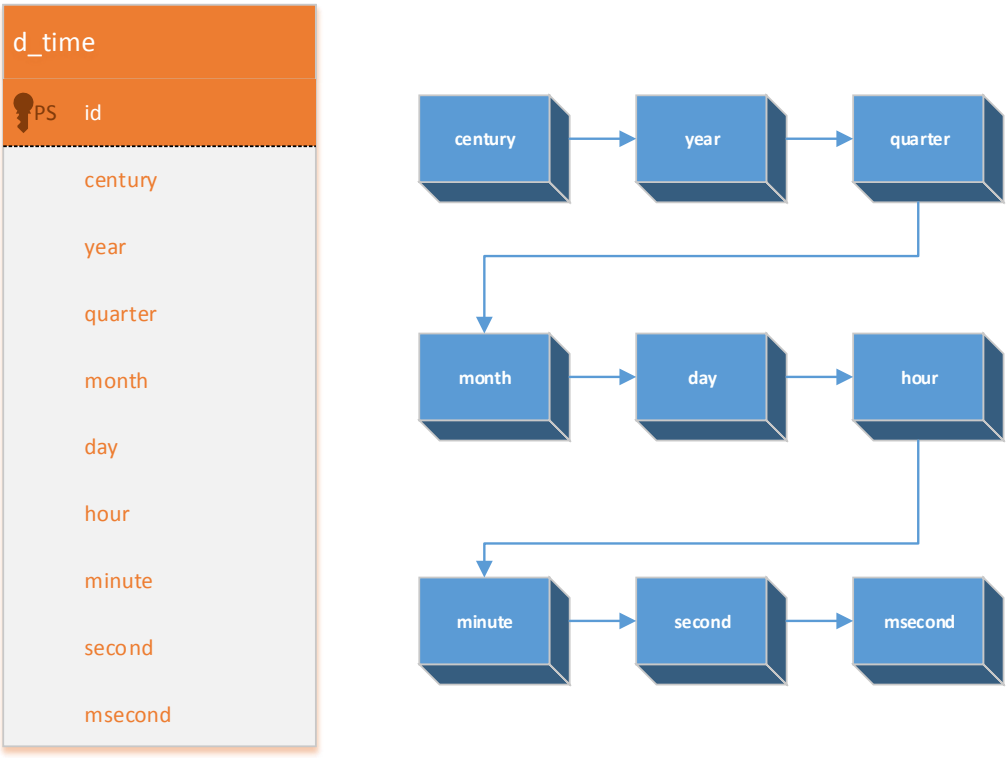

Abbildung 5.3.: Shared Dimensions Schema - Dimension Time

Common Dimension Space

Wie auch die temporale Dimension *d_time* gehört die spatiale Dimension *d_space* zu den *Common Dimensions*. Die Hierarchie der Attribute ist in Abbildung 5.4 (S. 38) dargestellt. Die höchsten Aggregationsstufen bilden *custom* und *geometry*. Im Folgenden werden die verschiedenen Attribute näher erläutert.

custom
: Eine beliebige alphanumerische Zeichenfolge, mit der ein spatiales Objekt markiert werden kann.

geometry
: Spezieller Datentyp, welcher für verschiedene geometrische Objekte stehen kann. Hierzu zählen Punkte, Linien und Polygone. Über datenbankspezifische

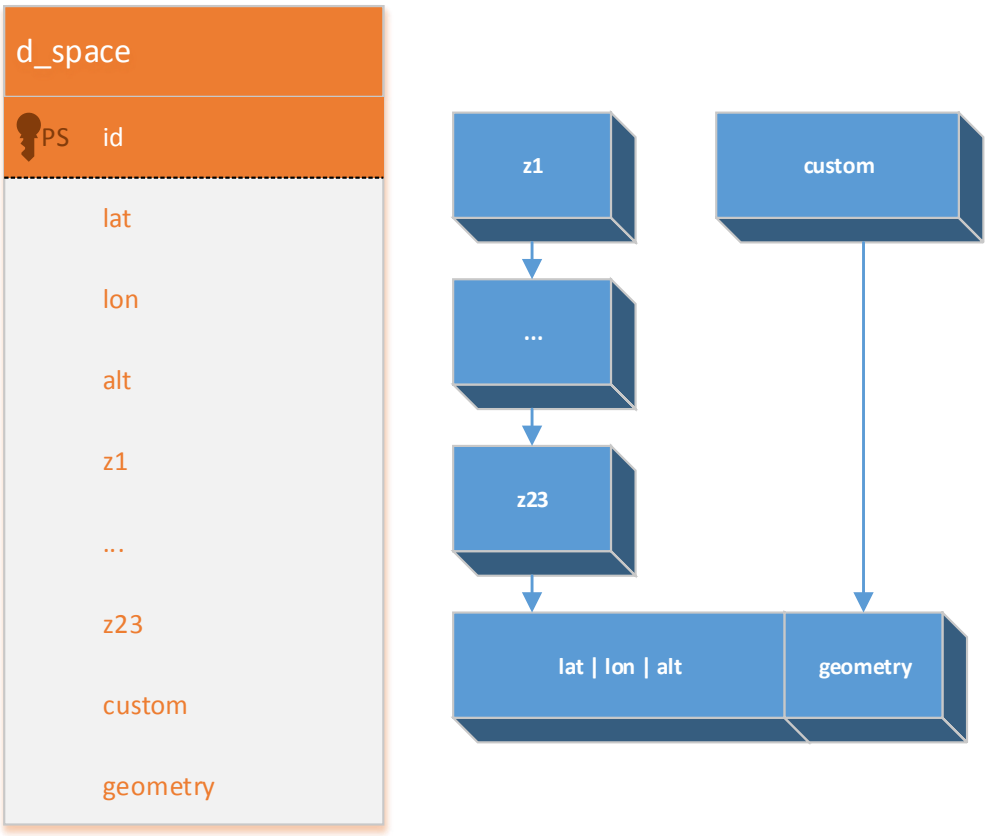

Abbildung 5.4.: Shared Dimensions Schema - Dimension Space

Abfragen können Typ, Position, Fläche und weitere Details des Objektes ermittelt werden.

z1-z23
Identifikationsnummern für QuadTiles. QuadTiles werden hier als Indizierungsstrategie von Geodaten verwendet (siehe AN-03). Der Präfix 'z' steht hier für Zoom und der numerische Suffix für das Zoomlevel. Das Konzept hinter der Indexierung über QuadTiles wird in Abschnitt 5.3.1.3 (S. 39) näher erläutert.

lat
Geographische Breite in dezimaler Darstellung. Entspricht der nördlichen bzw.

südlichen Entfernung eines Punktes der Erdoberfläche vom Äquator in Grad (siehe WGS84/EPSG:4326).

lon

Geographische Länge in dezimaler Darstellung. Entspricht der westlichen bzw. östlichen Entfernung eines Punktes der Erdoberfläche vom Nullmeridian in Grad (siehe WGS84/EPSG:4326).

alt

Entspricht der Höhe über dem Meeresspiegel als Näherung des Schwerefeldes der Erde, wie es bei GPS verwendet wird (siehe WGS84/EPSG:4326).

5.3.1.2. Dynamic Fact Tables

Die Referenzierung über die *Common Dimensions* bietet die Möglichkeit, Faktentabellen dynamisch abgestimmt auf die Anforderungen zu erzeugen. Ist die Datenbasis gewachsen und enthält Informationen, welche für die eigene Forschung nicht relevant sind, ist es wenig sinnvoll diese über eine einzige Faktentabelle zu verknüpfen.

DFTs existieren zunächst indirekt über Referenzen. Hierfür hält jede Dimension aus der Gruppe der DynDim Referenzen auf die beiden Dimensionen aus der Gruppe der ComDim. Über die SQL-Abfrage aus Quelltext 5.1 (S. 36) wird der Zusammenhang deutlich. Durch die Abfrage werden die benötigten Dimensionen verknüpft. Ein SQL-Beispiel für die Erstellung einer DFT mit mehreren DynDim befindet sich im Anhang A.1 (S. 77).

5.3.1.3. Georeferenzierung

Das Verwenden verschiedener Systeme zur Speicherung von Daten, konkret die Verwendung eines GIS im Zusammenspiel mit einem DWH, werfen das Problem der Georeferenzierung zwischen den beiden Systemen auf. Grundsätzlich ist es dem Schema des DWH geschuldet, dass spatiale Daten in einer hierarchischen Form abgespeichert werden. Die typischen Hierarchieebenen bei spatialen Dimensionen in einem DWH, wie es im Zusammenhang mit BI häufig verwendet wird, sind Kontinente, Länder, Kommunen und andere politische Grenzen bzw. Areale. Diese Einteilung ist für ökologische Daten jedoch nur bedingt geeignet. Politische Grenzen sind über die Zeit veränderlich und auch nicht weltweit eindeutig. Das bedeutet, dass Datensätze aus verschiedenen Epochen nicht in dieselbe Hierarchie gebracht werden könnten, falls sich Landesgrenzen zwischen den Datenerhebungen verändert haben. Durch eine von politischen Grenzen unabhängige Speicherung der Daten wird das Problem umgangen und es bleibt trotzdem möglich Daten auf Basis solcher Eingrenzungen abzufragen.

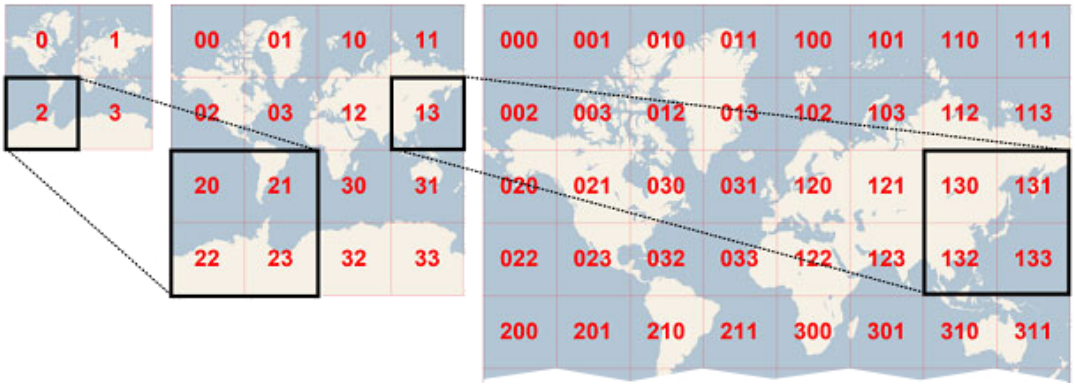

Abbildung 5.5.: Microsoft Bing Maps Tile System: Berechnung von QuadTile-IDs (Joe Schwartz)

Hierfür kommt ein Referenzsystem auf Basis von QuadTiles zum Einsatz. Konkret werden Geodaten mit Hilfe von QuadTrees (Finkel und Bentley, 1974) zur Referenzierung eingesetzt.

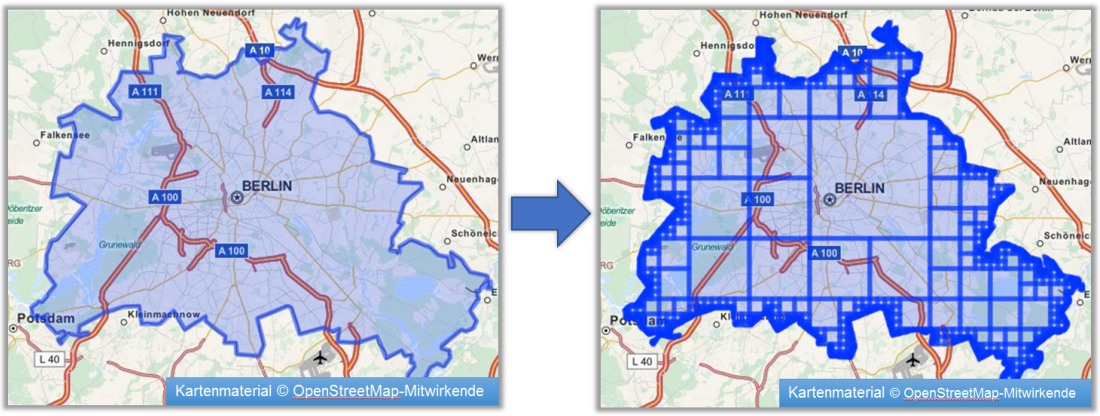

Abbildung 5.6.: Approximierung eines Polygons durch QuadTiles am Beispiel Berlin

Wie in Abbildung 5.5 (S. 40) zu erkennen ist, wird bei QuadTiles die Erdoberfläche auf Basis der Mercator-Projektion in Quadranten eingeteilt. Diese Unterteilung wird entsprechend der Definition im Bing Maps Tile System von Microsoft vorgenommen (Joe Schwartz). Dieses System sieht es vor, dass es verschiedene Ebenen mit unterschiedlicher Granularität gibt. Auf der ersten Ebene wird die Erdoberfläche in vier Quadranten mit eindeutiger Identifikationsnummer (ID) unterteilt. Auf jeder weiteren Ebene darunter unterteilt sich jeder Quadrant in weitere vier Quadranten. Daher auch die Bezeichnung QuadTree. Die Länge der ID entspricht hierbei der Ebene, auf der

sich der Quadrant befindet. So liegt der Quadrant mit der ID 131 auf der Ebene 3. Er hat den Quadranten mit den IDs 13 und 1 als Vorgänger. Dies ergibt die Hierarchie $1 => 13 => 131$. Nutzt man ein solches System zur Speicherung ergibt, sich ein weiteres Problem. Die Darstellung der Bereiche, für die Daten abgefragt werden sollen, liegen natürlicherweise nicht in Form von QuadTiles vor. Oft sind es Polygone, die erst umgewandelt werden müssen, um das Potential des DWH mit der vorliegenden Hierarchie verwenden zu können. Hierfür wurde im Rahmen dieser Arbeit ein Algorithmus entwickelt, welcher diese Umwandlung durchführt. Mit der Ergebnismenge von QuadTiles kann eine entsprechende Anfrage an das DWH gestellt werden, um die Daten für das Areal des Eingangspolygons zu erhalten. Diese Approximation des Polygons durch eine Menge von QuadTiles ist in Abbildung 5.6 (S. 40) dargestellt.

```
1  QuadTileSearch(geometry)
2  {
3    Set minimalZoomLevel to 1
4    Set startQuadTiles to null
5
6    minimalZoomLevel = DetermineLevelForArea(geometry.Area);
7    startQuadTiles = AllQuadTilesForZoom(minimalZoomLevel);
8
9    Set result to empty List of String
10
11   //Parallelized execution
12   for i=0 to startQuadTiles.Length
13     quadTile = startQuadTiles[i]
14     result += InspectQuadTile(quadTile, geometry)
15   return result;
16 }
```

Quelltext 5.2: Approximierung eines geometrische Objektes durch QuadTiles - Teil 1

Der vollständige Algorithmus ist als Pseudocode in den Quelltexten 5.2 (S. 41), 5.3 (S. 42) und 5.4 (S. 43) dargestellt. Als Eingabe benötigt der Algorithmus lediglich das zu approximierende geometrische Objekt. Zur Verringerung des Rechenaufwandes wird in Zeile 7 aus Quelltext 5.2 (S. 41) errechnet, auf welcher Ebene mit der Approximierung begonnen werden muss. Ist die Fläche des geometrischen Objektes kleiner als die Fläche eines QuadTiles auf der entsprechenden Ebene, wird die nächst tiefere Ebene untersucht. Dieser Vorgang wird so lange wiederholt bis eine Ebene erreicht ist, auf der die Fläche eines QuadTiles kleiner der Fläche des geometrischen Objektes ist oder die tiefste Ebene erreicht ist.

```
1  InspectQuadTile(quadTile, geometry){
2    Set result to empty list of string
3
4    switch DetermineVisibility(quadTile, geometry){
5      //Geoobject fully outside current QuadTile => return
6      case OUTSIDE:
7        break;
8      //QuadTile fully inside current geoobject => add current
           quadTile
9      case WITHIN:
10       result.Add(quadTile);
11       break;
12     //Geoobject smaller than this QuadTile or is only partially
         covered => examine children
13     case INTERSECTS:
14       if quadTile.ZoomLevel >= MAX_ZOOM_LEVEL
15         break;
16
17       //Parallelized execution
18       for i=0 to quadTile.Children.Length
19       {
20         childQuadTile = quadTile.Children[i]
21         result += InspectQuadTile(childQuadTile, geometry);
22       }
23       break;
24   }
25   return result;
26 }
```

Quelltext 5.3: Approximierung eines geometrische Objektes durch QuadTiles - Teil 2

Anschließend werden alle QuadTiles auf dem ermittelten Ebene untersucht. Die in Zeile 13 aus Quelltext 5.2 (S. 41) beginnende Schleife wird parallel ausgeführt, um die Performanz heutiger Mehrkernprozessoren auszunutzen. Innerhalb der Schleife wird die in Quelltext 5.3 (S. 42) dargestellte rekursive Funktion aufgerufen. Hier beginnt die eigentliche Arbeit des Algorithmus, nämlich die Untersuchung der Sichtbarkeit von QuadTiles mit Hinblick auf das gegebene geometrische Objekt. Bei der Sichtbarkeit gibt es die folgenden Möglichkeiten:

Outside Das geometrische Objekt liegt vollständig außerhalb des untersuchten QuadTiles.

Within Das untersuchte QuadTile liegt vollständig innerhalb des geometrischen Objektes.

Intersects Das geometrische Objekt und das untersuchte QuadTile schneiden sich.

Liegt das geometrische Objekt vollständig außerhalb des untersuchten QuadTiles wird die Suche hier abgebrochen und das QuadTile wird nicht der Ergebnismenge hinzugefügt. Ist es hingegen vollständig innerhalb des geometrischen Objektes, ist es nicht nötig auf tieferen Ebenen zu suchen und das QuadTile wird der Ergebnismenge hinzugefügt. Im Falle einer Überschneidung, ist es nötig, die Kinder des aktuellen QuadTiles zu betrachten. Hierfür werden rekursiv alle Kinder des aktuellen QuadTiles untersucht. Auch dieser Aufruf wird parallel ausgeführt. Als Abbruchkriterien des Algorithmus gelten folgende Kriterien:

- Das geometrische Objekt liegt vollständig außerhalb des untersuchten QuadTiles.
- Das untersuchte QuadTile liegt vollständig innerhalb des geometrischen Objektes.
- Das untersuchte QuadTile hat keine Kinder bzw. die tiefste Ebene wurde erreicht.

```
1 DetermineVisibility(quadTile, geometry){
2   if !quadTile.Bounds.Intersects(geometry) return OUTSIDE;
3   if quadTile.ZoomLevel == MAX_ZOOM_LEVEL return WITHIN;
4   if geometry.Contains(quadTile.Bounds) return WITHIN;
5   return INTERSECTS;
6 }
```

Quelltext 5.4: Approximierung eines geometrische Objektes durch QuadTiles - Teil 3

Spatiale Datenbanktypen

Neben einer datenbankunabhängigen Georeferenzierung durch QuadTiles, gibt es weitere Möglichkeiten, die Abfrage der Daten auf Basis von geometrischen Objekten zu realisieren. Hierfür bieten einige DBMS spatiale Typen für Spalten in Datenbanken. Zusätzlich bieten sie spatiale Indizes zur Beschleunigung der Abfragen. Das in Abschnitt 5.1.1 (S. 28) beschriebene DBMS MariaDB definiert eine Reihe spatialer Datentypen, welche in Tabelle 5.1 (S. 43) erläutert werden.

Datentyp	Beschreibung
GEOMETRY & GEOMETRYCOLLECTION	Abstrakter Datentyp und Basis aller anderen Typen.
POINT & MULTIPOINT	Zweidimensionaler Punkt bestehend aus X- und Y-Koordinate
LINESTRING & MULTILINESTRING	Linie bestehend aus n-Punkten mit Start- und Endpunkt.
POLYGON & MULTIPOLYGON	Polygon bestehend aus n-Punktringen.

Tabelle 5.1.: Spatiale Datentypen ab MariaDB 5.3.3 (MariaDB Corporation, 2014b)

Die Daten werden datenbankintern im Well-Known Binary Format (WKB) gespeichert. MariaDB bietet aus diesem Grund Funktionen zur Umwandlung von geometrischen

Informationen zum WKB-Format und vice versa. Hierfür müssen die geometrischen Informationen im Well-Known Text Format (WKT) vorliegen. Dieses simple Textformat dient dem Austausch von geometrischen Informationen in ASCII-Form. So können Daten im WKT- oder WKB-Format eingegeben und ausgelesen werden. Da geometrischen Daten in unterschiedlichen Projektionen vorliegen können, ist es in MariaDB möglich die Spatial Reference System ID zusammen mit den geometrischen Informationen zu speichern. Diese ID entspricht den Codes der European Petroleum Survey Group Geodesy (EPSG) und bietet so die Möglichkeit unterschiedliche Projektionen innerhalb der Datenbank zu verwalten.

Die Verwendung spatialer Datentypen in der Raum-Dimension ermöglicht die Erstellung von Datenbankabfragen mit der Einschränkung auf ein durch ein Polygon beschriebenes Gebiet. Die Geschwindigkeit durch spatiale Indizes ist hier besonders hoch und es muss keine aufwendige Berechnung seitens der API von Drill durchgeführt werden. Es ist möglich GIS-typische Operationen auf den geometrischen Objekten innerhalb der Datenbank auszuführen und somit komplexe Anfragen mit räumlichem Bezug zu erstellen.

5.3.2. Drill API

Als Schnittstelle zu MARS Rock wird die API Drill bereitgestellt. Sie dient als Abstraktion zur Datenbank, sodass sich Operationen wie das Einfügen von Daten oder das Erstellen von DFTs über die Verwendung der Programmiersprache C# durchführen lassen. Auch die Abfrage von Daten aus dem DWH ist hierüber gelöst. Anhand von verschiedenen Diagrammen wird das Konzept hinter Drill im Folgenden verdeutlicht. In Abbildung 5.7 (S. 45) ist ein Klassendiagramm der API Drill dargestellt. Es zeigt die wichtigsten Klassen, welche grundlegend für die Architektur und die Verwendung von Drill sind. Die zentrale Klasse ist gleich dem Namen der API *Drill*. Sie bietet eine Reihe statischer Methoden zur Verwaltung des DWH, welche in den folgenden Abschnitten näher erläutert werden.

Erstellung von Dimensionen

Das in Abbildung 5.8 (S. 46) dargestellte Sequenzdiagramm zeigt die Erstellung von Dimensionen. Zunächst wird Drill mit den Informationen über die zu verwendende Datenbank initialisiert. Hierfür wird die Methode *InitializeConnection* mit den entsprechenden Parametern für Benutzer, Passwort und IP-Adresse der Datenbank aufgerufen. Danach können alle Funktionen von Drill verwendet werden. Hierzu zählt beispielsweise das Anlegen von Dimensionen. Für diesen Vorgang muss sowohl ein Bezeichner als auch eine Liste von Attributen übergeben werden. Diese Attribute entsprechen

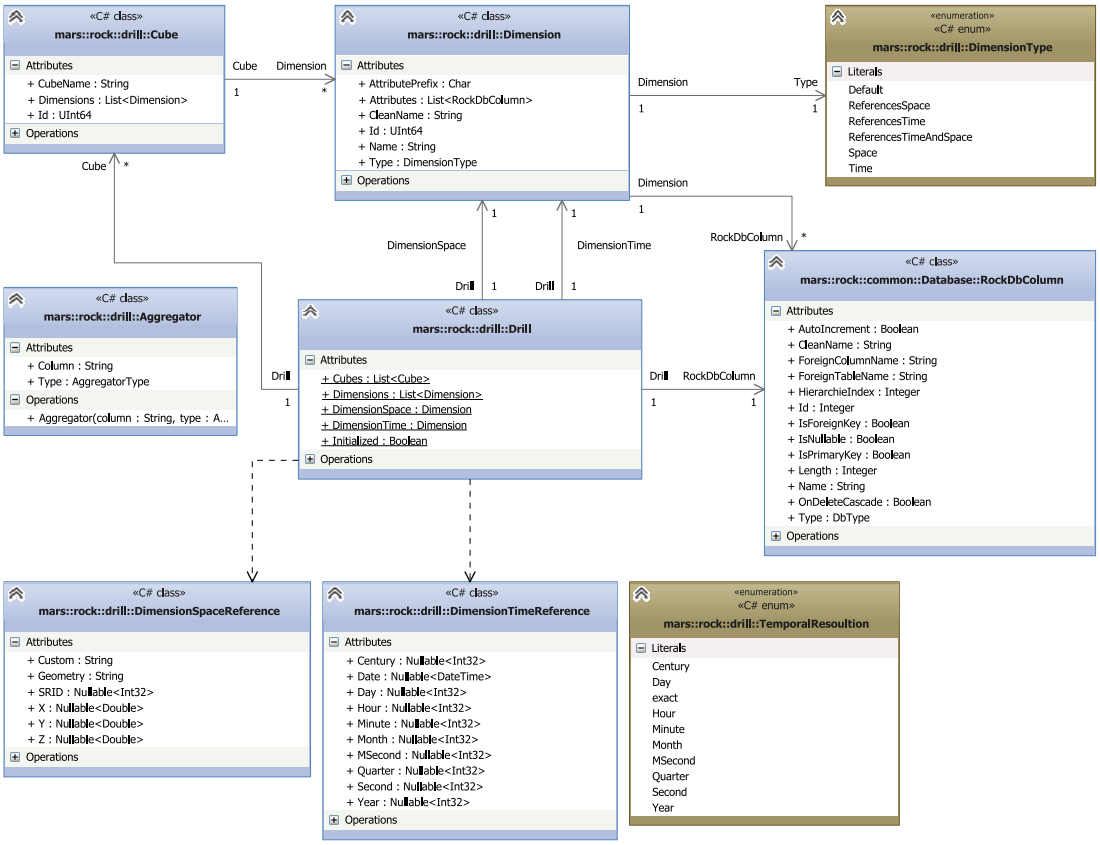

Abbildung 5.7.: Klassendiagramm Drill API

den Spalten der zu erzeugenden Dimension. Drill kommuniziert mit der Datenbank und nimmt die nötigen Einträge vor. Der Aufrufer der Methode erhält eine Instanz der Klasse *Dimension* zurück, welche dem aktuellen Stand der eingefügten Dimension in der Datenbank entspricht.

Import von Daten

Beim Einfügen von Daten werden neben der Dimension, in welche die Daten eingefügt werden sollen und den einzufügenden Daten, zwei weitere Parameter benötigt. Entsprechend dem SDS müssen Referenzen des einzufügenden Datensatzes auf Zeit und Raum angegeben werden. Hierzu dienen die Klassen *DimensionSpaceReference* und *DimensionTimeReference*, welche in Abbildung 5.7 (S. 45) dargestellt sind. Instanzen der beiden Klassen werden, wie in Abbildung 5.9 (S. 47) dargestellt, als Attribut der Methode *AddDataToDimension* übergeben. Sie implementieren verschiedene Operationen, welche die temporalen und spatialen Informationen automatisiert mit weiteren Informationen

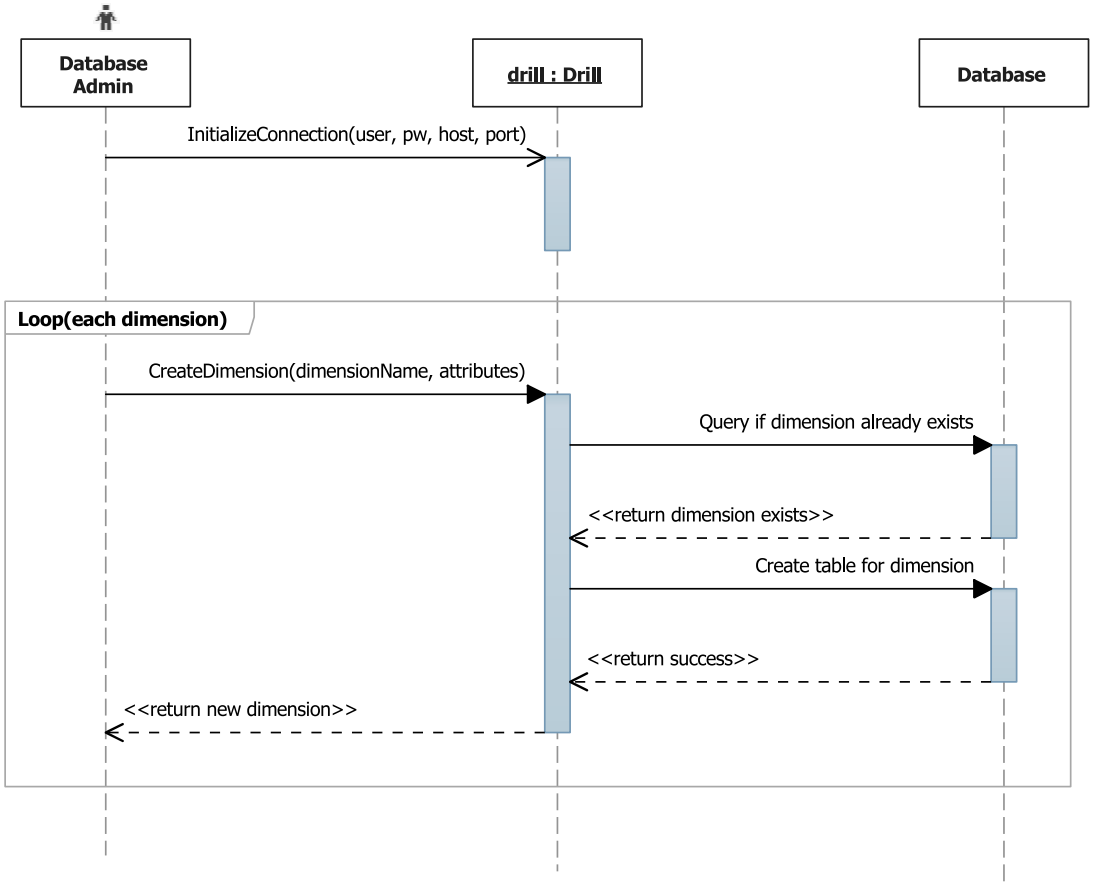

Abbildung 5.8.: Sequenzdiagramm - Erstellung von Dimensionen mit Drill

anreichern. So werden beispielsweise aus spatialen Informationen QuadTiles berechnet und aus temporalen Informationen das Jahrhundert, Quartal u.ä. vervollständigt.

Erstellung einer DFT

Die Erzeugung einer DFT ist in Abbildung 5.10 (S. 48) dargestellt. Hierfür muss die Erzeugung aller beteiligten Dimensionen vorausgegangen sein. Nach der obligatorischen Initialisierung der Datenbankverbindung wird der Methode *CreateCube* ein Bezeichner für die zu erstellende DFT sowie eine Liste von Dimensionen übergeben. Nach der Prüfung, ob der Bezeichner der DFT bereits vergeben ist, wird eine Sicht (englisch, SQL: View) entsprechend einer klassischen Faktentabelle mit den gegebenen Dimensionen erzeugt. Hierbei werden über die Referenzen der DynDims die zugehörigen Daten aus den ComDims hinzugefügt.

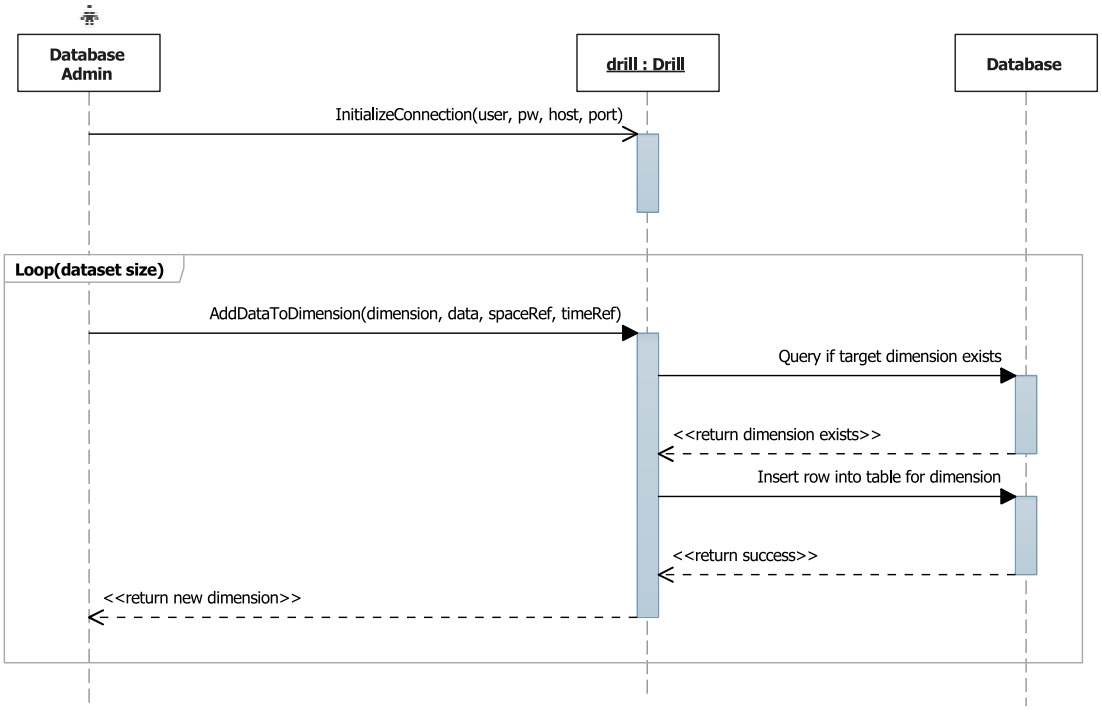

Abbildung 5.9.: Sequenzdiagramm - Einfügen von Daten

Neben der grundsätzlichen Erstellung einer DFT ist es weiterführend möglich, Dicing (siehe 2.5.4) durchzuführen. Hiermit können die Daten nach Zeit und Raum eingeschränkt werden und eventuell nicht benötigte Attribute herausgefiltert werden. So entsteht ein multidimensionaler Schnitt durch die Daten, welcher als Ergebnis eine Teilmenge der Daten der ursprünglichen DFT beinhaltet.

Abfrage von Daten

Die Datenabfrage verwendet die klassischen OLAP-Operationen auf einem Datenwürfel, wie er sich durch die DFTs definiert. Hierarchie- bzw. Aggregationsstufen entsprechen hierbei den Hierarchieebenen der spatio-temporalen Dimensionen. Verschiedene Methoden stehen zur Abfrage von Daten zur Verfügung. So können mit Hilfe der Methode *GetDimensionsByGeometry* Dimensionen ermittelt werden, welche Daten im Suchareal enthalten. Dies ist vor allem beim Explorieren der Daten für die Erstellung eines Szenariums innerhalb von MARS interessant. Nach der Erstellung einer DFT ist es möglich, mit verschiedenen Methoden die Daten der DFT abzufragen. Hierfür steht die Methode *GetData* mit diversen Überladungen zur Verfügung. Je nach

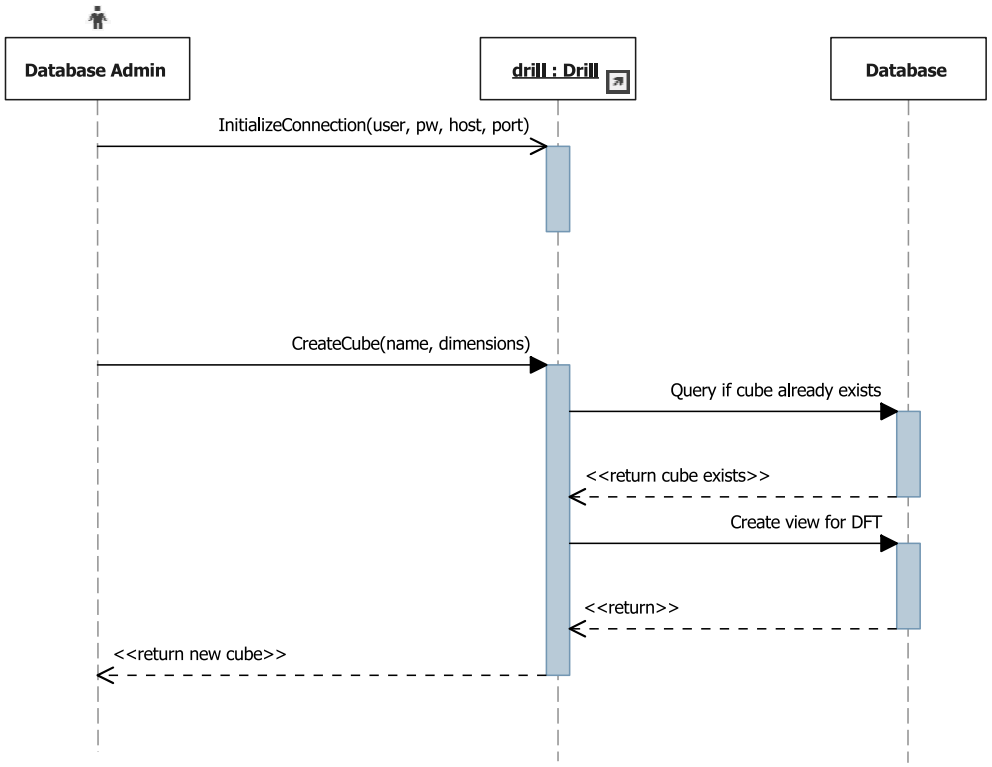

Abbildung 5.10.: Sequenzdiagramm - Erzeugung einer DFT

Überladung sind verschiedene Einschränkungen mit Hilfe spatialer und temporaler Parameter möglich. Hierfür kann der Raum mit QuadTiles bzw. einer Geometrie eingeschränkt werden. Zusätzlich kann der Zeitraum über die Angabe von Anfangs- und Endzeitpunkt eingeschränkt werden. Auch die Angabe eines genauen Zeitpunktes ist möglich. Die Abfrage von Daten und die möglichen Überladungen werden in Kapitel 6 eingehend erläutert.

6. Realisierung

Das folgende Kapitel beschreibt die Realisierung von MARS Rock und der zugehörigen API Drill, wie sie sich aus den Anforderungen aus Kapitel 4 (S. 22) und der Architektur aus Kapitel 5 (S. 28) ergeben. In Abschnitt 6.1 (S. 49) wird die Infrastruktur beschrieben, auf dessen Basis eine verteilte Datenbank realisiert wird. Darauf folgen Details zur Implementierung der Drill API, ihrer Anwendung und der Organisation von Metadaten zur Verwaltung von Rock.

6.1. Hardwareinfrastruktur

Dem Forschungsprojekt MARS stehen zur Zeit verschiedene Server für die Anwendungen des Frameworks zur Verfügung. Zur optimalen Nutzung der Hardwareressourcen sind alle Server in einem Virtualisierungscluster zusammengefasst. Die verwendete Hardware ist im Diagramm 6.1 (S. 50) der Infrastruktur von MARS dargestellt. Zum aktuellen Zeitpunkt, stehen vier Apple MacPro Server und ein Serverrechner aus Standardkomponenten für Forschungszwecke zur Verfügung. Zur Speicherung größerer Datenmengen steht ein Storagesystem mit einer Kapazität von vierzig Terabyte zur Verfügung.

Die Rechenknoten sind jeweils mit zwei Gigabit Ethernet Ports über Link Aggregation mit dem Netzwerk verbunden. Das Storagesystem ist hingegen mit zwei Ethernet Ports mit jeweils 10 Gigabit per Link Aggregation angebunden. Durch diese Hardwarekonstellation ist es jedem Knoten theoretisch möglich mit 250MB/s Daten über das Netzwerk zu übertragen. Durch die deutlich leistungsfähigere Anbindung des Storagesystems sind hier sogar Übertragungsraten von bis zu 2500MB/s möglich.

Für den Einsatz der verfügbaren Hardware als Cluster für Virtualisierungen ist auf jedem Knoten im Netzwerk das Betriebssystem Ubuntu 14.04.1 LTS mit KVM (siehe 5.1.2) installiert. Als Verwaltung kommt OpenNebula (siehe 5.1.3) und die dazugehörige Verwaltungsoberfläche Sunstone zum Einsatz, welche in Abbildung 6.2 (S. 51) dargestellt ist. Mit dieser Kombination aus Linux KVM und OpenNebula lässt sich die Hardware flexibel einsetzen und verwalten.

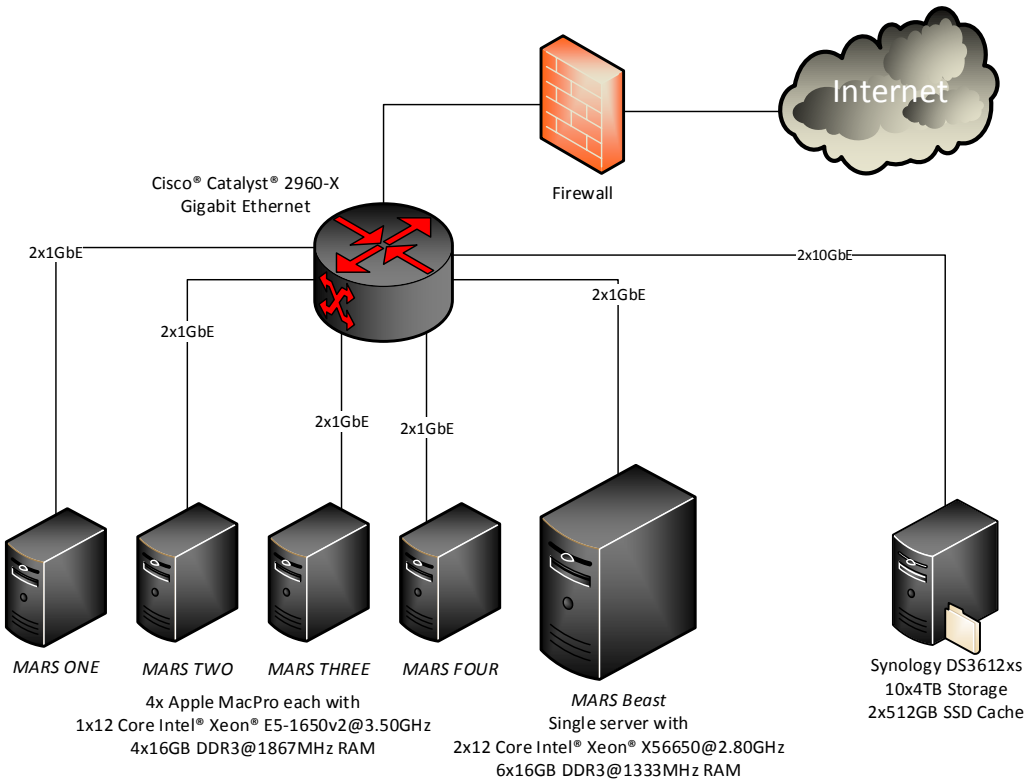

Abbildung 6.1.: MARS Infrastruktur

6.1.1. Virtuelles MariaDB Galera Cluster

Das zuvor beschriebene Cluster ist die Basis des Galera Clusters, welches aus mehreren virtuellen Maschinen besteht. Die Anzahl der virtuellen Knoten entspricht hierbei der Anzahl der physikalischen Knoten im Cluster. Diese Architekturentscheidung basiert darauf, dass die Performanz weniger von der Anzahl der Knoten als vielmehr von der Anzahl der CPU-Kerne je Knoten abhängt. So macht es Sinn, jeden physikalischen Knoten mit jeweils einem virtuellen Knoten voll auszureizen statt mehrere Knoten mit geringerer virtueller Hardware auszustatten. Die Konfiguration der virtuellen Maschinen ist in Tabelle 6.1 (S. 52) dargestellt.

Wie in Abschnitt 5.1.1.1 (S. 29) beschrieben ist, werden mindestens drei Knoten für den Einsatz eines MariaDB Galera Clusters benötigt, um eine konsistente Datenbasis zu gewährleisten. Jeder weitere Knoten erhöht die Ausfallsicherheit und die Performanz des Clusters. Es ist möglich, weitere Knoten zur Laufzeit hinzuzufügen und zu

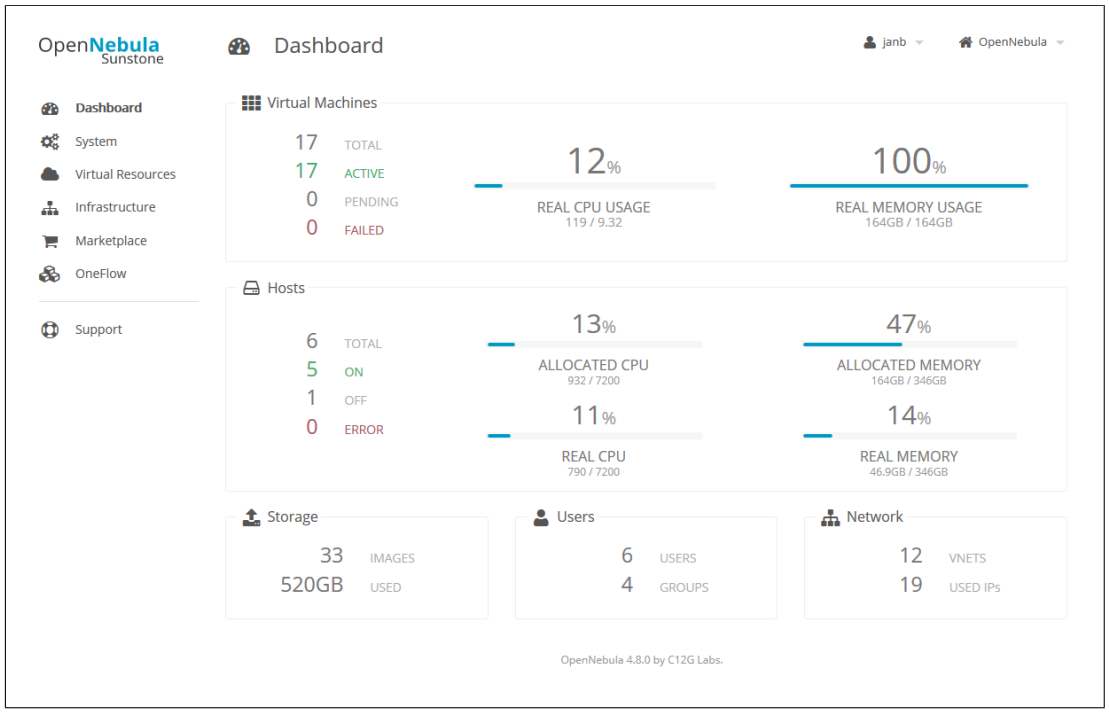

Abbildung 6.2.: Weboberfläche OpenNebula Sunstone

stoppen. So lässt sich das Datenbankcluster dynamisch an die Anforderungen anpassen.

Die virtuelle Maschine *Rock Balancer* aus Tabelle 6.1 (S. 52) dient als Lastverteiler zwischen den Knoten im Cluster. Hier kommt der GLB (siehe 5.1.1.2) zum Einsatz. Die Gewichtung für die Lastverteilung entspricht der verfügbaren virtuellen Hardware der Knoten und kann Tabelle 6.1 (S. 52) entnommen werden. Die virtuellen Maschinen *Rock One* und *Rock Balancer* werden parallel auf der physikalischen Maschine *MARS Beast* ausgeführt, da diese Maschine mehr CPU-Kerne zur Verfügung stellt als die restlichen physikalischen Knoten. Somit ist gewährleistet, dass die Lastverteilung selbst nicht einen der Knoten des Datenbankclusters in seiner Leistungsfähigkeit beeinflusst.

6.2. Drill API

Zur Verwendung von MARS Rock innerhalb des Frameworks stellt Drill diverse Schnittstellen zur Verfügung, mit welchen das DWH zur Speicherung von multiskaligen, heterogenen ökologischen Daten für Simulationszwecke angebunden werden kann. Der Ausgangspunkt innerhalb der Drill API ist die Klasse *Drill* im Namensraum

Knoten (virtuell)	Knoten (physikalisch)	vCPU Cores	vRAM	Gewicht (GLB)
Rock One	MARS Beast	12	4GB	1
Rock Two	MARS One	12	4GB	1
Rock Three	MARS Two	12	4GB	1
Rock Four	MARS Three	12	4GB	1
Rock Five	MARS Four	12	4GB	1
Rock Balancer	MARS Beast	12	2GB	n/a

Tabelle 6.1.: MARS Rock - Virtuelle Hardwarekonfiguration

mars.rock.drill. Sie bietet verschiedene statische Methoden zur Initialisierung und Verwendung der Datenbank. Verschiedene Anwendungsfälle werden in den folgenden Abschnitten näher erläutert.

6.2.1. Initialisierung

Für den Verbindungsaufbau mit der Datenbank müssen mindestens die IP-Adresse bzw. der Hostname, auf welchem sich die Datenbank befindet, angegeben werden sowie einen gültigen Benutzernamen mit Passwort. Weitere Parameter wie der zu verwendende Port und Datenbankprovider sind optional. Standardmäßig wird von einer MySQL/MariaDB ausgegangen, welche auf Port 3306 Anfragen entgegen nimmt. Abweichend davon werden weitere relationale DBMS unterstützt, welche im Namespace *mars.rock.common.Database.DbProviderFactoryDescription* liegen.

```
1 // Initializes the connection to the rock database. Default data
     provider is MySQL/MariaDB.
2 static void InitializeConnection(string host, string user, string
     password, int port = 3306, string providerInvariant =
     MySQLProviderDescription.ProviderInvariant);
3
4 //Example usage of method above
5 static void Main()
6 {
7   Drill.InitializeConnection("rock.mars.haw-hamburg.de", "user", "
       password");
8 }
```

Quelltext 6.1: Interface zur Initialisierung von Drill

Hier finden sich neben dem Datenprovider für MariaDB/MySQL weitere Provider für Microsoft SQL Server und PostgreSQL. Die Signatur der Methode ist in Quelltext 6.1 (S. 52) angegeben. Durch den Aufruf wird die *DbConnectionFactory* initialisiert und abhängig vom gewählten Datenprovider der entsprechende *connection string* generiert.

Es wird eine Testverbindung zur Datenbank aufgebaut, um die gegebenen Parameter zu überprüfen. Schlägt der Verbindungsaufbau fehl, wird ein Ausnahmefehler verursacht.

6.2.2. Erzeugung von Dimensionen

```
1  // Creates a new dimension with the given name, attributes and
       optional DimensionType.
2  // Throws an exception if the arguments are null or empty. Throws an
        exception if there's already a dimension with the given name.
3  static Dimension CreateDimension(string dimensionName, RockDbColumn
       [] attributes, Dimension.DimensionType type = Dimension.
       DimensionType.ReferencesTimeAndSpace);
4
5  //Example usage of method above
6  static void Main()
7  {
8    Dimension dimTree = Drill.CreateDimension(
9      "tree",
10     new RockDbColumn[] {
11         new RockDbColumn { Name = "patch", Type = DbType.String },
12         new RockDbColumn { Name = "species", Type = DbType.String },
13         new RockDbColumn { Name = "family", Type = DbType.String },
14         new RockDbColumn { Name = "height", Type = DbType.UInt16 }
15     }
16   );
17 }
```

Quelltext 6.2: Erstellen von Dimensionen

Die Erstellung von Dimensionen erfolgt über die statische Methode *CreateDimension* der Klasse *Drill*. Die Signatur und Verwendung ist in Quelltext 6.2 (S. 53) dargestellt. Die Methode benötigt den Namen für die zu erstellende Dimension sowie die Attribute der Dimension. Der Typ der Dimension ist optional und der Standardwert für diesen Parameter ist *ReferencesTimeAndSpace*, was bedeutet, dass die Dimension eine spatiale und temporale Referenz erhält. Für die Definition der Attribute einer Dimension kommt ein Array der Klasse *RockDbColumn* aus dem Namensraum *mars.rock.common.database* zum Einsatz. Die Klasse *RockDbColumn* implementiert gängige Parameter für Spalten in einer Datenbank. Hierzu gehören die Eigenschaften Primär-/Fremdschlüssel, das Erlauben von Nullwerten, Datentyp und Länge des Datentyps. Alle gängigen Datenbanktypen werden hierbei unterstützt.

6.2.3. Einfügen von Daten

Daten können über die Methode *AddDataToDimension* zu einer Dimension hinzugefügt werden. Hierzu müssen Dimension, die einzufügenden Werte und die räumlichen bzw. zeitlichen Referenzen angegeben werden. Zur Übergabe der Werte wird ein Array

der Klasse *RockDbValue* aus dem Namensraum *mars.rock.common.database* erwartet. Die Klasse *RockDbValue* enthält neben dem einzufügenden Wert Informationen über Datentyp und Namen der Spalte, in welche der Wert eingefügt werden soll.

```
1  // Adds the given datarow to the given dimension and time/space
       reference.
2  // Throws an exception if there's no dimension matching the given
       instance or another error occured.
3  static void AddDataToDimension(Dimension dimension, RockDbValue[]
       row, DimensionSpaceReference spaceRef, DimensionTimeReference
       timeRef);
4
5  //Example usage of method above
6  static void Main()
7  {
8    Dimension dimTree = Drill.GetDimension("tree");
9
10   Drill.AddDataToDimension(dimTree,
11     new RockDbValue[] {
12       new RockDbValue("patch", DbType.String, "R26"),
13       new RockDbValue("species", DbType.String, "Acacia␣
             erhenbergiana"),
14       new RockDbValue("family", DbType.String, "Mimosaceae"),
15       new RockDbValue("height", DbType.UInt16, 25),
16     },
17     new DimensionSpaceReference()
18     {
19       Z = 254.0,
20       X = 8.59321874,
21       Y = 1.262318562,
22       Custom = "awr"
23     },
24     new DimensionTimeReference()
25     {
26       Year = 2007,
27       Month = 5,
28       Day = 22,
29       Hour = 12,
30       Minute = 22,
31       Second = 44
32     }
33   );
34 }
```

Quelltext 6.3: Einfügen von Daten

6.2.4. Erzeugen einer Dynamic Fact Table

```
1  // Creates a cube/dft in the database with the given name and
       referencing the given dimensions.
2  // Throws an exception if there's already a cube with that name.
3  static Cube CreateCube(string name, Dimension[] dimensions);
4
5  //Example usage of method above
6  static void Main()
7  {
8    Dimension dimTree = Drill.GetDimension("tree");
9    Dimension dimAnimal = Drill.GetDimension("animal");
10   Dimension dimWeather = Drill.GetDimension("weather");
11
12   Cube testCube = Drill.CreateCube("test_cube", new Dimension[] {
         dimTree, dimAnimal, dimWeather });
13 }
```

Quelltext 6.4: Erzeugung eines Datenwürfels (DFT)

Bei der Erzeugung einer DFT werden als Parameter lediglich die beteiligten Dimensionen benötigt, wie Quelltext 6.4 (S. 55) zeigt. Hier wird ein Datenwürfel aus den Dimensionen *tree*, *animal* und *weather* gebildet. Zu beachten ist, dass die erzeugte DFT zusätzlich zu den hier angegebenen DynDims auch Referenzen auf die beiden ComDims *space* und *time* hält. Diese müssen nicht zusätzlich angegeben werden, da sie sich durch die jeweiligen Referenzen der DynDims ergeben.

Das Ergebnis der Methode ist eine Instanz der Klasse *Cube* (Namensraum *mars.rock.drill*), welche diverse Möglichkeiten zur Abfrage der in der DFT referenzierten Daten bietet. Diese Methoden werden in den folgenden Abschnitten behandelt.

6.2.5. Persistentes Dicing von DFTs

Neben der einfachen Erstellung von DFTs ist es möglich, Dicing-Operationen auf vorhandenen DFTs durchzuführen und das Ergebnis zu persistieren. Die verschiedenen Methodensignaturen hierzu sind in Quelltext 6.5 dargestellt. So wird durch die Dicing-Operation eine neue DFT erstellt, welche auf einer bestehenden basiert und diese über verschiedene Attribute einschränkt. Diese einschränkenden Attribute können sich auf Zeitraum und Ort beziehen, auf das Filtern einzelner Attribute von beteiligten Dimensionen und auf das Filtern ganzer Dimensionen. So lässt sich die Datenmenge auf einen bestimmten Bereich einschränken, welcher für Analyse und Simulation benötigt wird. Das Beispiel aus Quelltext 6.5 erzeugt einen kleineren Würfel, welcher neben den Standarddimensionen für Zeit und Raum nur noch die beiden Attribute *height* und *species* der Dimension *tree* referenziert. Zusätzlich ist der Zeitraum auf die Jahre 2007 bis 2012 eingeschränkt.

```
1  // Creates a cube/dft based on an existing one in the database. The
       resulting diced cube references only the given dimensions/column
       combination.
2  public static Cube DiceCube(Cube parent, string name, Dictionary<
       Dimension, List<RockDbColumn>> columns)
3
4  // Creates a cube/dft based on an existing one in the database. The
       resulting diced cube references only the given dimensions/column
       combination constrained to the given timerange.
5  public static Cube DiceCube(Cube parent, string name, Dictionary<
       Dimension, List<RockDbColumn>> columns, DateTime from, DateTime
       to, DimensionTimeReference.TemporalResoultion tempRes)
6
7  // Creates a cube/dft based on an existing one in the database. The
       resulting diced cube references only the given dimensions/column
       combination constrained to the given timerange.
8  public static Cube DiceCube(Cube parent, string name, Dictionary<
       Dimension, List<RockDbColumn>> columns, DateTime from, DateTime
       to, DimensionTimeReference.TemporalResoultion tempRes)
9
10 // Creates a cube/dft based on an existing one in the database. The
       resulting diced cube references only the given dimensions/column
       combination constrained to the given timerange and location.
11 public static Cube DiceCube(Cube parent, string name, Dictionary<
       Dimension, List<RockDbColumn>> columns, IGeometry position,
       DateTime from, DateTime to, DimensionTimeReference.
       TemporalResoultion tempRes, int srid = 4326)
12
13 //Example usage
14 static void Main()
15 {
16   Cube testCube = Drill.GetCube("test_cube");
17   var columns = new Dictionary<Dimension, List<RockDbColumn>>();
18   var tree = cube.Dimensions.First(dim => dim.CleanName == "tree");
19   var treeAttr = new List<RockDbColumn>
20   {
21     tree.Attributes.First(attr => attr.CleanName == "height"),
22     tree.Attributes.First(attr => attr.CleanName == "species")
23   };
24   columns.Add(tree, treeAttr);
25   Cube testCubeDiced = Drill.DiceCube(testCube, "test_cube_diced",
       columns, new DateTime(2007, 1, 1), new DateTime(2012, 1, 1),
       DimensionTimeReference.TemporalResoultion.Year);
26 }
```

Quelltext 6.5: Dicing eines Datenwürfels (DFT)

6.2.6. Approximation eines geometrischen Objektes

```
1 // Approximates a given geometric object with a list of quads
2 string[] FindQuads(IGeometry geometry, bool forceFullyCovered = true
      , double minPercCovered = 1, int maxZoomLvl = 23);
3 string[] FindQuads(string geoJsonGeometry, bool forceFullyCovered =
      true, double minPercCovered = 1, int maxZoomLvl = 23);
```

Quelltext 6.6: Approximation eines geometrischen Objektes

Für die Approximation eines geometrischen Objektes mit einer Menge von QuadTiles bietet die Klasse *MSQuadTile* im Namensraum *mars.rock.common.GIS* die statische Methode *FindQuads*. Der dahinter stehende Algorithmus ist in Kapitel 5.3.1.3 (S. 39) beschrieben. Als spatialen Parameter benötigt die Methode ein Objekt vom Typ *GeoAPI.Geometries.IGeometry*. Eine alternative Überladung akzeptiert ein geometrisches Objekt im GeoJSON-Standard[1]. Weitere Parameter sind optional und bieten die Möglichkeit den Algorithmus auf Kosten der Genauigkeit zu beschleunigen. So lässt sich mit dem Parameter *minPercCovered* angeben, zu welchem Prozentsatz eine Überdeckung eines QuadTiles durch die Ausgangsgeometrie gegeben sein muss, um in der Ergebnismenge enthalten zu sein und die Suche vorzeitig zu beenden. Die Methodensignaturen sind in Quelltext 6.6 (S. 57) aufgeführt.

6.2.7. Abfrage von Daten

Zur Abfrage von Daten einer DFT stellt die Klasse *Cube* verschiedenen Methoden zur Verfügung. Die Methodensignaturen finden sich in Quelltext 6.7 (S. 59). Die verschiedenen Abfragemöglichkeiten werden im Folgenden erläutert und Zeilenangaben beziehen sich auf den zuvor genannten Quelltext:

Zeile 7
Die Abfrage liefert die aggregierten Daten aller Dimensionen, welche vom Datenwürfel referenziert werden, ohne jedwede Einschränkungen.

Zeile 9
Mit der Angabe von Dimensionen wird eine Einschränkung der Daten vorgenommen. Es werden lediglich Daten der angegeben Dimensionen aus dem Datenwürfel zurückgegeben.

Zeile 11
Zusätzlich zu Dimensionen, werden hier temporale Einschränkungen bei der Datenabfrage berücksichtigt. So lassen sich Daten für einen Zeitpunkt abfragen und zusätzlich

[1] http://geojson.org/geojson-spec.html

die Auflösung des Zeitpunktes angegeben. Diese Auflösung entspricht den Hierarchieebenen der temporalen Dimension, wie sie in Abbildung 5.3 (S. 37) dargestellt sind. Die Angabe einer Aggregationsfunktion hat Auswirkungen, sobald zum gegebenen Zeitpunkt mehr als ein Datum existiert. So lässt sich beispielsweise der Durchschnitt oder das Maximum der abzufragenden Werte einer Spalte im Ergebnis abfragen.

Zeile 13
Diese Methode verhält sich entsprechend, jedoch gilt die temporale Einschränkung für einen Zeitraum statt eines Zeitpunkts.

Zeile 15
Diese Methode schränkt den Raum über die Angabe einer Menge von QuadTiles ein.

Zeile 17
Diese Methode schränkt den Raum über die Angabe eines geometrischen Objektes ein. Zusätzlich dazu kann optional eine Spatial Reference Identifier (SRID) angegeben werden. Dieser gibt das Koordinatensystem des geometrischen Objektes an.

Zeile 19
Entsprechend der vorangegangenen Methoden, werden hier die Einschränkung von Raum und Zeit vereint. Die Einschränkungen ergeben sich aus geometrischem Objekt und Zeitraum.

Zeile 21
Identisch mit der vorherigen Methode, nur die temporale Einschränkung wählt einen genauen Zeitpunkt.

```
#From class mars.rock.drill.Drill
// Returns the cube with the given name.
// Throws an exception if there's no cube with the given name.
static Cube GetCube(string name);

#From class mars.rock.drill.Cube
DataTable GetData();

DataTable GetData(IEnumerable<Dimension> dimensions);

DataTable GetData(IEnumerable<Dimension> dimensions, DateTime
    pointInTime, DimensionTimeReference.TemporalResoultion
    temporalResoultion, IEnumerable<Aggregator> aggregators = null);

DataTable GetData(IEnumerable<Dimension> dimensions, DateTime from,
    DateTime to, DimensionTimeReference.TemporalResoultion tempRes,
    IEnumerable<Aggregator> aggregators = null);

DataTable GetData(IEnumerable<Dimension> dimensions, IEnumerable<
    MsQuadTile> quads, IEnumerable<Aggregator> aggregators = null);

DataTable GetData(IEnumerable<Dimension> dimensions, IGeometry
    position, int srid = 4326, IEnumerable<Aggregator> aggregators =
    null);

DataTable GetData(IEnumerable<Dimension> dimensions, IGeometry
    position, DateTime from, DateTime to, DimensionTimeReference.
    TemporalResoultion tempRes, int srid = 4326, IEnumerable<
    Aggregator> aggregators = null)

DataTable GetData(IEnumerable<Dimension> dimensions, IGeometry
    position, DateTime pointInTime, DimensionTimeReference.
    TemporalResoultion tempRes, int srid = 4326, IEnumerable<
    Aggregator> aggregators = null)
```

Quelltext 6.7: Abfrage von Daten - Methodensignaturen

6.2.8. SQL Schnittstelle

Neben den zuvor beschriebenen Möglichkeiten Daten über die Abstraktion von Drill einzufügen und abzufragen ist es möglich, eigene Anfragen im SQL-Format zu erstellen. Liegen beim Nutzer dieser Schnittstelle die nötigen Kenntnisse vor, lassen sich somit komplexere Anfragen realisieren.

```
1 //Executes the query and returns an instance of IDataReader to
     consume data directly from the database
2 static IDataReader ExecuteQuery(string sql);
3
4 //Executes the query and returns the count of affected row
5 static int ExecuteNonQuery(string sql);
```

Quelltext 6.8: Benutzerdefinierte Anfragen - Methodensignaturen

6.3. Metadaten zur Verwaltung von Rock

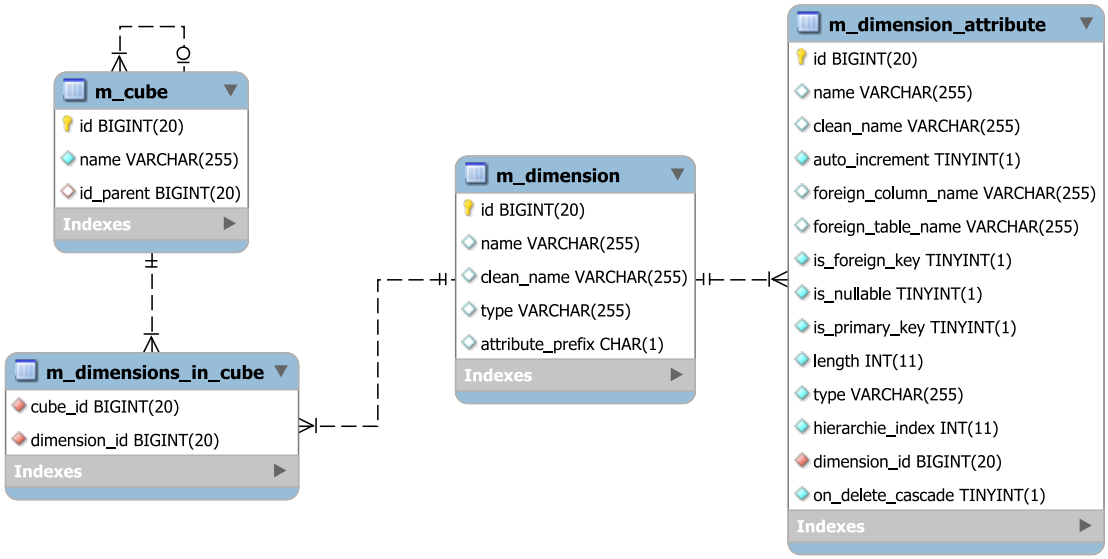

Abbildung 6.3.: Rock - Metadaten

Für die Verwaltung der Dimensions- und Faktentabellen in Rock existieren verschiedene Metadatentabellen. Mit Hilfe dieser Tabellen ist es der API Drill möglich, die vorhandenen Dimensionen und DFTs zu ermitteln, deren internen Aufbau zu inspizieren und Datenbankabfragen für diese Tabellen zu generieren. Das Schema dieser Metadaten ist in Abbildung 6.3 dargestellt. In dieser Abbildung sind die Abhängigkeiten der Tabellen untereinander zu erkennen und auch die vorhandenen Attribute der jeweiligen Tabellen.

Bei der Erzeugung von Dimensionen über Drill werden die entsprechenden Einträge in den Tabellen *m_dimension_attribute* und *m_dimension* vorgenommen. So werden die Attribute einer Dimension zusammen mit verschiedenen Informationen über diese Attribute erfasst. Hierzu zählen der Datentyp, Name und weitere Metainformationen,

welche Drill zur Erzeugung von Abfragen benötigt. Wichtig ist hier die *dimension_id*, welche die Referenz auf die zugehörige Dimension angibt.

Zur Dimension selbst werden, neben dem *clean_name*, welchen der Datenbankadministrator für die Dimension gewählt hat, der tatsächliche Name gespeichert. Dies ist nötig, da Dimensionen in der Datenbank immer mit dem Präfix „d_“ versehen werden. Auch die Attribute einer Dimension erhalten einen Präfix, um die Kollision von Attributnamen und Schlüsselwörtern in SQL zu vermeiden. Der Präfix für Attribute ergibt sich aus dem ersten Buchstaben des *clean_name* einer Dimension, sodass als Beispiel bei einer Dimension mit dem Namen „temperature“ alle Attribute den Präfix „t_“ erhalten. Die Dimension selbst wird in der Datenbank als „d_temperature“ gespeichert.

Bei der Definition einer DFT wird ein Eintrag in der Tabelle *m_cube* erzeugt mit dem Namen der DFT. Für die Referenzierung zwischen den DFTs und den Dimensionen werden Einträge in *m_dimensions_in_cube* erzeugt. Beim Dicing von DFTs wird ebenfalls ein Eintrag in der Tabelle *m_cube* erzeugt. Zusätzlich zu den bereits genannten Informationen, wird hier die ID der DFT im Feld *id_parent* gespeichert, auf welcher die durch das Dicing erzeugte DFT basiert.

7. Experimente

Das folgende Kapitel führt die Ergebnisse verschiedener Leistungsmessungen unterschiedlicher Komponenten auf, welche in Kapitel 5 (S. 28) und Kapitel 6 (S. 49) beschrieben sind. Weiterführend wird anhand des Anwendungsfalls von ARS AfricaE ein praktisches Beispiel für die Verwendung von MARS Rock gegeben.

7.1. Leistungsmessungen

Für die folgenden Leistungsmessungen wurden verschiedene Werkzeuge verwendet. So kommen bei der Leistungsmessung des Datenbankclusters der in Abschnitt 5.1.5 (S. 30) beschriebene SysBench OLTP Benchmark zum Einsatz. Des Weiteren wurden Messungen der Performanz des Algorithmus zur Approximierung von QuadTiles durchgeführt.

7.1.1. Algorithmus zur Approximierung von geometrischen Objekten

Der im Rahmen dieser Arbeit entwickelte Algorithmus zur Approximation von Geometrien durch QuadTiles muss im schlechtesten Fall $4^{23} = 70.368.744.177.664$ QuadTiles untersuchen. Dadurch ist die Komplexität des Algorithmus recht hoch. Mit Optimierungen wie Parallelisierung und Eingrenzung des Suchbereichs, d.h. der Bestimmung der Startebene, ist er jedoch praktikabel einsetzbar. Je nach System benötigt die Approximierung der Stadtgrenze von Berlin durch QuadTiles zwischen 18,01 und 23,3 Sekunden, wie in Abbildung 7.1 (S. 63) dargestellt ist. Die Berechnungsdauer hängt massiv von der Komplexität des Ausgangspolygons ab, welche bei politischen Grenzen meist sehr hoch ist. Approximierungen dieser Art lassen sich jedoch für wiederholte Anfragen zwischenspeichern bzw. für aktuell gültige Grenzen vorberechnen. Trotz dieser Möglichkeit bleibt der Nutzen des Algorithmus hinter den Erwartungen zurück, da auch die Anfrage an die Datenbank mit der logischen Disjunktion einer großen Anzahl von QuadTiles wenig performant ist.

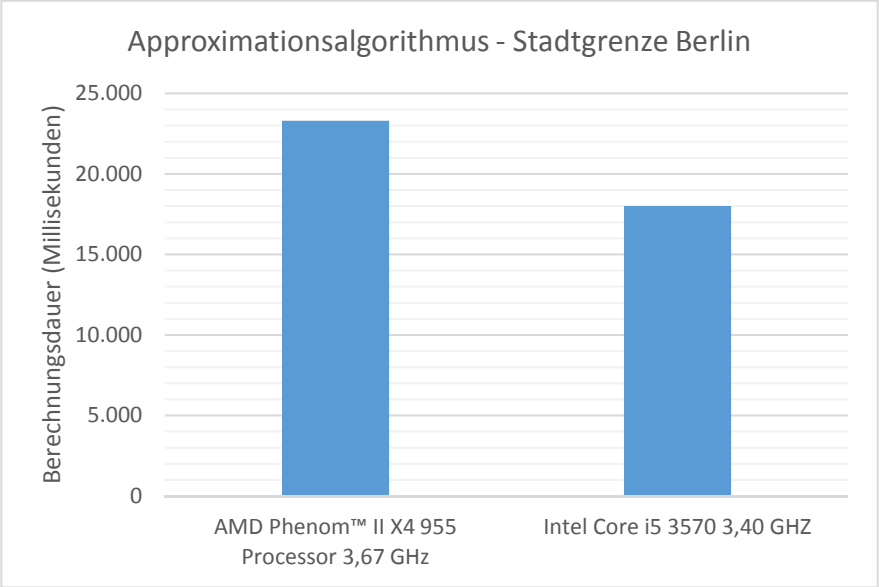

Abbildung 7.1.: Berechnungsdauer der Approximierung der Stadtgrenze von Berlin

7.1.2. MariaDB Galera Cluster

Die Leistungsmessung des Datenbankclusters wird mit Hilfe des etablierten SysBench OLTP Benchmarks durchgeführt, dessen Ergebnisse in Abbildung 7.2 (S. 64) dargestellt sind. Hier zeigt sich, dass die Performanz eines einzelnen Knoten die des Clusters mit 5 Knoten bis zu einem gewissen Grad an Parallelität übersteigt. Dieser Umstand ist eine logische Folge des Overheads, welcher bei der Lastverteilung entsteht. Steigt der Grad der Parallelität weiter an, übersteigt die Leistung des Cluster jedoch einen einzelnen Knoten und das Problem des Overheads der Lastverteilung wird durch die Verteilung selbst ausgeglichen. Somit ist zu erkennen, dass der Einsatz eines Clusters mit Hinblick auf die Performanz erst bei einem hohen Grad an Parallelität sinnvoll ist. Legt man das Augenmerk zusätzlich auf Ausfallsicherheit, ist der Einsatz eines Clusters auch bei einem geringeren Grad an Parallelität sinnvoll. Im Kontext von MARS treten während einer Simulation viele parallele Zugriffe auf, welche vom Einsatz eines Clusters profitieren.

Die Aussagekraft dieser Messung ist leider durch die Performanz von Netzwerk und Prozessor des anfragenden Clients limitiert, sodass angenommen werden kann, dass der parallele Zugriff von mehreren Clients ähnlicher Performanz den Geschwindigkeitsvorteil des Cluster deutlicher aufzeigen würde. Auch die Infrastruktur ist nicht

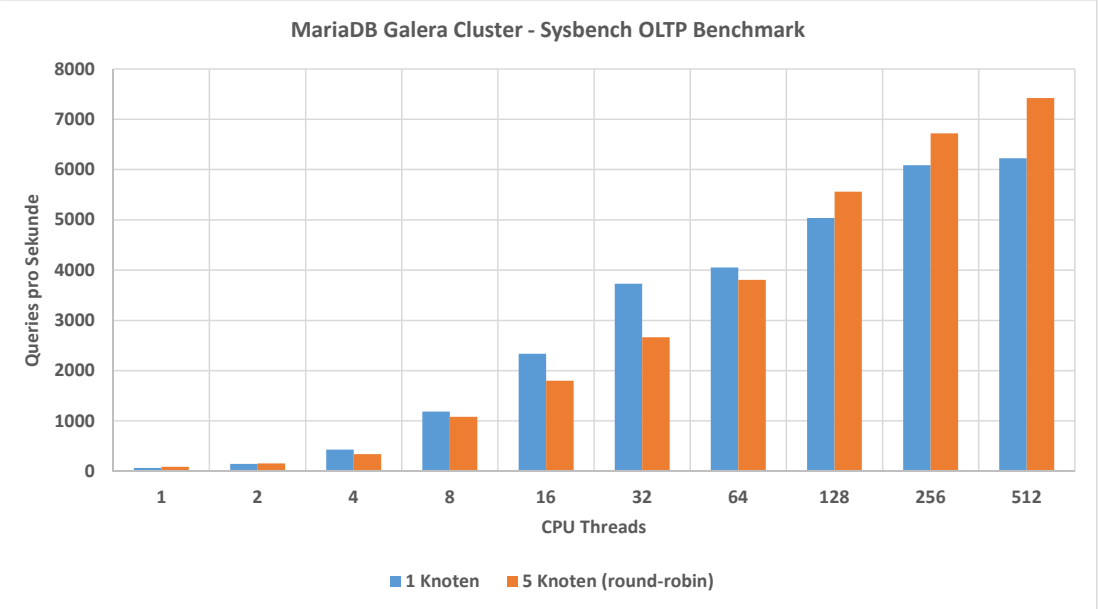

Abbildung 7.2.: Sysbench OLTP Benchmark - Abfragen pro Sekunde

optimal für eine Datenbanklösung. Die Anbindung des Speichers über das Netzwerk schafft einen Flaschenhals für die Leistungsfähigkeit des Clusters, der mit dedizierter Hardware, welche ausschließlich für den Betrieb der Datenbank vorgesehen ist, beseitigt werden könnte. Die Konkurrenz um Ressourcen mehrerer virtueller Maschinen mit unterschiedlichsten Diensten, welche parallel zu den Leistungsmessungen auf der vorhandenen Infrastruktur ausgeführt werden, hat ebenfalls negativen Einfluss auf die Datenbankperformanz.

7.2. Anwendungsfall ARS AfricaE

Das in Abschnitt 4.1.2 (S. 23) beschriebene Szenarium ARS AfricaE soll im folgenden Abschnitt als konkreter Anwendungsfall dienen, um die Hypothesen aus Kapitel 1 (S. 1) zu beweisen. Hierfür kommen die in dieser Arbeit geschaffenen Komponenten MARS Rock und die API Drill zur Anwendung. Auch einige Werkzeuge aus Abschnitt 5.1 (S. 28) werden für einen Teil des Anwendungsfalls verwendet.

7.2.1. Untersuchungsgebiet

Zum Aufbau eines individuenbasierten Modellsystems zur Vorhersage von Ökosystemdynamiken im Kontext von ARS AfricaE kommt MARS zum Einsatz. Hierfür stehen

Daten für einen Teilbereich des KNP in der Nähe des Hauptcamps Skukuza zur Verfügung. Der KNP hat eine Fläche von 20.000 km² und dient dem Erhalt der Tier- und Pflanzenwelt. Das Untersuchungsgebiet selbst liegt innerhalb des KNP zwischen 24°59'24.5"S 31°28'38.5"E und 25°01'37.5"S 31°31'03.5"E auf einer Höhe von 365m über Normalnull, mit einer Fläche von 16,7 km². Das Klima ist semiarid subtropisch, mit heißen, regnerischen Sommern und warmen, trockenen Wintern. Der durchschnittliche Niederschlag liegt bei 550mm pro Jahr. Dieses Gebiet und die beiden Messtürme für Temperatur- und Niederschlagsdaten sind in Abbildung 7.3 (S. 65) dargestellt. Hier ist zu erkennen, dass der Flux Tower für Temperaturdaten innerhalb des Untersuchungsgebietes liegt. Der Messturm für Niederschlagsdaten liegt in der näheren Umgebung innerhalb von Skukuza. (Scholes u. a., 2001)

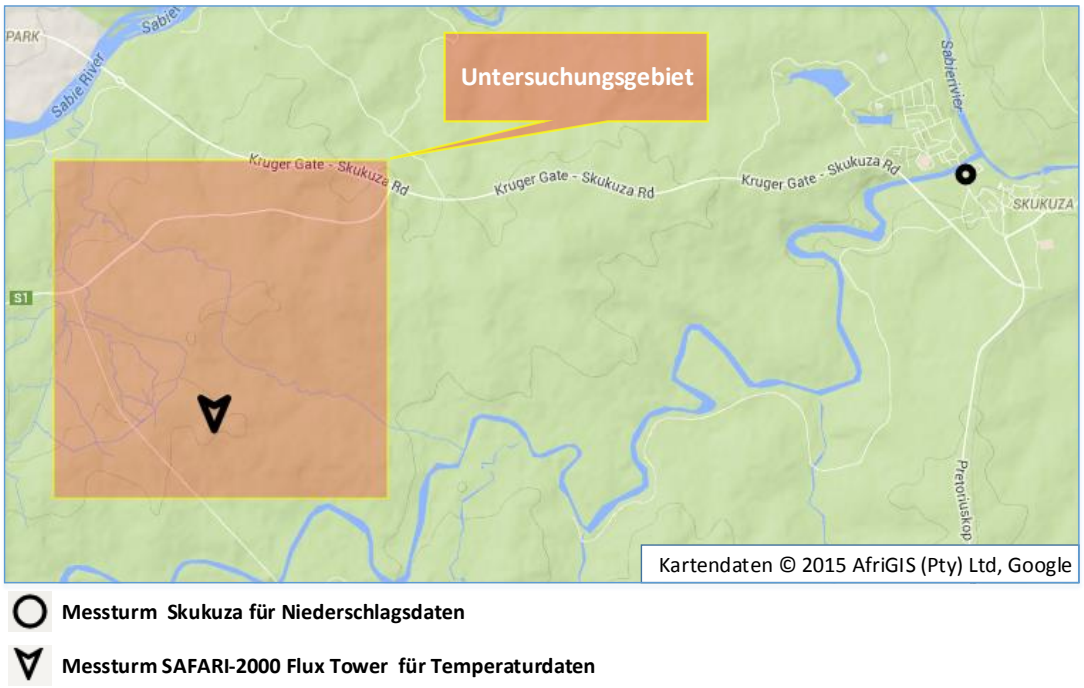

Abbildung 7.3.: Untersuchungsgebiet Skukuza

7.2.2. Zielsetzung und Vorgehensweise

Das Ziel ist es, ein Modell zu entwickeln, mit dessen Hilfe sich Vorhersagen über die Resilienz des Ökosystems, nach dem Eingriff durch den Menschen, treffen lassen. Hierbei soll ein besonderes Augenmerk auf die Resilienz der häufigsten Baumarten

und den Einfluss der Elefantenpopulation auf diese Baumarten gelegt werden. Hierzu entwickelt Ulfia Lenfers im Zuge Ihrer Dissertation die für die Simulation nötigen IBMs. Hierbei wird sie durch das MARS-Team unterstützt. Die Erarbeitung von Wachstumsmodellen für Bäume, auf Basis der vorliegenden Daten und Verhaltensmodelle der Elefanten, liegen im Fokus. Die Datenintegration und Anbindung an die Modelle ist ein wichtiger Punkt für das Erreichen des Ziels, welcher im Folgenden näher betrachtet wird.

Station	Year	Month	Day	Rainfall
SKZ	2009.00	1.00	1	
SKZ	2009.00	1.00	2	
SKZ	2009.00	1.00	3	1.60
SKZ	2009.00	1.00	4	0.10
SKZ	2009.00	1.00	5	0.00
SKZ	2009.00	1.00	6	

Tabelle 7.1.: Auszug aus den Niederschlagsdaten für Skukuza - Beispiel 1

Zunächst werden die vorliegenden Daten aus den verschiedenen Quellen für Niederschlag und Temperatur im Untersuchungsgebiet gesichtet. Die nötigen Bereinigungen und Transformationen der Daten sollen mit Hilfe des in Abschnitt 5.1.6 (S. 30) beschriebenen Werkzeugs Pentaho Kettle durchgeführt werden. Daraufhin werden die vorbereiteten Daten mit Hilfe der API Drill in das DWH Rock importiert. Erste Analysen der Daten werden durchgeführt und veranschaulicht, wie diese Daten in der Simulation verwendet werden.

Datenanalyse Niederschlagsdaten

Die Niederschlagsdaten liegen über mehrere Dateien verteilt vor. Auszüge aus diesen Dateien sind in den Tabellen 7.1 (S. 66), 7.2 (S. 67) und 7.3 (S. 67) dargestellt. In den Auszügen sind verschiedene Probleme zu erkennen. Die Darstellung von fehlenden Werten ist syntaktisch heterogen, d.h. fehlende Werte werden innerhalb der Daten unterschiedlich repräsentiert. In 7.1 (S. 66) sind sie durch Leerstellen gekennzeichnet. In 7.2 (S. 67) und 7.3 (S. 67) hingegen werden fehlende Werte sowohl durch Leerstellen als auch durch die Werte -999.90 und -777.70 repräsentiert. Die Daten müssen dahingehend bereinigt werden, dass eine einheitliche Darstellung fehlender Werte erreicht wird. Die Darstellung der Zeit innerhalb der Daten liegt ebenfalls syntaktisch heterogen vor. So wird ein Zeitpunkt in 7.1 (S. 66) und 7.2 (S. 67) durch getrennte Spalten für Tag, Monat und Jahr dargestellt, wohingegen eine alphanumerische Darstellung des

Datums in 7.3 (S. 67) vorliegt. Um eine einheitliche Darstellung der Zeit zu erreichen, müssen diese Daten transformiert werden.

Station	Code	Day	Month	Year	mm
Skukuza	SKZ	3	11	2010	
Skukuza	SKZ	4	11	2010	2
Skukuza	SKZ	5	11	2010	17
Skukuza	SKZ	6	11	2010	20.5
Skukuza	SKZ	7	11	2010	0.5
Skukuza	SKZ	8	11	2010	-999.90

Tabelle 7.2.: Auszug aus den Niederschlagsdaten für Skukuza - Beispiel 2

Datenanalyse Temperaturdaten

Daten zu gemessenen Temperaturen im Untersuchungsgebiet liegen in einer einzelnen Datei vor, deren Aufbau in 7.4 (S. 68) dargestellt ist. Die Temperatur wurde in Kelvin erfasst. Die Zeit ist alphanumerisch dargestellt, jedoch in einem anderen Format als zuvor. Die Daten sind frei von Nullwerten bzw. fehlen für Tage, an denen keine Messung vorliegt.

7.2.3. Durchführung

Basierend auf den zuvor gesichteten Daten wird nun mittels Pentaho Kettle eine Bereinigung und Transformation der Daten durchgeführt. Hierfür können verschiedene Operatoren über die Oberfläche von Kettle miteinander verbunden werden. In Abbildung 7.4 (S. 69) ist das Netzwerk aus Operatoren zur Transformation der Niederschlagsdaten dargestellt. Auf der linken Seite sind vier Operatoren zum Einlesen von CSV-Daten zu sehen. Auf der rechte Seite mündet das Netzwerk in einem Operator zum

STATION	DATE	MM
SKZ	9/1/1911 0:00:00	-777.70
SKZ	10/22/1911 0:00:00	0.30
SKZ	10/23/1911 0:00:00	0.30
SKZ	10/26/1911 0:00:00	1.50
SKZ	10/31/1911 0:00:00	4.30
SKZ	11/1/1911 0:00:00	-999.90

Tabelle 7.3.: Auszug aus den Niederschlagsdaten für Skukuza - Beispiel 3

date	value	quality	interpolated
2000-07-03	295.16	0	0
2000-07-11	296.92	0	0
2000-07-19	296.68	0	0
2000-07-27	296.1	0	0
2000-08-04	299.42	0	0
2000-08-12	289.04	65	0
2000-08-20	300.06	0	0
2000-08-28	307.06	0	0

Tabelle 7.4.: Auszug aus den Temperaturdaten für Skukuza

Schreiben von Text- bzw. CSV-Dateien. Die Operatoren zwischen Ein- und Ausgang bereinigen und transformieren Schritt für Schritt die Daten und bringen diese in eine einheitliche Struktur. Die „Select Value“- und „Concat“-Operatoren, welche nach dem Einlesen der Dateien folgen, transformieren die jeweiligen Datumsinformationen in eine einheitliche alphanumerische Darstellung mit dem Datumsformat „MM/dd/yyyy“. In einer der Dateien liegen die Datumsinformationen bereits im richtigen Format vor, sodass mit dem „Select Value“-Operator lediglich nicht benötigte Informationen entfernt werden. Mit dem „Filter“-Operator werden im folgenden Schritt die Daten anhand der Messstationsangabe gefiltert, sodass nur Daten der Station Skukuza weiterverwendet werden. Mit dem „Value Mapper“-Operator werden zum Schluss die Repräsentation von Nullwerten vereinheitlicht. Das Ergebnis dieser Transformation für Niederschlagsdaten ist in Tabelle 7.5 (S. 68) zu sehen.

Die Temperaturdaten müssen im Gegensatz dazu nicht weiter bereinigt bzw. transformiert werden. Das von den Niederschlagsdaten verschiedene Datumsformat stellt beim Importieren mit Drill kein Problem dar, da die verwendete Programmiersprache C# mit den verschiedenen Datumsformaten ohne Weiteres umgehen kann.

DATE	MM
2010/11/15	null
2010/11/16	3.5
2010/11/17	13.7
2010/11/18	3.8
2010/11/19	0.2
2010/11/20	null

Tabelle 7.5.: Auszug aus den transformierten Niederschlagsdaten für Skukuza

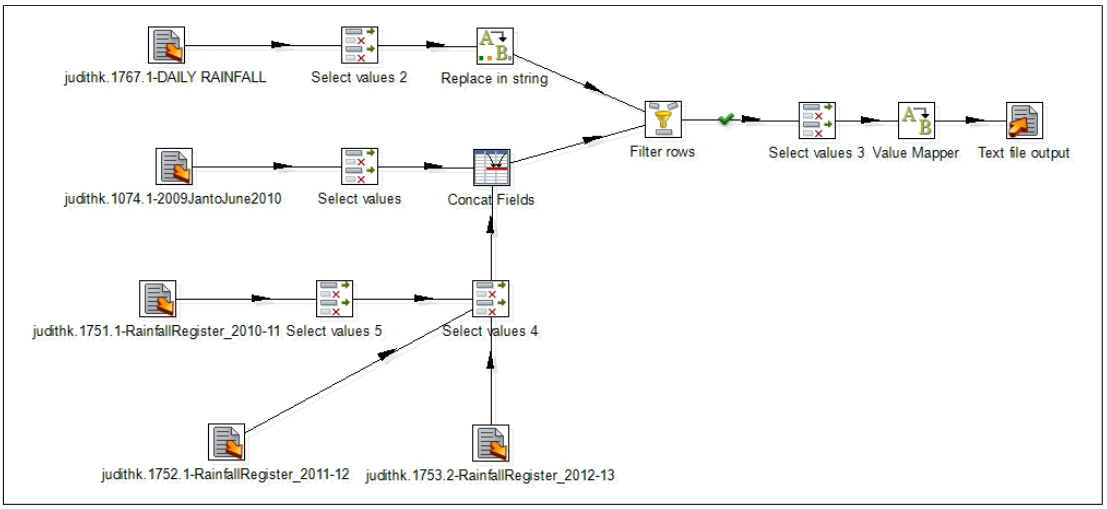

Abbildung 7.4.: Transformation der Niederschlagsdaten

Import der Daten

Da sich das Datenintegrationswerkzeug MARS Phobos (siehe 5.2.2) noch in der Entwicklung befindet, wurde für den konkreten Anwendungsfall ein simples Programm entwickelt, welches die Daten aus den CSV-Dateien einliest und über die Drill API in Rock speichert. Dieses Programm ist in C# geschrieben und ist ungekürzt in Anhang A.2 (S. 77) zu finden. Zeilenangaben im folgenden Abschnitt beziehen sich auf diesen Quellcode.

Zunächst wird in Zeile 4 die Datenbankverbindung mit entsprechenden Zugangsdaten und der IP-Adresse der Datenbank initialisiert. Daraufhin werden die benötigten Attribute für die Dimension definiert (Zeile 7-12), welche die Temperaturdaten aufnehmen wird. Die Attribute erhalten Namen, Datentyp und ggf. Länge. Der darauffolgende Schritt in Zeile 15 erzeugt mit Drill die Dimension innerhalb des DWH unter Angabe des Namen der zu erstellenden Dimension und der soeben definierten Attribute. In den Zeilen 18-24 werden Attribute und Dimension für die Niederschlagsdaten erzeugt. Die beiden erzeugten Dimensionen enthalten zu diesem Zeitpunkt noch keine Daten.

In Zeile 27 wird die CSV-Datei mit den Temperaturdaten eingelesen. Damit diese Daten in die Datenbank geschrieben werden können, werden in Zeile 30-52 die Daten der CSV-Datei zeilenweise abgearbeitet. Jede Zeile wird in eine Liste von Instanzen der Klasse *RockDbValue* konvertiert. Der etwas kryptische Ausdruck in Zeile 33 führt eben diese Konvertierung durch. Für das Einfügen von Daten in eine Dimension im

Shared Dimension Schema ist es erforderlich, Referenzen zu Zeit und Raum anzugeben. Hierfür muss zu jedem Datensatz eine solche Referenz angegeben werden. Die Definition der spatialen Referenz ist in Zeile 38 definiert. Das hier angegebene Areal entspricht dem Untersuchungsgebiet nahe Skukuza. Die temporale Referenz (Zeile 45) kommt hier aus den Daten der CSV-Datei selbst. In Zeile 51 werden die Daten über Drill unter Angabe von Dimension, spatio-temporaler Referenzierung und den Daten selbst in die Zieldimension eingefügt. Für die Niederschlagsdaten werden die gleichen Schritte in den Zeilen 55-86 durchgeführt.

Nachdem die Dimensionen für Niederschlag und Temperaturdaten erzeugt und mit den Daten aus den CSV-Dateien gefüllt wurden, wird nun eine Dynamic Fact Table für den Anwendungsfall ARS AfricaE erzeugt. Dies geschieht in Zeile 89 unter Angabe des Namens und der zugehörigen Dimensionen der DFT. Es ist nicht zwingend erforderlich vor der Erstellung der DFT die Daten zu importieren. Auch nach diesem Schritt lassen sich Daten zu den Dimensionen hinzufügen oder daraus entfernen. Auch die Anpassung der DFT, d.h. die Referenzierung von Dimensionen lässt sich jederzeit verändern.

Verwendung der Daten

Nachdem die Daten in Rock vorliegen, können diese nun innerhalb einer Simulation mit MARS Life (siehe 5.2.5) verwendet werden. Im konkreten Anwendungsfall werden die Temperatur- und Niederschlagsdaten für das Wachstumsmodell der Baumagenten benötigt. Hierfür werden die Daten über so genannte Layer innerhalb der Simulation zur Verfügung gestellt. Ein spezieller Layer für Daten aus dem DWH bietet hier eine Schnittstelle für andere Komponenten einer Simulation. Gemäß dem Anwendungsfall werden zwei Layer entwickelt, jeweils einer für Niederschlags- und Temperaturdaten. Innerhalb der Layer wird die Drill API verwendet. Als Schnittstellen zu anderen Layern und Agenten bieten diese Datenlayer verschiedene anwendungsspezifische Methoden. Diese Schnittstellen sind in Quellcode 7.1 und 7.2 dargestellt.

```
1 double GetTemperatureKelvin(IGeometry position, int year, int month,
      int day);
2 double GetTemperatureCelsius(IGeometry position, int year, int month
    , int day);
```

Quelltext 7.1: Schnittstelle Temperaturlayer

Eine mögliche Implementierung der Schnittstellen macht die temporale Auflösung an den Parametern für Jahr, Monat und Tag fest. Wird 0 als Parameter für den Tag angegeben, könnte z.B. der Monatsdurchschnitt zurückgegeben werden. Die Verwendung der Drill API lässt es dem Layerentwickler offen, wie er die Daten innerhalb der

Simulation verwendet.

```
double GetRainfallMillimeter(IGeometry position, int year, int month
    , int day);
```

Quelltext 7.2: Schnittstelle Niederschlagslayer

Eine Beispielimplementierung für die Methode *GetTemperatureCelsius* ist in 7.3 zu sehen. Hier wir das zuvor beschriebene Aggregieren der Daten umgesetzt. Gibt der Anfragende für den Monat oder den Tag 0 an, wird ein Durchschnittswert für die Temperatur berechnet. Es gilt zu beachten, dass hier keinerlei Fehlerprüfung stattfindet, was in einer konkreten Implementierung nicht ausgelassen werden sollte. Die vorliegende Implementierung ist vereinfacht dargestellt und nur im Zusammenhang mit den konkreten Daten aus dem Anwendungsfall ARS AfricaE funktional.

```
double GetTemperatureCelsius(IGeometry position, int year, int month
    = 0, int day = 0)
{
    var pointInTime = new DateTime(year, month == 0 ? 1 : month, day
        == 0 ? 1 : day);
    var tempResolution = day == 0 ?
      (month == 0 ? DimensionTimeReference.TemporalResoultion.Year :
          DimensionTimeReference.TemporalResoultion.Month)
      : DimensionTimeReference.TemporalResoultion.Day;
    var aggregation = tempResolution == DimensionTimeReference.
       TemporalResoultion.Day ? null : new[] { new Aggregator("
       t_value", Aggregator.AggregatorType.Average) };

    DataTable data = _cube.GetData(
      new[] { _tempDim },
      position,
      pointInTime,
      4326,
      tempResolution,
      aggregation);

    if (data.Rows.Count < 1) throw new Exception("No_data!");

    var result = Convert.ToDouble(data.Rows[0].ItemArray[data.
       Columns.IndexOf("t_value")]);//Kelvin
    result -= 273.15;//Celsius
    return result;
}
```

Quelltext 7.3: Beispielimplementierung für *GetTemperatureCelsius*

8. Diskussion

Dieses Kapitel diskutiert die Ergebnisse der Arbeit mit Hinblick auf die in Kapitel 1 (S. 1) aufgestellten Hypothesen über den Einsatz eines spatio-temporalen Data Warehouses für multiskalige, heterogene ökologische Daten. Ausgangspunkt ist der aktuelle Stand der Forschung beim Datenmanagement in der ökologischen Forschung und den Problemen sowohl mit der Datenmenge als auch mit der heterogenen Skalierung der Daten. Hierfür existiert bisher keine adäquate Lösung. (Cushing u. a., 2002)(Michener und Jones, 2012)

Das durchgeführte Experiment aus Kapitel 7.2 (S. 64), welches den Anwendungsfall ARS AfricaE nutzt, dient hierbei zur Prüfung der einführend aufgestellten Hypothesen aus Abschnitt 1.2 (S. 3). Das Experiment wurde mit der in Kapitel 5 (S. 28) und 6 (S. 49) beschriebenen Testumgebung durchgeführt. Es werden verschiedene Probleme der vorliegenden Lösung erläutert und ein Ausblick auf mögliche Weiterentwicklungen des Systems gegeben.

8.1. Heterogene Daten

Das Problemfeld heterogener Daten lässt sich in seiner Gesamtheit nicht mit dem Einsatz eines DWHs alleine lösen, jedoch bieten sich hier die Konzepte des ETL-Prozesses an, welche bereits durch deren Einsatz in der BI hinreichend erforscht sind. So können Daten bereinigt und transformiert werden, um mit dem DWH Analysen durchführen zu können.

Im Bereich der heterogenen Skalierung der Daten bietet das DWH mit seinen Aggregationsoperationen einen Weg die Daten sowohl spatial als auch temporal für die Auswertung aufzubereiten. Dies zeigt sich auch in Abschnitt 7.2.3 (S. 70) des Anwendungsbeispiels und stützt Hypothese 1 (F-AN-01). Diese Fähigkeit von DWHs ist einer der zentralen Punkte, welche für den Einsatz eines spatio-temporalen Data Warehouses für multiskalige, heterogene ökologische Daten sprechen.

Der Einsatz eines DWH bietet einen Lösungsweg die Probleme, welche in den Arbeiten von Michener und Jones (2012) und Cushing u. a. (2002) beschrieben und in Kapitel 3

(S. 18) näher erläutert sind, zu lösen. Die Analyse großer Datenmengen ist eins der Kernfunktionalitäten eines DWH. Die dynamische Zusammenstellung von Faktentabellen mit dem Shared Dimension Schema, welches Resultat dieser Arbeit ist, bietet die Möglichkeit unterschiedliche Datenquellen neu zu kombinieren und auszuwerten. Die hohe Variabilität der Skalierungen können durch Aggregation eingedämmt werden und die Datenhaltung wird in einem multidimensionalen Datenmodell vereinheitlicht.

8.2. Performanz & Skalierbarkeit

Die Leistungsmessungen aus Abschnitt 7.1 (S. 62) zeigen, dass ein Datenbankcluster basierend auf MariaDB Galera Cluster eine gute Basis hinsichtlich der Skalierbarkeit für die Komponente MARS Rock bietet und somit Hypothese 4 (NF-AN-01) stützt. Gerade die Geschwindigkeit von parallelen Datenbankanfragen mehrerer Clients ist im Kontext des MARS Frameworks wichtig, da eine Vielzahl verteilter Komponenten des Frameworks parallel Anfragen an die Datenbank stellen.

8.3. Verwendung in MARS

Die Lösung integriert sich in das MARS Framework und wird den Anforderungen bezüglich der Skalierbarkeit und Aufnahme großer Datenmengen gerecht. Die spatialen und temporalen Daten lassen sich über Datenwürfel, die auf Basis des Konzepts der DFT erstellt werden passgenau mit den Dimensionen verknüpfen, welche für die Simulation eines speziellen Modells benötigt werden. Über die Drill API können Modellentwickler die Daten in ihren Simulationen verwenden, ohne das spezielle Kenntnisse über Datenbanken und die SQL-Abfragesprache vorausgesetzt werden. Als Programmiersprache für die Umsetzung der API wurde C# eingesetzt. Die Anforderungen F-AN-03, NF-AN-02 und NF-AN-03 sind somit erfüllt.

8.4. Georeferenzierung

Die Referenzierung zwischen einem GIS und einem DWH mit spatialen Daten wurde von Beginn der Forschung an als Problem identifiziert. Primär wurde mit der Referenzierung über QuadTiles ein Lösungsweg angestrebt, der sowohl das Referenzierungsproblem löst als auch die Möglichkeit bietet, Drill-Down- und Roll-Up-Operationen anhand der Hierarchie der QuadTiles durchzuführen. Jedoch wurde durch den hohen Rechenaufwand, welcher für die Approximation von QuadTiles aus einem geometrischen Objekt entsteht, ein anderer Lösungsweg für die Referenzierung der spatialen Dimension angestrebt. Durch die Verwendung spatialer Datentypen ist es nunmehr

möglich ohne Umwege, d.h. auf Basis von Punkten, Polygonen und anderen geometrischen Formen, Daten aus dem DWH abzufragen (siehe Hypothese 3). Durch die Effizienz der spatialen Operationen der Datenbank bieten sich auch bei der Auswertung neue Möglichkeiten, Daten nach komplexen spatialen Faktoren zu analysieren.

8.5. Probleme

Durch den Einsatz eines spatio-temporalen DWHs lassen sich viele der angesprochenen Probleme lösen, jedoch ergeben sich dabei auch neue Probleme, welche im folgenden Abschnitt beleuchtet werden sollen.

8.5.1. Unsicherheit bei der Speicherung und Referenzierung

Da sowohl ein GIS als auch ein DWH für die Speicherung der Daten zum Einsatz kommt, ist es nicht immer eindeutig, welche Daten in welchem System gespeichert werden sollen. Die grobe Unterscheidung GIS-Datenformate in Ground zu speichern und tabellarische Daten in Rock abzulegen, ist in den meisten Fällen ausreichend. In bestimmten Fällen ist es jedoch nötig, die Daten genauer zu analysieren, um die Vorteile des jeweiligen Systems voll ausnutzen zu können. Hypothese 2 scheint an dieser Stelle nicht ganz erfüllt und es bedarf weiterer Forschung in diesem Bereich, um die beiden Systeme in Symbiose nutzen zu können.

Auf Basis der Leistungsmessungen aus 7.1.1 (S. 62) lässt sich feststellen, dass die Referenzierung über QuadTiles nur sinnvoll einsetzbar ist, wenn beide Systeme Daten über QuadTiles referenzieren. Die Approximierung eines Polygons mit Hilfe des Algorithmus durchzuführen ist relativ rechenintensiv, sodass es sinnvoller ist, die in Abschnitt 5.3.1.3 (S. 43) beschriebenen spatialen Datentypen für die Referenzierung anzuwenden. Hiermit wird es möglich Datenbankanfragen ohne vorherige Umwandlung der Polygone zu erstellen und somit größeren Rechenaufwand für die Approximation zu vermeiden.

8.5.2. Konzept SDS

Durch das im SDS definierte Konzept der ComDims für spatiale und temporale Daten ergibt sich ein höherer Speicheraufwand. Durch das verzögerte Erzeugen der DFTs kommt es zu einer großen Anzahl von Duplikaten innerhalb der spatialen und temporalen Dimension. Hier wird es mit zunehmenden Datenmengen wohl möglich zu Problemen kommen. Um dem entgegen zu wirken, ist es möglicherweise sinnvoll, temporale und spatiale Duplikate bereits beim Einfügen der Daten zu verhindern bzw.

zyklisch eine Bereinigung der Daten innerhalb der Datenbank durchzuführen und Duplikate zu eliminieren. Diese Bereinigung ließe sich zyklisch durchführen und somit wäre es möglich, die Leistungsfähigkeit der Datenbank erhalten.

8.5.3. Verteiltes Datenbankcluster

Die Verteilung der Datenbank auf mehrere Knoten bietet den Vorteil der Skalierbarkeit, hat jedoch auch den Nachteil, dass pro Knoten eine vollständige Kopie der Daten notwendig ist. Deshalb benötigt jeder Datenbankknoten neben der nötigen Rechenleistung auch dedizierte Speicherkapazitäten ausreichender Größe.

Die aktuelle Infrastruktur des MARS Teams (siehe 6.1) ist zur Zeit nur bedingt für eine verteilte Datenbank geeignet. Für ein hochperformantes Datenbankcluster wäre es nötig, dedizierte Knoten für die Datenbank zur Verfügung zu stellen, welche über leistungsstarke Hardware und jeweils direkt angebundene Speichermedien verfügen. So könnte ein optimaler Nutzen aus der Verteilung generiert werden und das Risiko von Engpässen durch das Netzwerk o.ä. könnte verringert werden.

8.6. Ausblick

Bei der weiteren Entwicklung von MARS Rock sollten verschiedene Punkte beachtet werden. Hierzu zählt insbesondere die Erfassung von Metadaten, welche essentiell für den produktiven Einsatz von MARS ist. Es gilt zu erfassen, woher Daten stammen, wann und wo sie erfasst wurden und andere Metainformationen, welche zur Zeit nicht erfasst werden können.

Die Drill API bietet diverse Funktionen zur Abfrage der Daten, ohne das SQL-Kenntnisse benötigt werden. Hier gilt es allerdings weitere Funktionalität zu schaffen, um komplexere Datenaggregation durchführen zu können. Die Fülle an möglichen Abfragen in SQL lässt sich nur schwer in Funktionen und Methoden abbilden. Eventuell ist es nötig eine vereinfachte Metasprache, ähnlich der Syntax von SQL zu entwickeln, mit deren Hilfe Daten aus Rock über Drill abgefragt werden können. Diese Metasprache sollte genau auf die benötigten Funktionalitäten für das MARS Framework beschränkt und einfach erlernbar sein, um Vorteile gegenüber SQL zu bieten.

Der Datenimport über Drill bietet Raum für Verbesserung. Aktuell ist es möglich Daten zeilenweise einzufügen, was je nach Datenmenge zu inperformant ist. Eine geeignete Erweiterung der Schnittstelle zum Einlesen großer Datenmengen wäre hier von Vorteil. Hier wäre es möglich ein Dateiformat zu definieren, was Drill akzeptiert

und automatisiert einlesen kann.

Das Management des virtuellen Datenbankclusters wird zwar durch die Werkzeuge OpenNebula und Sunstone (siehe 5.1.3) vereinfacht, ist aber nur für das grundsätzliche Starten und Stoppen der einzelnen Knoten zu verwenden. Eine Managementoberfläche integriert in Mars Mission Control (siehe 5.2.1) zur Verwaltung von Rock wäre optimal und würde es erleichtern, die Knoten des Clusters zu verwalten. Die Erweiterung der API Drill um Funktionalitäten zur Verwaltung der Knoten würde diese Integration in MMC vereinfachen.

A. Quelltexte

```
SELECT
    d_temperature.id AS f_temperature,
    d_rainfall.id AS f_rainfall,
    d_space.id AS f_space,
    d_time.id AS f_time
FROM
    d_temperature
        LEFT OUTER JOIN
    d_rainfall ON d_temperature.d_space = d_rainfall.d_space
        AND d_temperature.d_time = d_rainfall.d_time
        LEFT JOIN
    d_space ON d_space.id = d_temperature.d_space
        LEFT JOIN
    d_time ON d_time.id = d_temperature.d_time
UNION SELECT
    d_temperature.id AS f_temperature,
    d_rainfall.id AS f_rainfall,
    d_space.id AS f_space,
    d_time.id AS f_time
FROM
    d_rainfall
        LEFT OUTER JOIN
    d_temperature ON d_temperature.d_space = d_rainfall.d_space
        AND d_temperature.d_time = d_rainfall.d_time
        LEFT JOIN
    d_space ON d_space.id = d_rainfall.d_space
        LEFT JOIN
    d_time ON d_time.id = d_rainfall.d_time
```

Quelltext A.1: Erstellung einer DFT

```
void Main()
{
    // Initialize connection to rock
    Drill.InitializeConnection("10.1.1.2", "user", "12345");

    // Define attributes for dimension temperature
    var dimensionTemperatureAttributes = new RockDbColumn[]  {
        new RockDbColumn { Name = "value", Type = DbType.Double },
        new RockDbColumn { Name = "quality", Type = DbType.Int32 },
        new RockDbColumn { Name = "interpolated", Type = DbType.
            Boolean },
```

```
11          new RockDbColumn { Name = "unit", Type = DbType.String,
                Length = 2 }
12      };
13
14      // Create dimension temperature
15      var dimensionTemperature = Drill.CreateDimension("temperature",
            dimensionTemperatureAttributes);
16
17      // Define attributes for dimension rainfall
18      var dimensionRainfallAttributes = new RockDbColum[] {
19          new RockDbColumn { Name = "mm", Type = DbType.Double },
20          new RockDbColumn { Name = "unit", Type = DbType.String,
                Length = 2 }
21      };
22
23      // Create dimension rainfall
24      var dimensionRainfall = Drill.CreateDimension("rainfall",
            dimensionRainfallAttributes);
25
26      // Load csv with temperature data
27      var csvDocumentTemperature = Csv.Load("temperaturedata-skukuza-
            kelvin.csv");
28      var tableFromCsvTemperature = csvDocumentTemperature.ToDataTable
            ("skukuza-kelvin");
29
30      for (var i = 0; i < tableFromCsvTemperature.Rows.Count; i++)
31      {
32          // create row with values from csv
33          var row = (from DataColumn col in tableFromCsvTemperature.
                Columns select col.ColumnName)
34              .Where(colName => !string.IsNullOrEmpty(colName) &&
                    dimensionTemperature.ContainsAttributeWithoutPrefix(
                    colName))
35              .Select(colName => new RockDbValue(dimensionTemperature[
                    colName].Name, dimensionTemperature[colName].
                    CleanName, dimensionTemperature[colName].Type,
                    tableFromCsvTemperature.Rows[i].ItemArray[
                    tableFromCsvTemperature.Columns.IndexOf(colName)]))
36              .ToList();
37
38          var spaceRef = new DimensionSpaceReference
39          {
40              SRID = 4326, //EPSG:4326
41              Geometry = "POLYGON((-24.9901388888889␣31.4773611111111,
                    ␣-24.9901388888889␣31.5176388888889,␣
                    -25.0270833333333␣31.5176388888889,␣-25.0270833333333
                    ␣31.4773611111111,␣-24.9901388888889␣
                    31.4773611111111))", //Well-Known-Text
42              Custom = "skukuza" // Custom tag
43          };
44
45          var timeRef = new DimensionTimeReference
```

```
46          {
47              Date = DateTime.Parse(tableFromCsv.Rows[i].ItemArray[
                    tableFromCsv.Columns.IndexOf("date")].ToString())
48          };
49
50          // Add the data to dimension
51          Drill.AddDataToDimension(dimensionTemperature, row, spaceRef
                , timeRef);
52      }
53
54      // Load csv with rainfall data
55      var csvDocumentRainfall = Csv.Load("rainfall_combined.csv");
56      var tableFromCsvRainfall = csvDocumentRainfall.ToDataTable("
            skukuza-rainfall");
57
58      for (var i = 0; i < tableFromCsvRainfall.Rows.Count; i++)
59      {
60          // create row with values from csv
61          var row = (from DataColumn col in tableFromCsvRainfall.
                Columns select col.ColumnName)
62              .Where(
63                  colName =>
64                      !string.IsNullOrEmpty(colName) &&
                            dimensionRainfall.
                            ContainsAttributeWithoutPrefix(colName))
65              .Select(
66                  colName =>
67                      new RockDbValue(dimensionRainfall[colName].Name,
                            dimensionRainfall[colName].CleanName,
68                          dimensionRainfall[colName].Type,
69                          tableFromCsvRainfall.Rows[i].ItemArray[
                                tableFromCsvRainfall.Columns.IndexOf(
                                colName)]))
70              .ToList();
71
72          var spaceRef = new DimensionSpaceReference
73          {
74              SRID = 4326, //EPSG:4326
75              Geometry = "POLYGON((-24.9901388888889␣31.4773611111111,
                    ␣-24.9901388888889␣31.5176388888889,␣
                    -25.0270833333333␣31.5176388888889,␣-25.0270833333333
                    ␣31.4773611111111,␣-24.9901388888889␣
                    31.4773611111111))", //Well-Known-Text
76              Custom = "skukuza" // Custom tag
77          };
78
79          var timeRef = new DimensionTimeReference
80          {
81              Date = DateTime.Parse(tableFromCsv.Rows[i].ItemArray[
                    tableFromCsv.Columns.IndexOf("date")].ToString())
82          };
83
```

```
84          // Add the actual data to dimension
85          Drill.AddDataToDimension(dimensionRainfall, row, spaceRef,
                timeRef);
86      }
87
88      // Create DFT for ARS AfricaE data
89      var cube = Drill.CreateCube("ars_africae", new Dimension[] {
            dimensionRainfall, dimensionTemperature });
90  }
```

Quelltext A.2: Import der Daten für Skukuza mit Drill

B. Inhalt der DVD

Dieser Arbeit liegt eine DVD mit folgender Verzeichnisstruktur bei:

/Arbeit
: Enthält diese Arbeit als PDF.

/Arbeit/Literatur
: Enthält die zitierte Literatur innerhalb dieser Arbeit als PDF.

/Messergebnisse
: Enthält Messergebnisse aus Kapitel 7.

/Quellcode
: Enthält den Quellcode des entwickelten Systems MARS Rock.

Literaturverzeichnis

[Alexey Kopytov 2008] Alexey Kopytov: *sysbench - A modular, cross-platform and multi-threaded benchmark tool.* 2008. – URL http://manpages.ubuntu.com/manpages/utopic/man1/sysbench.1.html

[Baldowski 2014] Baldowski, Mariusz: *Räumliche Analyse mit spatialen Indikatoren.* Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Dissertation, 2014

[Baldowski u. a. 2014] Baldowski, Mariusz ; Busch, Jan ; Pereki, Hodabalo ; Thiel-Clemen, Thomas: Ermittlung der Waldbiomasse mit Hilfe eines spatial gemischten Indikators für den Abdoulaye Forest, Togo. In: Wittmann, Jochen (Hrsg.) ; Maretis, Dimitris K. (Hrsg.): *Simulation in Umwelt- und Geowissenschaften* Bd. 1. Aufl, Shaker, 2014, S. 37–49. – URL http://mars-group.org/wp-content/uploads/2014/09/Ermittlung-der-Waldbiomasse-M-Baldowsk-J-Busch-H-Pereki-T-Thiel-Clemen.pdf. – ISBN 9783844030327

[Butler u. a. 2014] Butler, Howard ; Schmidt, Christopher ; Springmeyer, Dane ; Livni, Josh: *Spatial Reference.* 2014. – URL http://spatialreference.org. – Zugriffsdatum: 31.10.2014

[Chaudhuri und Dayal 1997] Chaudhuri, Surajit ; Dayal, Umeshwar: An Overview of Data Warehousing and OLAP Technology. In: *SIGMOD Rec* 26 (1997), Nr. 1, S. 65–74. – URL http://doi.acm.org/10.1145/248603.248616. – ISSN 0163-5808

[Cushing u. a. 2002] Cushing, Judith B. ; Nadkarni, Nalini ; Delcambre, Lois ; Maier, Dave: Spatial Data Infrastructure for Ecological Research. In: *Proceedings of the 2002 Annual National Conference on Digital Government Research*, Digital Government Society of North America, 2002 (dg.o '02), S. 1–4

[Fegraus u. a. 2005] Fegraus, Eric H. ; Andelman, Sandy ; Jones, Matthew B. ; Schildhauer, Mark: Maximizing the Value of Ecological Data with Structured Metadata: An Introduction to Ecological Metadata Language (EML) and Principles for Metadata Creation. In: *Bulletin of the Ecological Society of America* 86 (2005), Nr. 3, S. 158–168. – URL http://dx.doi.org/10.1890/0012-9623(2005)86[158:MTVOED]2.0.CO;2. – ISSN 0012-9623

[Finkel und Bentley 1974] Finkel, R. A. ; Bentley, J. L.: Quad trees a data structure for retrieval on composite keys. In: *Acta Informatica* 4 (1974), Nr. 1, S. 1–9. – URL http://dx.doi.org/10.1007/BF00288933. – ISSN 0001-5903

[Grant und Swannack 2008] Grant, William E. ; Swannack, Todd M.: *Ecological modeling: A common-sense approach to theory and practice*. Malden and MA and Oxford : Blackwell Pub., 2008. – ISBN 1444308874

[Hall, Charles A. S und Day 1977] Hall, Charles A. S ; Day, John W.: *Ecosystem modeling in theory and practice: An introduction with case histories*. New York : Wiley, 1977. – ISBN 9780471341659

[Hochachka u. a. 2012] Hochachka, Wesley M. ; Fink, Daniel ; Hutchinson, Rebecca A. ; Sheldon, Daniel ; Wong, Weng-Keen ; Kelling, Steve: Data-intensive science applied to broad-scale citizen science. In: *Trends in ecology & evolution (Personal edition)* 27 (2012), Nr. 2, S. 130–137. – URL http://linkinghub.elsevier.com/retrieve/pii/S0169534711003296. – ISSN 0169-5347

[Hodabalo Pereki 2013] Hodabalo Pereki: *ODD Protocol for Abdoulaye Wildlife Reserve (AWR) Agent-Based Model*. 2013

[Hüning u. a. 2014] Hüning, Christian ; Wilmans, Jason ; Feyerabend, Nils ; Thiel-Clemen, Thomas: MARS - A next-gen multi-agent simulation framework. In: Wittmann, J. (Hrsg.) ; Müller, M. (Hrsg.): *Simulation in Umwelt- und Geowissenschaften, Workshop Leipzig*, Shaker, 2014. – URL http://mars-group.org/wp-content/uploads/2014/10/MARS-A-next-gen-simulation-framework.pdf

[Jennings 2000] Jennings, Nicholas R.: On agent-based software engineering. In: *Artificial Intelligence* 117 (2000), Nr. 2, S. 277–296. – URL http://www.sciencedirect.com/science/article/pii/S0004370299001071. – ISSN 0004-3702

[Jensen u. a. 2004] Jensen, Christian S. ; Kligys, Augustas ; Pedersen, Torben B. ; Timko, Igor: Multidimensional Data Modeling for Location-based Services. In: *The VLDB Journal* 13 (2004), Nr. 1, S. 1–21. – URL http://dx.doi.org/10.1007/s00778-003-0091-3. – ISSN 1066-8888

[Joe Schwartz] Joe Schwartz ; Microsoft (Hrsg.): *Bing Maps Tile System*. – URL http://msdn.microsoft.com/en-us/library/bb259689.aspx. – Zugriffsdatum: 01.02.2014

[Leser und Naumann 2007] Leser, Ulf ; Naumann, Felix: *Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen*. 1. Aufl. Heidelberg : Dpunkt-Verl., 2007. – ISBN 3898644006

[MariaDB Corporation 2014a] MariaDB Corporation: *MariaDB Enterprise Cluster: Getting Started Guide v1.0.2*. 2014. – URL https://mariadb.com/sites/default/files/MariaDB%20Enterprise%20Cluster%20-%20Getting%20Started%20Guide%20-%20v1.0.2%20-%202014-06-04-002.pdf. – Zugriffsdatum: 08.01.2015

[MariaDB Corporation 2014b] MariaDB Corporation: *MariaDB Knowledge Base*. 2014. – URL https://mariadb.com/kb/en/. – Zugriffsdatum: 22.12.2015

[Marshall u. a. 2008] Marshall, Andrew R. ; Lovett, Jon C. ; White, Piran C L: Selection of line-transect methods for estimating the density of group-living animals: lessons from the primates. In: *American journal of primatology* 70 (2008), Nr. 5, S. 452–462. – ISSN 1098-2345

[McGuire u. a. 2008] McGuire, Michael ; Gangopadhyay, Aryya ; Komlodi, Anita ; Swan, Christopher: A user-centered design for a spatial data warehouse for data exploration in environmental research. In: *Ecological Informatics* 3 (2008), Nr. 4–5, S. 273–285. – URL http://www.sciencedirect.com/science/article/pii/S1574954108000496. – ISSN 1574-9541

[Michener und Jones 2012] Michener, William K. ; Jones, Matthew B.: Ecoinformatics: supporting ecology as a data-intensive science. In: *Trends in ecology & evolution (Personal edition)* 27 (2012), Nr. 2, S. 85–93. – URL http://linkinghub.elsevier.com/retrieve/pii/S0169534711003399. – ISSN 0169-5347

[Niazi und Hussain 2011] Niazi, Muaz ; Hussain, Amir: Agent-based computing from multi-agent systems to agent-based models: a visual survey. In: *Scientometrics* 89 (2011), Nr. 2, S. 479–499. – ISSN 0138-9130

[Pereki u. a. 2013] Pereki, Hodabalo ; Wala, Kperkouma ; Thiel-Clemen, T.: Woody species diversity and important value indices in dense dry forests in Abdoulaye Wildlife Reserve (Togo, West Africa). In: *International Journal* 5 (2013), Nr. June, S. 358–366. – URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Woody+species+diversity+and+important+value+indices+in+dense+dry+forests+in+Abdoulaye+Wildlife+Reserve+(+Togo+,+West+Africa+)#0

[Pestana u. a. 2005] Pestana, G. ; da Silva, M. M. ; Bedard, Y.: Spatial OLAP modeling: an overview base on spatial objects changing over time. In: *IEEE 3rd International Conference on Computational Cybernetics, 2005. ICCC 2005*, 2005, S. 149–154

[Red Hat 2014] Red Hat, Inc.: *Linux KVM*. 2014. – URL http://www.linux-kvm.org. – Zugriffsdatum: 22.12.2014

[Russell u. a. 2010] Russell, Stuart J. ; Norvig, Peter ; Davis, Ernest: *Artificial intelligence: A modern approach*. 3rd ed. Upper Saddle River, NJ : Prentice Hall, 2010 (Prentice Hall series in artificial intelligence). – ISBN 9780132071482

[Scholes u. a. 2001] Scholes, R. J. ; Gureja, N. ; Giannecchinni, M. ; Dovie, D. ; Wilson, B. ; Davidson, N. ; Piggott, K. ; McLoughlin, C. ; Van der Velde, K. ; Freeman, A. ; Bradley, S. ; Smart, R. ; Ndala, S.: The environment and vegetation of the flux measurement site near Skukuza, Kruger National Park. In: *Koedoe* 44 (2001), Nr. 1. – ISSN 2071-0771

[Thiel-Clemen 2013] Thiel-Clemen, Th.: Information Integration in Ecological Informatics and Modelling. In: *Workshop on Environmental Informatics, German Society of Computer Science, Leipzig, Germany, April 2013, pp.* 2013

[Vollmert u. a. 2003] Vollmert, Patrick ; Fink, Andreas H. ; Besler, Helga: Ghana Dry Zone" und "Dahomey Gap": Ursachen für eine Niederschlagsanomalie im tropischen Westafrika. In: *Erde* 134 (2003), Nr. 4, S. 375–393

Abkürzungsverzeichnis

ARS AfricaE Adaptive Resilience of Southern African Ecosystems
ABM Agentenbasierte Modelle
API Application Programming Interface
ASCII American Standard Code for Information Interchange
AWR Abdoulaye Wildlife Reserve (Togo)
BI Business Intelligence
ComDim Common Dimensions
CSV Character Seperated Value
DBMS Datenbankmanagementsystem
DFT Dynamic Fact Table
DM Data Mart
DynDim Dynamic Dimensions
DWH Data Warehouse
EM Ecological Modelling
EML Ecological Metadata Language
EPSG European Petroleum Survey Group Geodesy
ETL Extract-Transform-Load
GIS Geoinformationssystem
GLB Galera Load Balancer
GPS Global Positioning System
GPL GNU General Public Licencse
HAW Hochschule für Angewandte Wissenschaften
IBM Individuenbasierte Modelle
ID Identifikationsnummer
KNP Krüger National Park, Südafrika
KVM Kernel-based Virtual Machine
LDK Life Developement Kit

MARS Multi Agent Research and Simulation

MAS Multi-Agenten-Simulation

MMC MARS Mission Control

NDVI Normalized Difference Vegetation Index

OLAP Online Analytical Processing

ROI Return on investment

SDS Shared Dimensions Schema

SRID Spatial Reference Identifier

SQL Structured Query Language

TCP Transmission Control Protocol

WGS World Geodetic System

WKB Well-Known Binary, ein Format für die binäre Speicherung geographischer und geometrischer Daten.

WKT Well-Known Text, ein Format für die Repräsentation geographischer und geometrischer Daten in ASCII-Form.

ÖD Ökologische Daten (engl.: Ecological Data)

Danksagung

Ich möchte an dieser Stelle allen Danken, die mich während meines gesamten Studiums begleitet und unterstützt haben.

Hierbei geht mein besonderer Dank an meine Verlobte, die mich zu jedem Zeitpunkt unterstützt hat.

Mein Dank geht auch an die anderen Personen, ohne die diese Arbeit nicht möglich gewesen wäre:

- meine Familie
- meinen Betreuer Herrn Prof. Dr. Thomas Thiel-Clemen
- meinen Zweitprüfer Herrn Prof. Dr. Stefan Sarstedt
- das gesamte MARS-Team
- meine Korrekturleser
- meine Freunde

und an alle, die vergessen wurden.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 23. März 2015 Jan Busch