JOHANNES KEPLER
UNIVERSITÄT LINZ

Eingereicht von

Simon Staudinger, BSc

Angefertigt am

Institut für Wirtschaftsinformatik – Data & Knowledge Engineering

Beurteiler / Beurteilerin

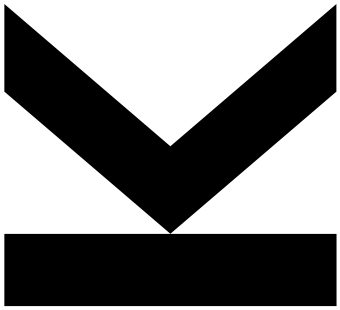
o.Univ.-Prof. DI Dr. Michael Schrefl

Mitbetreuung

Ass.-Prof. Mag. Dr. Christoph Schütz

Oktober 2019

Eine grafische Benutzungsschnittstelle für Online Analytical Processing auf Basis von Analysesituationen und inkrementellen Manipulationsoperationen



Masterarbeit

zur Erlangung des akademischen Grades

Master of Science

im Masterstudium

Wirtschaftsinformatik

JOHANNES KEPLER

UNIVERSITÄT LINZ

Altenberger Straße 69

4040 Linz, Österreich

EIDESSTATTLICHE ERKLÄRUNG

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

Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Ort, Datum

Unterschrift

Kurzfassung

Online Analytical Processing ist ein Teilbereich der Business Intelligence und ein wesentlicher Bestandteil, um Abfragen auf ein Data Warehouse durchzuführen. Diese Abfragen sollen dabei helfen die im Unternehmen vorhanden Daten bestmöglich zum Vorteil des Unternehmens einzusetzen. Entscheidungsträger können auf Basis der Ergebnisse dieser Abfragen die innovativsten und effektivsten Entscheidungen zum Vorankommen des Unternehmens treffen.

Mitarbeiter eines Unternehmens brauchen für die Erfüllung ihrer Aufgaben häufig Informationen, die sie mittels Abfragen aus dem Data Warehouse des Unternehmens erhalten. Da nicht jeder Mitarbeiter Kenntnis einer Abfragesprache besitzt, müssen Werkzeuge bereitgestellt werden, die es Mitarbeitern dennoch ermöglichen, selbstständig an die benötigten Informationen zu gelangen.

Diese Arbeit beschreibt den Prototyp eines Clients für Online Analytical Processing, mit dessen Hilfe Benutzer ohne Kenntnis einer Abfragesprache Abfragen über ein Data Warehouse mittels inkrementellen Manipulationsoperationen nach dem Analysegraph-Navigationsmodell durchführen können. Jede Änderung des Benutzers wird als Manipulationsoperation in einem Analysegraphen gespeichert. Die aktuelle Instanz des Graphen kann vom Benutzer angewendet werden, um zu beliebigen vorherigen Abfragen zu wechseln und die Datenanalyse dort fortzusetzten.

Abstract

Online analytical processing is a sub-area of business intelligence and an essential component for executing queries against a data warehouse. These queries should help to use the available company data in the best possible way for the benefit of the company. Based on the results of these queries, decision-makers are able to come to the most innovative and effective decisions about the company’s priorities.

In order to perform their tasks, the employees of a company often need information which they receive via queries from the company's data warehouse. Since not every employee has knowledge of a query language, tools must be provided that enable these employees to access the information on their own.

This work describes the prototype of a client for online analytical processing, which allows users without knowledge of a query language to query a data warehouse using incremental manipulation operations according to the analysis graph navigation model. Each modification made by the user to the query is stored as a manipulation operation in an analysis graph. The current instance of the graph can be used by the user to jump to any previous query and continue the data analysis there.

Inhaltsverzeichnis

Abbildungsverzeichnis:

Tabellenverzeichnis:

Codeverzeichnis:

1. Einleitung

Business Intelligence (BI) beschreibt die Anwendung von Werkzeugen, die es ermöglichen, Informationen aus bestehenden Daten herauszuarbeiten und diese in weiterer Folge als Grundlage für Entscheidungen aufzubereiten (Aruldoss, Lakshmi Travis, & Prasanna Venkatesan, 2014). Online Analytical Processing (OLAP) ist ein Teilbereich der BI (Golfarelli, Rizzi, & Cella, 2004). Es unterstützt Führungskräfte eines Unternehmens dabei, Entscheidungen auf Basis der im Unternehmen vorhandenen Daten zu treffen (Watson & Wixom, 2007). Eigens für OLAP entworfene Abfragesprachen wie MDX ermöglichen es dem Benutzer verschiedenste Abfragen auf eine darunterliegende Datenbasis durchzuführen (Giacometti, Marcel, & Negre, 2008).

Neuböck & Schrefl (2015) schlagen Analysegraphen für die proaktive Modellierung interessanter Analyseprozesse vor. Ein Analysegraph besteht aus Analysesituationen, die die Knoten des Analysegraphen darstellen und aus Navigationsschritten, die als gerichtete Kanten im Analysegraph abgebildet werden. Jede Analysesituation repräsentiert eine Abfrage auf eine Datenbasis. Diese wird in eine Abfragesprache übersetzt. Jeder Navigationsschritt verbindet eine Quell-Analysesituation mit einer Ziel-Analysesituation. Ein Navigationsschritt beschreibt dabei die Änderungen, die durchgeführt werden müssen, um die Quell-Analysesituation in die Ziel-Analysesituation zu überführen. Diese Änderungen werden mittels Manipulationsoperatoren z.B. Roll-Up oder Drill-Down, ausgedrückt. Durch den Einsatz von Variablen ist es möglich Analysegraph-Schemata zu definieren, die als Vorgabe für die Durchführung einer Analyse dienen. Hilal, Schuetz, & Schrefl (2017) beschreiben einen Prototyp für die praktische Anwendung von Analysegraphen im Zusammenhang mit im Web verfügbaren Daten auf Basis des Ressource Description Framework (W3C, 2014), wie sie beispielsweise von Wikidata (Vrandecic & Krötzsch, 2014) angeboten werden. Dem Benutzer wird dabei ein vordefiniertes Analysegraph-Schema zur Verfügung gestellt. Die Variablen im Schema können dann zur Laufzeit an konkrete Werte gebunden werden. Der Benutzer kann den vorher definierten Navigationsschritten folgen, um so neue Abfragen zu formulieren.

Das Analysegraph-Navigationsmodell und insbesondere die dazugehörigen Manipulationsoperatoren können auch abseits der proaktiven Modellierung von Analyseprozessen genutzt werden, um damit einen interaktiven OLAP-Client zu erstellen. In dieser Arbeit wird ein Prototyp vorgestellt, der nicht vordefinierte Analysegraph-Schemata verwendet, sondern es dem Benutzer stattdessen erlaubt, interaktive Analysesituationen zusammenzustellen. Dazu verwendet der Benutzer die Manipulationsoperatoren aus dem Analysegraph-Navigationsmodell, ohne dabei an ein vorher definiertes Schema gebunden zu sein. Im Hintergrund wird der Analyseprozess als Analysegraph-Instanz aufgezeichnet. Der Benutzer wird durch den OLAP-Client in die Lage versetzt, ohne Kenntnis einer Abfragesprache genau die Abfragen zusammenzustellen, die er benötigt, um seine Aufgaben zu erfüllen.

Das Ziel dieser Masterarbeit ist die Implementierung einer prototypischen Webanwendung, die es dem Benutzer ermöglicht, selbstständig Abfragen auf eine Datenbasis zusammenzustellen. Dabei wird das Analysegraph-Navigationsmodell verwendet und vom Benutzer, ohne Einschränkung durch vordefinierte Pfade, instanziiert. Der so erstellte Graph kann sohin vom Benutzer an beliebigen Stellen erweitert und durch eine eigene Ansicht visualisiert werden. Dieser Graph wird im Backend hinterlegt, wodurch eine weitere Verarbeitung, beispielsweise mit Methoden des Data-Mining, möglich ist. Weiters sollen Tests zu der Benutzbarkeit des Prototyps durchgeführt werden.

Der Aufbau dieser Masterarbeit gliedert sich wie folgt. Kapitel 2 beschreibt grundlegende Elemente und Konzepte, die in dieser Arbeit ihre Anwendung finden. Kapitel 3 befasst sich mit der Beschreibung und dem Aufbau des Analysegraphen und der dazugehörigen Analysesituationen sowie Manipulationsoperatoren. Kapitel 4 erläutert die Architektur des Prototyps. Der praktische Einsatz aus Sicht des Benutzers wird in Kapitel 5 genauer erläutert und bildlich dargestellt. Kapitel 6 bietet Informationen und Einblicke in die Implementierung des Prototyps. Kapitel 7 fasst alle Erkenntnisse aus der durchgeführten Evaluierung zusammen und Kapitel 8 bietet eine abschließende Zusammenfassung und einen Ausblick auf mögliche Erweiterungen.

2. State of the Art

Dieses Kapitel der Masterarbeit beschäftigt sich mit den Grundlagen, die für die Umsetzung des Prototyps notwendig waren. Dabei wird genauer auf die Bereiche Data Warehousing und Online Analytical Processing, der Wissensrepräsentation und den Grundlagen einer Webanwendung eingegangen. Diese bilden die Basis, auf der der Prototyp aufgebaut ist.

2.1. Data Warehousing und Online Analytical Processing

Unternehmen sind einer Vielzahl von Herausforderungen ausgesetzt, um am Markt bestehen zu können. Im operativen Betrieb fallen dabei täglich eine Vielzahl an Daten an, die wichtige Informationen und Indizien über das aktuelle Betriebsgeschehen liefern. Diese Informationen bieten die Grundlage, auf der Entscheidungsträger des Unternehmens ihre Entscheidungen treffen können, um so das bestmögliche Ergebnis für das Unternehmen zu erzielen. Das System, das für die Speicherung all dieser erhobenen Daten verwendet wird, wird als Data Warehouse bezeichnet. Dabei ist es entscheidend die gesammelten Informationen in einer Form zur Verfügung zu stellen, die genau auf das Informationsbedürfnis des Entscheidungsträgers abgestimmt ist. Für weitere Informationen zum Thema Data Warehousing wird auf Vaisman & Zimányi (2014) verwiesen.

Um die benötigten Informationen aus einem Data Warehouse zu extrahieren, bedarf es verschiedener Operationen, die es einem Benutzer ermöglichen, die Daten einzuschränken und auf einen gewissen Detailgrad zu bringen. Diese Operationen werden unter dem Begriff Online Analytical Processing (OLAP) zusammengefasst siehe Vaisman & Zimányi (2014).

Inmon (2005) definiert ein Data Warehouse als eine „subject-oriented, integrated, nonvolatile and time-variant collection of data in support of management's decision". Dabei bezieht sich das subjektorientiert auf die einzelnen Entitäten, die je nach Unternehmensgegenstand gebraucht werden. Im klassischen Handel sind das die Produkte oder Vertriebsniederlassungen, oder in einem Krankenhaus die Patienten bzw. verschiedene Medikamente. Diese sind die zentralen Punkte zu denen Daten abgespeichert werden sollen. Integration bezieht sich auf das Konsolidieren von verschiedenen Datenquellen. In einem Unternehmen können an verschiedenen Stellen oder in verschiedenen Systemen Daten anfallen, die danach zentral in einem Data Warehouse im gleichen Format gespeichert werden sollten. Nicht flüchtig bedeutet, dass sich die Daten in einem Data Warehouse, im Gegensatz zu den operativen Daten, grundsätzlich nicht mehr ändern. Die operativen Daten werden meist zyklisch, gebündelt in das Data Warehouse geladen und für das Berichtswesen verfügbar gemacht. Zeitvariabel bedeutet, dass alle Daten im Data Warehouse zu einem angegebenen Zeitpunkt gültig und deswegen mit einem Zeitstempel versehen sind.

Um ein Data Warehouse planen zu können, bedarf es einer konzeptuellen Beschreibungssprache. Golfarelli, Maio, & Rizzi (1998) beschreiben in ihrer Arbeit ein dimensionales Faktenmodell, das genau für den Entwurf eines solchen Data Warehouse verwendet werden kann. Dabei besteht so ein Modell aus den zentralen Elementen Fakten, Kennzahlen, Dimensionen und Hierarchien. Jedes dieser Faktenschemata bildet dabei einen Teil des Unternehmens ab, das dann in weiterer Folge abgefragt werden kann. Zu diesem Zweck kann jeder Fakt mehrere Kennzahlen beinhalten, die für das Unternehmen wichtig sind. Diese Kennzahlen können dann über die definierten Dimensionen aggregiert werden, um den Detailgrad der Kennzahlen zu ändern. Die Dimensionen beinhalten die Hierarchien der einzelnen Detailgrade die festlegen, in welcher Beziehung diese Grade zueinanderstehen und wie dann in weiterer Folge die Kennzahlen aggregiert werden sollen. Hierbei wird zwischen dimensionalen und nicht-dimensionalen Attributen (Detailgraden) unterschieden. Ein dimensionales Attribut bietet die Möglichkeit die Kennzahl über dieses Attribut zu aggregieren, während ein nicht-dimensionales Attribut lediglich zusätzliche Informationen zu einem dimensionalen Attribut bietet. In Abbildung 1 ist ein Beispiel für eine solches dimensionales Faktenmodell abgebildet.

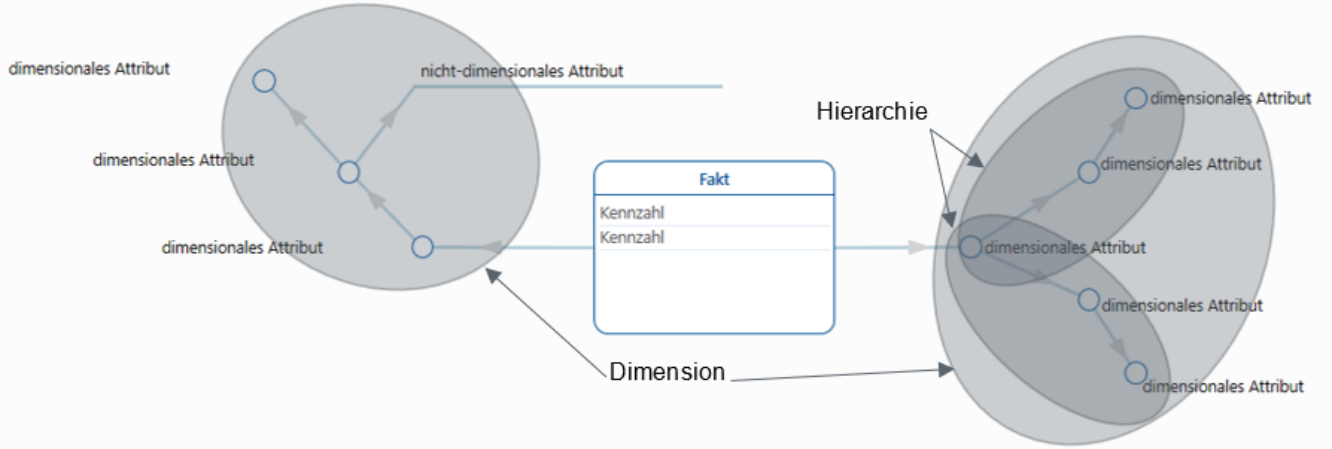


Abbildung 1: Dimensionales Faktenmodell (Golfarelli et al., 1998)

Vaisman & Zimányi (2014) beschreiben OLAP-Operationen, die die Möglichkeit bieten durch dieses dimensionale Faktenmodell zu navigieren und die gesamten detaillierten Daten auf einen Teilbereich in einem benötigten Detailgrad zu bringen. In anderen Worten, sie erlauben dem Benutzer die Daten aus verschiedenen Sichtweisen zu betrachten und so eine bessere Einsicht in diese zu bekommen. Die zentralen Operationen die OLAP bietet, sind Roll-Up, Drill-Down, Pivot, Slice und Dice.

Um die OLAP-Operationen besser darstellen zu können, wird das dimensionale Faktenmodell oft als Würfel dargestellt. Dabei enthält es 3 Dimensionen, die je auf einer Achse des Würfels dargestellt werden. Der Roll-Up ändert den Detailgrad einer Dimension auf eine allgemeinere Ebene. Der Drill-Down führt ebenso eine Änderung des Detailgrades durch, jedoch auf eine detailliertere anstatt auf eine allgemeinere Ebene. Der Roll-Up und der Drill-Down sind in Abbildung 2 zu sehen.

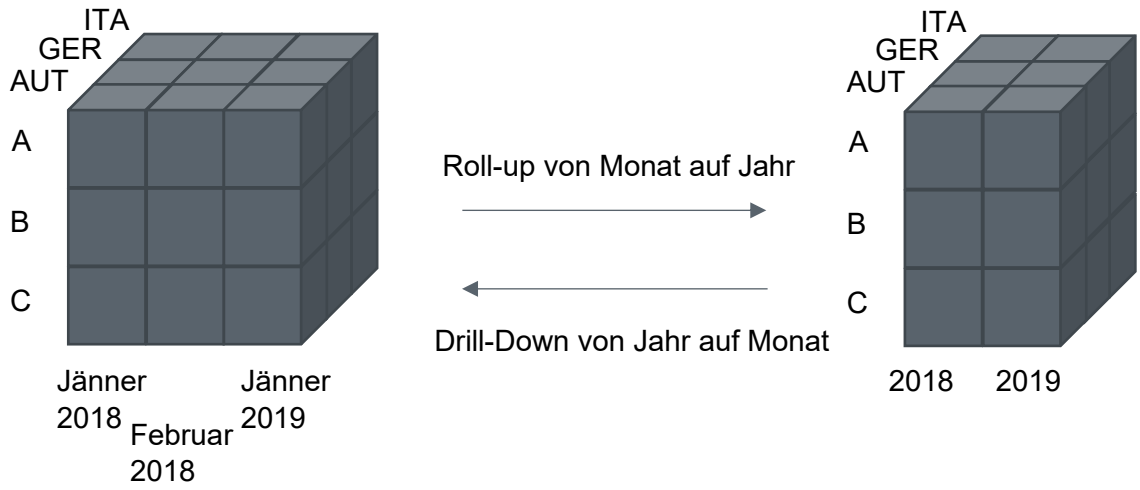


Abbildung 2: Roll-Up und Drill-Down

Mit der Pivot-Operation werden die Dimensionen auf den Achsen des Würfels vertauscht und der Anwender hat so eine alternative Sicht auf die Kennzahlen im Würfel. In Abbildung 3 wird die x-Achse mit der y-Achse vertauscht und so eine Pivot-Operation durchgeführt.

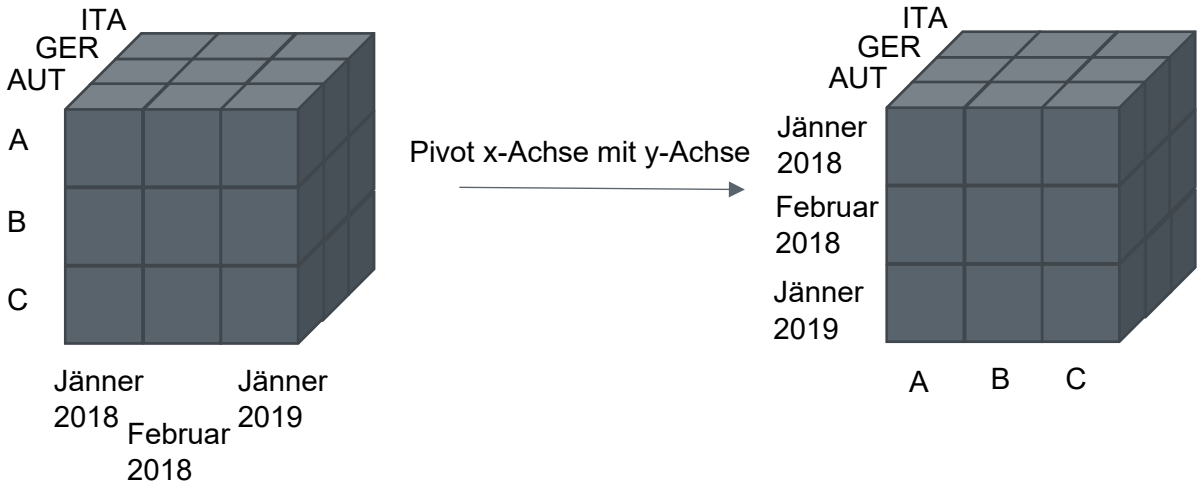


Abbildung 3: Pivot

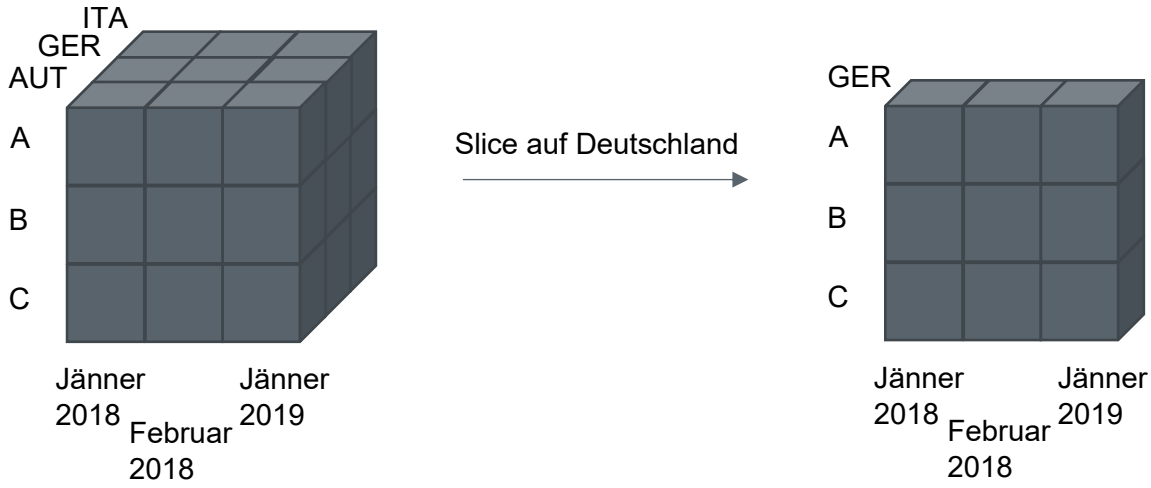


Abbildung 4: Slice

Der Slice-Operator selektiert eine definierte Anzahl von Dimensionen aus dem eigentlichen Faktenmodell. Ein beispielhafter Slice ist in Abbildung 4 zu sehen. Der Dice-Operator selektiert einen Teilwürfel aus dem gesamten Würfel, der eine vorgegebene Bedingung erfüllt.

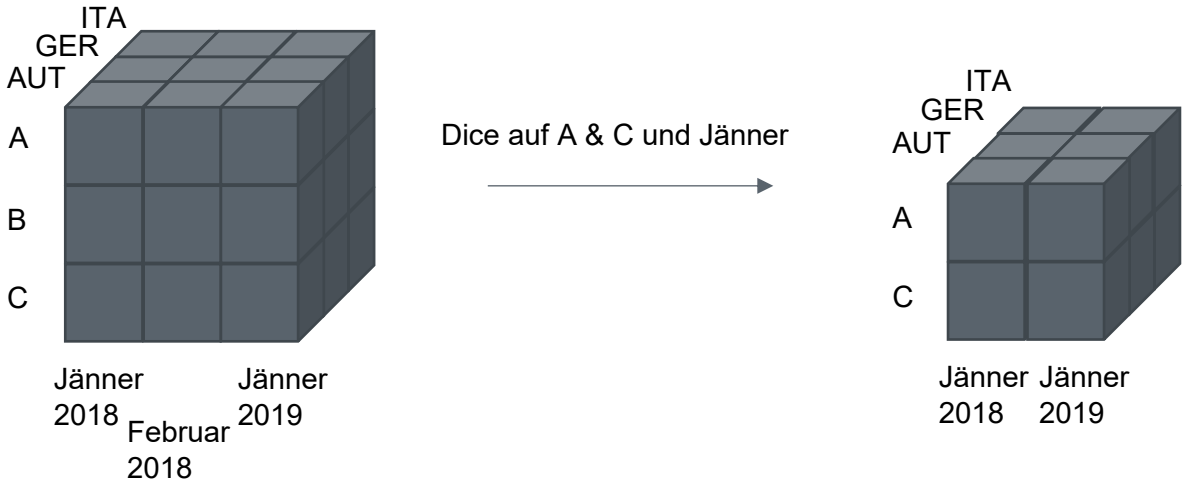


Abbildung 5: Dice

Online Transaction Processing (OLTP) befasst sich im Gegensatz zu OLAP mit den aktuellen Daten des Unternehmens (Chaudhuri & Dayal, 1997; Vaisman & Zimányi, 2014). Diese enthalten detaillierte aktuelle Informationen ohne die Aufzeichnung der bereits vergangenen Daten. Auf dieser Datenbasis ist es aufwändiger komplizierte Abfragen durchzuführen, die zu den benötigten Informationen führen sollen. Der Focus beim OLTP liegt in der bestmöglichen Unterstützung des operativen Geschäfts und der Sicherstellung der Mehrbenutzerfähigkeit und der Konsistenz. Weiters ist es auch möglich, dass die Informationen, die benötigt werden, um eine Abfrage beantworten zu können, auf verschiedenen Stellen verteilt sind und vorher erst zusammengeführt werden müssen.

2.2. Wissensrepräsentation

Davis, Shrobe, & Szolovits (1993) beschreiben Wissensrepräsentation als einen zentralen Baustein im Bereich der künstlichen Intelligenz. Dabei wird ein Teil der realen Welt in einer gewissen vordefinierten Form abgebildet. Diese Abbildung soll sowohl vom Menschen gelesen als auch von einer Maschine verarbeitet werden können. Sie ist eine Vereinfachung der realen Welt und abstrahiert den jeweiligen gewünschten Ausschnitt. Durch diese Abstrahierung gehen zwangsweise Informationen über den jeweiligen Ausschnitt verloren. Weiters soll die Wissensrepräsentation es erlauben, neue Schlussfolgerungen aus den bereits vorhandenen Informationen abzuleiten.

Eine Art der Wissensrepräsentation bietet das semantische Web. Dieses bezeichnet eine von Tim Berners-Lee ins Leben gerufene Idee das Internet flexibler zu gestalten und es mit mehr maschinenlesbaren Informationen auszustatten. Dabei soll es dem System selbstständig möglich sein, bei Informationsbedarf alle vorhandenen Informationen zu durchsuchen und so ohne Aufwand durch

den Benutzer auf ein Ergebnis zu kommen. Weiters soll es dem System möglich sein, zusammengehörende Daten selbstständig miteinander zu verknüpfen und zu kombinieren. Dazu ist es notwendig, zusätzliche Information über die einzelnen vorhandenen Elemente zu hinterlegen, die es dem System ermöglichen diese komplexen Aufgaben ohne Eingreifen des Anwenders durchzuführen. Das System soll durch diese Informationen die Daten nicht mehr nur wiedergeben, sondern auch verstehen was diese bedeuten und wie sie im Zusammenhang zueinanderstehen. Durch die Hinterlegung einer Bedeutung für die Daten im Web wird eine automatische Verarbeitung dieser möglich. Diese zusätzlichen Informationen werden auch als Metadaten bezeichnet. (Matthews, 2005)

Viele der jetzt im Internet verfügbaren Daten sind nur für die Benutzung durch Menschen gedacht. Der Mensch kann diese lesen und verstehen und so deren Bedeutung interpretieren, jedoch ist diese Aufgabe für eine Maschine nicht trivial. Das Semantische Web soll als Erweiterung des klassischen Webs fungieren und so die menschenlesbaren Daten mit maschinenlesbaren Informationen ausstatten. Es soll Regeln definieren, die es ermöglichen, die verschiedensten Wissensrepräsentationen in das klassische Web einzubetten. Dazu bieten sich die Extensible Markup Language (XML) und das Ressource Description Framework (RDF) an. Für weitere Informationen zum Semantischen Web siehe Berners-Lee, Hendler, & Lassila (2001). Das Ressource Description Framework (RDF) beschreibt eine Sammlung von Tripeln aus Subjekt, Prädikat und Objekt. Dabei verbindet jeder dieser Tripel zwei Ressourcen (Subjekt und Objekt) über eine Beziehung (Prädikat) zueinander. Jede dieser Ressourcen und Beziehungen ist eindeutig mit einem Universal Ressource Identifier (URI) identifizierbar. Diese URI macht es möglich, Ressourcen oder Beziehungen eine eindeutige Bedeutung zu geben. Diese Bedeutung sollte für alle möglichen Anwender im Web verfügbar sein.

Die aktuelle Version von RDF wurde vom W3C (2014) als Empfehlung veröffentlicht. Dort wird ein RDF-Graph als Sammlung von Knoten und deren Beziehungen definiert. Dabei kann ein Knoten ein Identifier, ein Literal oder ein Platzhalter sein. Identifier und Literale beziehen sich auf eine Abbildung aus der realen Welt. Diese bilden die Ressourcen, die mit den jeweiligen Beziehungen in Verbindung gesetzt werden. Der grobe Aufbau ist in Abbildung 6 zu sehen. Der Unterschied zwischen einem Identifier und einem Literal ist, dass ein Literal im Gegensatz zum Identifier als konstant angesehen wird.

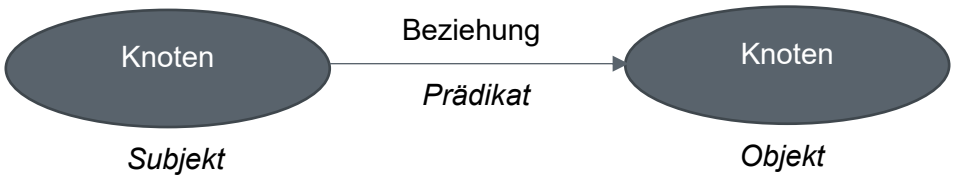


Abbildung 6: RDF-Tripel (W3C, 2014)

Jeder dieser RDF-Graphen ist eine mathematisch definierte Menge von Triple (W3C, 2014). Werden neue Triple in dieses Set hinzugefügt oder gelöscht, so entsteht ein neuer Graph. Weiters werden verschiedene RDF-Formate verwendet. Dazu zählen unter anderem, Turtle, RDFa, JSON-LD oder TriG.

Diese können jeweils in die anderen Formate umgewandelt werden und dienen als Austauschformat zwischen mehreren verschiedenen Anwendungen.

Um Daten in einem RDF-Graphen abfragen zu können, bedarf es einer eigenen Abfragesprache. Die SPARQL Protocol and RDF Query Language (SPARQL) definiert verschiedene Konstrukte, um Informationen aus RDF-Daten abzufragen oder diese Daten zu verändern (W3C, 2013). Ein Select-Konstrukt liefert als Ergebnis eine Menge von Tupel der ausgewählten Variablen. Diese Tupel können mittels einer Where-Bedingung eingeschränkt werden. Eine Construct-Abfrage liefert als Ergebnis einen RDF-Graphen, der dynamisch in der Abfrage zusammengestellt werden kann. Zum Verändern der Daten wird ein Insert-Data und ein Delete angeboten. So kann in einer Abfrage ein Tripel zum Graphen hinzugefügt oder gelöscht werden.

Um RDF-Daten zu speichern wird ein eigener Triple-Store verwendet. Dieser speichert die Daten und die dazugehörenden Schemainformationen. Diese Triple-Stores bieten gewöhnlich ein Administrationsmodul, ein Abfragemodul und eine Exportfunktion. Im Hintergrund werden die RDF-Daten selbstständig vom Triple-Store in eine relationale oder objektrelationale Datenbank gespeichert. Das Schema dieser Datenbank ist optimiert für die Speicherung und Abfrage von RDF-Daten. (Hertel, Broekstra, & Stuckenschmidt, 2009)

Ein Beispiel für einen Tripel-Store ist Fuseki (The Apache Software Foundation, 2019). Dieser bietet als Stand-Alone Anwendung eine Administrationssicht in der sämtliche Datensets verwaltet werden. Weiters bietet der Store ein Abfrageinterface, um direkt über das Userinterface des Stores SPARQL-Abfragen auf die Daten durchführen zu können. Es wird auch ein SPARQL-Endpunkt zur Verfügung gestellt, der von anderen Anwendungen genutzt werden kann, um die Daten abzufragen oder zu verändern.

2.3. Webanwendungen

Eine Webanwendung ist eine Applikation, die das Web nutzt, um ihre Aufgabe zu erfüllen. Darunter fällt Software, die extra für den Einsatz im Web entwickelt wurde, als auch Software, die nur die Infrastruktur des Webs zur Ausführung verwendet. (Gellersen & Gaedke, 1999)

Representational State Transfer (REST) ist eine Technologie, die in Webanwendungen verwendet werden kann. Sie bietet eine Alternative zum Simple Object Access Protocol (SOAP). Dabei basiert REST auf der Übertragung von Daten und bietet keinen direkten Aufruf von entfernt gespeicherter Funktionalität wie SOAP. REST verwendet sogenannte Ressourcen, die über einen Uniform Resource Locator (URL) erreicht werden. Die zentralen Operationen, die mit REST auf die jeweiligen Ressourcen durchgeführt werden, sind Erzeugen, Lesen, Updaten und Löschen (CRUD; Create, Read, Update und Delete). Diese werden mit den Hypertext Transfer Protocol (HTTP) Befehlen GET, PUT, POST und DELETE durchgeführt. Durch die Verwendung dieser Befehle kann die jeweilige gewünschte CRUD-

Operation auf eine Ressource durchgeführt werden. REST bietet somit für alle clientseitigen Operationen eine eigene Ressource, die genau die gebrauchten Daten mit der jeweiligen CRUD-Operation zur Verfügung stellt und weist jeder Ressource eine eindeutige URL zu, über die sie angesprochen werden kann. REST eignet sich unter anderem gut in der Kombination mit dem Semantischen Web. Die dort eingeführten URIs können direkt in das REST Schema übernommen und als URL für die jeweiligen Ressourcen genützt werden. (Battle & Benson, 2008)

Fielding & Taylor (2002) beschreiben REST als eine Technologie, die die Daten an den Ort ihrer Verarbeitung bringt und nicht wie in früheren Systemen die Verarbeitung zu den Daten. Jede REST Ressource liefert Daten, die semantisch an diese Ressource gebunden sind. Eine Ressource kann dabei auch eine leere Menge oder mehrere Ressourcen die gleichen Daten zurückliefern.

3. Analysegraphen

Dieses Kapitel beschreibt den Aufbau eines Business-Intelligence-Analysegraphen. Es wird erklärt wie sich dieser Graph zusammensetzt. Auf die einzelnen Bestandteile wird genauer eingegangen. Weiters wird die genaue Wissensrepräsentation des Graphen dargelegt, die in dem Prototyp verwendet wurde. Beispiele werden aus dem Use Case des Prototyps angeführt.

Ein Business-Intelligence-Analysegraph besteht aus einzelnen Analysesituationen, die über Navigationsschritte miteinander verbunden sind. Der Analysegraph ist ein gerichteter Graph, der Analysesituationen als Knoten und Navigationsschritte als Verbindungslinien dieser Knoten beinhaltet. Dabei beschreibt jede Analysesituation eine Abfrage auf eine Datenbasis. Jeder Navigationsschritt beinhaltet die Veränderung zwischen den jeweiligen verbundenen Analysesituationen. Mit diesem Graphen ist es möglich bereits durchgeführte Analyseprozesse nachzuvollziehen und wiederverwendbar bzw. erweiterbar zu machen. Durch die Verwendung von Variablen können Vorlagen von Abfragen erstellt werden, die zur Laufzeit mit den konkreten gewünschten Werten befüllt werden. (Neuböck, Neumayr, Rossgatterer, Anderlik, & Schrefl, 2012)

Der Analysegraph teilt sich in die Schemaebene und die Instanzebene auf. Die Schemaebene beschreibt einen möglichen Analyseprozess, der verschiedene Variablen beinhaltet, die dann zur Ausführungszeit vom Benutzer an konkrete Werte gebunden werden. Des Weiteren kann anhand eines Analysegraphen auf Schemaebene der mögliche Analysefluss gesteuert werden. Der Benutzer kann den vordefinierten Pfaden des Graphen folgen und die jeweiligen geforderten Variablen an konkrete Werte binden. Dadurch, dass die Variablen erst zur Laufzeit an konkrete Werte gebunden werden, ist es möglich den Analysegraphen bzw. Teile daraus wiederzuverwenden. Ein Analysegraph auf Instanzebene bezeichnet einen Graphen, der keine Variablen mehr beinhaltet, sondern nur mehr konkrete Werte enthält. (Neuböck et al., 2012)

Das Repräsentationsmodell für Analysegraphen, das im Prototyp verwendet wird, ist in Abbildung 7 zu sehen. Dabei zeigt sich, dass sich dieser Graph aus einer oder mehreren Analysesituationen und keinem oder mehreren Navigationsschritten zusammensetzt. Der Analysegraph, der im Prototyp verwendet wird, befindet sich nur auf Instanzebene. Der Benutzer kann anhand dieses Graphen die bisher durchgeführten Schritte nachvollziehen. Der Graph zeigt den Prozess der Erstellung der aktuell im Prototyp gezeigten Abfrage und ist an kein vordefiniertes Schema gebunden. Wird im Prototyp auf einen Analysegraphen verwiesen, ist immer ein Graph auf Instanzebene ohne dazugehörige Schemadefinition gemeint. Als Ausgangspunkt jeder Analyse dient ein Analysegraph, der genau eine Analysesituation beinhaltet, die alle Daten ohne Einschränkungen abfragt. Der Aufbau der Analysesituationen und der Navigationsschritte wird in den folgenden Kapiteln noch genauer beschrieben.

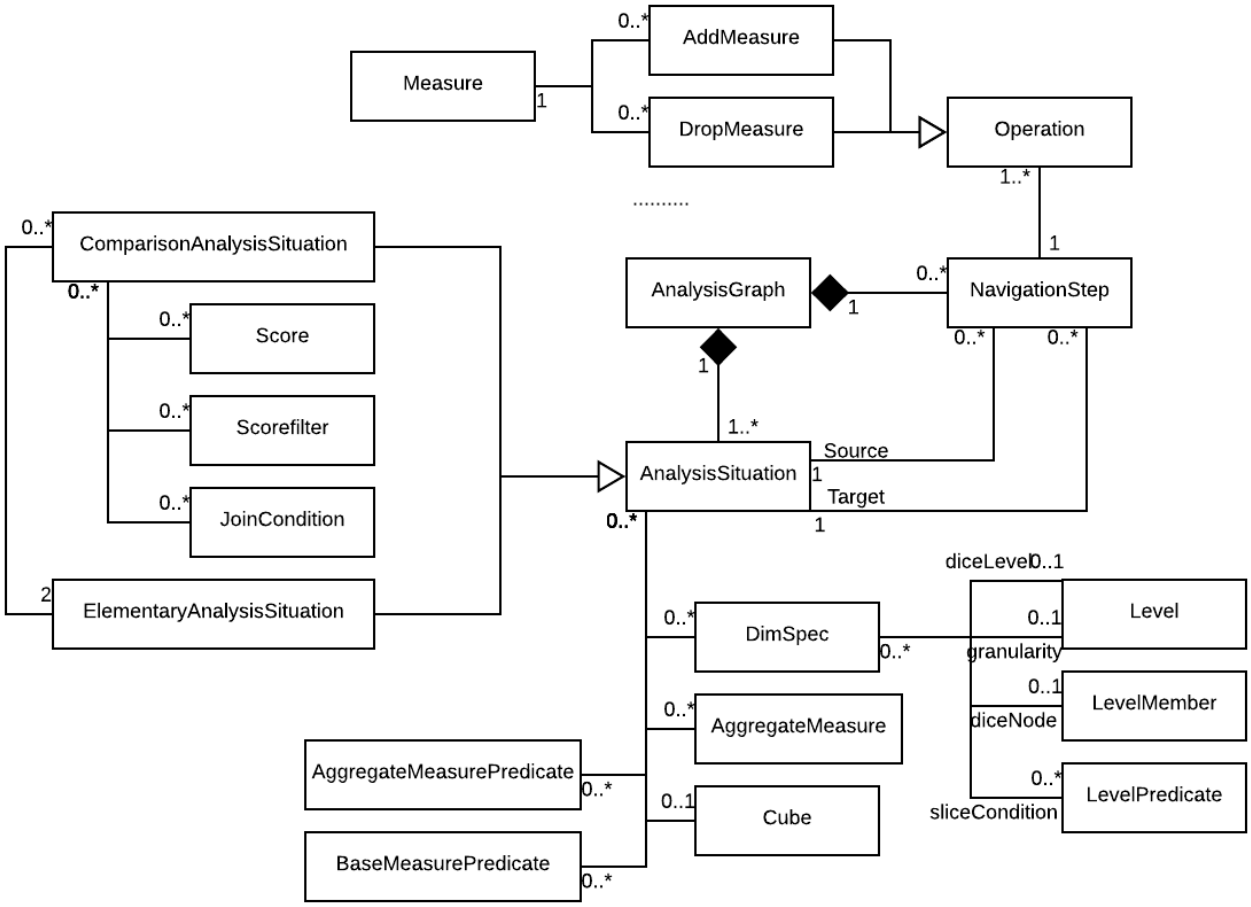


Abbildung 7: Repräsentationsmodell für Analysegraphen

3.1. Analysesituationen

Eine Analysesituation ist ein Knoten in einem Analysegraph. Dabei beschreibt jede Analysesituation eine multidimensionale Abfrage auf eine Datenbasis. Jede Analysesituation bezieht sich auf einen gewissen vordefinierten Datencube. Dieser Datencube definiert den Aufbau der abzufragenden Datenbasis. Eine klassische Analysesituation, wie von Neuböck & Schrefl (2015) beschrieben, kann Kennzahlen und Einschränkungen auf Dimensionsebene beinhalten. Die hinzugefügten Kennzahlen werden beim Ausführen der Abfrage über die vorhandenen Dimensionen mithilfe einer Aggregatsfunktion aggregiert. Jede vorhandene Dimension kann in weiterer Folge auf die benötigten Daten eingeschränkt werden. Dazu können ein Dice-Level, ein Dice-Node, verschiedene Slice-Conditions und ein Granularity Level vergeben werden. Als Dice und Granularity-Level einer Dimension kann jeweils einer der vorhandenen Levels aus dem dimensionalen Faktenmodell des Cubes verwendet werden. Die Dice-Node ist eine Instanz des jeweiligen ausgewählten Dice-Levels. Dabei können die Werte dieser Instanz noch weiter mit vordefinierten Slice-Conditions eingeschränkt werden.

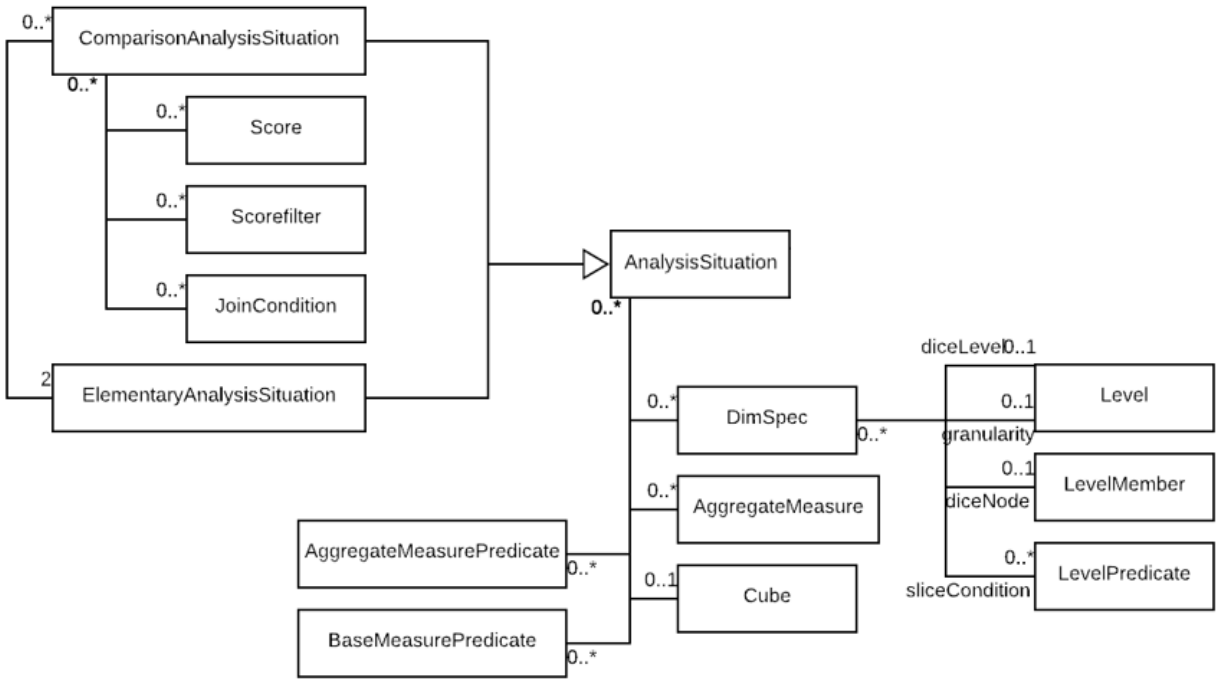


Abbildung 8: Repräsentationsmodell einer Analysesituation

Abbildung 8 zeigt den Aufbau der Analysesituation im Prototyp. Es wird in zwei Arten von Analysesituationen unterschieden. Die einfachen Analysesituationen beschreiben eine Abfrage auf eine Datenbasis. Im Gegensatz dazu beschreibt eine vergleichende Analysesituation den Zusammenhang bzw. Unterschied zwischen zwei einfachen Analysesituationen. (Neuböck, 2018)

Eine einfache Analysesituation kann aus folgenden Bestandteilen bestehen:

- Einem Verweis auf einen Cube
- Aggregierten Kennzahlen
- Prädikate auf die aggregierten Kennzahlen
- Prädikate auf die Basiskennzahlen
- Spezifikation der einzelnen Dimensionen eines Cubes

Der Cube ist eine vorher definierte multidimensionale Datenstruktur. Diese beinhaltet verschiedenste Elemente, die einer Analysesituation hinzugefügt werden können. Die aggregierten Kennzahlen sind entweder Kennzahlen die mithilfe einer Aggregatsfunktion (Sum, Avg, usw.) über die jeweiligen Dimensionen aggregiert wurden, oder eine Kombination von bereits definierten aggregierten Kennzahlen. Prädikate sind vordefinierte Filterbedingungen. Diese werden im erweiterten Faktenschema definiert und schränken die Ergebnismenge der Analysesituation ein. Einschränkungen können sowohl für die Basiskennzahlen eines Cubes als auch für die aggregierten Kennzahlen hinzugefügt werden. Der letzte Bestandteil sind die Dimensionsspezifikationen. Jede vorhandene Dimension setzt sich, wie oben beschrieben, aus vier Bestandteilen zusammen. Dazu gehören einerseits die Angabe der Level und eine Instanz des Levels, auf den die Kennzahlen aggregiert werden

sollen. Weiters einen Level der Dimension der die Granularität der Daten festlegt und eine Liste von Slice-Conditions. Slice-Conditions sind Prädikate (Einschränkungen) auf der Levelebene. Werden für Dice-Level, Dice-Node und Granularität keine Werte angegeben, so wird automatisch die allgemeinste Option ausgewählt.

Die vergleichende Analysesituation besteht aus folgenden Bestandteilen:

- Zwei einfachen Analysesituationen
- Vergleichenden Kennzahlen
- Prädikate für die vergleichenden Kennzahlen
- Join-Conditions

Das Ziel der vergleichenden Analysesituation ist es, den Unterschied zwischen zwei einfachen Analysesituationen aufzuzeigen. Dazu können vergleichende Kennzahlen in die Analysesituation aufgenommen werden, die dann mit Prädikaten weiter eingeschränkt werden. Weiters können auch Join-Bedingungen hinzugefügt werden, die festlegen, wie die beiden Analysesituationen miteinander verglichen werden sollen.

Als Grundlage für die Befüllung von Analysesituationen mit Werten dient ein speziell dafür entwickeltes multidimensionales Datenschema. Dieses Schema beinhaltet verschiedene vordefinierte Elemente, die in den Analysesituationen benutzt werden können. Neuböck (2018) beschreibt in seiner Arbeit eine erweiterte Form eines dimensionalen Faktenmodells (originales Faktenmodell siehe Abschnitt 2.1.), die die Grundlage des Datenschemas ist. Dabei wird das ursprüngliche Modell aus Fakten, Dimensionen, Levels und Hierarchien um weitere Elemente ergänzt, die dann dem Benutzer beim Zusammenstellen der Analysesituationen zur Verfügung stehen. Der Prototyp verwendet eine Form dieses erweiterten Faktenschemas als Definition der einzelnen Cubes, die für eine Analyse verfügbar sind. Hinzugefügt werden beispielsweise aggregierte Kennzahlen und verschiedene Prädikate, die Einschränkungen der Daten enthalten können. Eine genaue Definition mit Beispielen dieses Datenschemas ist in Abschnitt 6.2. zu finden.

3.2. Navigationsschritte

Ein Manipulationsoperator beschreibt den Unterschied zwischen einer Startanalysesituation und einer Zielanalysesituation. Durch die Angabe einer Startanalysesituation und dem Navigationsschritt kann die Zielanalysesituation ermittelt werden. Der Navigationsschritt beinhaltet verschiedene Veränderungsoperatoren und deren Parameter. Werden mehr als ein Operator in einem Navigationsschritt angewendet, so muss die Reihenfolge der Operatoren angegeben werden, um mögliche durch Umsortierung entstandene Fehler zu verhindern. Abbildung 9 beschreibt den Aufbau eines Navigationsschrittes im Prototyp. Als Operatoren können verschiedene weiter unten genauer beschriebenen Aktionen verwendet werden. (Neuböck, 2018)

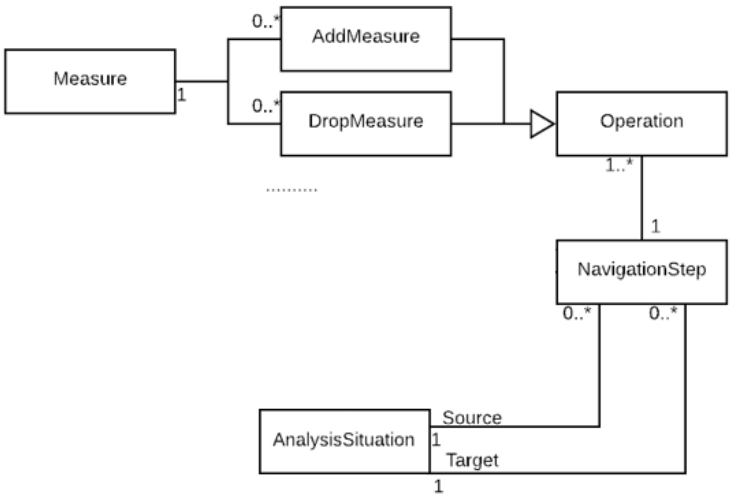


Abbildung 9: Wissensrepräsentation eines Navigationsschritts im Prototyp

3.3. Use Case

Der Prototyp verwendet neben einem Beispiel für Medikamente derzeit hauptsächlich ein von Julien Hyde (Julian Hyde, 2016) publiziertes und für die Anwendung im Prototyp verändertes und angepasstes Beispiel. Das Beispiel wird unter dem Namen Foodmart geführt und ist Teil der Penthaho Mondrian OLAP Engine (Hitachi Vantara, 2017). Dieses Beispiel wurde aufgrund seiner Bekanntheit gewählt und dementsprechend als Datenstruktur für den Prototyp umgebaut und angepasst, jedoch bleiben die grundlegende Idee und die Daten gleich.

Das Foodmart-Beispiel wurde ins Backend eingebunden und dem Prototyp als Beispiel im erweiterten dimensionalen Faktenmodell zur Verfügung gestellt. Dabei wurde die Sales-Datenstruktur als zentrale Datenquelle in das neue Datenformat adaptiert und Beispiele für alle relevanten Testfälle zum Beispiel hinzugefügt. Für jegliche Testzwecke wurde nur der Sales Cube aus dem Foodmart Beispiel entnommen und angepasst. Dieser enthält eine Produktdimension, eine Geschäftsstellendimension und eine Zeitdimension mit zwei verschiedenen Hierarchien. Als Kennzahlen werden Beispiele wie Anzahl der Kunden, Verkaufserlöse und Verkaufszahlen verwendet.

4. Architektur

Der Prototyp ist als Webanwendung aufgebaut. Als Vorbild und Vergleichsobjekt dient dabei der OLAP-Client Saiku, der als Webanwendung realisiert ist. Saiku bietet eine grafische Oberfläche, die es ermöglicht OLAP-Abfragen auf ein Data Warehouse zusammenzustellen und durchzuführen. Der hier erstellte Prototyp arbeitet im Gegensatz zu Saiku mit vordefinierten Manipulationsoperatoren, die der Anwender benützen kann, um seine Abfrage zu erstellen.

Die Architektur gliedert sich in zwei Teilbereiche. Der erste Bereich ist ein Backend, das nicht Teil dieser Masterarbeit ist. Der zweite Bereich ist ein grafisches Frontend das durch Zugriff auf die vom Backend zur Verfügung gestellten Schnittstellen verschiedene Aufgaben für den Benutzer erledigt.

Den zentralen Bestandteil des Backends bildet ein Jersey-Webserver. Dieser stellt verschiedene spezifizierte REST-Schnittstellen zur Verfügung, die es einem Client ermöglichen die damit verbundenen Funktionen zu nutzen. Weiters beinhaltet das Backend einen RDF-Triple-Store. Dieser speichert einerseits die Informationen über den Aufbau der im Data Warehouse gespeicherten Cubes und andererseits die Analysegraph-Instanzen aller über die Schnittstelle durchgeführten Analysen. Zusätzlich beinhaltet das Backend ein Data Warehouse, das die eigentlichen Cubes mit den entsprechenden Daten enthält.

Das Frontend ist eine Weboberfläche, die mithilfe eines Browsers aufgerufen wird. Dabei unterteilt sich das Frontend in zwei Bereiche. Eine Oberfläche zum Zusammenstellen der Abfrage und eine Oberfläche, die die aktuelle Analysegraph-Instanz grafisch darstellt. Der Benutzer hat die Möglichkeit, mit den beiden Oberflächen zu interagieren, um so zu seiner gewünschten Abfrage zu gelangen.

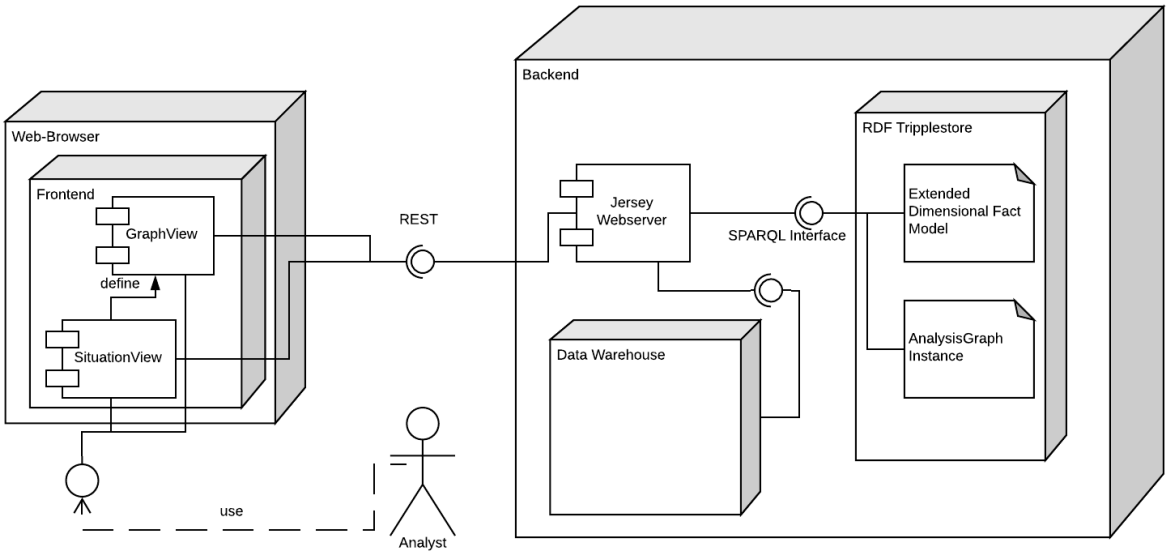


Abbildung 10: Architektur des Prototyps

Abbildung 10 zeigt eine grafische Darstellung der Architektur. Auf der rechten Seite befindet sich Backend mit dem Data Warehouse und den Cube bzw. Instanzinformationen. Die linke Seite zeigt das Frontend mit den beiden zentralen Oberflächen für den Benutzer. Die REST-Schnittstellen des Backends verwenden JSON-Dateien für die Kommunikation zwischen Front- und Backend. Genauer gesagt handelt es sich um JSON-LD-Dokumente. JSON-LD ist eine Form für die Darstellung von RDF-Daten und zugleich eine gültige Form einer normalen JSON-Datei. Somit kann das gängige JSON-Format für den Austausch von RDF-Daten verwendet werden. Die Entscheidung der Verwendung von RDF-basierten Daten wurde aufgrund von Interoperabilitätsaspekten mit anderen Arbeiten (Hilal et al., 2017; Hilal, Schuetz, & Schrefl, 2018; Straßer, 2019) getroffen und als Anforderung an den Prototyp definiert. Die Implementierung des Backend ist nicht Teil dieser Arbeit. Im Rahmen dieser Arbeit wird jedoch ein Backend für Demonstrationszwecke bereitgestellt, das Beispieldaten zurückliefert, die es ermöglichen, die Benutzbarkeit des Prototyps zu testen.

Das Frontend selber besteht aus mehreren zusammengehörenden HTML-Seiten, die mit verschiedenen JavaScript-Funktionen ergänzt wurden, um so die benötigte Funktionalität in die ansonsten statischen Webseiten zu bringen. Für die Erstellung einzelner User-Interface-Elemente wurde auf das Bootstrap-Framework zurückgegriffen. Dieses erlaubt es bereits vordefinierte grafische Elemente zu nutzen und diese mit entsprechenden Funktionen zu hinterlegen.

5. Benutzersicht

Dieses Kapitel beschreibt die Benutzung des Prototyps aus der Sicht eines Anwenders. Dieser möchte mithilfe diese Prototyps Abfragen auf vordefinierte Datenstrukturen erstellen. Das Ziel ist es, mit der Anwendung von Operatoren einen gewissen gegebenen Informationsbedarf zu decken oder eine Exploration der Daten zu ermöglichen. Die Symbole, die hier für die Operatoren verwendet werden, sind von Neuböck (2018) übernommen und angepasst.

5.1. Allgemeiner Aufbau

Der Prototyp ist eine Webanwendung, daher kann er über eine URL im Browser aufgerufen werden. Diese kann beim Veröffentlichen im jeweiligen Webserver definiert werden. Für Testzwecke kann die Webseite auch lokal abgelegt und dann aus dem lokalen Verzeichnis geöffnet werden.

Nach dem Aufrufen der URL im Browser wird dem Benutzer der in Abbildung 11 angeführte Bildschirm angezeigt.

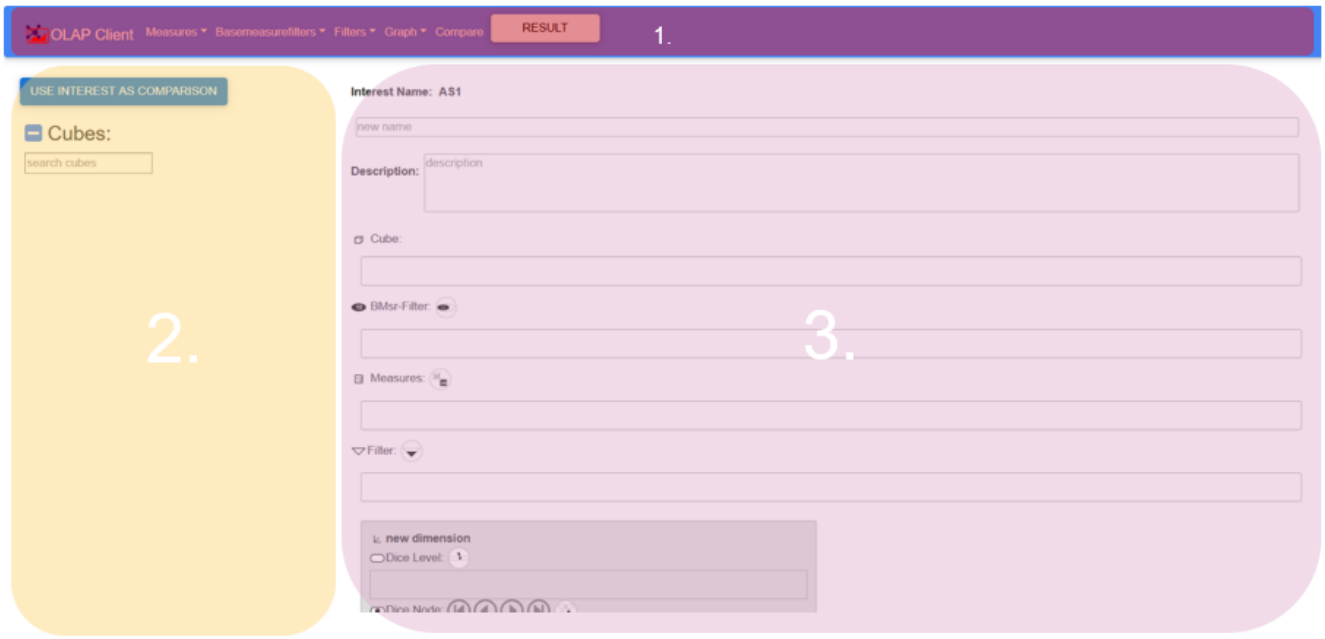


Abbildung 11: Aufbau einfache Analysesituation

Der Startbildschirm teilt sich grob in drei Bereiche auf. Im Bereich 1 befindet sich eine Navigationsleiste. Diese wird verwendet, um verschiedene Operatoren auf die aktuelle Analysesituation anzuwenden. Weiters befindet sich hier jeweils ein Button, um von einer einfachen Analysesituation auf eine vergleichende zu wechseln, oder die Auswahl der beiden Analysesituationen einer vergleichen Analysesituation zu wechseln. Zusätzlich existieren Buttons, um auf die Graph-Ansicht umzuschalten oder das Ergebnis der aktuellen Analysesituation zu erhalten. Der 2. Bereich beinhaltet die Auswahl des aktuellen Cubes für die Analysesituation und zeigt nach dessen Auswahl alle möglichen Elemente, die dieser Cube beinhaltet. Beispiele dafür sind die Kennzahlen, Dimensionslevels oder Prädikate zum

Einschränken der Analysesituationen. Im 3. Bereich befindet sich die aktuelle Analysesituation mit allen bisher ausgewählten Elementen. Zusätzlich zu den durch Klicken, Hinzuziehen oder durch die Anwendung verschiedener Operatoren hinzugefügten Elemente kann hier ein Name und eine Beschreibung der Analysesituation hinterlegt werden. Wie in der Navigationsleiste befinden sich verschiedene Buttons neben den einzelnen Feldern, die wiederum an definierte Operatoren gebunden sind und so das Hinzufügen oder Löschen von Elementen der Analysesituation ermöglichen.

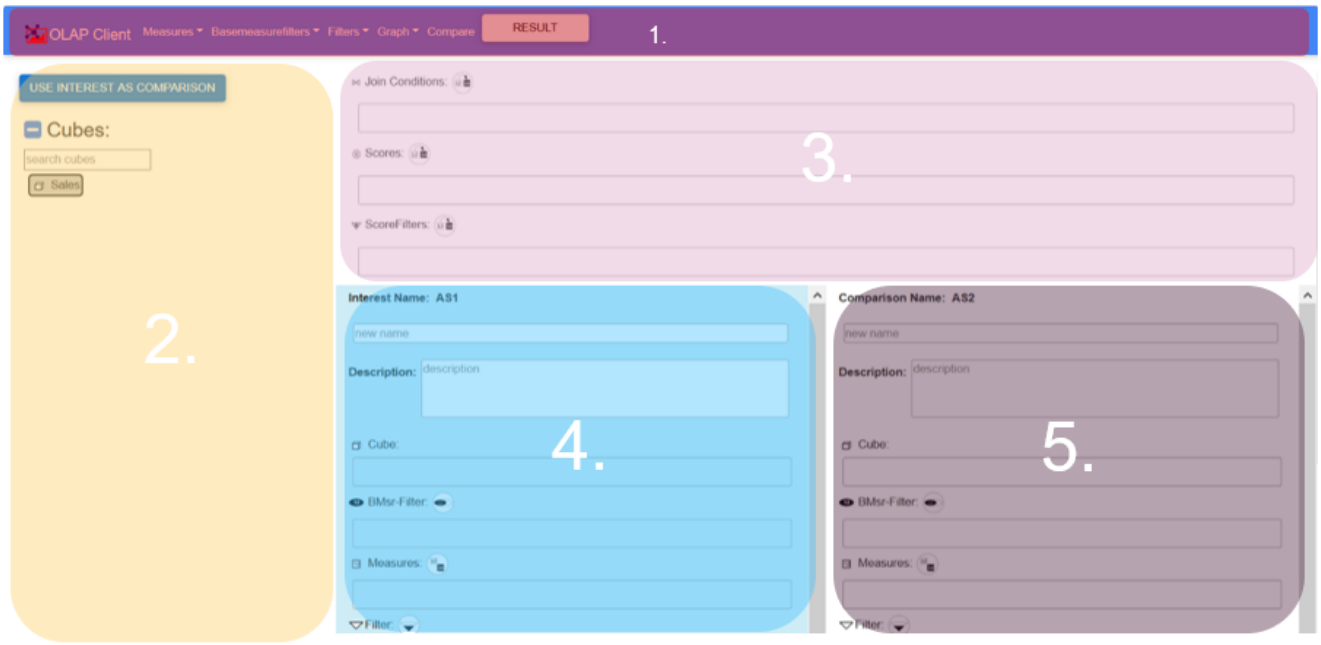


Abbildung 12: Aufbau einer vergleichenden Analysesituation

Durch das Klicken auf „Compare“ in der Navigationsleiste wird der in Abbildung 12 gezeigte Aufbau geöffnet. Die Bereiche der Navigationsleiste und der Cube-Informationen bleiben gleich. Der Bereich der aktuellen Analysesituation wird aufgeteilt. Dabei befinden sich im Bereich 3, Elemente, die sich auf die gesamte vergleichende Analysesituation beziehen mit entsprechenden Buttons zum Aufrufen von dazugehörigen Operatoren. Der Bereich 4 zeigt alle Elemente, die sich in der ersten der zu vergleichenden Analysesituationen befinden. Diese Analysesituation wird von der einfachen Analysesituation übernommen. Der Bereich 5 zeigt eine zweite Analysesituation, die als Vergleichsanalysesituation verwendet wird.

Die Graph-Ansicht kann durch das Klicken auf den Button „Graph“ in der Navigationsleiste aufgerufen werden. Abbildung 13 zeigt eine beispielhafte Graph-Ansicht. Diese ist aufgeteilt in eine Navigationsleiste (Bereich 1) und einen Bereich für den Graphen (Bereich 2). Jede Änderung, die ein Benutzer an einer Analysesituation durchgeführt hat, wird im Graphen mitgespeichert und kann so im Nachhinein nachvollzogen werden. Eine einfache Analysesituation wird mit einem blauen Kreis dargestellt, eine vergleichende Analysesituation mit einem roten Kreis. Zwei gestrichelte Linien verbinden eine vergleichende Analysesituation mit den beiden einfachen Analysesituationen, die darin enthalten sind. Die Analysesituationen können mit einem Operator miteinander verbunden sein. Das

Symbol des jeweiligen Operators ist auf der Verbindungslinie zwischen den Analysesituationen zu sehen. Durch Klicken auf eine Analysesituation öffnet sich ein Menü, in dem der Benutzer zu einer vergangenen Analysesituation zurückwechseln kann. Die aktuelle Analysesituation, die momentan ausgewählt ist, ist im Graphen zusätzlich durch eine grüne Umrandung gekennzeichnet, um so eine bessere Übersicht für den Benutzer zu bieten.

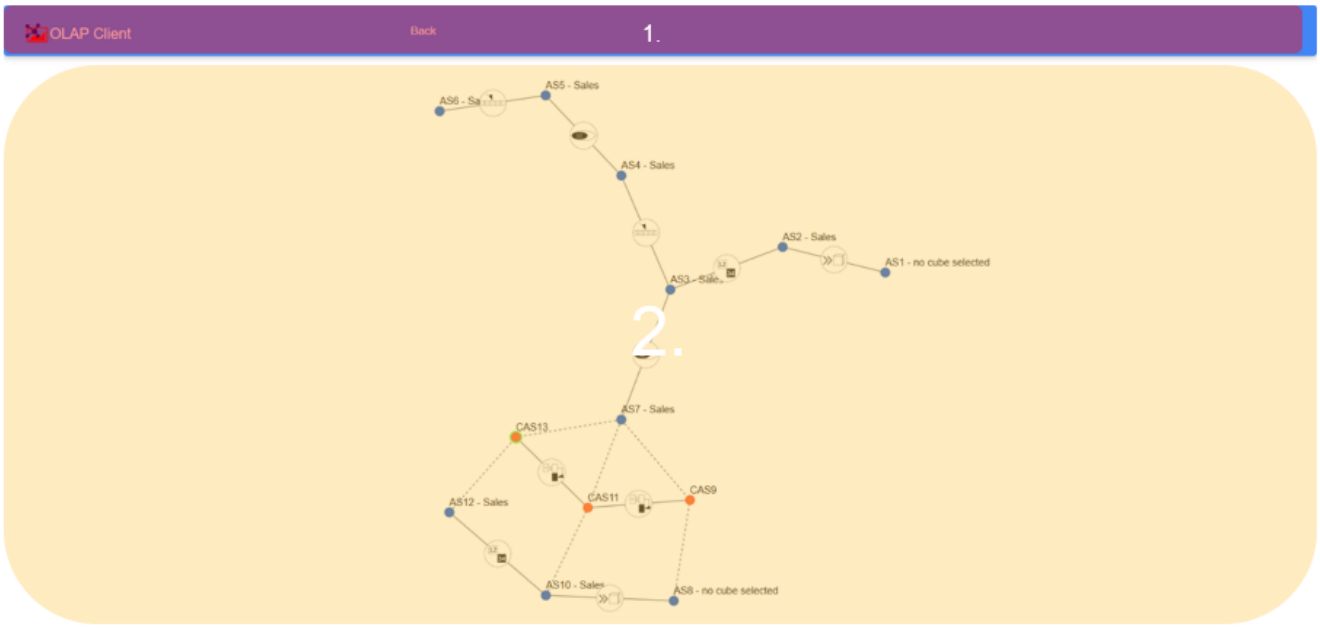


Abbildung 13: Aufbau der Graph-Ansicht

Durch das Klicken auf den Button „Result“ in der Navigationsleiste wird die aktuelle Analysesituation im Backend ausgewertet und das Ergebnis an den Benutzer gesendet. Es öffnet sich ein neuer Tab, der das Ergebnis tabellarisch anzeigt. Diese Tabelle kann dann anschließend durch Drag & Drop genau an das Informationsbedürfnis des Benutzers anpasst werden. Ein Beispiel für eine Ergebnistabelle ist in Abbildung 14 zu sehen.

Table | Sum | Unit Sales | Month | Unit Sales | Product Category

Month \ Product Category	Baking Goods	Beer and Wine	Bread	Carbonated Beverages	Dairy	Drinks	Hot Beverages	Pure Juice Beverages	Totals
1	197.00	149.00	177.00	107.00	16.00	144.00	109.00	184.00	**1,083.00**
2	64.00	174.00	258.00	228.00	169.00	157.00	15.00	72.00	**1,137.00**
3	235.00	299.00	222.00	247.00	186.00	198.00	38.00	280.00	**1,705.00**
4	76.00	46.00	34.00	214.00	29.00	201.00	12.00	108.00	**720.00**
5	259.00	249.00	105.00	237.00	259.00	228.00	191.00	163.00	**1,691.00**
6	154.00	262.00	43.00	236.00	152.00	152.00	58.00	128.00	**1,185.00**
7	279.00	237.00	138.00	267.00	275.00	88.00	63.00	54.00	**1,401.00**
8	232.00	8.00	183.00	145.00	143.00	153.00	138.00	244.00	**1,246.00**
9	8.00	284.00	269.00	68.00	128.00	0.00	239.00	261.00	**1,257.00**
10	73.00	97.00	112.00	146.00	297.00	211.00	159.00	240.00	**1,335.00**
11	85.00	6.00	215.00	296.00	279.00	33.00	167.00	261.00	**1,342.00**
12	189.00	148.00	288.00	283.00	184.00	182.00	92.00	235.00	**1,601.00**
Totals	**1,851.00**	**1,959.00**	**2,044.00**	**2,474.00**	**2,117.00**	**1,747.00**	**1,281.00**	**2,230.00**	**15,703.00**

Abbildung 14: Ergebnistabelle

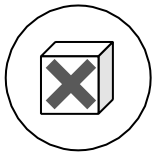
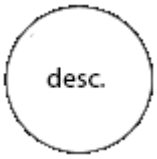
5.2. Einfache Analysesituationen

Dieser Abschnitt der Arbeit befasst sich mit den Manipulationsoperatoren, die auf eine einfache Analysesituation angewendet werden können. Die einzelnen Operatoren werden genau erläutert und mit grafischen Beispielen hinterlegt.

5.2.1. Grundlegende-Operatoren

Dieser Abschnitt befasst sich mit der Benützung der Oberfläche für die einfachen Analysesituationen. Es werden alle Operatoren, die auf diese Analysesituationen angewendet werden können, vorgestellt und beschrieben. Weiters sollen Symbole dabei helfen, die Anwendung und Veränderung der jeweiligen Operatoren darzustellen. Diese Symbole werden sowohl im UI der Webanwendung, sowie in der Graphen-Ansicht verwendet. Die Tabelle 1 beschreibt die grundlegenden Operatoren, die verwendet werden, um eine Analysesituation mit einem Cube und Metadaten zu befüllen.

Tabelle 1: Grundlegende-Operatoren (Neuböck, 2018)

Operator	Symbol	Beschreibung
SelectCube		Dieser Operator wählt für die jeweilige Analysesituation einen Cube aus, für den eine Abfrage erstellt werden soll. Parameter: ID des Cubes
DropCube		Dieser Operator entfernt einen Cube mitsamt allen zugehörigen Elementen aus der Analysesituation. Parameter: ID des Cubes
ChangeName		Dieser Operator benennt eine Analysesituation mit einem übergebenen Namen. Parameter: Situationsname
ChangeDescription		Dieser Operator ändert die Beschreibung einer Analysesituation auf eine neue übergebene Beschreibung. Parameter: Beschreibung
DropDimension		Dieser Operator entfernt sämtliche ausgewählte Elemente einer Dimension und setzt die Granularität zurück auf den Ausgangslevel. Parameter: ID der Dimension

Operator SelectCube

Der erste Operator, der hier beschrieben wird, ist der SelectCube-Operator. Dieser fügt einer Analysesituation einen Cube hinzu, der dann in weiterer Folge die Grundlange für die zusammengebaute Abfrage bildet. Ein Benutzer hat zwei Möglichkeiten einen Cube einer Analysesituation hinzuzufügen. Durch einen Klick auf einen in den Cube-Informationen angezeigten Cube oder mittels Drag & Drop in das dafür vorgesehene Feld in der Analysesituation. Befinden wir uns in einer vergleichenden Analysesituation wird der Cube beim Klicken in die aktuell ausgewählte Analysesituation hinzugefügt. Abbildung 15 zeigt das Hinzufügen eines Cube zur Analysesituation.

Operator DropCube

Um einen Cube aus einer Analysesituation zu entfernen und somit den Operator DeleteCube anzuwenden, kann der Benutzer den Cube in der Analysesituation anklicken, oder per Drag and Drop aus der Analysesituation in den freien Bereich um die Felder der Analysesituation ziehen. Wird versucht einen Cube zu entfernen obwohl noch Elemente in der Analysesituation sind, wird ein Dialog angezeigt. Dieser ermöglicht es dem Benutzer auszuwählen, ob der Cube mitsamt allen Elementen entfernt oder weiterhin in der Analysesituation belassen werden soll. In Abbildung 15 ist das Löschen eines Cubes aus einer Analysesituation dargestellt.

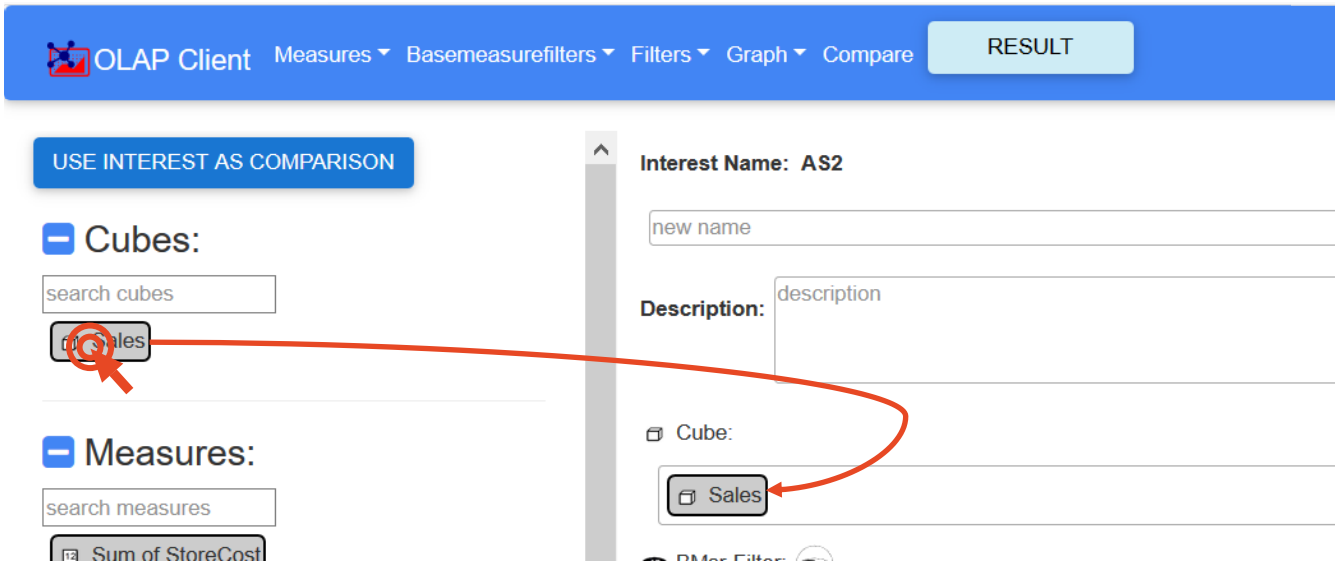


Abbildung 15: Operator SelectCube

Operator ChangeName

Die Analysesituationen sind programmseitig mit einem automatisch generierten, durchnummernierten Namen versehen. Dieser kann vom Benutzer geändert werden um so einer Analysesituation einen speziellen Namen zuzuweisen. Dazu muss einfach der jeweilige Name in das entsprechnede Feld eingetragen werden. Nachdem der Benutzer seine Eingabe mit Enter bestätigt, wird der eingegebene Name für die nächste Analysesituation übernommen.

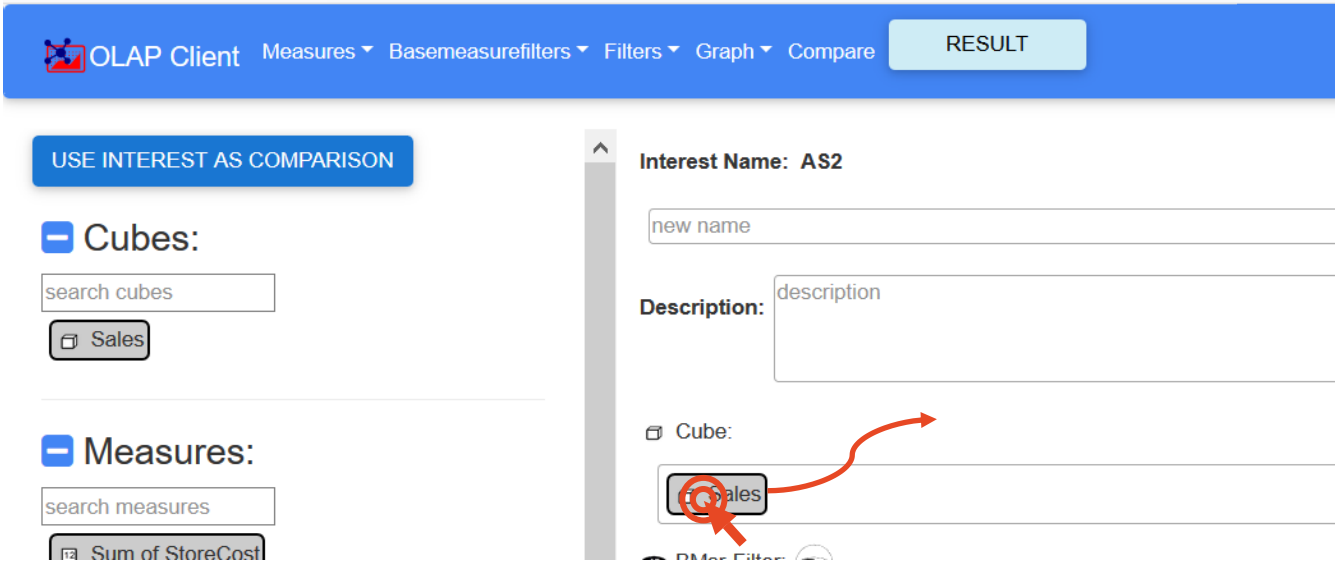


Abbildung 16: Operator DropCube

Operator ChangeDescription

Ähnlich zum Ändern des Names einer Analysesituation kann der Benutzer genauso eine Beschreibung zu einer Analysesituation hinzufügen und bearbeiten. Darin können weitere Gedanken oder Erklärungen und Zusammenhänge der Elemente in der Analysesituation angeführt werden. Die Beschreibung wird geändert sobald der Benutzer nach einer Änderung das Feld der Beschreibung verlässt.

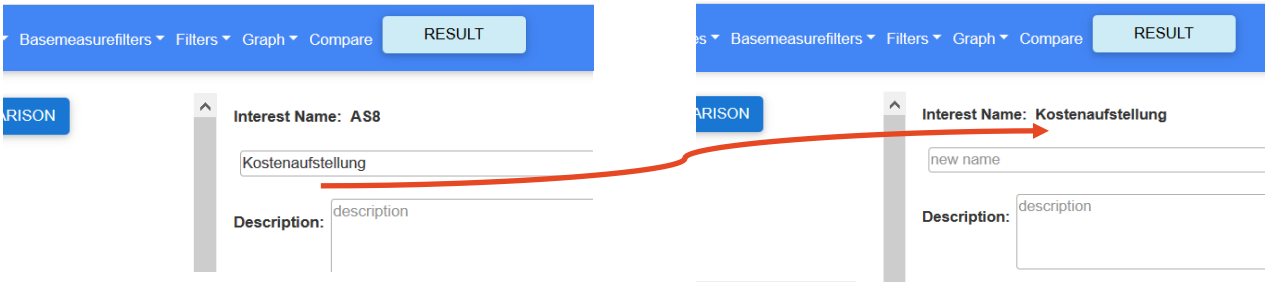


Abbildung 17: Operator ChangeName

Operator DropDimension

Der Operator DropDimension entfernt alle Inhalte einer Dimension aus einer Analysesituation. Jede Dimension, die einer Analysesituation hinzugefügt wird, besitzt in der rechten oberen Ecke ein kleines Mülltonnensymbol. Durch einen Klick auf dieses wird die Dimension mitsamt all ihrer Elemente aus der Analysesituation entfernt. Abbildung 18 zeigt ein Beispiel für eine Dimension und den genauen Platz des Mülltonnensymbols in der Dimension.

5.2.2. Kennzahlen-Operatoren

Die nun folgenden Operatoren befassen sich mit den Kennzahlen die einer Analysesituation hinzugefügt oder entfernt werden können. Die Tabelle 2 zeigt einen Überblick aller Operatoren, die mit einer Kennzahl arbeiten.

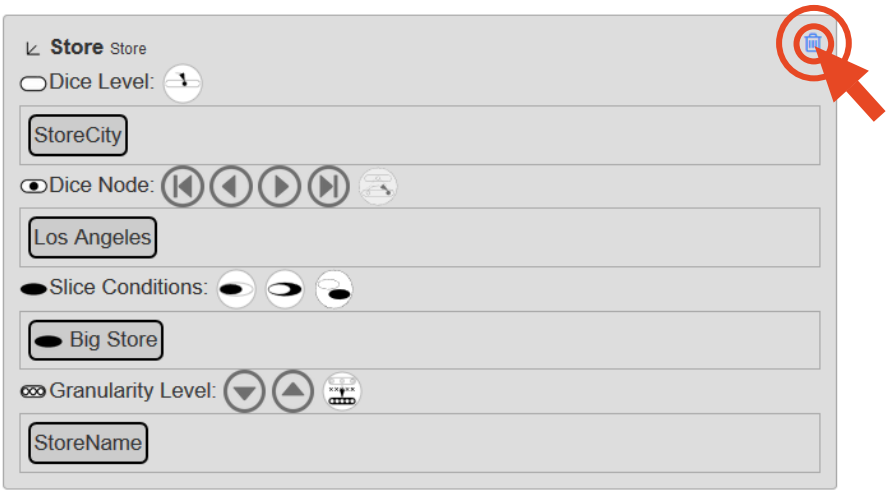
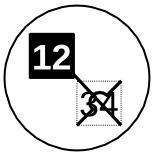
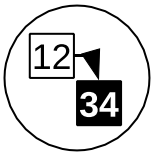


Abbildung 18: Operator DeleteDimension

Tabelle 2: Kennzahlen-Operatoren (Neuböck, 2018)

Operator	Symbol	Beschreibung
AddMeasure		Dieser Operator fügt einer Analysesituation eine Kennzahl hinzu. Parameter: ID der Kennzahl
DropMeasure		Dieser Operator entfernt eine Kennzahl aus einer Analysesituation. Parameter: ID der Kennzahl
RefocusMeasure		Dieser Operator fügt eine Menge von Kennzahlen der Analysesituation hinzu und entfernt eine zweite Menge von Kennzahlen aus der Analysesituation Parameter: Menge von Kennzahlen, die hinzugefügt werden sollen; Menge von Kennzahlen, die entfernt werden sollen
MoveDownToMeasure		Dieser Operator bietet die Möglichkeit eine Kennzahl in verschiedene, zur Berechnung dieser Kennzahl verwendete Kennzahlen, aufzuteilen. Parameter: Menge von Kennzahlen in die die ausgewählte Kennzahl aufgeteilt werden soll

Operator AddMeasure

Der erste Operator, der hier beschrieben wird, ist der AddMeasure Operator. Dieser ermöglicht es dem Benutzer eine Kennzahl zu einer Analysesituation hinzuzufügen. Eine Kennzahl kann über mehrere Arten zu einer Analysesituation hinzugefügt werden. Der Benutzer kann die Kennzahl aus dem Bereich

der Cube-Informationen per Drag & Drop in den Kennzahlenbereich der Analysesituation ziehen. Weiters ist es auch möglich, durch einen Klick auf die Kennzahl im Bereich der Cube-Informationen diese der aktuell ausgewählten Analysesituation hinzuzufügen. Zu sehen in Abbildung 19. Die dritte Möglichkeit zum Hinzufügen einer Kennzahl besteht über das Dropdown Menü in der Navigationsleiste. Im Menüpunkt „Measures“ befindet sich der Unterpunkt „Add Measure“. Durch einen Klick auf diesen Unterpunkt öffnet sich ein Fenster, in dem die hinzuzufügende Kennzahl ausgewählt werden kann. Das dort angezeigte Dropdown Menü zeigt alle vorhandenen Kennzahlen und bietet zusätzlich eine Suchfunktion, um so den Suchvorgang bei einer großen Anzahl an Kennzahlen zu erleichtern. Der genaue Vorgang ist in Abbildung 20 ersichtlich.

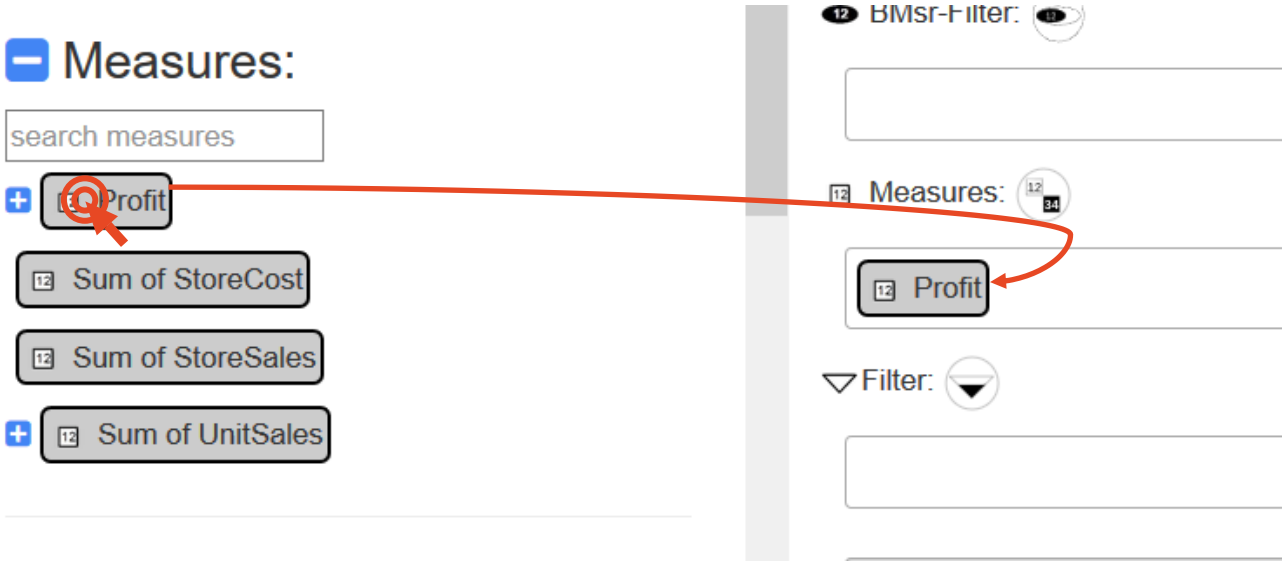


Abbildung 19: Operator AddMeasure (1)

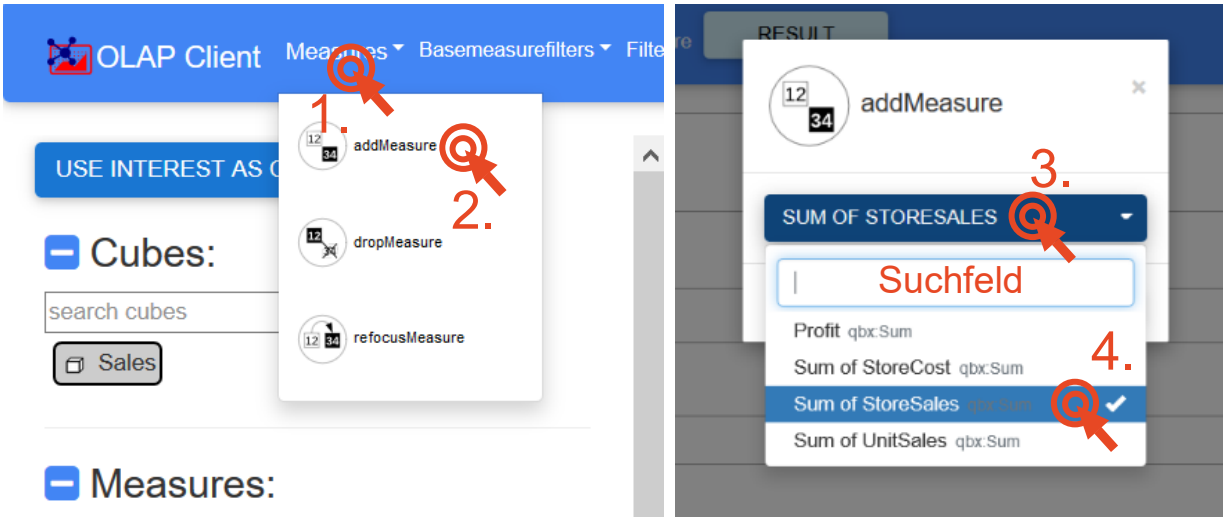


Abbildung 20: Operator AddMeasure (2)

Operator DropMeasure

Um eine Kennzahl aus der Analysesituation zu löschen, kann der Operator DropMeasure verwendet werden. Dieser befindet sich in der Navigationsleiste unter dem Menüpunkt „Measures“. Es öffnet sich

ein Fenster mit einem Dropdown-Menü, das alle in der Analysesituation vorhandenen Kennzahlen anzeigt. Zusätzlich beinhaltet das Dropdown-Menü ein Suchfeld, um so eine spezielle Kennzahl schneller zu finden. Die ausgewählte Kennzahl wird sohin aus der aktuell ausgewählten Analysesituation entfernt. Eine zweite Methode eine Kennzahl aus einer Analysesituation zu entfernen, ist diese direkt anzuklicken und die direkte Auswahl des Menüpunkts „Drop Measure“.

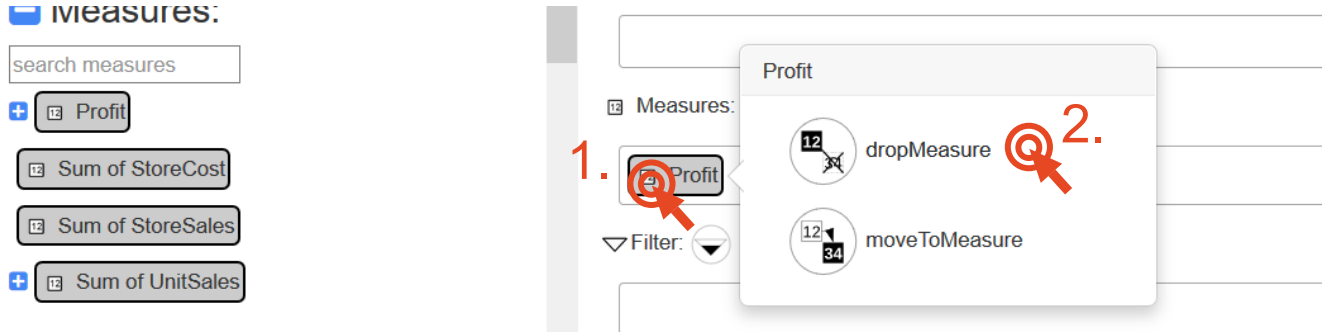


Abbildung 21: Operator DropMeasure (1)

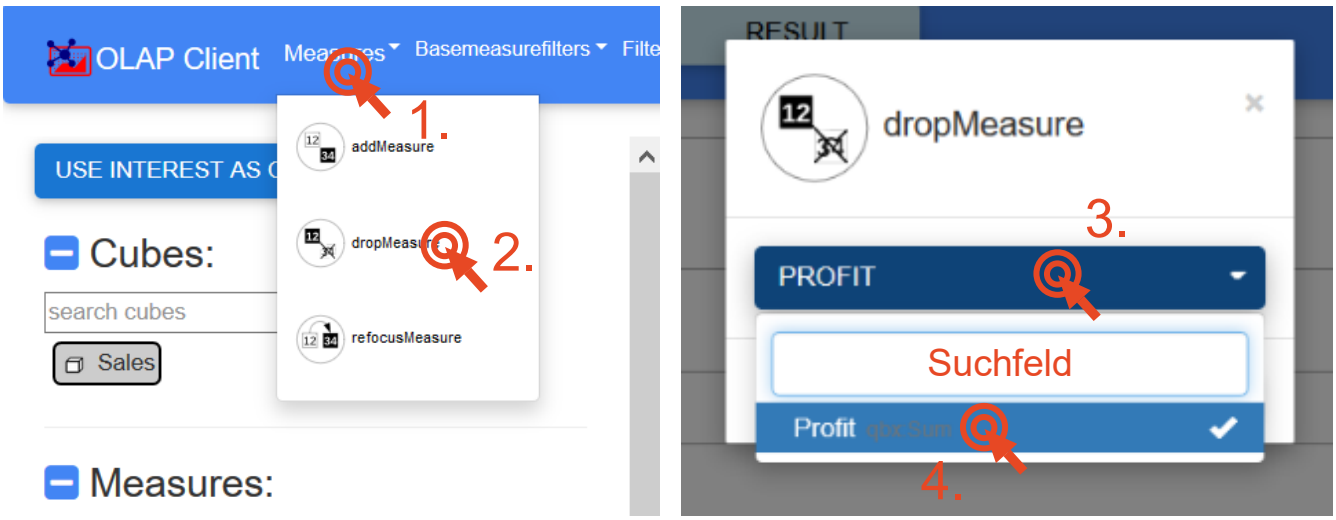


Abbildung 22: Operator DropMeasure (2)

Operator RefocusMeasure

Der AddMeasure- und der DropMeasure-Operator beziehen sich immer jeweils auf eine Kennzahl. Da es aber oft notwendig ist gleich mehrere Kennzahlen hinzuzufügen oder zu entfernen bzw. auszutauschen, vereint der RefocusMeasure Operator diese Aufgaben. Er ermöglicht in einem Schritt eine beliebige Menge an Kennzahlen durch eine andere Menge zu ersetzen. Der Operator kann durch einen Klick auf den Button „Refocus Measure“ in der Navigationsleiste unter dem Punkt „Measures“ aufgerufen werden. Abbildung 23 zeigt das Fenster zum Anpassen der Kennzahlen. Dabei sind auf der linken Seite alle Kennzahlen, die im Cube verfügbar sind und auf der rechten Seite alle Kennzahlen, die bereits in der Analysesituation enthalten sind. Sowohl die rechte als auch die linke Seite kann separat mit einem Filter eingeschränkt werden, um so eine bessere Übersicht über die vorhandenen Kennzahlen zu ermöglichen. Nachdem die Kennzahlen jeweils durch Anklicken auf die richtigen Seiten (Hinzufügen

oder Entfernen) gebracht wurden, werden nach Betätigen das „Refocus Measures“-Button die jeweiligen Änderungen an der Analysesituation durchgeführt.

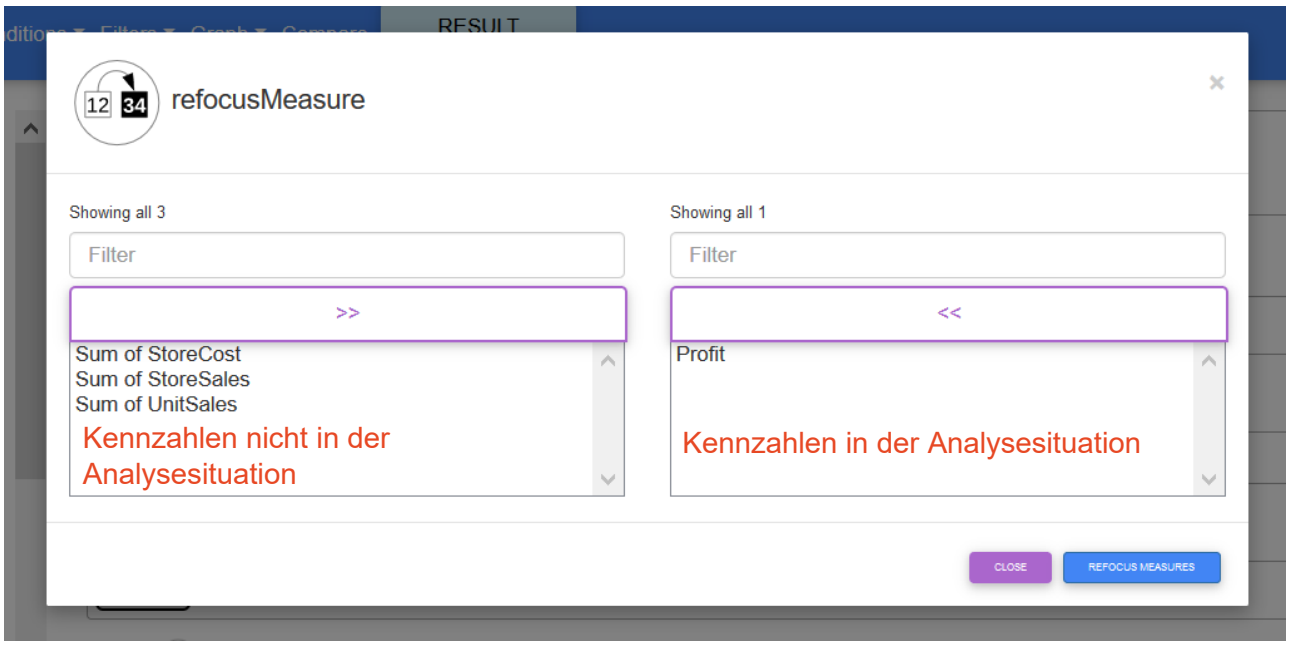


Abbildung 23: Operator RefocusMeasure

Operator MoveToMeasure

Kennzahlen können sich nicht nur auf die Basiskennzahlen eines Cube beziehen, sondern auch aus anderen bereits definierten Kennzahlen zusammensetzen. Beispielsweise setzt sich die Kennzahl Profit aus der Kennzahl Summe der Einnahmen – Summe der Ausgaben zusammen. Um diese zusammengesetzten Kennzahlen einfach und unkompliziert in ihre Bestandteile teilen zu können, bietet der Operator MoveToMeasure die Funktionalität alle oder nur einzelne ausgewählte Kennzahlen, anstatt der zusammengesetzten in die Analysesituation einzufügen. Aufgerufen wird der Operator durch einen Klick auf die Kennzahl in der jeweiligen Analysesituation und mit Auswählen des Menüpunktes „Move to measure“. Im danach erscheinenden Fenster können im Dropdown-Menü alle gewünschten für die zusammengesetzte Kennzahl verwendeten Kennzahlen ausgewählt werden. Nach dem Auswählen kann durch einen Klick auf „Move to Measure“ der Operator angewendet werden.

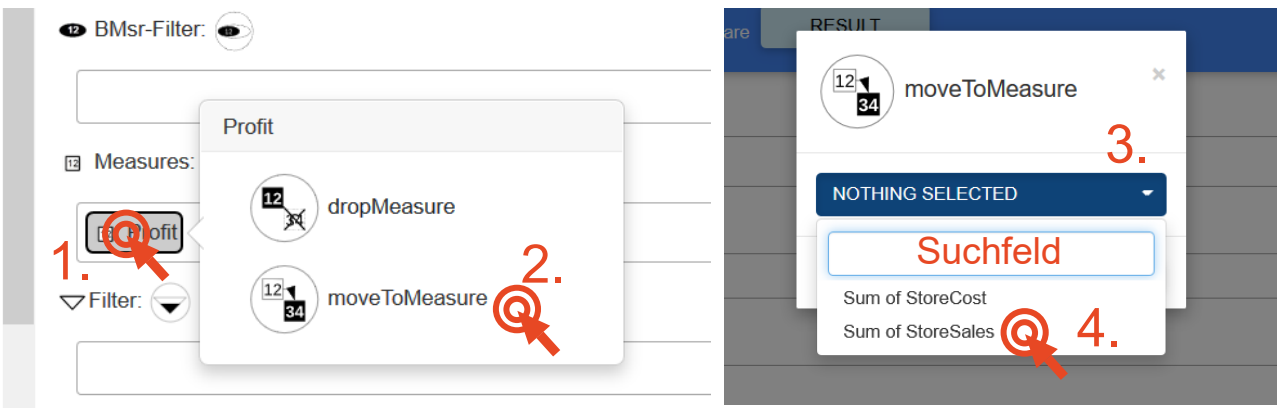


Abbildung 24: Operator MoveToMeasure

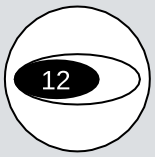
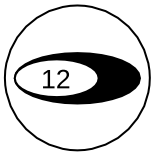
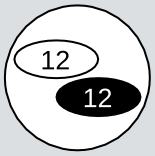
5.2.3. Basiskennzahlenfilter-Operatoren

Die Tabelle 15 zeigt einen Überblick über alle Operatoren, die verwendet werden können, um die Basiskennzahlenfilter einer Analysesituation zu verändern. Es handelt sich dabei um die Operatoren zum Hinzufügen und Löschen eines Basiskennzahlenfilters, sowie einem Operator, um eine Menge von Basiskennzahlenfiltern durch eine andere Menge auszutauschen.

Operator AddBaseMeasureFilter

Durch Anwenden des Operators AddBaseMeasureFilter kann ein Benutzer einen Filter auf die Basiskennzahlen der aktuellen Analysesituation hinzufügen. Es bieten sich dem Benutzer mehrere Arten, um diesen Operator anzuwenden. Durch einen Klick auf den jeweiligen Basiskennzahlenfilter im Bereich der Cube-Informationen wird dieser in die aktuell ausgewählte Analysesituation eingefügt. Weiters ist es möglich, diesen mittels Drag & Drop in das jeweilige dafür vorgesehene Feld in der Analysesituation zu ziehen. Diese Schritte sind genauso für den Operator AddMeasure möglich und daher in Abbildung 19 ersichtlich. Es besteht genauso die Möglichkeit den Operator aus der Navigationsleiste des Webclients auszuwählen. Dazu das Menü „Basemeasurefilter“ öffnen und den Menüpunkt „add BMsrFilter“ auswählen. Dieser öffnet ein Fenster, in dem ein vorhandener Basiskennzahlenfilter ausgewählt werden kann, der dann zur Analysesituation hinzugefügt wird. Es besteht die Möglichkeit den Basiskennzahlenfilter mittels eines Suchfeldes aus allen bekannten Basiskennzahlenfiltern herauszufiltern. Die Durchführung dieses Prozesses entspricht genau dem in Abbildung 20 gezeigten Prozess für die Kennzahlen.

Tabelle 3: Basiskennzahlenfilter-Operatoren (Neuböck, 2018)

Operator	Symbol	Beschreibung
AddBaseMeasreFilter		Dieser Operator fügt der Analysesituation einen Basiskennzahlenfilter hinzu. Parameter: ID des Basiskennzahlenfilters
DropBaseMeasreFilter		Dieser Operator entfernt einen Basiskennzahlenfilter aus der entsprechenden Analysesituation. Parameter: ID des Basiskennzahlenfilters
Refocus BaseMeasureFilter		Dieser Operator fügt der Analysesituation eine Menge von Basiskennzahlenfiltern hinzu und entfernt eine zweite Menge von Basiskennzahlenfiltern aus der Analysesituation. Parameter: Menge von Basiskennzahlenfiltern, die hinzugefügt werden sollen; Menge von Basiskennzahlenfiltern, die entfernt werden sollen

Operator DropBaseMeasureFilter

Der Operator DropBaseMeasureFilter entfernt einen Basiskennzahlenfilter aus der Analysesituation. Der Benutzer hat verschiedene Möglichkeiten diesen anzuwenden. Durch Anklicken des Basiskennzahlenfilters in der Analysesituation wird dieser aus der Analysesituation gelöscht. Es ist genauso möglich, den Basiskennzahlenfilter per Drag & Drop aus dem dafür vorgesehen Feld zu ziehen. Dieser Prozess ähnelt dem zum Entfernen des Cubes und ist in Abbildung 16 abgebildet. Der Anwender hat auch die Möglichkeit den Operator über die Navigationsleiste auszuwählen. Dieser befindet sich im Menü „Basemeasurefilters“. Durch Auswählen des Unterpunktes „dropBMsrFilter“ öffnet sich ein Fenster, in dem alle derzeit in der Analysesituation enthaltenen Basiskennzahlenfilter aufgelistet sind. Es besteht die Möglichkeit, einen gesuchten Basiskennzahlenfilter in dem dafür vorgesehenen Suchfeld einzugeben, um so die Auswahl zu erleichtern. Der ausgewählte Basiskennzahlenfilter wird nach dem Bestätigen aus der Analysesituation gelöscht. Diese Form der Anwendung ähnelt den in Abbildung 22 gezeigten Schritten zum Entfernen einer Kennzahl.

Operator RefocusBaseMeasureFilter

Da oft mehrere Änderungen an den Basiskennzahlenfiltern vorgenommen werden müssen, besteht die Möglichkeit diese Änderungen durch einmaliges Anwenden des Operators RefocusBaseMeasreFilter durchzuführen, anstatt dem mehrmaligen Anwenden der add und drop Operatoren. Um den Operator anzuwenden, muss der Anwender das Menü „Basemeasurefilters“ in der Navigationsleiste anklicken und dann den Menüpunkt „refocusBMsrFilter“ wählen. Es öffnet sich ein Fenster, das in zwei Spalten unterteilt ist. Die linke Seite zeigt dabei alle im Cube vorhandenen Basiskennzahlenfilter. Die rechte Seite zeigt alle bereits in der Analysesituation befindlichen Basiskennzahlenfilter. Durch einen Klick auf den jeweiligen Basiskennzahlenfilter kann dieser hinzugefügt oder entfernt werden. Die beiden Pfeile oberhalb der Spalten erlauben alle Basiskennzahlenfilter der jeweiligen Seite auf die andere Seite zu verschieben. Das Filterfeld über den Spalten erlaubt eine Einschränkung der angezeigten Basiskennzahlenfilter anhand einer eingegebenen Filterbedingung. Durch Bestätigen der Eingabe mit dem „Refocus BMsr Filter“-Button werden die Änderungen an der Analysesituation vorgenommen. Abbildung 23 zeigt genau diesen Vorgang am Beispiel des Operators RefocusMeasure.

5.2.4. Filter-Operatoren

Dieser Abschnitt befasst sich mit den Änderungen an den Filtern für die Kennzahlen. Eine Übersicht aller Operatoren ist in Tabelle 4 ersichtlich. Eine genaue Beschreibung der einzelnen Operatoren folgt anschließend an diese Tabelle.

Operator Add Filter

Um einen Filter zu einer Analysesituation hinzuzufügen, kann der Benutzer den Operator Add Filter verwenden. Es gibt verschiedene Möglichkeiten diesen Operator aufzurufen. Der Benutzer kann einen Filter aus dem Bereich der Cube-Informationen per Drag & Drop in das dafür vorgesehene Feld in der

Analysesituation ziehen. Weiters kann dieser Filter durch einen Klick auf das jeweilige Element in den Cube-Informationen zur aktuell ausgewählten Analysesituation hinzugefügt werden. Abbildung 1 zeigt diesen Vorgang am Beispiel einer Kennzahl. Der Benutzer kann ebenso die Navigationsleiste verwenden, um den Operator aufzurufen. Dazu wird aus dem Menü „Filter“ der Unterpunkt „addFilter“ ausgewählt. Dieser öffnet ein neues Fenster, in dem einer der vordefinierten Filter aus einem Drop-Down-Menü ausgewählt werden kann. Ein Suchfeld erleichtert das Finden des gewünschten Filters. Nach Bestätigen des Fensters wird der ausgewählte Filter der Analysesituation hinzugefügt. Eine bildliche Darstellung dieses Prozesses kann am Beispiel einer Kennzahl in Abbildung 20 nachvollzogen werden.

Tabelle 4: Filter-Operatoren (Neuböck, 2018)

Operator	Symbol	Beschreibung
AddFilter		Dieser Operator fügt der Analysesituation einen Filter hinzu. Parameter: ID des Filters
DropFilter		Dieser Operator entfernt einen Filter aus der Analysesituation. Parameter: ID des Filters
RefocusFilter		Dieser Operator fügt der Analysesituation eine Menge von Filtern hinzu und entfernt gleichzeitig eine zweite Menge von Filtern aus der Analysesituation. Parameter: Menge von Filtern, die hinzugefügt werden sollen; Menge von Filtern, die entfernt werden sollen

Operator DropFilter

Das Entfernen eines Filters aus einer Analysesituation kann mithilfe des Operators DropFilter durchgeführt werden. Dieser kann durch Klicken auf den jeweiligen Filter in der Analysesituation angewendet werden. Weiters kann der Benutzer den Filter per Drag & Drop aus dem dafür vorgesehen Feld ziehen, um ihn zu entfernen. Diese Möglichkeiten zum Entfernen des Filters sind in Abbildung 16 anhand eines Cubebeispiels zu sehen. Der Benutzer kann den Operator genauso aus dem Navigationsmenü des Webclients auswählen. Dazu wird im Menü „Filters“ der Unterpunkt „dropFilter“ ausgewählt. Dieser öffnet ein neues Fenster in dem der Benutzer in einem Drop-Down-Menü einen, der in der Analysesituation befindlichen Filter auswählen kann. Dieser wird nach Bestätigen des Fensters aus der Analysesituation entfernt. Ein Suchfeld erleichtert das Auffinden von speziell gesuchten Filtern. Abbildung 22 verbildlicht diesen Prozess mit dem Entfernen einer Kennzahl.

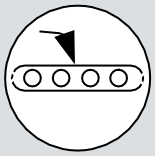
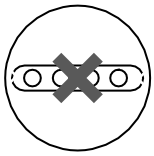
Operator RefocusFilter

Um das Hinzufügen und Entfernen von mehreren Filtern auf einmal zu ermöglichen, wird der Operator RefocusFilter angeboten. Dieser kann über das Navigationsmenü des Webclients aufgerufen werden. Durch einen Klick auf den Unterpunkt „refocusFilter“ im Menü „Filters“ öffnet sich ein neues Fenster. Dieses ist in zwei Spalten unterteilt. Die linke Seite zeigt alle definierten Filter, die sich aktuell nicht in der ausgewählten Analysesituation befinden und die rechte Seite alle Filter, die derzeit in der Analysesituation enthalten sind. Durch einen Klick auf den jeweiligen Filter kann dieser auf die andere Seite verschoben und so hinzugefügt oder entfernt werden. Die Pfeile oberhalb der Spalten ermöglichen es, alle Filter in der Spalte auf die jeweils andere Seite zu verschieben. Jede Spalte bietet zusätzlich ein Suchfeld, um die Anzeige der jeweiligen Filter einschränken zu können. Abbildung 23 zeigt die Anwendung dieses Operators an dem Beispiel von Kennzahlen.

5.2.5. Dice-Level-Operatoren

Die hier beschriebenen Operatoren befassen sich mit dem Hinzufügen und Löschen bzw. Ändern eines Dice-Levels einer Dimension. Eine Übersicht der vorhandenen Operatoren ist in Tabelle 5 näher beschrieben. Eine Änderung des Dice-Levels zieht auch gleichzeitig eine Änderung der dazugehörigen Dice-Node nach sich.

Tabelle 5: Dice-Level-Operatoren (Neuböck, 2018)

Operator	Symbol	Beschreibung
ChangeDiceLevel		Dieser Operator ändert den Dice-Level einer Dimension in einer Analysesituation. Parameter: ID des Levels
DropDiceLevel		Dieser Operator entfernt den Dice-Level und die Dice-Node aus einer Dimension einer Analysesituation. Parameter: ID des Levels

Operator ChangeDiceLevel

Um den Dice-Level einer Dimension zu ändern und so die Anzeige des Ergebnisses zu verändern, kann der Benutzer den Operator ChangeDiceLevel anwenden. Ein Level kann aus dem Bereich der Cube-Informationen in den Bereich des Dice-Level mittels Drag & Drop gezogen werden. Dabei spielt es keine Rolle, ob die Dimension, in die der Level gezogen wurde, die Dimension ist, der der Level angehört. Der Level wird automatisch der richtigen Dimension zugeordnet. Falls die Dimension des Levels noch nicht in der Analysesituation angezeigt wird, wird diese erzeugt und in die Analysesituation eingefügt. Abbildung 25 veranschaulicht die Anwendung des Operators. Es wird der Level Product Name in das Feld des Dice-Levels gezogen. Da die Dimension Product noch nicht in der Analysesituation enthalten ist, wird diese hinzugefügt. Der fett-gedruckte Name ist die Bezeichnung der Dimension. Der kleinere

der beiden Namen weist auf die Hierarchie hin, in welcher sich der Level in der Dimension befindet. Es kann immer nur eine Hierarchie pro Ebene ausgewählt werden.

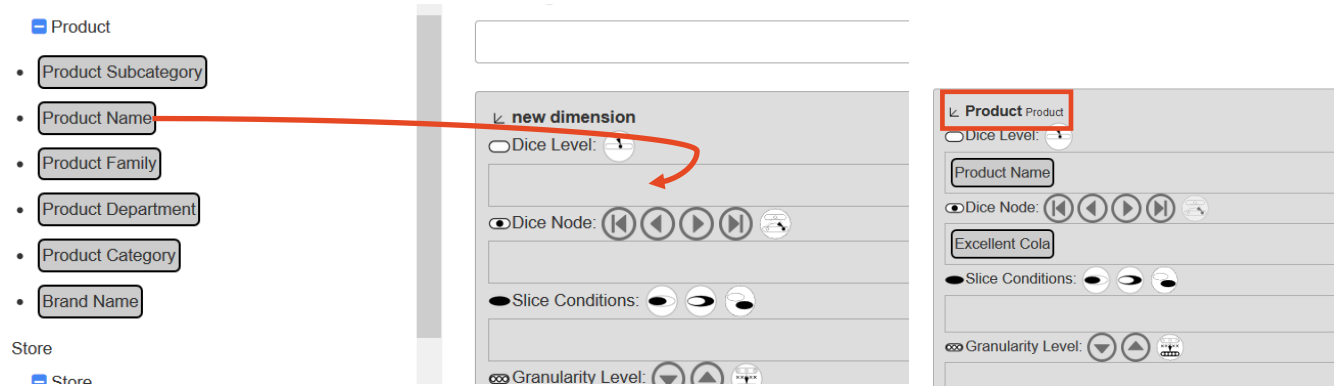


Abbildung 25: Operator ChangeDiceLevel

Operator DropDiceLevel

Um einen Dice-Level aus einer Dimension zu entfernen und so wieder auf den allgemeinsten Level zurückzukehren, kann der Benutzer den Operator DropDiceLevel anwenden. Durch einen Klick auf den Dice-Level in der Dimension wird dieser, mit seinen zugehörigen Dice-Node aus der Analysesituation entfernt. Es ist auch möglich den Dice-Level mittels Drag & Drop aus dem dafür vorgesehenen Fenster zu ziehen, um ihn so zu entfernen. Die Anwendung des Operators wird in Abbildung 26 verbildlicht.

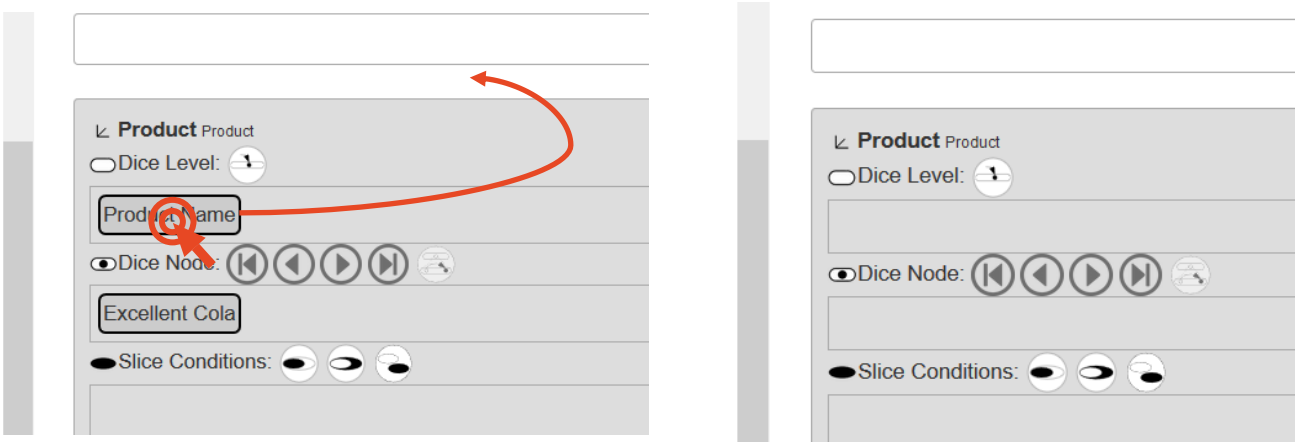


Abbildung 26: Operator DropDiceLevel

5.2.6. Dice-Node-Operatoren

Nachdem einer Dimension ein Dice-Level zugeordnet wurde, kann die automatisch gewählte Dice-Node auf den gewünschten Wert gebracht werden. Dazu stehen dem Benutzer verschiedene Operatoren zur Verfügung. Eine Übersicht dieser ist in Tabelle 6 ersichtlich.

Operator ChangeDiceNode

Um eine Dice-Node auf eine beliebige andere zu wechseln, kann der Benutzer den Operator ChangeDiceNode verwenden. Um diesen aufzurufen, muss der Benutzer auf das Symbol über dem Feld für die Dice-Node klicken. Es öffnet sich ein neues Fenster, in dem aus einem Drop-Down-Menü

eine neue Dice-Node aus allen vorhandenen Dice-Nodes ausgewählt wird. Ein Suchfeld unterstützt den Anwender beim Auffinden von gesuchten Werten. Nach Bestätigen des Fensters wird die ausgewählte Dice-Node in die Dimension übernommen. Abbildung 27 zeigt diesen Vorgang.

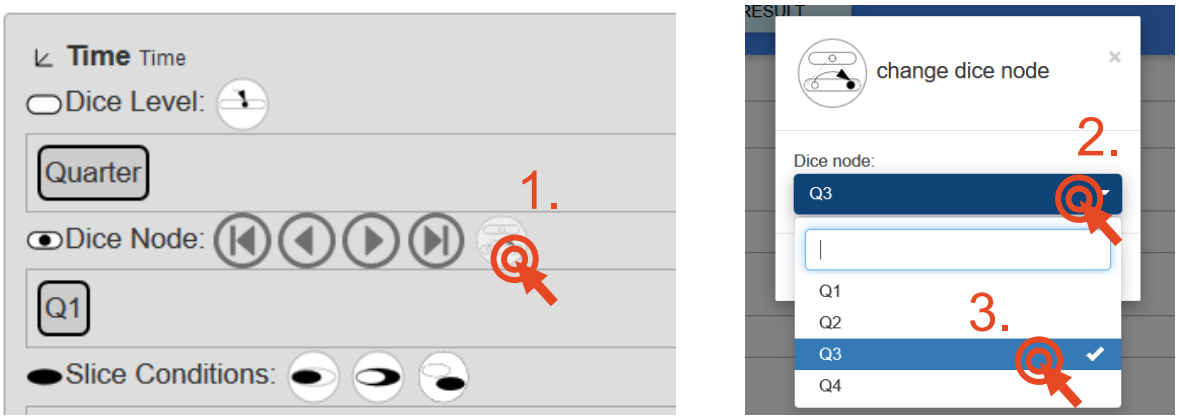
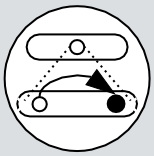
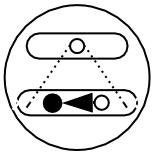
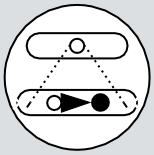
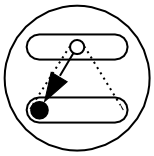
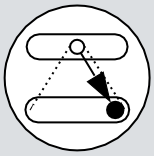


Abbildung 27: Operator ChangeDiceNode

Tabelle 6: Dice-Node-Operatoren (Neuböck, 2018)

Operator	Symbol	Beschreibung
ChangeDiceNode		Dieser Operator ändert die Dice-Node einer Dimension in einer Analysesituation. Parameter: ID der Dice-Node
MoveToPrevNode		Dieser Operator ändert die Dice-Node einer Dimension in einer Analysesituation auf die vorhergehende Dice-Node. Parameter: ID der Dimension; ID der Dice-Node
MoveToNextNode		Dieser Operator ändert die Dice-Node einer Dimension in einer Analysesituation auf die nachfolgende Dice-Node. Parameter: ID der Dimension; ID der Dice-Node
MoveToFirstNode		Dieser Operator ändert die Dice-Node einer Dimension in einer Analysesituation auf die erste bekannte Dice-Node. Parameter: ID der Dimension; ID der Dice-Node
MoveToLastNode		Dieser Operator ändert die Dice-Node einer Dimension in einer Analysesituation auf die letzte bekannte Dice-Node. Parameter: ID der Dimension; ID der Dice-Node

Operator MoveToPrevNode / MoveToNextNode

Um die Navigation durch die einzelnen Dice Nodes zu erleichtern, kann der Benutzer die Operatoren MoveToPrevNode bzw. MoveToNextNode benützen. Diese ändern die Dice-Node auf den jeweils vorherigen bzw. nächsten Wert. Dadurch kann mit nur einem Klick durch die Reihenfolge der Dice-Nodes durchgewechselt werden. Ist die Dice-Node bereits ein Ende der Reihenfolge z.B. Jänner für ein Dice-Level Monat, dann erfolgt eine Meldung, dass das Ende erreicht ist. Aufgerufen werden diese Operatoren durch einen Klick auf die einfachen Pfeile oberhalb des Felds für die Dice-Node in der Dimension. Die jeweiligen Buttons sind in Abbildung 28 mit rot gekennzeichnet.

Operator MoveToFirstNode / MoveToLast Node

Die Operatoren MoveToFirstNode und MoveToLastNode bieten einen schnellen Weg, in der Reihenfolge der Dice-Nodes zu wechseln. Dabei wird durch die Anwendung des MoveToFirstNode Operators die in der Reihenfolge erste Dice-Node ausgewählt und beim Operator MoveToLastNode die in der Reihenfolge letzte. Der Anwender kann diese Operatoren benutzen indem er auf die äußeren Pfeile oberhalb des Feldes für die Dice-Nodes klickt. Abbildung 28 verweist auf die jeweiligen Buttons in blauer Farbe.

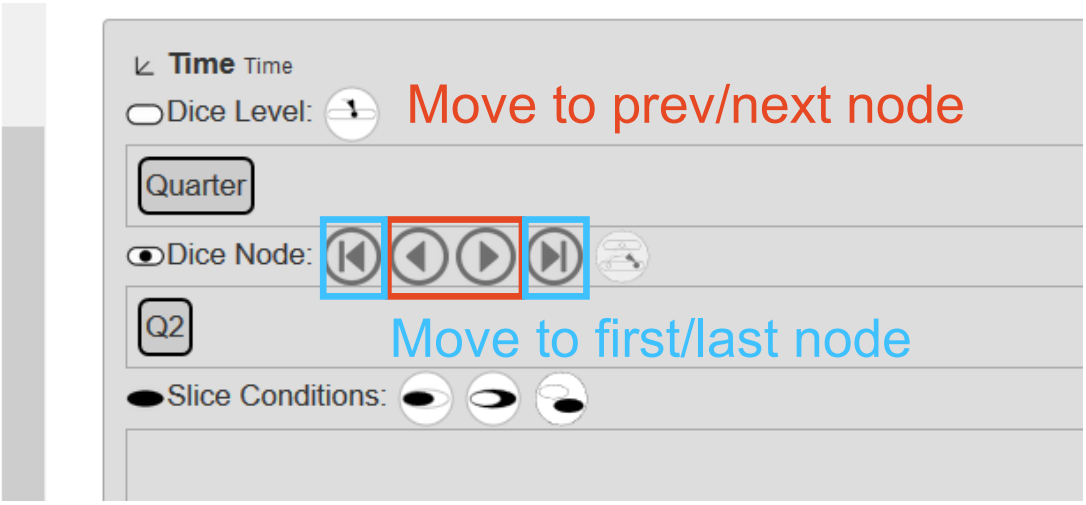


Abbildung 28: MoveTo-Operatoren

5.2.7. Slice-Condition-Operatoren

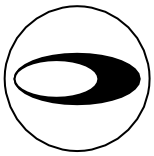
Eine Dimension kann mithilfe von Slice-Conditions eingeschränkt werden. Der Benutzer kann die vordefinierten Slice-Conditions zu einer Dimension hinzufügen oder entfernen. Tabelle 7 gibt einen Überblick über alle vorhanden Operatoren, die mit der Slice-Condition arbeiten.

Operator AddSliceCondition

Um eine Slice-Condition einer Dimension hinzuzufügen kann der Operator AddSliceCondition verwendet werden. Ein Benutzer kann eine Slice-Condition zu der ausgewählten Analysesituation hinzufügen indem er auf die jeweilige Slice-Condition im Bereich der Cube-Informationen klickt. Weiters ist es möglich, diese mittels Drag & Drop in das jeweilige Feld in der Dimension zu ziehen. Abbildung

25 zeigt diese Anwendung anhand des Dice-Levels. Der Anwender hat auch die Möglichkeit durch Drücken eines Buttons über dem Feld der Slice-Conditions ein Fenster zu öffnen. In diesem Fenster kann eine Slice-Condition aus allen vorhandenen Slice-Conditions ausgewählt werden. Ein Suchfeld unterstützt beim Auffinden der gebrauchten Slice-Condition. Nach Bestätigen des Fensters wird die gewählte Slice-Condition der Dimension hinzugefügt. Eine beispielhafte Anwendung dieses Operators kann in Abbildung 27 anhand des Beispiels der Dice-Node nachvollzogen werden.

Tabelle 7: Slice-Condition-Operatoren (Neuböck, 2018)

Operator	Symbol	Beschreibung
AddSliceCondition		Dieser Operator fügt einer Dimension einer Analysesituation eine Slice-Condition hinzu. Parameter: ID der Slice-Condition
DropSliceCondition		Dieser Operator entfernt eine Slice-Condition aus einer Dimension einer Analysesituation. Parameter: ID der Slice-Condition
RefocusSliceCondition		Dieser Operator fügt eine Menge von Slice-Conditions einer Dimension einer Analysesituation hinzu und entfernt eine zweite Menge von Slice-Conditions aus der Dimension. Parameter: Menge von Slice-Conditions, die hinzugefügt werden sollen; Menge von Slice-Conditions, die entfernt werden sollen

Operator DropSliceCondition

Um eine Slice-Condition aus einer Dimension zu entfernen, kann der Operator DropSliceCondition verwendet werden. Ein Benutzer kann durch einen Klick auf eine Slice-Condition in einer Dimension diese aus der Dimension entfernen. Weiters ist es möglich, eine Slice-Condition mittels Drag & Drop aus dem dafür vorgesehenen Feld zu ziehen und diese so zu entfernen. Der Benutzer hat auch die Möglichkeit mittels eines Buttons über dem Feld für die Slice-Conditions eine auszuwählen und diese aus der Dimension zu entfernen. Durch den Klick auf diesen Button öffnet sich ein eigenes Fenster mit einem Drop-Down-Menü indem die zu entfernende Analysesituation ausgewählt werden kann. Ein Suchfeld unterstützt beim Auffinden einer speziellen Slice-Condition. Abbildung 27 zeigt diesen Vorgang anhand einer Dice-Node.

Operator RefocusSliceCondition

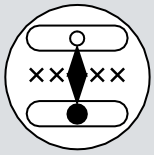
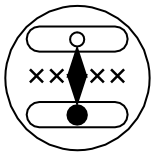
Da oftmals mehr als eine Änderung an den Slice-Conditions notwendig ist, bietet der Operator RefocusSliceConditions eine Möglichkeit nicht nur eine Änderung auf einmal vorzunehmen. Der

Anwender kann durch einen Klick auf einen Button über dem Feld für die Slice-Conditions das Fenster zum Ändern der Slice-Conditions öffnen. In der linken Spalte werden alle Slice-Conditions angezeigt, die im erweiterten Faktenschema definiert wurden. Die rechte Spalte zeigt alle Slice-Conditions die bereits in der Dimension enthalten sind. Durch einen Klick auf die jeweilige Slice-Condition kann diese auf die andere Seite gebracht werden. Die Pfeile über den Spalten ermöglichen es dem Anwender alle Slice-Conditions auf die jeweils andere Seite zu bringen. Ein Suchfeld bietet die Möglichkeit die angezeigten Slice-Conditions zu filtern. Nach Bestätigen des Fensters werden die Änderungen an den Slice-Conditions in der Dimension vorgenommen. Ein Beispiel dieses Operators kann anhand von Abbildung 23 gezeigt werden, in dem der Vorgang an den Kennzahlen erläutert wird.

5.2.8. Granularitäts-Operatoren

Die folgenden Operatoren befassen sich mit der Granularität einer Dimension. Die Tabelle 8 gibt einen Überblick über alle Operatoren, die verwendet werden können, um die Granularität einer Dimension zu ändern. Die verschiedenen Operatoren werden nachfolgend genauer beschrieben und erklärt.

Tabelle 8: Granularitäts-Operatoren (Neuböck, 2018)

Operator	Symbol	Beschreibung
Change Granularity		Dieser Operator ändert den Granularitätslevel einer Dimension der Analysesituation. Parameter: ID des Levels
Drop Granularity		Dieser Operator entfernt den Granularitätslevel einer Dimension der Analysesituation und setzt ihn auf den allgemeinsten Level zurück. Parameter: ID der Dimension; ID des Levels

Operator ChangeGranularity

Um die Granularität einer Dimension zu ändern, kann ein Anwender den Operator ChangeGranularity verwenden. Dieser erlaubt es den Detailgrad einer Dimension zu ändern. Dazu kann der Benutzer einen Level in den Cube-Informationen anklicken. Dieser wird dann automatisch als Granularitätslevel seiner Dimension übernommen. Es besteht genauso die Möglichkeit, den Level mittels Drag & Drop in das Feld für die Granularität zu ziehen und so eine Änderung vorzunehmen. In Abbildung 29 ist die Anwendung des Operators mittels dieser Methoden zu sehen. Eine weitere Art diesen Operator anzuwenden, ist der Aufruf über den Button, der sich über dem Feld für den Granularitätslevel befindet. Dieser öffnet ein neues Fenster mit einem Drop-Down-Menü. Der Benutzer kann nun aus allen vorhandenen Leveln einen als Granularität für die Dimension auswählen. Nach dem Bestätigen des

Fensters wird der Level in die Dimension übernommen. Eine beispielhafte Anwendung dieses Prozesses ist in Abbildung 23 zu sehen, und zwar anhand dem Operator RefocusMeasures.

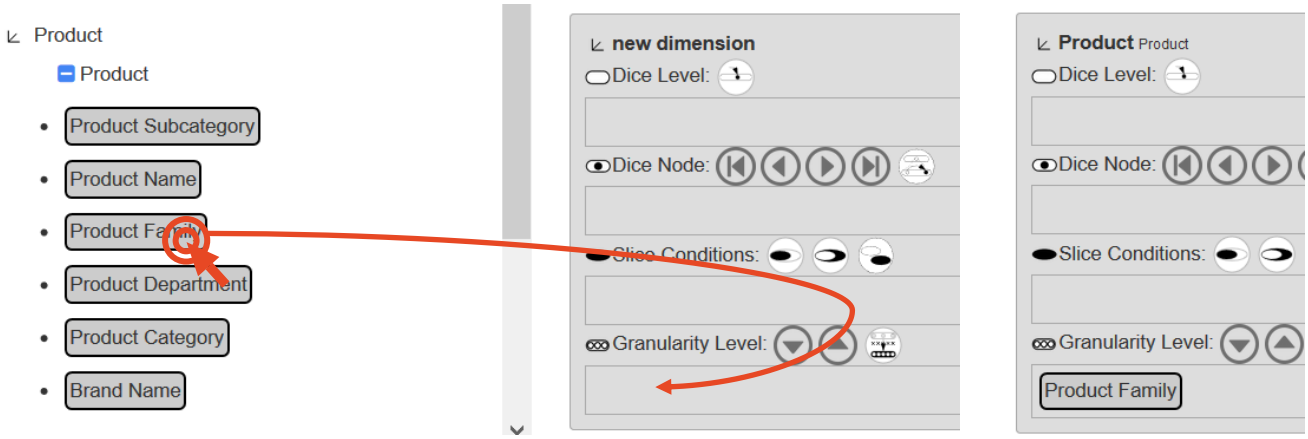


Abbildung 29: Operator ChangeGranularity

Operator DropGranularity

Um ein Granularitätslevel aus einer Dimension zu entfernen, kann ein Anwender den Operator DropGranularity verwenden. Er kann durch einen Klick auf das Level im Granularitätsfeld dieses Level aus der Dimension entfernen. Weiters ist es möglich, das Granularitätslevel mittel Drag & Drop aus dem dafür vorgesehen Feld zu ziehen und so zu entfernen. Eine beispielhafte Anwendung dieses Operators ist in Abbildung 26 anhand des Entfernens eines Dice-Levels gezeigt.

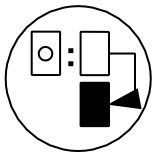
5.3. Vergleichende Analysesituationen

Der folgende Abschnitt befasst sich mit der Benutzung der vergleichenden Analysesituationen. Es werden sämtliche verfügbare Operatoren beschrieben und vorgestellt. Zusätzlich wird deren Anwendung bildlich dargestellt und beschrieben.

5.3.1. ChangeSet-Operatoren

Die ChangeSet-Operatoren befassen sich mit Änderungen der beiden einfachen Analysesituationen, die in einer vergleichenden Analysesituation enthalten sind. Zur Anwendung kommen die beiden Operatoren sobald der Benutzer eine Änderung an einer einfachen Analysesituation vornimmt, die wiederum für einen Vergleich in einer vergleichenden Analysesituation verwendet wird. Beispielsweise existiert eine vergleichende Analysesituation mit der ID VA 1. Diese vergleicht die beiden einfachen Analysesituationen EA 1 und EA 2. Wird nun der einfachen Analysesituation EA 1 eine Kennzahl mit dem Operator AddMeasure hinzugefügt, erzeugt dies eine neue einfache Analysesituation EA 3. Gleichzeitig wird auch eine neue vergleichende Analysesituation VA 2 angelegt. Diese ist über einen Change Set Operator mit der vergleichenden Analysesituation VA 1 verbunden und tauscht die einfache Analysesituation EA 1 mit der neu erstellten EA 3 aus. Je nachdem ob die linke Seite (Set of interest) oder die rechte Seite (Set of comparison) einer vergleichenden Analysesituation ausgetauscht wird, wird der Operator ChangeSetOfInterest oder der Operator ChangeSetOfComparison angewendet.

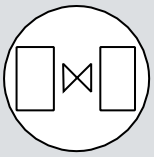
Tabelle 9: ChangeSet-Operatoren (Neuböck, 2018)

Operator	Symbol	Beschreibung
ChangeSetOfInterest		Dieser Operator ändert die Interest Analysesituation einer vergleichenden Analysesituation. Parameter: ID der Analysesituation
ChangeSetOfComparison		Dieser Operator ändert die Comparison Analysesituation einer vergleichenden Analysesituation. Parameter: ID der Analysesituation

5.3.2. Join-Condition-Operatoren

Die nun folgenden Operatoren befassen sich mit den Änderungen der Join-Conditions. Dabei kann ein Benutzer diese zu einer vergleichenden Analysesituation hinzufügen oder wieder entfernen. Je nachdem welche Granularitätslevels in den Dimensionen der zu vergleichenden Analysesituationen gewählt wurden, können verschiedene Join-Conditions ausgewählt werden. In Tabelle 10 ist eine Auflistung aller vorhandenen Operatoren zu sehen.

Tabelle 10: Join-Condition-Operatoren (Neuböck, 2018)

Operator	Symbol	Beschreibung
AddJoinCondition		Dieser Operator fügt einer vergleichenden Analysesituation eine Join-Condition hinzu. Parameter: ID der Join-Condition
DropJoinCondition		Dieser Operator entfernt eine Join-Condition aus einer vergleichenden Analysesituation. Parameter: ID der Join-Condition
RefocusJoinCondition		Dieser Operator fügt einer vergleichenden Analysesituation eine Menge von Join-Conditions hinzu und entfernt eine zweite Menge von Join-Conditions. Parameter: Menge von Join-Conditions, die hinzugefügt werden soll; Menge von Join-Conditions, die entfernt werden soll

Operator AddJoinCondition

Um eine Join-Condition zu einer vergleichenden Analysesituation hinzuzufügen, kann der Benutzer den Operator AddJoinCondition verwenden. Es ist möglich eine Join-Condition aus dem Bereich der Cube-

Informationen mittels Drag & Drop in das Feld der Join-Conditions in der Analysesituation zu ziehen. Weiters besteht die Möglichkeit, diese durch einen Klick auf das Element in den Cube-Informationen zur aktuellen Analysesituation hinzuzufügen. In Abbildung 15 ist dieser Vorgang anhand des Hinzufügens eines Cubes zu sehen.

Die Join-Conditions sind von den Granularitätslevels der Dimensionen der zu vergleichenden Analysesituationen abhängig. Ändert der Benutzer ein Granularitätslevel in einer Dimension wird automatisch überprüft, ob dem Benutzer neue Join-Conditions zur Auswahl stehen bzw. ob es einen Konflikt mit aktuellen Join-Conditions in der Analysesituation gibt. Sollte ein Konflikt auftreten, kann der Benutzer entscheiden, ob er die Join-Conditions, die zu dem Konflikt geführt haben, löscht oder den alten Granularitätslevel behält. Dieser Dialog ist in Abbildung 30 zu sehen.

Operator DropJoinCondition

Der Benutzer kann eine bereits zu einer Analysesituation hinzugefügte Join-Condition auch wieder daraus entfernen. Dazu muss er nur auf die Join-Condition in der Analysesituation klicken. Weiters besteht auch die Möglichkeit, die Join-Condition mittels Drag & Drop einfach aus dem dafür vorgesehen Feld zu ziehen. Abbildung 16 beschreibt diese Anwendung anhand des Entfernens eines Cubes.

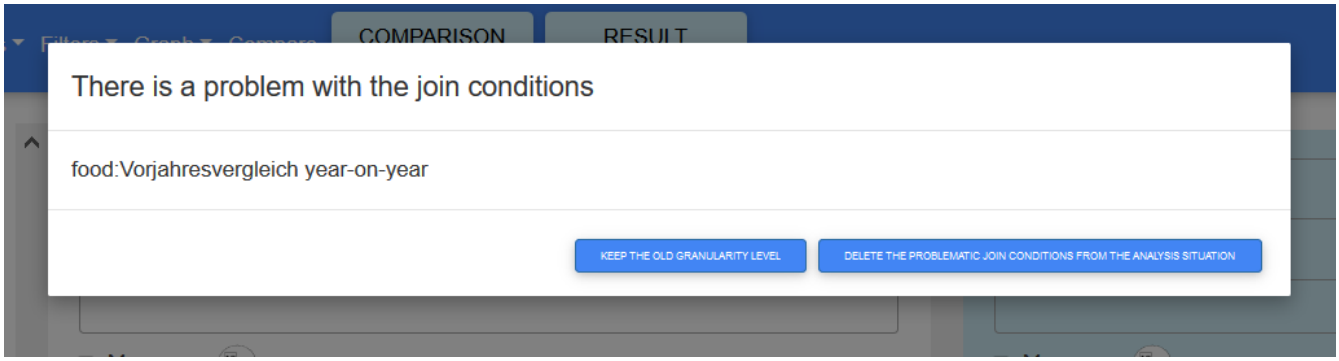


Abbildung 30: Problematische Join-Condition

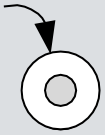
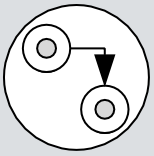
Operator RefocusJoinCondition

Um dem Benutzer eine einfache Methode zu bieten, gleichzeitig mehrere Änderungen an den Join-Conditions durchzuführen, kann der Operator RefocusJoinCondition verwendet werden. Dieser kann durch einen Klick auf das Symbol des Operators neben dem dafür vorgesehen Feld in der Analysesituation aufgerufen werden. Es öffnet sich ein neues Fenster mit zwei Spalten. Die linke Spalte zeigt alle zur Auswahl vorhanden, noch nicht in der Analysesituation befindlichen Join-Conditions. Auf der rechten Seite sind alle bereits in der Analysesituation befindlichen Join-Conditions zu sehen. Durch einen Klick auf eine Join-Condition kann diese hinzugefügt oder entfernt bzw. auf die andere Seite der beiden Spalten gebracht werden. Die Pfeile über den Spalten bieten die Möglichkeit, alle Join-Conditions der jeweiligen Spalte auf die andere Spalte zu bringen. Jede Spalte verfügt zusätzlich noch über ein Suchfeld. Dieses ermöglicht es dem Anwender das Auffinden einer gesuchten Join-Condition zu erleichtern. Ein Beispiel dieses Refocus-Fensters ist in Abbildung 23 zu sehen.

5.3.3. Vergleichende Kennzahlen-Operatoren

Um vergleichende Kennzahlen einer vergleichenden Analysesituation hinzuzufügen oder zu entfernen, kann ein Benutzer verschiedene Operatoren auf verschiedene Arten anwenden. Eine Übersicht aller vorhandenen Operatoren ist in Tabelle 11 zu sehen. Die Anwendung der einzelnen Operatoren wird Schritt für Schritt erklärt.

Tabelle 11: Vergleichende Kennzahlen-Operatoren (Neuböck, 2018)

Operator	Symbol	Beschreibung
AddComparativeMeasure		Dieser Operator fügt einer vergleichenden Analysesituation eine vergleichende Kennzahl hinzu. Parameter: ID der vergleichenden Kennzahl
DropComparativeMeasure		Dieser Operator entfernt eine vergleichende Kennzahl aus einer vergleichenden Analysesituation. Parameter: ID der vergleichenden Kennzahl
RefocusComparative Measure		Dieser Operator fügt einer vergleichenden Analysesituation eine Menge von vergleichenden Kennzahlen hinzu und entfernt eine zweite Menge von vergleichenden Kennzahlen. Parameter: Menge von vergleichenden Kennzahlen, die hinzugefügt werden sollen; Menge von vergleichenden Kennzahlen, die entfernt werden sollen

Operator AddComparativeMeasure

Um einer vergleichenden Analysesituation eine vergleichende Kennzahl hinzuzufügen, kann der Benutzer den Operator AddComparativeMeasure verwenden. Um diesen Operator anzuwenden, kann eine vergleichende Kennzahl aus dem Bereich der Cube-Informationen mittels Drag & Drop in das Feld der Analysesituation gezogen werden. Weiters besteht die Möglichkeit, durch einen Klick auf das Element im Bereich der Cubeninformationen diese in die Analysesituation einzufügen. In Abbildung 15 ist eine beispielhafte Anwendung dieses Operators anhand des Hinzufügens eines Cubes zu sehen.

Operator DropComparativeMeasure

Der Operator DropComparativeMeasure kann verwendet werden, um eine vergleichende Kennzahl aus einer Analysesituation zu entfernen. Der Aufruf dieses Operators erfolgt durch einen Klick auf die vergleichende Kennzahl in der Analysesituation. Die vergleichende Kennzahl kann auch mittels Drag &

Drop aus dem dafür vorgesehen Feld gezogen werden. Abbildung 16 zeigt eine beispielhafte Anwendung dieses Operators beim Entfernen eines Cubes.

Operator RefocusComparativeMeasure

Der Benutzer hat die Möglichkeit durch die Nutzung des Operators RefocusComparativeMeasure mehrere Änderungen an den vergleichenden Kennzahlen gleichzeitig durchzuführen. Dieser wird durch einen Klick auf dessen Symbol neben dem dafür vorgesehenen Feld in der Analysesituation aufgerufen. Dadurch öffnet sich ein neues Fenster, das in zwei Spalten unterteilt ist. Die linke Spalte zeigt alle im Cube verfügbaren vergleichenden Kennzahlen und die rechte Seite alle bereits in der Analysesituation befindlichen vergleichenden Kennzahlen. Über jeder Spalte befindet sich ein Button mit einem Pfeil. Dieser ermöglicht es dem Benutzer, alle vergleichenden Kennzahlen einer Spalte auf die jeweils andere Spalte zu verschieben. Ein Suchfeld kann für die Einschränkung der angezeigten vergleichenden Kennzahlen verwendet werden. Das Refocus-Fenster ist in Abbildung 23 abgebildet.

Tabelle 12: Vergleichende Kennzahlenfilter-Operatoren (Neuböck, 2018)

Operator	Symbol	Beschreibung
AddComparative Measurefilter		Dieser Operator fügt einer vergleichenden Analysesituation einen vergleichenden Kennzahlenfilter hinzu. Parameter: ID des vergleichenden Kennzahlenfilters
DropComparative Measurefilter		Dieser Operator entfernt einen vergleichenden Kennzahlenfilter aus einer vergleichenden Analysesituation. Parameter: ID des vergleichenden Kennzahlenfilters
RefocusComparative Measurefilter		Dieser Operator fügt einer vergleichenden Analysesituation eine Menge von vergleichenden Kennzahlenfilter hinzu und entfernt eine zweite Menge von vergleichenden Kennzahlenfiltern. Parameter: Menge von vergleichenden Kennzahlenfiltern, die hinzugefügt werden sollen; Menge von vergleichenden Kennzahlenfilter, die entfernt werden sollen.

5.3.4. Vergleichende Kennzahlenfilter-Operatoren

Die in diesem Abschnitt beschriebenen Operatoren befassen sich mit den Filtern für die vergleichenden Kennzahlen. Eine Übersicht der Operatoren ist in Tabelle 12 zu sehen. Anschließend werden diese Operatoren im Detail beschrieben und erklärt.

Operator AddComparativeMeasurefilter

Der Operator AddComparativeMeasurefilter erlaubt es dem Benutzer einen Filter auf eine vergleichende Kennzahl zur Analysesituation hinzuzufügen. Dabei gibt es zwei Möglichkeiten diesen Operator anzuwenden. Ein Benutzer kann durch einen Klick auf einen vergleichenden Kennzahlenfilter diesen der Analysesituation hinzufügen oder diesen mittels Drag & Drop in das dafür vorgesehene Feld ziehen. Die Anwendung dieses Operators ist beispielhaft in Abbildung 15 am Beispiel des Hinzufügens eines Cubes zu sehen.

Operator DropComparativeMeasurefilter

Um einen vergleichenden Kennzahlenfilter wieder aus einer Analysesituation zu entfernen, kann der Operator DropComparativeMeasurefilter verwendet werden. Dieser kann durch einen Klick auf den vergleichenden Kennzahlenfilter in der Analysesituation angewendet werden. Zudem ist es möglich, den Filter einfach mittels Drag & Drop aus dem dafür vorgesehen Feld zu ziehen. Der Vorgang zum Entfernen eines vergleichen Kennzahlenfilters ist in Abbildung 16 am Beispiel des Entfernens eines Cubes ersichtlich.

Operator RefocusComparativeMeasurefilter

Der Operator RefocusComparativeMeasurefilter ermöglicht es einem Benutzer mehrere Änderungen an den vergleichenden Kennzahlenfiltern einer Analysesituation gleichzeitig vorzunehmen. Ein Benutzer kann durch das Drücken des Symbols neben dem Feld für die vergleichenden Kennzahlenfilter ein neues Fenster öffnen. Dieses Fenster besteht aus zwei Spalten. Die linke Spalte zeigt alle vorhandenen vergleichenden Kennzahlenfilter an. In der rechten Spalte werden nur jene angezeigt, die sich bereits in der Analysesituation befinden. Jede dieser beiden Spalten verfügt zusätzlich noch über einen Button mit einem Pfeil und einem Suchfeld. Der Button mit dem Pfeil ermöglicht es, alle Filter einer Spalte in die jeweils andere Spalte zu bringen (Hinzufügen bzw. Entfernen von Filtern). Das Suchfeld ermöglicht eine eingeschränkte Darstellung auf einen Suchbegriff. Nach dem Bestätigen des Fensters werden alle vorgenommenen Änderungen der Analysesituation übernommen. Eine beispielhafte Darstellung des Refocus-Fensters ist in Abbildung 23 ersichtlich.

5.4. Analysegraph-Instanz

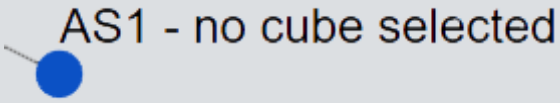
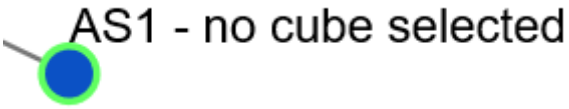
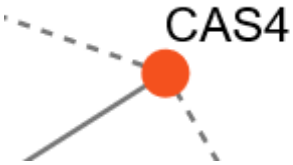
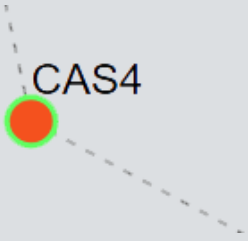
Dieser Abschnitt der Arbeit befasst sich mit der Benützung der Graph-Ansicht des Prototyps. In dieser werden alle Änderungen des Benutzers im genauen zeitlichen Verlauf mitgespeichert. Der Benutzer hat dann die Möglichkeit, den Weg hin zu einer Analysesituation nachzuvollziehen und zu einer bereits

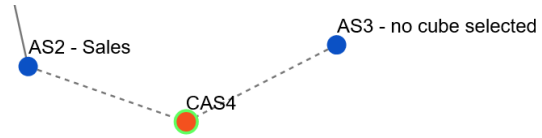
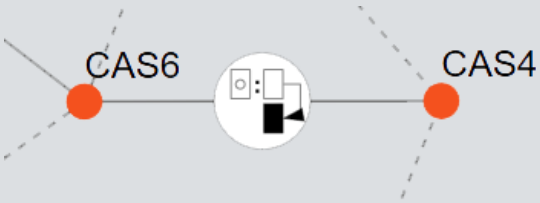
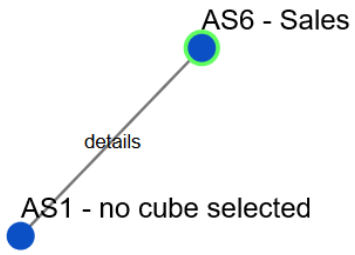
vorher erstellten Analysesituation zurückzuwechseln. Um eine bessere Übersicht über den Analyseprozess zu erlangen, ist es möglich den Graphen in zwei verschiedenen Detailstufen anzeigen zu lassen. Um in die Graph-Ansicht des Prototyps zu wechseln, kann ein Benutzer den Menüpunkt Graph in der Navigationsleiste des Prototyps auswählen.

5.4.1. Zentrale Elemente des Graphen

Der im Prototyp gezeigte Instanz-Graph besteht aus verschiedenen Elementen. Eine Übersicht und Beschreibung dieser Elemente sind in Tabelle 13 abgebildet. Diese Elemente werden sowohl im einfachen wie auch im detaillierten Graphen verwendet.

Tabelle 13: Zentrale Elemente des Graphen

Darstellung	Beschreibung
	Eine einfache Analysesituation wird mittels eines blauen Kreises dargestellt. Rechts über dem blauen Kreis werden der jeweilige Name und der ausgewählte Cube der Analysesituation angezeigt.
	Ist eine einfache Analysesituation (blauer Kreis) mit einem grünen Kreis umrandet, so heißt das, dass der Benutzer genau diese Analysesituation gerade in der Situationsansicht offen hat.
	Die Verbindung zwischen zwei einfachen Analysesituationen wird mit einer durchgezogenen Linie dargestellt. In der Mitte dieser Linie befindet sich das Symbol des jeweiligen Manipulationsoperators, der angewendet wurde, um von einer auf die andere Analysesituation zu kommen.
	Eine vergleichende Analysesituation wird mittels eines roten Kreises dargestellt. Rechts über dem roten Kreis wird der jeweilige Name der vergleichenden Analysesituation angezeigt.
	Ist eine vergleichende Analysesituation (roter Kreis) mit einem grünen Kreis umrandet, so heißt das, dass der Benutzer genau diese Analysesituation gerade in der Situationsansicht offen hat.

	Jede vergleichende Analysesituation ist mit den beiden einfachen Analysesituation, die im Vergleich verwendet werden, mit einer strichlierten Linie verbunden.
	Eine Verbindung zwischen zwei vergleichenden Analysesituationen wird mit einer durchgezogenen Linie dargestellt. In der Mitte dieser Linie befindet sich das Symbol des jeweiligen Manipulationsoperators, der angewendet wurde, um von einer auf die andere Analysesituation zu kommen.
	In der kombinierten Ansicht des Graphen wird nicht mehr jeder Navigationsschritt als eigene durchgezogene Linie angezeigt. Das Symbol des Operators ist durch die Beschriftung „details“ ersetzt. Durch einen Klick auf diese Beschriftung öffnet sich ein Fenster, das alle darin enthalten Operatoren auflistet.

5.4.2. Einfacher Graph

Der einfache oder detaillierte Graph stellt alle bisherigen Schritte zum Zusammenstellen der aktuellen Analysesituation dar. Dabei wurde jede Änderung, die der Benutzer in der Situationsansicht durchgeführt hat, in den Graphen als Einzelschritt übernommen. Der Benutzer hat so die Möglichkeit, zu jedem vorherigen Punkt in der Analyse zurückzuwechseln. Mit einem Klick auf eine einfache oder vergleichende Analysesituation kann ein neues Fenster geöffnet werden. Dieses Fenster bietet Informationen über die jeweilige Analysesituation mit der Möglichkeit für den Benutzer diese erneut zu laden. Handelt es sich um eine einfache Analysesituation, kann diese als Basis oder als zu vergleichende Analysesituation geladen werden, siehe Abbildung 31. Handelt es sich um eine vergleichende, kann diese nur als ganze in die Situationsansicht geladen werden. Nach dem Laden wird

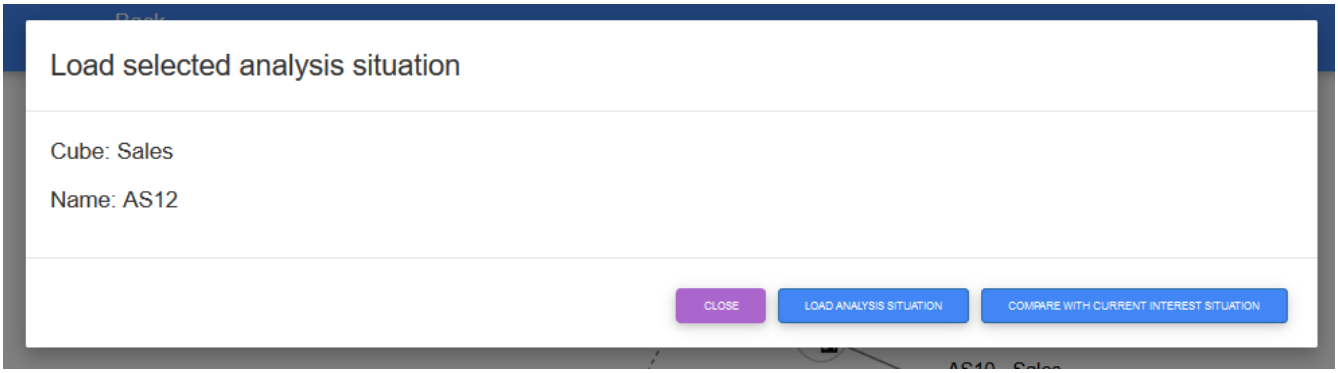


Abbildung 31: Dialog zum Laden einer Analysesituation

die Situationsansicht mit der ausgewählten Analysesituation aktualisiert. Ein Beispiel für eine einfache Analysegraph-Instanz ist in Abbildung 32 zu sehen.

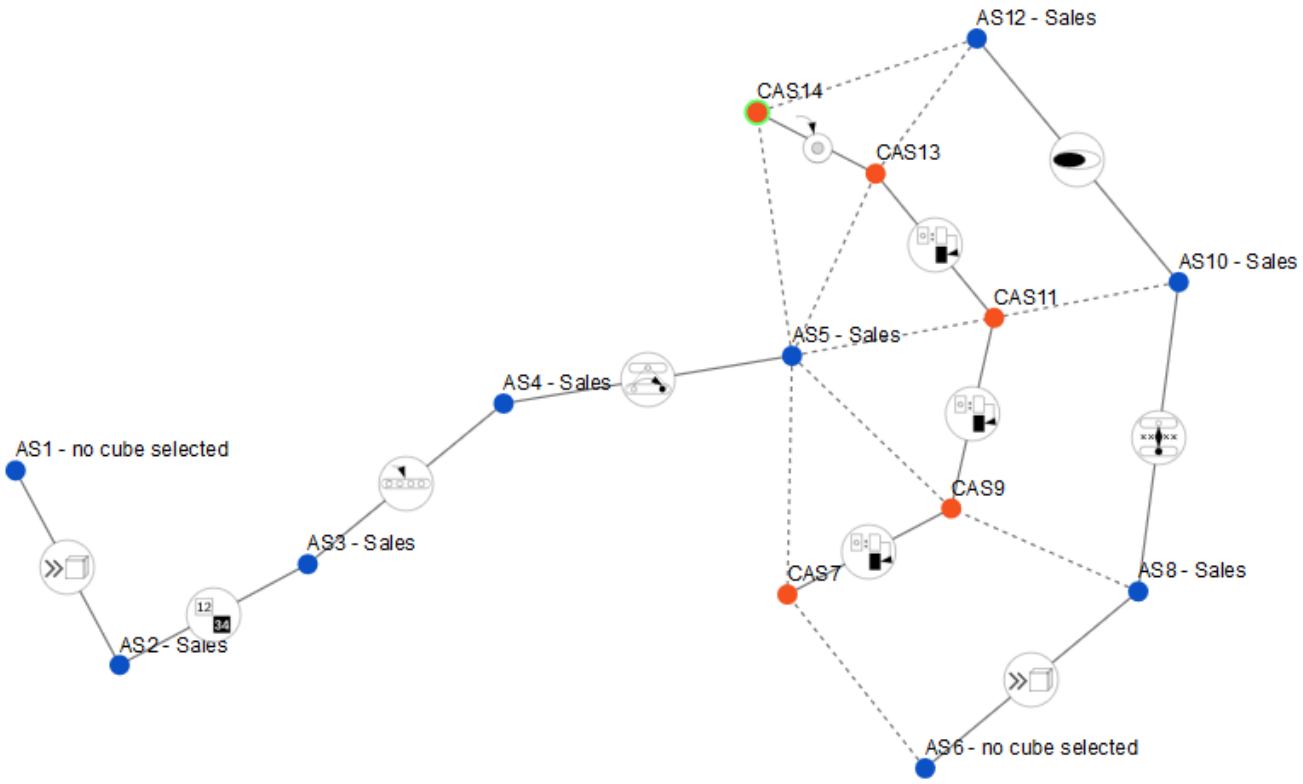


Abbildung 32: Beispiel für eine einfache Analysegraph-Instanz

5.4.3. Kombinierter Graph

Da bei größeren Analysen der einfache Graph aufgrund der vielen verwendeten Operatoren sehr schnell sehr unübersichtlich werden kann, besteht die Möglichkeit diesen in einer kompakteren Form anzuzeigen. Dabei werden verschiedene Navigationsschritte in einen Schritt zusammengefasst und so die Lesbarkeit des Graphen erhöht. Im kombinierten Graphen werden folgende Analysesituationen angezeigt: Die leere Startsituation bildet sowohl im einfachen wie auch im kombinierten Graphen die Startsituation. Wird einer Analysesituation ein neuer Name vergeben, so wird diese in den kombinierten Graphen aufgenommen. Abbildung 33 zeigt einen kombinierten Graphen, nach dem eine Analysesituation benannt wurde. Der Startanalysesituation wurde der Cube Sales hinzugefügt und zwei der Kennzahlen des Cubes. Diese Schritte können beim Klicken auf „details“ in dem sich öffnenden Fenster nachgeschlagen werden. Weiters wird jede Analysesituation, für die ein Benutzer ein Ergebnis angefragt hat, in den kombinierten Graphen eingefügt. Wird eine Analysesituation aus dem einfachen Graphen in die Situationsansicht geladen, wird diese Analysesituation auch im kombinierten Graphen als neuer Startpunkt eingetragen, sofern sie nicht schon enthalten ist. Erstellt der Benutzer eine neue vergleichende Analysesituation, so wird die aktuelle einfache Analysesituation sowie eine neue einfache

als Vergleichssituation in den kombinierten Graphen übernommen. Die Operatoren, die sich zwischen zwei Analysesituationen befinden, können durch einen Klick auf das „details“ auf der Verbindung der beiden angezeigt werden. In Abbildung 34 ist ein kombinierter Graph zu sehen nachdem eine neue vergleichende Analysesituation erstellt wurde.

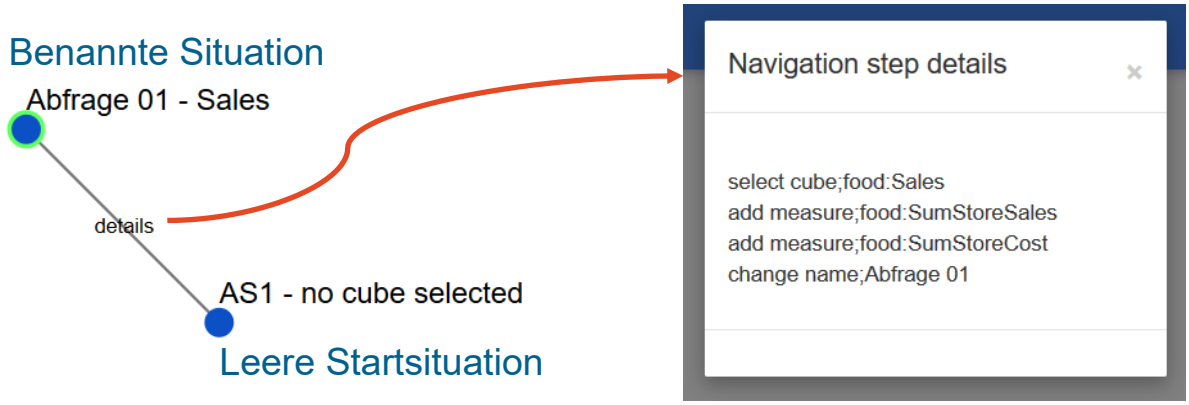


Abbildung 33: Kombinierter Graph beim Benennen einer Analysesituation

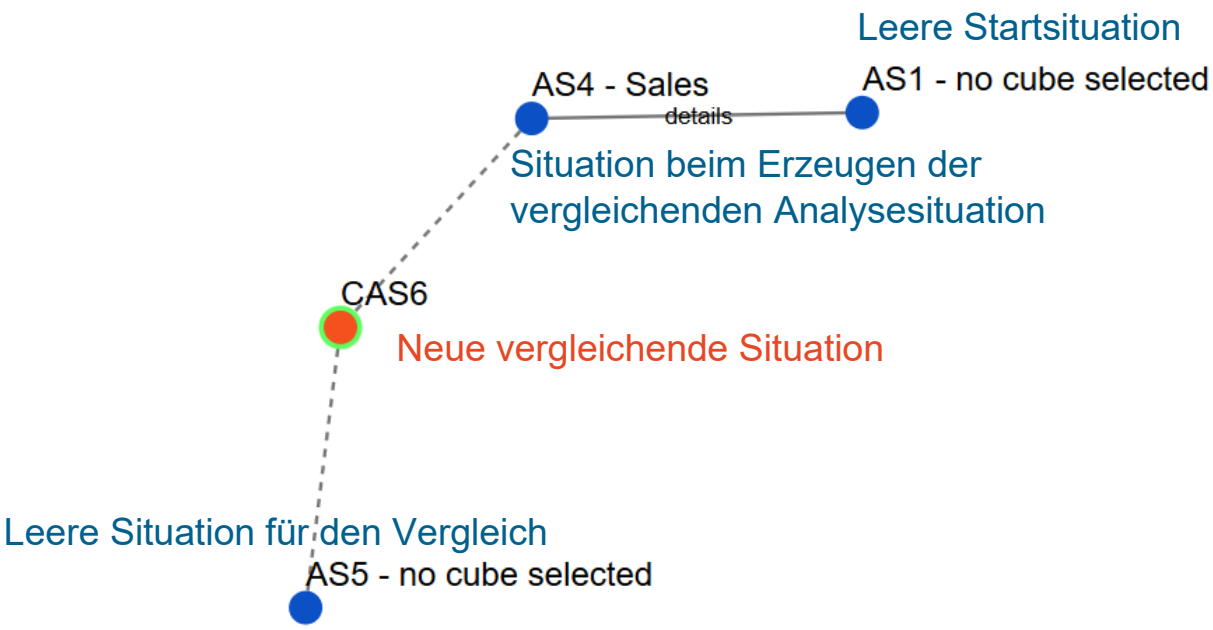


Abbildung 34: Kombinierter Graph beim Erstellen einer vergleichenden Analysesituation

Um den Unterschied zwischen diese beiden Anzeigeformaten deutlich zu machen, zeigt Abbildung 35 einen einfachen Graphen und Abbildung 36 den dazugehörigen kombinierten Graphen. Das Beispiel wurde folgendermaßen zusammengestellt. Zuerst wurde der Startsituation (AS1) der Cube Sales hinzugefügt (AS2). Im nächsten Schritt wurden zwei Kennzahlen aus dem Cube zu dieser einfachen Analysesituation hinzugefügt (AS3 und AS4). Der nächste Schritt war das Erstellen einer vergleichenden Analysesituation (CAS6) und damit auch das Erstellen einer zweiten einfachen Analysesituation (AS5). Dieser zweiten Analysesituation wurde ebenfalls der Sales-Cube hinzugefügt (AS7). Durch die Änderung an der einfachen Analysesituation wurde eine neue vergleichende Analysesituation (CAS8) mit der geänderten einfachen Analysesituation erstellt. Schlussendlich wurde

der vergleichenden Analysesituation noch ein vergleichender Kennzahlenfilter hinzugefügt (CAS9) und ein Ergebnis für diese Analysesituation angefordert. Im einfachen Graphen ist jeder der durchgeführten Schritte ersichtlich. Im kombinierten Graphen werden nur die oben beschriebenen Analysesituationen angezeigt und der Rest hinter der Detailansicht verborgen. Die angezeigten Analysesituationen sind die Startsituation AS1. Die Situationen AS4, AS5 und CS6 werden durch das Erstellen der vergleichenden Analysesituation in den kombinierten Graphen aufgenommen. Die Situationen AS7 und CAS9 werden durch das Anfordern des Ergebnisses für CAS9 hinzugefügt. Die Schritte zwischen den angezeigten Situationen können durch einen Klick auf „details“ der jeweiligen Verbindung angezeigt werden.

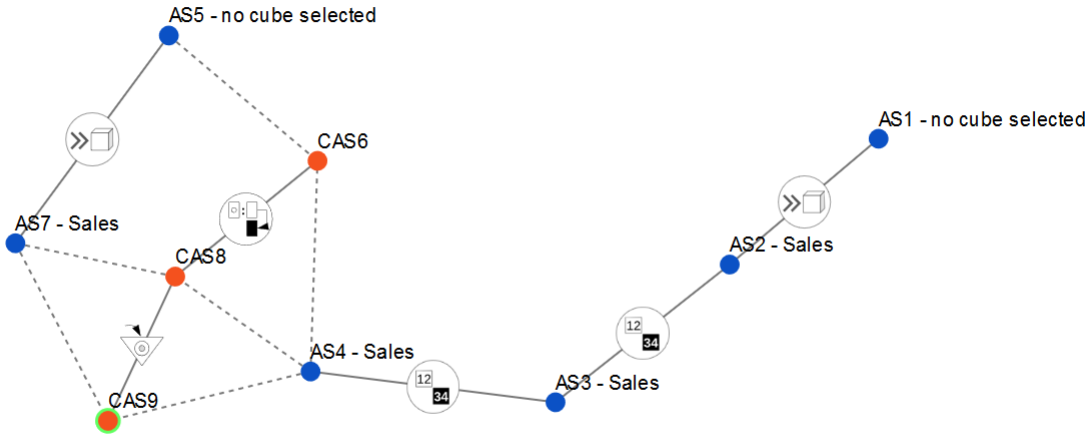


Abbildung 35: Beispiel für einen einfachen Graphen

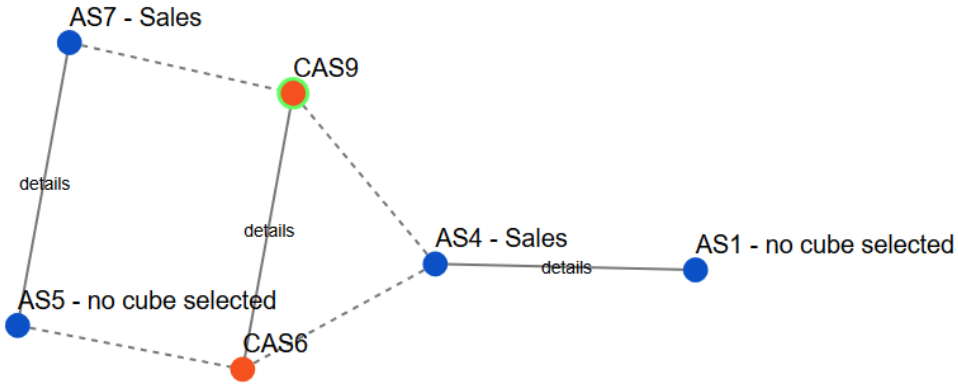
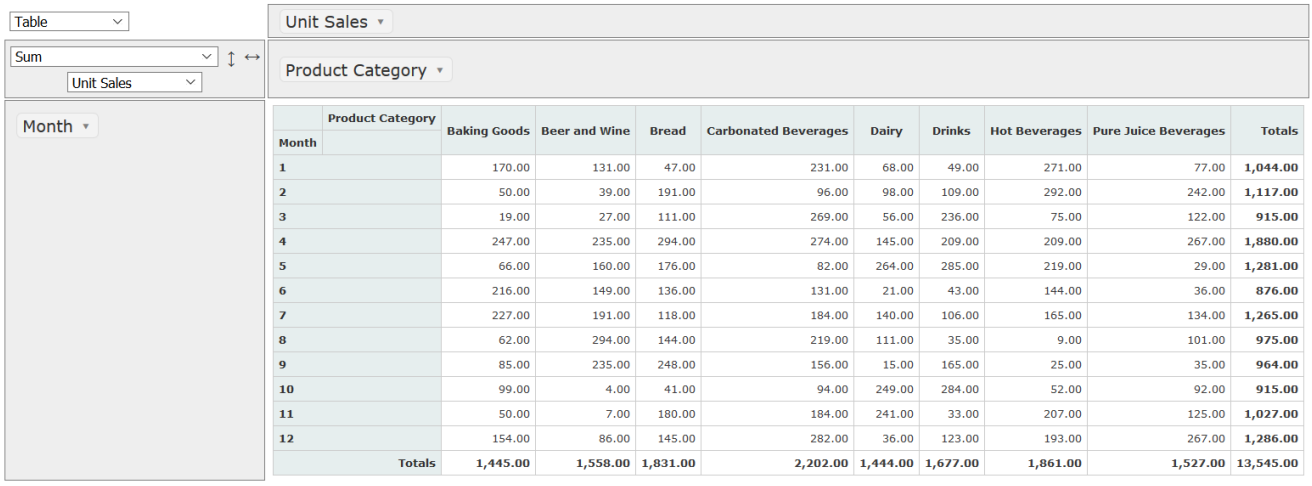


Abbildung 36: Beispiel für einen kombinierten Graphen auf Basis der detaillierten Darstellung in Abbildung 35

5.5. Ergebnissicht

Nachdem der Benutzer das Ergebnis einer Analysesituation durch Drücken des Buttons „Result“ in der Navigationsleiste angefordert hat, öffnet sich ein neuer Tab im Browser, der die Ergebnistabelle

beinhaltet. Dort sollte je nach zusammengestellter Analysesituation das jeweilige Ergebnis aus dem Backend angezeigt werden. Da diese Verarbeitung jedoch noch nicht im Backend implementiert ist, liefert jede Abfrage lediglich Beispieldaten einer Abfrage zurück.



Month \ Product Category	Baking Goods	Beer and Wine	Bread	Carbonated Beverages	Dairy	Drinks	Hot Beverages	Pure Juice Beverages	Totals
1	170.00	131.00	47.00	231.00	68.00	49.00	271.00	77.00	**1,044.00**
2	50.00	39.00	191.00	96.00	98.00	109.00	292.00	242.00	**1,117.00**
3	19.00	27.00	111.00	269.00	56.00	236.00	75.00	122.00	**915.00**
4	247.00	235.00	294.00	274.00	145.00	209.00	209.00	267.00	**1,880.00**
5	66.00	160.00	176.00	82.00	264.00	285.00	219.00	29.00	**1,281.00**
6	216.00	149.00	136.00	131.00	21.00	43.00	144.00	36.00	**876.00**
7	227.00	191.00	118.00	184.00	140.00	106.00	165.00	134.00	**1,265.00**
8	62.00	294.00	144.00	219.00	111.00	35.00	9.00	101.00	**975.00**
9	85.00	235.00	248.00	156.00	15.00	165.00	25.00	35.00	**964.00**
10	99.00	4.00	41.00	94.00	249.00	284.00	52.00	92.00	**915.00**
11	50.00	7.00	180.00	184.00	241.00	33.00	207.00	125.00	**1,027.00**
12	154.00	86.00	145.00	282.00	36.00	123.00	193.00	267.00	**1,286.00**
Totals	**1,445.00**	**1,558.00**	**1,831.00**	**2,202.00**	**1,444.00**	**1,677.00**	**1,861.00**	**1,527.00**	**13,545.00**

Abbildung 37: Beispielergebnis einer Abfrage

Der Benutzer hat die Möglichkeit, das in Abbildung 37 gezeigte Ergebnis weiter zu analysieren. Dazu kann er die Art der Tabelle beispielsweise auf eine Heatmap mit speziell hervorgehobenen Werten ändern. Er hat auch die Möglichkeit, die dem Ergebnis mitgelieferten Elemente selbstständig auf den Achsen zu verteilen. Dazu kann mittels Drag & Drop jedes der Elemente in den oberen bzw. den seitlichen Bereich gezogen werden, um es so der X bzw. Y-Achse zuzuordnen und dort anzuzeigen. Aktuell nicht angezeigte Elemente können in das oberste Feld gezogen werden, in dem sich die Unit Sales gerade befinden. Der Benutzer kann auch verschiedene Aggregatsfunktionen auf verschiedene mitgelieferte Elemente anwenden und so das Ergebnis weiter untersuchen.

6. Implementierungssicht

Dieser Abschnitt der Arbeit befasst sich mit den interessantesten Aspekten der Implementierung. Dabei wird auf die verwendeten Technologien, den Aufbau der Webanwendung, der Definition von Cubes und einige weitere Implementierungsdetails genauer eingegangen. Eine genaue Spezifikation der verwendeten REST-Schnittstellen befindet sich im Anhang: Dokumentation der REST-Schnittstellen des Backends.

6.1. Allgemeiner Aufbau

Der Prototyp ist eine Webanwendung. Sie besteht aus verschiedenen HTML-Seiten die zusätzlich mit Erweiterungen und JavaScript-Funktionen um weitere Funktionalitäten ergänzt wurden. Zum Erstellen des Codes wurde die Entwicklungsumgebung Webstorm verwendet. Das User-Interfaces wurde mit Hilfe des Bootstrap-Frameworks erstellt. Dieses Framework unterstützt beim Zusammenstellen des User-Interfaces und ermöglicht die Einbindung von vordefinierten Elementen, die nur noch mit der jeweiligen Logik hinterlegt werden müssen.

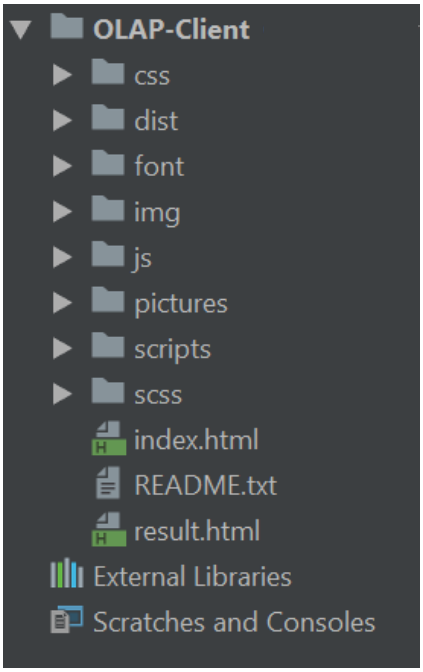


Abbildung 38: Gliederung der Webanwendung

Abbildung 38 zeigt die Gliederung der Webanwendung. Der Ordner css enthält verschiedene in Bootstrap enthaltene css-Dateien, sowie einige selbst definierte grafische Einstellungen des User-Interfaces. Der Ordner pictures beinhaltet sämtliche in der Webanwendung verwendeten Grafiken. Im Ordner scripts sind alle JavaScript-Dateien, die die eigentliche Logik und die Funktionalität des Prototyps beinhalten, enthalten. Die Seite index.html ist die Startseite der Webanwendung und gleichzeitig auch die zentrale Seite, in der die jeweilige Analysesituation zusammengestellt wird. Um das Ergebnis einer zusammengestellten Abfrage anzuzeigen, wird die Seite result.html verwendet. Als

Ausgangspunkt der Webanwendung wird ein Bootstrapprojekt (MDBootstrap, 2018) verwendet. Dieses wurde dann schrittweise bis zur aktuellen Version verbessert und erweitert. Zusätzlich zum Standardpaket von Bootstrap wird noch ein Plugin für eine Dual-List-Box (István Ujj-Mészáros, 2014) und ein Plugin für die live-Suche in den Menüs verwendet (SnapAppointments, 2015). Für die Visualisierung der Analysegraph-Instanz wurde auf das D3 Framework zurückgegriffen (Mike Bostock, 2019). Die Tabelle in der das Ergebnis angezeigt wird, wurde mit Hilfe des pivotTable.js erstellt (Nicolas Kruchten, 2012).

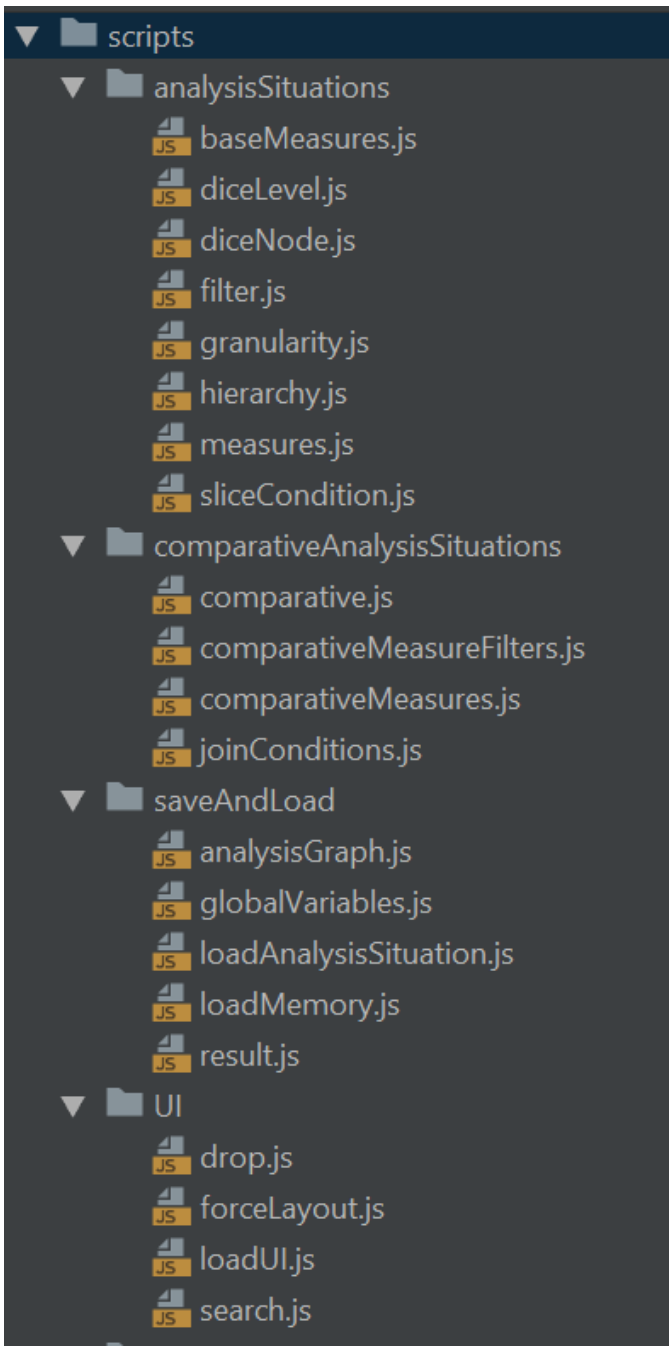


Abbildung 39: Gliederung der JavaScript-Dateien

Die JavaScript-Dateien werden in vier große Unterbereiche eingeteilt. Einem Bereich für die Analysesituationen, einem Bereich für die vergleichenden Analysesituationen, einem Bereich für den Speicher und das Laden von Daten und einem Bereich für das User-Interface (UI).

Der UI Bereich setzt sich zusammen aus dem drop.js, das sämtliche Funktionalität im Zusammenhang mit der Drag & Drop Funktion des Prototyps beinhaltet. Im forceLayout.js befindet sich die Funktionalität des D3.js das dafür zuständig ist, die Analysegraph-Instanz im Prototyp darzustellen. Das loadUI.js beinhaltet alle Methoden die Daten aus dem Speicher des Prototyps in das UI zu laden, sowie weitere UI-Funktionen wie das Wechseln von der Graph-Ansicht zur Situationsansicht. Search.js enthält Methoden, die von den Suchfeldern im Bereich der Cube-Informationen genutzt werden.

Der Bereich saveAndLoad befasst sich mit der Zwischenspeicherung von Daten im Prototyp bzw. mit der Speicherung der Analysesituationen und deren Navigationsschritte im Backend. Im analysisGraph.js befindet sich die Logik zum Erstellen der Analysesituationen aus der Situationsansicht. Diese Analysesituationen werden nach Erstellung, genauso wie die Navigationsschritte, an das Backend gesendet. Weiters werden hier auch die Daten gesammelt, die benötigt werden, um die aktuelle Analysegraph-Instanz grafisch darzustellen. Die Datei globalVariables.js bildet den internen Speicher des Prototyps ab. Dieser enthält alle global benötigten Daten, die aus dem Backend zurückgeliefert wurden bzw. während der Benützung des Prototyps entstanden sind. Sämtliche Methoden, die benötigt werden, um eine zuvor erstellte Analysesituation wieder in die Situationsansicht zu laden werden in loadAnalysisSituation.js verwaltet. Der interne Speicher des Prototyps wird mit Hilfe der in loadMemory.js hinterlegten Methoden befüllt. Das Skript result.js ist für die Anzeige des Ergebnisses in der dafür vorgesehenen HTML-Seite verantwortlich.

Der Bereich analysisSituations beinhaltetet die Ausprogrammierung sämtlicher Operatoren, die benützt werden können, um eine Aktion mit einer einfachen Analysesituation durchzuführen. Dabei sind die einzelnen Operatoren nochmal unterteilt in die jeweiligen Bereiche, die sie an einer Analysesituation verändern.

Der letzte Bereich ist der comparativeAnalysisSituation Bereich. Dieser enthält sämtliche Funktionalitäten in Bezug auf die vergleichende Analysesituation sowie sämtliche Operatoren der vergleichenden Kennzahlen, deren Filtern und die Join-Conditions betreffen.

6.2. Cube-Definition

Die Aufgabe des Prototyps ist es, Abfragen auf Datenstrukturen durchführen zu können. Dabei bedarf es einer definierten Form, in der die jeweiligen Datenstrukturen zur Verfügung gestellt werden müssen. In 2.1. wird eine grundlegende Form einer multidimensionalen Datenstruktur beschrieben die von Neuböck (2018) adaptiert und als erweitertes dimensionales Faktenmodell publiziert wurde. Dieses dimensionale Faktenmodell ist die Grundlage der Datenstrukturen, die dem Prototyp aus dem Backend zur Verfügung gestellt und in weiterer Folge für die Analyse verfügbar gemacht werden.

Das Eingabeformat für einen Cube wird vom Prototyp im RDF-Format erwartet. Dabei wird ein erweitertes qb-Vokabular verwendet. Dieses Vokabular wird unter dem Prefix qbx (qb extended) geführt und beinhaltet eine angepasste Version des originalen qb-Formats (Etcheverry, 2015/2018).

Das Eingabeformat für den Prototyp besteht aus folgenden Bestandteilen, wobei jeder von ihnen ein Label und einen Kommentar hat. Das Label ist der Name der später im Prototyp angezeigt wird und der Kommentar die Information, wenn sich die Maus über das Element bewegt.

Tabelle 14: Cube-Definition

Cube	
rdf:type	qbx:BaseCube
rdfs:label	„Textuelle Darstellung des Cube“
rdfs:comment	„Beschreibung des Cube“
qbx:measure	Kennzahl des Cube
qbx:dimension	Dimension des Cube

```
food:Sales a qbx:BaseCube ;
  rdfs:label "Sales"@en ;
  rdfs:comment "Sales"@en ;
  qbx:measure food:UnitSales ;
  qbx:measure food:StoreCost ;
  qbx:dimension food:DimensionStore ;
  qbx:dimension food:DimensionStoreSizeinSQFT.
```

Code 1: Beispiel für einen Cube

Ein Cube ist die zentrale Speichereinheit eines Data Warehouse und als Typ qbx:BaseCube definiert. Jeder Cube kann keine, eine oder mehrere Kennzahlen enthalten, die mit qbx:measure an das BaseCube Element angeheftet sind. Die im Cube enthaltenen Dimensionen werden mit dem Prädikat qbx:dimension im BaseCube angegeben.

Die Basiskennzahlen eines Cubes sind als Typ qbx:BaseMeasure gekennzeichnet. Sie können eine Einheit, in der sie gemessen werden und einen Datentyp enthalten.

Tabelle 15: Basiskennzahlen-Definition

Basiskennzahl	
rdf:type	qbx:BaseMeasure
rdfs:label	„Textuelle Darstellung der Basiskennzahl“
rdfs:comment	„Beschreibung der Basiskennzahl“
qbx:unitOfMeasurement	Einheit der Kennzahl (z.B. qbx:Euro)
qbx:dataType	Datentyp (z.B. xsd:number)

```
food:UnitSales a qbx:BaseMeasure ;
  rdfs:label "UnitSales"@en ;
  rdfs:comment "UnitSales"@en ;
  qbx:dataType xsd:integer .
```

Code 2: Beispiel für eine Base Measure

Aggregierte Kennzahlen beziehen sich auf eine Basiskennzahl oder eine andere aggregierte Kennzahl. Die Aggregatsfunktion gibt an wie die Basiskennzahl/aggregierte Kennzahl, von der sie abgeleitet ist, aggregiert wird.

Tabelle 16: Aggregierte Kennzahlen-Definition

Aggregierte Kennzahl	
rdf:type	qbx:AggregateMeasure
rdfs:label	„Textuelle Darstellung der aggregierten Kennzahl“
rdfs:comment	„Beschreibung der aggregierten Kennzahl“
qbx:aggreagtionFunction	Aggregationsfunktion (z.b. qbx:Sum, qbx:Avg usw.)
qbx:derivedFrom	Basiskennzahl oder aggregierte Kennzahl von dem die aggregierte Kennzahl abgeleitet ist

```
food:SumUnitSales a qbx:AggregateMeasure ;
  rdfs:label "Sum of UnitSales"@en ;
  rdfs:comment "Sum of UnitSales"@en ;
  qbx:aggregationFunction qbx:Sum ;
  qbx:derivedFrom food:UnitSales .
```

Code 3: Beispiel für eine aggregierte Kennzahl

Um einen Basiskennzahlenfilter zu definieren, muss dieser vom Typ qbx:BaseMeasurePredicate sein. Die Verbindung qbx:over definiert über welche Basiskennzahl der Filter angewendet werden soll.

Tabelle 17: Basiskennzahlenfilter-Definition

Prädikat für Basiskennzahlen	
rdf:type	qbx:BaseMeasurePredicate
rdfs:label	„Textuelle Darstellung des Prädikates der Basiskennzahl“
rdfs:comment	„Beschreibung des Prädikates der Basiskennzahl“
qbx:over	Basiskennzahl

```
food:HighCostsPerUnitPredicate a qbx:BaseMeasurePredicate ;
  rdfs:label "High Costs per Unit"@en ;
  rdfs:comment "High Costs per Unit"@en ;
  qbx:over food:UnitSales .
```

Code 4: Beispiel für einen Basiskennzahlenfilter

Um einen Filter für eine aggregierte Kennzahl zu definieren, muss ein Element vom Typ qbx:AggregateMeasurePredicate erzeugt werden. Dieser wird mit qbx:over einer aggregierten Kennzahl zugewiesen.

Tabelle 18: Filter-Definition

Prädikat für aggregierte Kennzahl	
rdf:type	qbx:AggregateMeasurePredicate
rdfs:label	„Textuelle Darstellung des Prädikates der aggregierten Kennzahl“
rdfs:comment	„Beschreibung des Prädikates der aggregierten Kennzahl“
qbx:over	aggregierte Kennzahl

```
food:HighUnitSalesPredicate a qbx:AggregateMeasurePredicate ;
  rdfs:label "High Unit Sales"@en ;
  rdfs:comment "High Unit Sales"@en ;
  qbx:over food:SumUnitSales .
```

Code 5: Beispiel für einen Filter

Eine vergleichende Kennzahl wird mit dem Typ qbx:ComparativeMeasure gekennzeichnet. Sie bezieht sich jeweils auf eine Kennzahl aus den beiden einfachen Analysesituationen, die in einer vergleichenden Analysesituation miteinander verglichen werden.

Tabelle 19: Vergleichende Kennzahlen-Definition

Vergleichende Kennzahl	
rdf:type	qbx:ComparativeMeasure
rdfs:label	„Textuelle Darstellung der vergleichenden Kennzahl“
rdfs:comment	„Beschreibung der vergleichenden Kennzahl“
qbx:setOfInterestMeasure	Kennzahl
qbx:setOfComparisonMeasure	Kennzahl

```
food:ChangeOfSumUnitSales a qbx:ComparativeMeasure ;
  rdfs:label "Change of costs"@en ;
  rdfs:comment "Description"@en ;
  qbx:setOfInterestMeasure food:SumUnitSales;
  qbx:setOfComparisonMeasure food:SumUnitSales.
```

Code 6: Beispiel für eine vergleichende Kennzahl

Ein Filter für eine vergleichende Kennzahl wird mit dem Typ qbx:ComparativeMeasurePredicate gekennzeichnet. Dabei definiert qbx:over auf welche der definierten vergleichenden Kennzahlen sich der Filter bezieht.

Tabelle 20: Vergleichende Kennzahlenfilter-Definition

Prädikat für vergleichende Kennzahl	
rdf:type	qbx:ComparativeMeasurePredicate
rdfs:label	„Textuelle Darstellung des Prädikates der vergleichenden Kennzahl“
rdfs:comment	„Beschreibung des Prädikates der vergleichenden Kennzahl“
qbx:over	vergleichende Kennzahl

```
food:SignificantChange a qbx:ComparativeMeasurePredicate ;
  rdfs:label "Change of unit sales"@en ;
  rdfs:comment "Change over 10%"@en ;
  qbx:over food: SumUnitSales.
```

Code 7: Beispiel für einen vergleichenden Kennzahlenfilter

Um eine Join-Condition zu definieren, muss ein Element vom Typ qbx:JoinConditionPredicate erstellt werden. Diese Join-Condition verweist auf die Granularitätslevels der beiden zu vergleichenden einfachen Analysesituationen. Es kann mehr als ein Level für qbx:overLevelInSetOfInterest und qbx:overLevelInSetOfComparison definiert werden. Sobald in jeder der beiden Analysesituationen ein Level als Granularität angegeben ist, kann die Join-Condition in der vergleichenden Analysesituation verwendet werden.

Tabelle 21: Join-Condition-Definition

Join Condition	
rdf:type	qbx:JoinConditionPredicate
rdfs:label	„Textuelle Darstellung der Join Condition“
rdfs:comment	„Beschreibung der Join Condition“
qbx:overLevelInSetOfInterest	Level
qbx:overLevelInSetOfComparison	Level

```
food:Vorjahresvergleich a qbx:JoinConditionPredicate ;
  rdfs:label "year-on-year"@en ;
  rdfs:comment "Descritpion"@en ;
  qbx:overLevelInSetOfInterest food:TimeDimensionYearLevel ;
  qbx:overLevelInSetOfComparison  food:TimeDimensionYearLevel ;
```

Code 8: Beispiel für eine Join-Condition

Die Slice-Conditions teilen sich in zwei Typen auf. Der Typ qbx:LevelPredicate wird mit qbx:over an ein Level gebunden. Der Typ qbx:ConjuncttiveLevelPredicate schränkt eine andere Slice-Condition weiter ein. Dazu wird mit qbx:conjunct die einzuschränkenden Slice-Condition angegeben.

Tabelle 22: Slice-Condition-Definition

Slice-Condition	
rdf:type	qbx:LevelPredicate
rdfs:label	„Textuelle Darstellung der Slice-Condition“
rdfs:comment	„Beschreibung der Slice-Condition“
qbx:over	Level

```
food:BigCity a qbx:LevelPredicate ;
  rdfs:label "Big city "@en ;
  rdfs:comment "Big city"@en ;
  qbx:over food:LevelStoreCity.
```

Code 9: Beispiel für eine Slice-Condition

Tabelle 23: Zusammengesetzte Slice-Condition-Definition

Slice-Condition	
rdf:type	qbx:ConjunctiveLevelPredicate
rdfs:label	„Textuelle Darstellung der Slice Condition“
rdfs:comment	„Beschreibung der Slice Condition“
qbx:conjunct	LevelPredicate / ConjunctiveLevelPredicate

```
food:BigCityWithDistributionCenter a qbx:ConjuctiveLevelPredicate ;
  rdfs:label "Big city with distribution center"@en ;
  rdfs:comment "Big city with distribution center"@en ;
  qbx:conjunct food:BigCity ;
  qbx:conjunt food:DistributionCenter .
```

Code 10: Beispiel für zusammengesetzte Join-Condition

Um eine Dimension für einen Cube zu definieren, kann ein Element vom Typen qbx:Dimension erstellt werden. Einer Dimension können mit qbx:hasHierarchy mehrere verschiedene Hierarchien zugeteilt werden.

Tabelle 24: Dimensions-Definition

Dimension	
rdf:type	qbx:Dimension
rdfs:label	„Textuelle Darstellung der Dimension“
rdfs:comment	„Beschreibung der Slice Condition“
qbx:hasHierarchy	Hierarchie

```
food:DimensionStore a qbx:Dimension ;
  rdfs:label "Store"@en ;
  rdfs:comment "Store"@en;
  qbx:hasHierarchy food:DimensionStoreHierarchyMain .
```

Code 11: Beispiel einer Dimension

Um eine neue Hierarchie zu definieren, muss eine Element vom Typ qbx:Hierarchy erstellt werden. Einer Hierarchie werden später einzelne Hierarchieschritte hinzugefügt, die die Levels der Hierarchie enthalten.

Tabelle 25: Hierarchie-Definition

Hierarchie	
rdf:type	qbx:Hierarchy
rdfs:label	„Textuelle Darstellung der Hierarchie“
rdfs:comment	„Beschreibung der Hierarchie“

```
food:DimensionStoreHierarchyMain a qbx:Hierarchy ;
  rdfs:label "Store"@en ;
  rdfs:comment "Store Hierarchy“@en .
```

Code 12: Beispiel einer Hierarchie

Die Levels der Hierarchien werden mit dem Typ qbx:Level versehen. Sie werden dann über die Hierarchieschritte den einzelnen Dimensionen zugeordnet.

Tabelle 26: Level-Definition

Level	
rdf:type	qbx:Level
rdfs:label	„Textuelle Darstellung des Levels“
rdfs:comment	„Beschreibung des Levels“

```
food:LevelStoreState a qbx:Level ;
  rdfs:label "StoreState"@en ;
  rdfs:comment "StoreState"@en .
```

Code 13: Beispiel für ein Level

Ein Hierarchieschritt wird mit dem Typ qbx:HierarchyStep gekennzeichnet. Ein Hierarchieschritt hat immer ein Eltern- und ein Kind-Level. Zusätzlich wird eine Kardinalität zwischen diesen beiden Leveln definiert. Jeder Hierarchieschritt wird mit qbx:inHierarchy einer Hierarchie und somit einer Dimension zugeordnet.

Tabelle 27: Hierarchieschritt-Definition

Hierarchieschritt	
rdf:type	qbx:HierarchyStep
qbx:inHierarchy	Hierarchie
qbx:pcCardinality	Kardinalität der Level (z.B. qbx:OneToMany)
qbx:childLevel	Level
qbx:parentLevel	Level

```
food:DimensionHierarchyLevelStoreStateToLevelStoreCountry a qbx:HierarchyStep ;
  qbx:inHierarchy food:DimensionStoreHierarchyMain ;
  qbx:pcCardinality qbx:OneToMany ;
  qbx:childLevel food:LevelStoreState ;
  qbx:parentLevel food:LevelStoreCountry .
```

Code 14: Beispiel eines Hierarchieschrittes

Jedes Level kann in der Analysesituation als Dice-Level verwendet werden. Dadurch ist es notwendig die entsprechenden Dice-Nodes zusätzlich zu den Levels zu definieren. Jedes Level muss mindestes ein Level-Member definiert haben. Durch qbx:inLevel wird der Level-Member einem Level zugeordnet.

Tabelle 28: Level-Member-Definition

Level-Member	
rdf:type	qbx:LevelMember
rdfs:label	„Textuelle Darstellung des Levels Members“
rdfs:comment	„Beschreibung des Levels Members“
qbx:inLevel	Level

```
food:Coffee a qbx:LevelMember ;
  rdfs:label "Coffee"@en ;
  rdfs:comment "Coffee"@en ;
  qbx:inLevel food:LevelProductSubcategory .
```

Code 15: Beispiel eines Level-Members

6.3. Speicher des Prototyps

Der Prototyp hat einen internen Speicher in Form von JSON-Elementen, die sämtliche Informationen speichert die länger benötigt werden bzw. von Nutzen sind. Diese Elemente werden in der Datei globalVariables.js abgelegt und nach jedem Neustart der Webanwendung neu initialisiert bzw. beim Schließen der Webanwendung gelöscht.

Die Variable `httpURL` definiert die URL des Webservers, auf dem das Backend gehostet wird. Wird diese verändert, so muss sie auch dort im Client geändert werden. Dies kann hier zentral für alle Schnittstellen des Prototyps durchgeführt werden. Weiters beinhaltet der Speicher sämtliche Information über die Cubes der Analysesituationen (Dimensionen, Level, Kennzahlen, Filter, usw.). Zusätzlich zu den Cube-Informationen werden auch die aktuellen IDs der Analysesituationen sowohl für das Backend als auch zu Erstellung der Analysegraph-Instanz gespeichert.

Dieser Speicher wird nun während der Benützung des Prototyps nach und nach aus den im Anhang gelisteten Schnittstellen befüllt. Beim Öffnen der Webanwendung werden alle verfügbaren Cubes vom Backend geholt (Schnittstelle A), im Speicher abgelegt und dem Benutzer angezeigt. Durch einen Klick auf einen dieser Cubes werden die einzelnen Elemente dieses Cube in den Speicher geladen und so für alle späteren Verwendungen verfügbar gemacht, ohne jedes Mal einen extra REST-Aufruf durchführen zu müssen. Geladen werden Basiskennzahlen (Schnittstelle B), Basiskennzahlenfilter (Schnittstelle D), Kennzahlen (Schnittstelle C), Filter (Schnittstelle E), Dimensionen (Schnittstelle G), Hierarchien (Schnittstelle Q), deren Levels (Schnittstelle S) und Slice-Conditions (Schnittstelle T). Diese werden nach dem Laden des Cube nicht mehr aktualisiert. Eventuelle Änderungen im Backend werden erst durch ein erneutes Klicken auf den Cube in den Speicher des Prototyps übernommen. Dadurch, dass die Anzahl der Level-Members in einer Datenstruktur eine sehr große Zahl annehmen kann, werden diese nicht schon beim Laden des Cube mitgeladen. Sie werden nur gebraucht sobald ein Level als Dice-Node ausgewählt wird. Sobald ein Benutzer ein Level in den Bereich der Dice-Node bringt, wird im Speicher überprüft, ob die Members bereits vorhanden sind. Falls dies nicht der Fall sein sollte, werden die Members für dieses Level vom Backend nachgeladen und im Speicher hinterlegt (Schnittstelle U). Die vergleichenden Kennzahlen (Schnittstelle N), deren Filter (Schnittstelle O) und die Join-Conditions (Schnittstelle P) werden je nach Änderung der Cubes bzw. der Granularitäten im Speicher aktualisiert.

6.4. Analysesituationen

Durch die Verwendung des Prototyps kann ein Benutzer eine Analysesituation zusammenstellen. Für diese Analysesituation kann der Anwender jederzeit ein Ergebnis anfordern. Dazu wird die Analysesituation in das in Kapitel 3. vorgestellte Format gebracht und an das Backend gesendet. Dort wird aus dieser Analysesituation dann eine Abfrage erstellt und das Ergebnis dieser Abfrage als CSV-Datei zur Verfügung gestellt. Jede Änderung des Benutzers in der Webanwendung wird intern als Navigationsschritt verwaltet. Genauer gesagt wird mit jeder Änderung eine neue Analysesituation (Schnittstelle J) und der dazugehörige Navigationsschritt (Schnittstelle K) erstellt und an das Backend gesendet. Dort wird der gesamte Analysegraph verwaltet und mitgespeichert. Die Analysesituationen können wieder aus dem Backend geholt werden (Schnittstelle M) bzw. kann ein Ergebnis für alle Analysesituationen angefordert werden (Schnittstelle H).

```
createNodeInGraph(interest,name,parameter);
```

Code 16: Aufruf zum Erstellen einer Analysesituation

Um eine neue Analysesituation inklusive dem dazugehörigen Navigationsschritt zu erstellen, muss lediglich der in Code 16 gezeigte Aufruf gemacht werden. Der erste Parameter ist ein Boolean der angibt, in welcher Analysesituation die Änderung gemacht werden soll. True bedeutet, dass die Änderung an der Interest-Analysesituation durchgeführt werden soll (einfache Analysesituation / linke Analysesituation in der vergleichenden Analysesituation). False zeigt an, dass die Änderung an der Comparison-Analysesituation (die rechte in einer vergleichenden Analysesituation) ausgeführt werden

```
    <urn:uuid:7d7e02e0-26a9-401f-acae-736d15d43ffa>
            a       <http://www.dke.jku.at.ac/ag#ElementaryAnalysisSituation> ;
            <http://www.w3.org/2000/01/rdf-schema#label>
                    "AS12"@en ;
            <http://dke.jku.at/inga/cubes#hasAggregateMeasure>
                    <food:SumStoreCost> , <food:SumStoreSales> ;
            <http://dke.jku.at/inga/cubes#hasCube>
                    <food:Sales> ;
            <http://dke.jku.at/inga/cubes#hasDimensionSpec>
                    [ <http://dke.jku.at/inga/cubes#diceLevel>
                              <food:LevelProductCategory> ;
                      <http://dke.jku.at/inga/cubes#diceNode>
                              <food:BeerAndWine> ;
                      <http://dke.jku.at/inga/cubes#dimension>
                              "food:DimensionProduct" ;
                      <http://dke.jku.at/inga/cubes#hierarchy>
                              "food:DimensionProductHierarchyMain"
                    ] ;
            <http://dke.jku.at/inga/cubes#time>
                    "8.10.2019, 18:09:10" .
```

Code 17: Beispiel einer Analysesituation im Backend

soll. Der Name ist ein String und gibt den Namen des Operators wieder. Dieser Name kann im anlysisGraph.js weiter angeführt werden und so den Operator genauer definieren. Der Parameter gibt den Übergabewert des Operators an und kann auch im analysisGraph.js weiterverarbeitet werden, falls dieser mehr als einen Wert enthalten sollte.

Durch diesen Aufruf wird automatisch eine neue Analysesituation und der dazugehörige Navigationsschritt erstellt. Diese Analysesituation wird sowohl in der Analysegraph-Instanz des Prototyps als auch im Backend zum aktuellen Analysegraphen hinzugefügt. Um eine Analysesituation in der in Kapitel 3. vorgestellten Form erstellen zu können, werden die IDs der JavaScript-Objekte genutzt. Jedes Element, dass in der Situationsansicht in der gezeigten Analysesituation ist, beginnt mit einem speziellen ID-Präfix. Dabei gibt der Präfix „form“ Elemente aus der Interest-Analysesituation, „comparisonForm“ Elemente aus der Comparison-Analysesituation und „comparativeForm“ Elemente aus der vergleichenden Analysesituation an. Der Präfix wird um das jeweilige Element erweitert. So beginnt die ID einer Kennzahl in der einfachen Analysesituation mit „formMeasure“ bzw. eine Join-Condition mit „comparativeFormJoinCondition“. In diesem Format könnten weitere Elemente in die Analysesituation hinzugefügt werden. Diese Elemente müssen dann im File analysisGraph.js in das Switch eingebaut und ausspezifiziert werden. Zusätzlich muss das neue Element beim Laden einer Analysesituation im File loadAnalysisSituation.js berücksichtigt werden. Ein Beispiel einer Analysesituation, wie sie im Backend in der in Kapitel 3. gezeigten Form gespeichert ist, ist in Code 17 zu sehen. Diese enthält alle Elemente, die ein Benutzer beim Zusammenstellen im Prototyp hinzugefügt hat. Die urn:uuid ist eine eindeutige Kennung die vom Backend über eine REST-Schnittstelle bezogen wird (Schnittstelle I) um so eine Doppelvergabe auszuschließen. Sie wird verwendet, um die Analysesituation später wieder zu finden und um die beiden Analysesituationen eines Navigationsschritts festzulegen.

6.5. Veränderungsoperatoren

Der Unterschied zwischen zwei aufeinanderfolgenden Analysesituationen wird mit Hilfe eines Navigationsschritts dargestellt. Dieser beinhaltet einen Operator und die dazugehörigen Parameter. Die Erstellung eines Navigationsschrittes passiert automatisch mit dem in Code 16 gezeigten Aufruf. Dabei wird zuerst die neue Analysesituation erstellt. Die IDs der alten und der neuen Analysesituation werden im Speicher der Webanwendung mitgespeichert. Mit diesen Informationen und dem Namen des Operators und seiner Parameter kann nun der Navigationsschritt erstellt werden. Durch den Namen kann der Operator in der Datei analysisGraph.js noch genauer spezifiziert werden. Der Navigationsschritt wird nach Fertigstellung an das Backend gesendet und dort gespeichert. Weiters wird der Navigationsschritt auch in die aktuelle Analysegraph-Instanz des Prototyps aufgenommen und so für das D3.js-Framework zur Visualisierung zur Verfügung gestellt. Code 18 zeigt ein Beispiel eines Navigationsschritts wie er im Speicher des Backends liegt.

```
<urn:uuid:bbfa6b42-3bc0-42b0-8951-9a50ae7b0bbb>
            a       <http://www.dke.jku.at.ac/ag#NavigationStep> ;
            <http://www.w3.org/2000/01/rdf-schema#label>
                    "Name"@en ;
            <http://www.dke.jku.at.ac/ag#hasOperation>
                    [ a       <http://www.dke.jku.at.ac/ag#AddMeasure> ;
                      <http://www.w3.org/2000/01/rdf-schema#label>
                              "add measure"@en ;
                      <http://www.dke.jku.at.ac/ag#hasParameter>
                              <food:SumStoreCost>
                    ] ;
            <http://www.dke.jku.at.ac/ag#source>
                    <urn:uuid:415cce1b-72f1-470e-a966-6a54a35c6750> ;
            <http://www.dke.jku.at.ac/ag#target>
                    <urn:uuid:eff982a4-c1bf-40fd-924a-f82c286778d3> .
```

Code 18: Beispiel eines Navigationsschritts im Backend

6.6. Analysegraph-Instanz

Die Analysegraph-Instanz, die im Prototyp angefertigt wird, kann in der Graph-Ansicht des Prototyps dargestellt werden. Dazu wird das D3.js-Framework verwendet. Dieses arbeitet aufgrund von zwei JSON-Dateien, die im Speicher des Prototyps abgelegt sind.

```
var graph = {
    "nodes": [
    ],
    "links": [
    ]
};
```

Code 19: JSON-String für die Analysegraph-Instanz

Code 19 zeigt den Aufbau des JSONs der als Eingabe verwendet wird. Dieser enthält alle Analysesituationen in den `nodes` und alle Navigationsschritte in den `links`. Es wird ein solcher JSON für den gesamten Graphen und ein JSON für den kombinierten Graphen im Speicher des Prototyps hinterlegt. Dabei kann der Aufbau der Elemente im `nodes` Array selbstständig gewählt werden. Die Elemente im `links` Array müssen ein `source` und ein `traget` Attribut besitzen. Diese verweisen auf die jeweiligen Analysesituationen in den `nodes`. Die `source` stellt den Startpunkt, das `target` den Endpunkt der Verbindung dar. Diese verweisen jeweils auf den Index der in `nodes` enthaltenen Elemente. Die Elemente in den beiden Arrays können beliebig viele weitere Attribute enthalten. Diese können später genutzt werden, um die angezeigten Elemente des Graphen zu personalisieren. Im Prototyp gibt beispielsweise ein Attribut `type` an, ob es sich um eine einfache oder um eine vergleichende Analysesituation handelt. Demensprechend wird dann ein roter oder ein blauer Kreis im Graphen gezeichnet.

In der Datei forceLayout.js können sämtliche Einstellungen zur grafisch dargestellten Analysegraph-Instanz gemacht werden. Dazu zählt die weitere Personalisierung der vorhandenen Elemente bzw. das Hinzufügen neuer Elemente. Weiters können auch allgemeine Einstellungen zum verwendeten D3 Force-Layout gemacht werden. Hierzu zählen beispielweise die Länge der Verbindungen oder die Kräfte mit denen die einzelnen Elemente angezogen oder abgestoßen werden.

7. Evaluierung

Im Sinne der Design Science sollten angefertigte Artefakte stets auf ihre praktische Benutzbarkeit zum Lösen des angedachten Problems getestet werden (Hevner, March, Park, & Ram, 2008). Dazu wurden im Rahmen dieser Masterarbeit erste Tests mit Probanden durchgeführt. Diese Tests wurden in Form von Usability Tests abgehalten. Das Ziel dieser Tests lag nicht darin das Wissen der Probanden zum Thema zu untersuchen, sondern das Hauptaugenmerk lag auf dem Interface zum Ausführen von OLAP-Abfragen und den möglichen Stärken und Schwächen im Umgang mit dem Prototyp (Lazar, 2017).

Die Probanden waren elf Bachelor-Studierende die den Kurs Data & Knowledge Engineering im Sommersemester 2018 belegten. Diese waren grundsätzlich mit dem Fachgebiet vertraut, jedoch bestand wenig bis keine Erfahrung im Bereich Business Intelligence. Um dies auszugleichen, erhielten die Probanden eine kurze Einführung zum Thema Data Warehousing und OLAP. Die grundlegenden Operationen wurden erklärt und anhand von Beispielen näher erläutert. Danach wurden Beispiele für die Verwendung des Prototyps und die Verwendung des OLAP-Client Saiku, der als Vergleichsobjekt dient, vorgeführt und die wesentlichen Vorgänge zu einfachen Aufgaben erläutert. Danach sollten die Probanden selbstständig spontan gestellte Aufgaben lösen. Ein Teil der Versuchspersonen hat dabei zuerst mit Saiku begonnen und danach mit dem Prototyp gearbeitet und der andere Teil zuerst mit dem Prototyp und dann mit Saiku. Die Aufgaben haben sich auf das Arbeiten mit der einfachen Analysesituation beschränkt. Die Funktionalität der vergleichenden Analysesituation war nicht Thema der Evaluierung. Die Probanden sollten alle für sie nicht intuitiven Vorgänge zum Durchführen der Aufgaben anmerken und eventuelle Wünsche für andere Abläufe und Funktionen äußern. Dabei wurden erste Schwächen bei der Verwendung des Prototyps identifiziert und diese nach der Benutzerstudie wie folgt behoben.

Als allgemeine Anmerkung zum Prototyp wurde festgestellt, dass die Schriftgröße teilweise zu klein gehalten wurde und Symbole aufgrund ihrer Größe nicht erkannt werden konnten. Weiters wurden Abkürzungen von Begriffen im Feld der Analysesituation als Beschriftung verwendet. Diese wurden von den Probanden nicht sofort erkannt. Um die Lesbarkeit zu steigern, wurde die Schriftgröße erhöht und verwendete Abkürzungen durch die vollen Namen ersetzt.

Eine weitere aufgetauchte Schwäche war, dass alle Elemente in den Cube-Informationen durch einen Klick in ein Feld der Analysesituation übertragen werden konnten, dies jedoch bei den Levels einer Dimension nicht möglich war. Diese Funktion wurde aufgrund der nicht bekannten Zuordnung zu Dice-Node oder Granularität einer Dimension nicht implementiert. Durch diese Anmerkung der Probanden wird ein Level nun standardmäßig durch einen Klick als Granularität in die jeweilige Dimension eingefügt.

Weiters wurde festgestellt, dass Saiku die Möglichkeit bietet, Elemente mittels Drag & Drop nicht nur zur Abfrage hinzuzufügen, sondern diese durch Wegziehen aus den Feldern wieder zu entfernen. Diese Funktionalität wurde nach der Benutzerstudie ebenfalls in den Prototyp übernommen. Sämtliche Elemente in einer Analysesituation können nun auch wieder aus den dafür vorgesehenen Feldern gezogen und so entfernt werden.

Die Überschriften der Buttons in der Navigationsleiste waren teilweise nicht verständlich genug gestaltet. Diese wurden angepasst und mit besser verständlichen Beschriftungen versehen. Dadurch ist eine einfachere Navigation mit diesen Buttons möglich.

Zusätzlich zu den Abfragen, für die ein Ergebnis angefordert wurde, werden nun auch vom Benutzer benannte Analysesituationen im kombinierten Graphen angezeigt. Dies war zur Zeit des Tests noch nicht der Fall und sorgte für Unverständnis bei den Teilnehmern, die ihre gerade erstellten und benannten Analysesituationen nicht im kombinierten Graphen finden konnten.

Die Symbole der Operatoren, die die Granularität bzw. die Dice-Node einer Dimension um einen Schritt in eine Richtung ändern, wurden von den Probanden als nicht intuitiv genug bemängelt. Einfache Richtungspfeile anstatt der Symbole sollen diesen Vorgang vereinfachen. Die jeweiligen Symbole wurden im UI im Bereich der Analysesituationen durch Pfeilsymbole ausgetauscht. Intern wird die Anwendung dieser Pfeilsymbole jedoch mit den jeweiligen Operatoren fortgeführt.

Die ersten Tests lieferten Einblicke in die Benutzbarkeit des erstellten Prototyps. Die Berechnung einer quantitativen Kennzahl für die Benutzbarkeit, wie beispielsweise ein System-Usability-Score (SUS) war jedoch kein Ziel der Evaluierung. Um aussagekräftige Werte zu erhalten, bedarf es einer größeren Anzahl an Teilnehmern sowie fest definierte Aufgaben und Feedback-Bögen. Nichtsdestotrotz hat bereits diese einfache Evaluierung wesentlich zur Verbesserung der Benutzbarkeit beigetragen.

8. Fazit und Ausblick

Diese Masterarbeit beschreibt die Konzeption und Erstellung eines prototypischen OLAP-Clients basierend auf einem Analysegraph-Navigationsmodell. Diese Analysegraphen bestehen aus Analysesituationen, die über Navigationsschritte miteinander verbunden sind. Diese Navigationsschritte enthalten Manipulationsoperatoren, die die jeweils durchgeführten Änderungen zwischen der Quell- und der Ziel-Analysesituation beinhalten. Benutzer haben die Möglichkeit, durch die Anwendung des Clients eine Abfrage zusammenzustellen, ohne dazu die weitere Kenntnis einer Datenabfragesprache zu benötigen. Der Prototyp ist eine Webanwendung, die ihre Informationen über REST-Schnittstellen aus einem Backend bezieht und den erstellen Analysegraphen dort speichert. Erste durchgeführte Benutzbarkeitstest weisen darauf hin, dass diese Form eines OLAP-Clients eine Alternative zu bisher vorhandenen OLAP-Clients darstellen könnte.

Im Zusammenhang mit der Benutzbarkeit des Prototyps sind weitere Versuche und eine ausgedehntere Benutzerstudie notwendig, um so einen besseren Einblick in die Einsetzbarkeit des in dieser Arbeit entstandenen Prototyps, auch im Vergleich mit anderen OLAP-Clients, zu erhalten. Weiters existieren Funktionen und Erweiterungsmöglichkeiten, die einen Mehrwert für den Anwender bieten können. Dazu zählen die Einbindung von OLAP-Patterns (Schuetz, Schausberger, Kovacic, & Schrefl, 2017), die komplexere Abfragen ermöglichen. Eine andere Möglichkeit könnte die Zusammenführung des Prototyps mit bereits bestehenden OLAP-Clients (Hilal et al., 2017), die mit proaktiv modellierten Analysegraph-Schemata arbeiten, sein.

Quellenverzeichnis

Aruldoss, M., Lakshmi Travis, M., & Prasanna Venkatesan, V. (2014). A survey on recent research in business intelligence. *Journal of Enterprise Information Management*, *27*(6), 831–866. https://doi.org/10.1108/JEIM-06-2013-0029

Battle, R., & Benson, E. (2008). Bridging the semantic Web and Web 2.0 with Representational State Transfer (REST). *Journal of Web Semantics*, *6*(1), 61–69. https://doi.org/10.1016/j.websem.2007.11.002

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*, *284*(5), 28–37.

Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, *26*(1), 65–74. https://doi.org/10.1145/248603.248616

Davis, R., Shrobe, H., & Szolovits, P. (1993). *What Is a Knowledge Representation?* 17.

Etcheverry, L. (2018). *A Vocabulary for Business Intelligence over Linked Data*. Abgerufen von https://github.com/lorenae/qb4olap (Original work published 2015)

Fielding, R. T., & Taylor, R. N. (2002). Principled design of the modern Web architecture. *ACM Transactions on Internet Technology*, *2*(2), 115–150. https://doi.org/10.1145/514183.514185

Gellersen, H.-W., & Gaedke, M. (1999). Object-oriented Web application development. *IEEE Internet Computing*, *3*(1), 60–68. https://doi.org/10.1109/4236.747323

Giacometti, A., Marcel, P., & Negre, E. (2008). A framework for recommending OLAP queries. *Proceeding of the ACM 11th International Workshop on Data Warehousing and OLAP - DOLAP '08*, 73. https://doi.org/10.1145/1458432.1458446

Golfarelli, M., Maio, D., & Rizzi, S. (1998). THE DIMENSIONAL FACT MODEL: A CONCEPTUAL MODEL FOR DATA WAREHOUSES. *International Journal of Cooperative Information Systems*, *07*(02n03), 215–247. https://doi.org/10.1142/S0218843098000118

Golfarelli, M., Rizzi, S., & Cella, I. (2004). Beyond data warehousing: What's next in business intelligence? *Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP - DOLAP '04*, 1. https://doi.org/10.1145/1031763.1031765

Hertel, A., Broekstra, J., & Stuckenschmidt, H. (2009). RDF Storage and Retrieval Systems. In S. Staab & R. Studer (Hrsg.), *Handbook on Ontologies* (S. 489–508). https://doi.org/10.1007/978-3-540-92673-3_22

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2008). Design Science in Information Systems Research. *Management Information Systems Quarterly*, *28*(1), 32.

Hilal, M., Schuetz, C. G., & Schrefl, M. (2017). An OLAP Endpoint for RDF Data Analysis Using Analysis Graphs. *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks Co-Located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017.*, 4. Abgerufen von http://ceur-ws.org/Vol-1963/paper515.pdf

Hilal, M., Schuetz, C. G., & Schrefl, M. (2018). Using superimposed multidimensional schemas and OLAP patterns for RDF data analysis. *Open Computer Science*, *8*(1), 18–37. https://doi.org/10.1515/comp-2018-0003

Hitachi Vantara. (2017). Mondrian. Abgerufen 19. August 2019, von https://community.hitachivantara.com/s/article/mondrian

Inmon, W. H. (2005). *Building the Data Warehouse*. John Wiley & Sons.

István Ujj-Mészáros. (2014). Bootstrap Dual Listbox. Abgerufen 26. August 2018, von https://www.virtuosoft.eu/code/bootstrap-duallistbox/

Julian Hyde. (2016). Foodmart data set in hsqldb format. Abgerufen 19. August 2019, von https://github.com/julianhyde/foodmart-data-hsqldb

Lazar, J. (2017). *Research methods in human computer interaction* (2nd edition). Cambridge, MA: Elsevier.

Matthews, B. (2005). Semantic web technologies. *E-learning*, *6*(6), 8.

MDBootstrap. (2018). Getting Started with MBD. Abgerufen 14. August 2018, von https://mdbootstrap.com/docs/jquery/getting-started/download/

Mike Bostock. (2019). D3.js. Abgerufen von https://d3js.org/

Neuböck, T. (2018). *Analysis Process Modeling Notation For Business Intelligence; nicht-publizierte Arbeitsversion*.

Neuböck, T., Neumayr, B., Rossgatterer, T., Anderlik, S., & Schrefl, M. (2012). Multi-dimensional Navigation Modeling Using BI Analysis Graphs. In S. Castano, P. Vassiliadis, L. V. Lakshmanan, & M. L. Lee (Hrsg.), *Advances in Conceptual Modeling* (Bd. 7518, S. 162–171). https://doi.org/10.1007/978-3-642-33999-8_20

Neuböck, T., & Schrefl, M. (2015). Modelling knowledge about data analysis processes in manufacturing. *IFAC-PapersOnLine*, *48*(3), 277–282.

Nicolas Kruchten. (2012). PivotTable.js. Abgerufen von https://pivottable.js.org/examples/

Schuetz, C. G., Schausberger, S., Kovacic, I., & Schrefl, M. (2017). Semantic OLAP patterns: Elements of reusable business analytics. *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 318–336. Springer.

SnapAppointments. (2015). Bootstrap select. Abgerufen 15. Mai 2018, von http://silviomoreto.github.io/bootstrap-select

Straßer, M. (2019). *Dialogbasierte Benutzungsschnittstelle für interaktive Datenanalyse in natürlicher Sprache* (Masterarbeit). Johannes Kepler Universität.

The Apache Software Foundation. (2019). Apache Jena Fuseki. Abgerufen 8. Juli 2019, von https://jena.apache.org/documentation/fuseki2/

Vaisman, A., & Zimányi, E. (2014). *Data Warehouse Systems*. https://doi.org/10.1007/978-3-642-54655-6

Vrandecic, D., & Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. *Commun. ACM*, *57*(10), 78–85. https://doi.org/10.1145/2629489

W3C. (2013). SPARQL 1.1 Overview. Abgerufen von http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/

W3C. (2014). RDF 1.1 Concepts and Abstract Syntax. Abgerufen von http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

Watson, H. J., & Wixom, B. H. (2007). The Current State of Business Intelligence. *Computer*, *40*(9), 96–99. https://doi.org/10.1109/MC.2007.331

Anhang: Dokumentation der REST-Schnittstellen des Backends

GET

A) /cubes

Liefert alle verfügbaren Cubes.

Parameter:

Code:

200: Erfolg

```
@graph*   [{
          @type   string
          comment {
                  @value*    string
                  @language* string
                  }
          @id     string
          label   {
                  @value*    string
                  @language* string
                  }
          }]
@context* {
          qb*      string
          drugs*   string
          rdf*     string
          qbx*     string
          qb4o*    string
          xsd*     string
          comment* string
          rdfs*    string
          label*   string
          food*    string
          }
```

500: Unbekannter Fehler

GET

B) /cubes/{cubeName}/baseMeasures

Liefert alle Basiskennzahlen zu einem übergebenen Cube.

Parameter:

cubeName: string (path) *notwendig

identifiziert den Cube

Code:

200: Erfolg

```
@context* {
          dataType* {
                    @id*   string
                    @type* string
                    }
          label*    string
          qb*       string
          drugs*    string
          rdf*      string
          qbx*      string
          qb4o*     string
          xsd*      string
          rdfs*     string
          food*     string
          }
@graph*   [{
          @id       string
          @type     string
          dataType  string
          label     {
                    @language* string
                    @value*    string
                    }
          }]
```

500: Unbekannter Fehler

GET

C) /cubes/{cubeName}/aggregateMeasures

Liefert alle aggregierten Kennzahlen zu einem übergebenen Cube.

Parameter:

cubeName: string (path) *notwendig

identifiziert den Cube

Code:

200: Erfolg

```
@context* {
          derivedFrom*          {
                                @id*   string
                                @type* string
                                }
          aggregationFunction*  {
                                @id*   string
                                @type* string
                                }
          label*                string
          comment*              string
          qb*                   string
          drugs*                string
          rdf*                  string
          qbx*                  string
```

```
          qb4o*                string
          xsd*                 string
          rdfs*                string
          food*                string
         }
@graph*   [{
          @id                 string
          @type               string
          aggregationFunction string
          derivedFrom         [string]
          comment             {
                               @language* string
                               @value*    string
                               }
          label               {
                               @language* string
                               @value*    string
                               }
         }]
```

500: Unbekannter Fehler

GET

D) /cubes/{cubeName}/baseMeasures/{measureName}/predicates

Liefert alle Filter zu der Basiskennzahl eines Cube.

Parameter:

cubeName: string (path) *notwendig

identifiziert den Cube

measureName: string (path) *

Identifiziert die Basiskennzahl

Code:

200: Erfolg

```
@context* {
          label*   string
          comment* string
          qb*      string
          drugs*   string
          rdf*     string
          qbx*     string
          qb4o*    string
          xsd*     string
          rdfs*    string
          food*    string
          }
@graph*   [{
          @id     string
          comment {
                  @language* string
```

```
                @value*    string
                }
          label   {
                @language* string
                @value*    string
                }
         }]
```

500: Unbekannter Fehler

GET

E) /cubes/{cubeName}/aggregateMeasures/{measureName}/predicates

Liefert alle Filter zur übergebenen aggregierten Kennzahl.

Parameter:

cubeName: string (path) *notwendig

identifiziert den Cube

measureName: string (path) *

identifiziert die aggregierte Kennzahl

Code:

200: Erfolg

```
@context* {
          derivedFrom         {
                              @id*   string
                              @type* string
                              }
          aggregationFunction {
                              @id*   string
                              @type* string
                              }
          label*              string
          comment*            string
          qb*                 string
          drugs*              string
          rdf*                string
          qbx*                string
          qb4o*               string
          xsd*                string
          rdfs*               string
          food*               string
          }
@graph*   [{
          @id     string
          comment {
                  @language* string
                  @value*    string
                  }
          label   {
                  @language* string
```

```
                @value*    string
                }
        }]
```

500: Unbekannter Fehler

GET

F) /cubes/{cubeName}/aggregateMeasures/{measureName}/ composedMeasures

Liefert alle aggregierten Kennzahlen, aus denen die übergebene Kennzahl zusammengestellt ist.

Parameter:

cubeName: string (path) *notwendig

identifiziert den Cube

measureName: string (path) *

identifiziert die aggregierte Kennzahl

Code:

200: Erfolg

```
@context* {
          derivedFrom*          {
                                 @id*   string
                                 @type* string
                                }
          aggregationFunction* {
                                 @id*   string
                                 @type* string
                                }
          label*                string
          comment*              string
          qb*                   string
          drugs*                string
          rdf*                  string
          qbx*                  string
          qb4o*                 string
          xsd*                  string
          rdfs*                 string
          food*                 string
         }
@graph*   [{
          @id                  string
          @type                string
          aggregationFunction string
          derivedFrom          string
          comment              {
                                @language* string
                                @value*    string
                                }
          label                {
                                @language* string
                                @value*    string
```

```
                        }
        }]
```

500: Unbekannter Fehler

GET

G) /cubes/{cubeName}/dimensions

Liefert alle Dimensionen zu einem übergebenen Cube.

Parameter:

cubeName: string (path) *notwendig

identifiziert den Cube

Code:

200: Erfolg

```
@context* {
          pcCardinality* {
                         @id*   string
                         @type* string
                         }
          parentLevel*   {
                         @id*   string
                         @type* string
                         }
          inHierarchy*   {
                         @id*   string
                         @type* string
                         }
          childLevel*    {
                         @id*   string
                         @type* string
                         }
          label*         string
          comment*       string
          hasHierarchy*  {
                         @id*   string
                         @type* string
                         }
          rdf*           string
          qbx*           string
          rdfs*          string
          food*          string
          }
@graph*   [{
          @id           string
          @type         string
          childLevel    string
          inHierarchy   string
          parentLevel   string
          pcCardinality string
          }]
```

500: Unbekannter Fehler

GET

H) **/analysis/situations/{situationID}/result**

Liefert das Ergebnis einer Analysesituation als Link zu einer CSV-Datei.

Parameter:

situationID: string (path) *notwendig

identifiziert die Analysesituation

Code:

200: Erfolg

```
vals*           [string]
csv*            string
aggregatorName* string
rows*           [string]
cols*           [string]
```

500: Unbekannter Fehler

GET

I) **/analysis/situation/new**

Liefert eine neue UID.

Parameter:

Code:

200: Erfolg

```
string
```

500: Unbekannter Fehler

POST

J) **/analysis/situations/{sessionID}/add**

Fügt den Inhalt des RequestBodies zum Graphen der SessionID hinzu.

Parameter:

sessionID: string (path) *notwendig

identifiziert den Analysegraphen

element: string (body) *

Analysesituation die zum Graphen hinzugefügt werden soll

```
String*
```

Code:

201: Inhalt hinzugefügt

500: Unbekannter Fehler

POST

K) /analysis/situations/{sessionID}/navigationStep/add

Fügt den Inhalt des RequestBodies zum Graphen der SessionID hinzu.

Parameter:

sessionID: string (path) *notwendig

identifiziert den Analysegraphen

element: string (body) *

Navigationsschritt der zum Graphen hinzugefügt werden soll

```
String*
```

Code:

201: Inhalt hinzugefügt

500: Unbekannter Fehler

GET

L) /analysis/situations/{sessionID}/{uuid}

Liefert die jeweilige Analysesituation zurück.

Parameter:

sessionID: string (path) *notwendig

identifiziert den Cube

uuid . string (Path) *

identifiziert die Analysesituation

Code:

200: Erfolg

```
@context* {
          sliceCondition*              {
                                       @id*   string
                                       @type* string
                                       }
          hierarchy*                   string
          granularity*                 {
                                       @id*   string
                                       @type* string
```

```
                                    }
          dimension*                string
          diceNode*                 {
                                    @id*   string
                                    @type* string
                                    }
          diceLevel*                {
                                    @id*   string
                                    @type* string
                                    }
          time*                     string
          hasDimensionSpec*         {
                                    @id*   string
                                    @type* string
                                    }
          hasCube*                  {
                                    @id*   string
                                    @type* string
                                    }
          hasBaseMeasurePredicate*  {
                                    @id*   string
                                    @type* string
                                    }
          hasAggregateMeasurePredicate* {
                                    @id*   string
                                    @type* string
                                    }
          hasAggregateMeasure*      {
                                    @id*   string
                                    @type* string
                                    }
          label*                    string
          drugs*                    string
          rdf*                      string
          qbx*                      string
          rdfs*                     string
          food*                     string
         }
@graph*  [{
          @id            string
          diceLevel      string
          diceNode       string
          dimension      string
          granularity    string
          hierarchy      string
          sliceCondition string
          }]
```

500: Unbekannter Fehler

GET

M) /analysis/situations/{sessionID}/{uuid}/information

Liefert die jeweilige Analysesituation zurück.

Parameter:

sessionID: string (path) *notwendig

identifiziert den Cube

uuid . string (Path) *

identifiziert die Analysesituation

Code:

200: Erfolg

```
@context* {
          hasCube*                       string
          hasBaseMeasurePredicate*       string
          hasAggregateMeasurePredicate*  string
          hasAggregateMeasure*           string
          label*                         string
          sliceCondition*                string
          granularity*                   string
          dimension*                     string
          diceNode*                      string
          diceLevel*                     string
          rdf*                           string
          qbx*                           string
          rdfs*                          string
          food*                          string
         }
@graph*   [{
          @id             string
          diceLevel       {
                          @language* string
                          @value*    string
                          }
          diceNode        {
                          @language* string
                          @value*    string
                          }
          dimension       string
          granularity     {
                          @language* string
                          @value*    string
                          }
          sliceCondition  {
                          @language* string
                          @value*    string
                          }
         }]
```

500: Unbekannter Fehler

GET

N) /analysis/situations/comparative/measures

Liefert für die beiden Cubes der vergleichenden Analysesituation alle verfügbaren vergleichenden Kennzahlen.

Parameter:

interestCube:	string (query)	
	Cube der ersten Analysesituation	
comparisonCube	string (query)	
	Cube der zweiten Analysesituation	

Code:

200: Erfolg

```
@context* {
          setOfInterestMeasure*   {
                                   @id*   string
                                   @type* string
                                   }
          setOfComparisonMeasure* {
                                   @id*   string
                                   @type* string
                                   }
          label*                  string
          comment*                string
          qb*                     string
          drugs*                  string
          rdf*                    string
          qbx*                    string
          qb4o*                   string
          xsd*                    string
          rdfs*                   string
          food*                   string
         }
@graph*   [{
          @id                    string
          @type                  string
          setOfComparisonMeasure string
          setOfInterestMeasure   string
          comment                {
                                  @language* string
                                  @value*    string
                                  }
          label                  {
                                  @language* string
                                  @value*    string
                                  }
         }]
```

500: Unbekannter Fehler

GET

O) /analysis/situations/comparative/filterConditions

Liefert für die beiden Cubes der vergleichenden Analysesituation alle verfügbaren vergleichenden Kennzahlenfilter.

Parameter:

interestCube: string (query)

Cube der ersten Analysesituation

comparisonCube string (query)

Cube der zweiten Analysesituation

Code:

200: Erfolg

```
@context* {
          over*    {
                    @id*   string
                    @type* string
                   }
          label*   string
          comment* string
          qb*      string
          drugs*   string
          rdf*     string
          qbx*     string
          qb4o*    string
          xsd*     string
          rdfs*    string
          food*    string
         }
@graph*  [{
          @id     string
          @type   string
          over    string
          comment {
                   @language* string
                   @value*    string
                   }
          label   {
                   @language* string
                   @value*    string
                   }
         }]
```

500: Unbekannter Fehler

POST

P) /analysis/situations/compative/joinConditions

Liefert für einen JSON-String mit den aktuellen Granularitätslevels die gültigen Join-Conditions.

Parameter:

granularityLevels: object (body) *notwendig

JSON-String mit allen Granularitätslevels

```
{
  "interest": [
    "string"
  ],
  "comparison": [
    "string"
  ]
}
```

Code:

200: Erfolg

```
@context* {
          over*    {
                    @id*   string
                    @type* string
                   }
          label*   string
          comment* string
          qb*      string
          drugs*   string
          rdf*     string
          qbx*     string
          qb4o*    string
          xsd*     string
          rdfs*    string
          food*    string
         }
@graph*  [{
          @id     string
          @type   string
          over    string
          comment {
                   @language* string
                   @value*    string
                   }
          label   {
                   @language* string
                   @value*    string
                   }
         }]
```

500: Unbekannter Fehler

GET

Q) /dimensions/{dimensionName}/hierarchies

Liefert alle Hierarchien einer Dimension

Parameter:

dimensionName: string (path) *notwendig

identifiziert eine Dimension

Code:

200: Erfolg

```
@context* {
          label*          string
          comment*        string
          pcCardinality*  {
                          @id*    string
                          @type*  string
                          }
          parentLevel*    {
                          @id*    string
                          @type*  string
                          }
          inHierarchy*    {
                          @id*    string
                          @type*  string
                          }
          childLevel*     {
                          @id*    string
                          @type*  string
                          }
          qb*             string
          drugs*          string
          rdf*            string
          qbx*            string
          qb4o*           string
          xsd*            string
          rdfs*           string
          food*           string
          }
@graph*   [{
          @id             string
          @type           string
          childLevel      string
          inHierarchy     string
          parentLevel     string
          pcCardinality string
          }]
```

500: Unbekannter Fehler

GET

R) /dimensions/hierarchies/{hierarchyName}

Liefert alle Hierarchieschritte einer Hierarchie.

Parameter:

hierarchyName: string (path) *notwendig

identifiziert eine Hierarchie

Code:

200: Erfolg

```
@context* {
          label*          string
          comment*        string
```

```
          pcCardinality* {
                         @id*   string
                         @type* string
                         }
          parentLevel*   {
                         @id*   string
                         @type* string
                         }
          inHierarchy*   {
                         @id*   string
                         @type* string
                         }
          childLevel*    {
                         @id*   string
                         @type* string
                         }
          qb*            string
          drugs*         string
          rdf*           string
          qbx*           string
          qb4o*          string
          xsd*           string
          rdfs*          string
          food*          string
         }
@graph*  [{
          @id           string
          @type         string
          childLevel    string
          inHierarchy   string
          parentLevel   string
          pcCardinality string
         }]
```

500: Unbekannter Fehler

GET **S) /dimensions/hierarchies/steps/{hierarchyStepName}/levels**

Liefert die beiden Levels zu einem Hierarchieschritt.

Parameter:

hierarchyStepName: string (path) *notwendig

identifiziert einen Hierarchieschritt

Code:

200: Erfolg

```
@context* {
          pcCardinality* {
                         @id*   string
                         @type* string
                         }
          parentLevel*   {
```

```
                    @id*    string
                    @type*  string
                    }
          inHierarchy*    {
                    @id*    string
                    @type*  string
                    }
          childLevel*     {
                    @id*    string
                    @type*  string
                    }
          label*          string
          comment*        string
          qb*             string
          drugs*          string
          rdf*            string
          qbx*            string
          qb4o*           string
          xsd*            string
          rdfs*           string
          food*           string
         }
@graph*  [{
          @id             string
          @type           string
          childLevel      string
          inHierarchy     string
          parentLevel     string
          pcCardinality   string
         }]
```

500: Unbekannter Fehler

GET **T) /dimensions/levels/{levelName}/levelPredicates**

Liefert die Slice-Conditions zu einem übergebenen Level.

Parameter:

levelName: string (path) *notwendig

identifiziert einen Level

Code:

200: Erfolg

```
@context* {
          label*    string
          comment*  string
          qb*       string
          drugs*    string
          rdf*      string
          qbx*      string
          qb4o*     string
          xsd*      string
```

```
          rdfs*    string
          food*    string
         }
@graph*  [{
          @id     string
          @type   string
          comment {
                  @language* string
                  @value*    string
                  }
          label   {
                  @language* string
                  @value*    string
                  }
         }]
```

500: Unbekannter Fehler

GET

U) /levelMembers/{level}

Liefert die Dice-Nodes zu einem übergebenen Level.

Parameter:

level: string (path) *notwendig

identifiziert einen Level

Code:

200: Erfolg

```
@context* {
          inLevel* {
                   @id*   string
                   @type* string
                  }
          label*   string
          comment* string
          qb*      string
          drugs*   string
          rdf*     string
          qbx*     string
          qb4o*    string
          xsd*     string
          rdfs*    string
          food*    string
         }
@graph*  [{
          @id     string
          @type   string
          inLevel [string]
          comment {
                  @language* string
                  @value*    string
                  }
```

```
        label   {
                @language* string
                @value*    string
                }
        }]
```

500: Unbekannter Fehler