Visuelle Suchanfragen auf graphbasierten Datenstrukturen

Von der Fakultät Informatik, Elektrotechnik und
Informationstechnik der Universität Stuttgart
zur Erlangung der Würde eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

Vorgelegt von

Florian Haag

aus Stuttgart

Hauptberichter:	Prof. Dr. Thomas Ertl
Mitberichter:	Prof. Dr. Thomas Schlegel
Tag der mündlichen Prüfung:	26. Februar 2016

Institut für Visualisierung und Interaktive Systeme
der Universität Stuttgart

2016

Abstract

The total amount of available data is steadily increasing. Standardized data formats allow for connecting different data sources, which can include merging of different data items depending on the use case. This creates even more comprehensive datasets that render finding a particular piece of information difficult.

If the data consist of unstructured of homogenous information, searching can only be done by matching patterns with data items or parts thereof—for instance, character strings or regular expressions that match parts of textual data items. However, the availability of structured data is increasing. This kind of data is either stored as distinct facets of each data item from the outset, or originally unstructured data has been enriched to form a structure. As each facet can indicate a link to another data item, the entire dataset forms a graph that is suitable for faceted search conepts. At this point, some interoperability across data sources can be achieved by employing Semantic Web approaches and techniques.

Numerous works have attempted to visualize an overview of the entire dataset, or details of a particular excerpt of the dataset. Finding specific data remains a problem, however, as the precise specification of search criteria is difficult. As these criteria can be connected in complex ways, just like the overview of datasets, this issue lends itself to using visual representations. A special trait of this application of visualization is the possible absence of any data. Instead, the visualization must be capable of conveying the conceptual idea of a search query without displaying any data. Former works related to this problem focused on the visual representation of search queries and filter expressions for relational and object-oriented databases. More recent works increasingly address a Semantic Web context. Various of these concepts, however, lack a clear abstract definition. Also, scalability issues appear in the case of complex queries. Furthermore, little attention was paid to how to connect several concepts in order to combine advantages of different query visualizations.

This dissertation considers the described problems and presents six concepts for query visualization. Both generic visualizations—that is, for filtering any kind of structured data—and domain-specific or type-specific visualizations are addressed. In part, existing approaches have been adapted to the particularities of graph-based data structures. Likewise, several new approaches specifically designed for this kind of data are presented. For each of these concepts, the necessity of a dataset is discussed. Moreover,

options for providing a preview on query results from such a dataset, if available, are considered. Finally, ways for connecting the query visualization concepts are presented. This connection approach is suitable for systematically linking together arbitrary query visualizations.

By means of the connection approach, users can express different parts of a query using different visualization concepts, in order to benefit from the advantages of several query visualizations at a time. Like this, queries that include complex criteria as well as complex relations between criteria can now be defined and displayed visually without losing the visual overview of any of these aspects.

Zusammenfassung

Die Menge an verfügbaren Daten nimmt stetig zu. Durch standardisierte Datenformate wird die Verknüpfung verschiedener Datenquellen und dadurch auch die Zusammenführung unterschiedlicher Datenelemente je nach Anwendungszweck ermöglicht. Dies führt wiederum zu noch umfassenderen Datenbeständen, in denen die eigentlich gewünschten Informationen teilweise nur schwer gefunden werden können.

Handelt es sich bei den Daten um unstrukturierte oder gleichförmige Informationen, so beschränken sich Suchmöglichkeiten auf die Suche nach Übereinstimmungen von Mustern mit Datenelementen oder Teilen davon – beispielsweise Zeichenketten oder regulären Ausdrücken, die mit Teilen von textuellen Datenelementen übereinstimmen. In zunehmendem Maß stehen jedoch auch strukturierte Daten zur Verfügung. Bei diesen wird entweder von Anfang an zwischen unterschiedlichen Facetten pro Datenelement unterschieden, oder es wurden ursprünglich unstrukturierte Daten entsprechend angereichert. Da die einzelnen Facetten auch Verknüpfungen zu anderen Datenelementen darstellen können, entstehen hierbei Graphstrukturen, welche sich für Ansätze der facettierten Suche eignen. Eine Interoperabilität zwischen Datenquellen wird hier unter anderem über die Konzepte und Techniken des Semantic Web erreicht.

Zahlreiche Arbeiten haben sich mit der Darstellung der gesamten Datenmengen als Übersicht oder von festgelegten Ausschnitten der Datenmengen im Detail auseinandergesetzt. Jedoch ist das Auffinden bestimmter Daten nach wie vor ein Problem. Die Schwierigkeit liegt dabei darin, die Suchkriterien präzise auszudrücken. Da sich zwischen den einzelnen Kriterien komplexe Zusammenhänge ergeben können, bietet sich auch hier genau wie bei der Übersicht der Datenmengen eine visuelle Darstellung an. Eine Besonderheit dieses Einsatzszenarios für Visualisierungen besteht darin, dass nicht zwangsläufig Daten vorliegen. Statt dessen muss die Visualisierung auch ohne verfügbare Daten die konzeptuelle Idee einer Suchanfrage ausdrücken. Frühere Arbeiten zu diesem Problem befassen sich mit der visuellen Repräsentation von Suchanfragen und Filterausdrücken in Bezug auf relationale Datenbanken und Objektdatenbanken. Viele neuere Arbeiten gehen vermehrt auch auf den Kontext des Semantic Webs ein. Einige dieser Konzepte sind jedoch nicht auf abstrakte Weise klar definiert. Bei komplexeren Anfragen treten zum Teil auch Skalierungsprobleme auf. Zudem wurde bisher kaum betrachtet, wie sich unterschiedliche Konzepte miteinander in Verbindung bringen lassen, um die Vorteile aus unterschiedlichen

Anfragevisualisierungen nutzen zu können.

Diese Dissertation adressiert die beschriebenen Probleme und stellt sechs Konzepte für die visuelle Darstellung von Suchanfragen vor. Es wird sowohl auf Visualisierungen für allgemeine Einsatzzwecke – also für die Filterung beliebiger strukturierter Informationen –, als auch für spezielle Domänen oder Arten von Informationen eingegangen. Bestehende Ansätze wurden teilweise auf die Gegebenheiten graphbasierter Datenstrukturen angepasst. Ebenso werden neue Ansätze präsentiert, die gezielt auf diese Art von Datenstrukturen ausgelegt sind. Dazu wird jeweils erörtert, inwiefern sich die Anfragevisualisierungen auch ohne Vorhandensein einer zu filternden Datensammlung einsetzen lassen. Zudem wird erklärt, wie bei Vorhandensein einer solchen eine Vorschau auf die Ergebnisse des Filtervorgangs gewährt werden kann. Abschließend werden Verbindungsmöglichkeiten der unterschiedlichen Visualisierungskonzepte präsentiert. Dieser Verbindungsansatz eignet sich dazu, beliebige Anfragevisualisierungen systematisch miteinander zu kombinieren.

Mit dem Verbindungskonzept können Benutzer verschiedene Bestandteile einer Anfrage mittels unterschiedlicher Visualisierungskonzepte ausdrücken, um gleichzeitig von den Stärken unterschiedlicher Anfragevisualisierungen zu profitieren. Auf diese Weise können nun Anfragen visuell definiert und dargestellt werden, die sowohl komplexe Bedingungen als auch komplexe Zusammenhänge zwischen den Bedingungen aufweisen, ohne die visuelle Übersicht über einen dieser Aspekte zu verlieren.

Danksagung

Ich möchte mich in aller Form bei meinem Hauptberichter Thomas Ertl für die Chance bedanken, einige Jahre lang Teil des Instituts für Visualisierung und Interaktive Systeme (VIS) sein zu dürfen. Ebenso danke ich ihm und meinem Mitberichter Thomas Schlegel dafür, dass sie mich regelmäßig mit hilfreichen Vorschlägen unterstützt haben, mir jedoch gleichzeitig den Freiraum gelassen haben, meine eigenen Ideen zu verfolgen.

Meinen zahlreichen Kollegen am VIS, allen voran meinen dienstälteren Bürokollegen Martin Falk, Michael Raschke und Steffen Lohmann, danke ich für die Hinweise und Unterstützung beim Einstieg in die Forschung und dem Aufbau meines wissenschaftlichen Portfolios. Auch den anderen, die an Veröffentlichungsvorhaben teilnahmen, die stets für themenbezogene Diskussionen offen waren, die sich gelegentlich als Teilnehmer für Testdurchläufe von Benutzerstudien zur Verfügung stellten oder die auch einfach nur dafür sorgten, die Forschungs- und Lehrtätigkeiten am VIS angenehm und vielseitig zu gestalten, bin ich zu Dank verpflichtet. Dies schließt Kollegen am VIS ebenso wie an anderen Instituten mit ein, wie Tanja Blascheck, Harald Bosch, Klaus Bosse, Qi Han, Philipp Heim, Florian Heimerl, Dominik Herr, Andreas Hub, Markus John, Steffen Koch, Robert Krüger, Kuno Kurzhals, Hermann Pflüger, Bernhard Schmitz, Christoph Stach, Christiane Taras, Dennis Thom und Michael Wörner und weitere. Den Trägern und Partnern im Projekt IP-KOM-ÖV danke ich für die Finanzierung eines Teils meiner Forschung und die Einblicke in die Welt des öffentlichen Verkehrs, die mir das Projekt ermöglicht hat.

Ich möchte mich auch bei den von mir betreuten Studenten bedanken, die in ihren Abschlussarbeiten themenverwandte Fragen beleuchtet haben. Dies gilt gleichermaßen für meine studentischen Hilfskräfte, die eine wertvolle Unterstützung bei der Entwicklung von Forschungsprototypen darstellten, allen voran Sukanya Bhowmik, Gundolf Kuhn, Hao Li und Corvin Schapöhler.

Auch meiner Frau und meinen Eltern möchte ich dafür danken, dass sie mich ermutigt haben, das Vorhaben der Promotion zu verfolgen. Ebenso danke ich ihnen sowie meiner Tochter dafür, dass sie die Geduld hatten, mich auf dem langen Weg bis zur Fertigstellung der Dissertation zu begleiten.

Letztendlich danke ich all den Autoren der in Anhang B aufgeführten Werkzeuge, ohne deren Existenz meine Forschung und diese Arbeit in der vorliegenden Form nicht möglich gewesen wären.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Quelltextbeispiele

Algorithmenverzeichnis (Pseudocode)

Verwendete Abkürzungen

DAML	**D**ARPA **A**gent **M**arkup **L**anguage
DNF	**D**isjunktive **N**ormal**f**orm
Gtk#	**G**IMP **T**ool**k**it für .NET
IRI	**I**nternational **R**esource **I**dentifier
OWL	**W**eb **O**ntology **L**anguage
RDFS	**R**esource **D**escription **F**ramework **S**chema
RDF	**R**esource **D**escription **F**ramework
SPARQL	**SPARQL P**rotocol **a**nd **R**DF **Q**uery **L**anguage
SQL	**S**tructured **Q**uery **L**anguage
UML	**U**nified **M**odelling **L**anguage
URI	**U**niform **R**esource **I**dentifier
VQL	**V**isual **Q**uery **L**anguage
VQS	**V**isual **Q**uery **S**ystem
WPF	**W**indows **P**resentation **F**oundation
XML	E**x**tensible **M**arkup **L**anguage

Teil I

Motivation und Stand der Technik

1 Einleitung und Motivation

In den letzten Jahren wurden immer größere Mengen an Daten verfügbar. Die Datenmengen, auf die gesammelt zugegriffen werden kann, wachsen noch schneller an, da zunehmend Dienste miteinander verknüpft werden. Beispielsweise werden über die Produktsuche des Online-Händlers Amazon [Ama15] nicht nur die von dem Unternehmen selber vertriebenen Produkte zugreifbar. Auch solche Produkte, die von Drittanbietern geliefert werden, werden aufgelistet. Durch die Verwendung standardisierter Datenaustauschformate wird diese Entwicklung weiter vorangetrieben.

Insbesondere im Zusammenhang mit dem Semantic Web hat sich eine Reihe von De-facto-Standardformaten um die Technologie RDF [CDN+14] herausgebildet, welche den einheitlichen Austausch und die Verknüpfung von Daten ermöglicht. Auf dieser Grundlage ist die Idee von Linked Data entstanden – dem Ansatz, dezentralisiert Datensammlungen anzubieten, welche auf einer Reihe einheitlicher Vokabulare aufbauen. Somit können diese Datensammlungen zu einem quellenüberspannenden Objektgraphen verschmolzen werden [BHB09]. Derartige Datenquellen sind mittlerweile zu unterschiedlichsten Themengebieten vorhanden. Zunehmend stellen Regierungen statistische und organisatorische Daten zu ihrem jeweiligen Land als Linked Open Data bereit [DPH12; DLE+11]. Zahlreiche weitere themenspezifische Datensammlungen sind abrufbar [BNT+08; DB08; HC09; KCL+15]. Auch themenübergreifende Enzyklopädien von allgemeinem Interesse sind als Linked Data umgesetzt, wie das DBpedia-Projekt [ABK+07; LIJ+15].

Obwohl nun zahlreiche Datensammlungen verfügbar sind, werden dennoch bessere Möglichkeiten benötigt, diese zu durchsuchen [SOB+12]. Gleichermaßen kann für die reine Visualisierung von Daten bereits eine Filterung notwendig sein, wenn die Daten zunächst vereinfacht werden müssen [MLL+13]. Schon jetzt existieren zahlreiche Ansätze, welche die Navigation und die Entdeckung von Zusammenhängen in den Objektgraphen mit visuellen Mitteln unterstützen [MG14]. Es bestehen jedoch noch Schwierigkeiten dabei, gezielt nach Informationen zu suchen, die bestimmte Kriterien erfüllen. Je nach Benutzer können unterschiedliche Auswahlkriterien davon bevorzugt werden. Insbesondere, wenn diese Kriterien in boolesche Verknüpfungen wie Disjunktionen oder Konjunktionen eingeschlossen werden sollen, fällt Nutzern die präzise Angabe der Filterkriterien schwer: Werden die Anfragen natürlichsprachlich ausgedrückt, kann es aufgrund sprachlicher Mehrdeutigkeiten zu Missverständnissen bei Begriffen wie „und“ und

„oder“ kommen [GDC+90; Jon98]. Andererseits sind formale Anfragesprachen wie SQL für Benutzer mit eingeschränkten Informatik- oder Mathematikkenntnissen nur bedingt geeignet.

Eine vielversprechende Möglichkeit, mit diesen Schwierigkeiten umzugehen, besteht darin, die Suchanfrage visuell auszudrücken. Ein Beispiel für solch eine Darstellungsform ist in Abbildung 1.1 zu sehen.

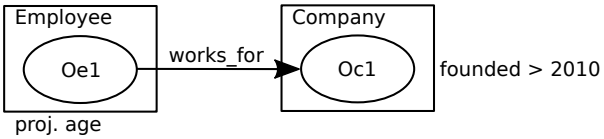


Abbildung 1.1: Visuelle Darstellung einer Suchanfrage nach dem Alter von Mitarbeitern von Unternehmen, die nach 2010 gegründet wurden, hier unter Verwendung der visuellen Notation OQD [KM93].

Erfahrungen aus dem Feld der relationalen Datenbanken können dabei nur begrenzt weiterverwendet werden. Bei Objektgraphen kann nämlich nicht von einer festen Tabellenstruktur ausgegangen werden. Zudem kommen durch zunehmend unterschiedliche Umgebungen neue Anforderungen hinzu. Beispielsweise ist eine platzsparende Darstellung erforderlich, um Anfragevisualisierungen auf kleineren Bildschirmen oder sogar auf mobilen Geräten anzeigen zu können. Eine solche kompaktere Darstellung ist ohnehin von Vorteil, da Anfragevisualisierungen nicht immer für sich alleine stehen. Häufig sind sie in die Oberflächen größerer Systeme eingebettet. Diese Systeme können die Anfrage ausführen und ihre Ergebnisse darstellen. Dies ist insbesondere bei Visual-Analytics-Systemen der Fall, in denen der Zyklus zwischen Verfeinerung der Anfrage und Erkennung von Mustern in den Ergebnissen mehrmals iterativ verlaufen kann. Es bleibt also weniger Platz für die Anfragevisualisierung, da noch andere Inhalte Platz finden müssen [BH86; WPQ+08; RS08; SGJ+13].

1.1 Forschungsfragen

Konkret betrachtet die vorliegende Arbeit die folgenden Fragen:

- Wie können Suchanfragen für graphbasierte Datenstrukturen mit und ohne Vorhandensein einer Datensammlung verständlich visualisiert werden?
- Wie kann diese Darstellung auf möglichst kompakte und dennoch verständliche Weise erfolgen?
- Wie können unterschiedliche Filtervisualisierungskonzepte miteinander kombiniert werden, um die Vorteile unterschiedlicher Ansätze zu vereinen?

1.2 Aufbau und wissenschaftlicher Beitrag

Das vorliegende Dokument ist in drei große Teile untergliedert. Der erste Abschnitt beschreibt den aktuellen Entwicklungsstand. Im Rahmen dieses Teils werden in Kapitel 1.3 zunächst einige wiederholt verwendete Begriffe definiert. Anschließend erfolgt eine Betrachtung verwandter Arbeiten in Kapitel 2. In letzterem enthalten ist auch eine knappe Analyse der Anwendungsgebiete, in denen in der Literatur bislang komplexe Suchanfragen und vor allem Anfragevisualisierungen verwendet wurden.

Der zweite Teil des Dokuments präsentiert mehrere Konzepte für Anfragevisualisierungen, die im Rahmen dieses Promotionsvorhabens mit unterschiedlichen Zielsetzungen entwickelt wurden.

Einerseits werden dazu in Kapitel 3 Anfragevisualisierungen betrachtet, die sich allgemein verwenden lassen und nicht auf bestimmte Datentypen oder Anwendungsfälle zugeschnitten sind. Das Filter/Flow-Konzept [YS93] wird erweitert, um unterschiedliche Anwendungsgebieten gerecht zu werden [HRE12][1]. Ebenso dienen die Erweiterungen dazu, Filter/Flow-Graphen auf kompaktere Weise darstellen zu können [HLE12; HLE13a]. Dabei wird auch mittels einer Benutzerstudie untermauert, dass die kompaktere Darstellung die Lesbarkeit der Anfragevisualisierung nicht verschlechtert [HLE13b]. Zudem wird eine Abbildung auf Objektgraphen und insbesondere die Anfragesprache SPARQL definiert [HLB+14; HLE14].

Filter/Flow-basierte Konzepte richten den Fokus auf die logische Verknüpfung von Filterkriterien. Im Gegensatz dazu wird mit QueryVOWL eine Visualisierung des gesuchten Subgraphmusters in einem Objektgraphen vorgeschlagen [HLS+; HLS+15b]. QueryVOWL stützt sich grafisch auf die visuelle Ontologienotation VOWL [NLH14], deren Lesbarkeit im Rahmen mehrerer qualitativer Auswertungen evaluiert und bestätigt wurde [LNH+14; LNH+16]. Magnetic Querying bildet den Ansatz Dust & Magnet [SMS+05] auf Objektgraphen ab. Es stellt damit ein Konzept dar, welches im Gegensatz zu den zwei zuvor genannten dazu eingesetzt werden kann, mehrere Gruppen zueinander ähnlicher Datenelemente zu identifizieren. Letztendlich kann die Visualisierungstechnik der Filter Dials verwendet werden, um einen Überblick über den Umfang der Teilmengen von Datensammlungen zu erlangen, die verschiedenen Kombinationen von Filterkriterien entsprechen [HE14].

Kapitel 4 befasst sich mit Visualisierungskonzepten, welche Anfragen für spezielle Anwendungsgebiete visualisieren könnten. Zunächst wird hier

[1] Die in dieser Einleitung referenzierten publizierten Arbeiten enthalten jeweils Grundlagen der genannten Aspekte. Die entsprechenden Kapitel in dieser Arbeit enthalten jedoch größtenteils Informationen, die über diese Veröffentlichungen hinausgehen.

Aspect Grid vorgestellt, welches im speziellen Fall der Kundenrezensionen dabei helfen kann, einen Überblick über die aggregierten Bewertungen pro Aspekt zu gewinnen [HHJ+14]. Außerdem wird auf VESPa eingegangen. Dabei handelt es sich um eine Anfragevisualisierung, welche die Modellierung und das Erkennen von Mustern in spatiotemporalen Kontexten erlaubt [KHH+15].

Abschließend wird in Kapitel 5 als zusätzlicher Beitrag dieser Arbeit aufgezeigt, wie sich die vorgestellten Visualisierungstechniken für Anfragen miteinander kombinieren lassen: Abschnitt 5.1 stellt die Konzepte einander gegenüber. Abschnitt 5.2 zeigt auf, wie sich die Konzepte gegenseitig ergänzen können. Abschnitt 5.3 definiert letztendlich ein generisches Modell, mit welchem Anfragevisualisierungen für graphbasierte Datenstrukturen miteinander verknüpft werden können.

Über diese Beiträge hinaus wurden im Verlauf dieses Promotionsvorhabens auch einige weitere Arbeiten durchgeführt, die über das Themenspektrum dieses Dokuments hinausgehen. Eine vollständige Auflistung der veröffentlichten und anderweitig betreuten Arbeiten inklusive Angaben zum jeweiligen Typ der Veröffentlichung befindet sich in Anhang A.

1.3 Begriffsdefinitionen

Im Zusammenhang mit dem Thema dieser Arbeit werden bestimmte Konzepte immer wieder aufgegriffen und erwähnt. Einige davon liegen nah beieinander und oft hat sich kein einheitlicher Benennungsstandard herausgebildet[2]. Deshalb sollen in diesem Abschnitt die wichtigsten Konzepte voneinander abgegrenzt und mit eindeutigen Benennungen assoziiert werden, die im Rest des Dokuments einheitlich verwendet werden.

Datenmenge

Eine *Datenmenge* ist eine beliebige Ansammlung von Daten. Die Daten sind entweder in ihrem ursprünglichen Zustand, bereits weiterverarbeitet oder gefiltert, oder schon in ihren angedachten Zielzustand überführt.

[2]Hinzu kommt, dass diese Arbeit auf Deutsch verfasst ist, während hauptsächlich aus englischsprachiger Literatur zitiert wird. Somit existieren einerseits zum Teil keine griffigen allgemein akzeptierten deutschen Begriffe, andererseits können sich schon allein durch die Übersetzung von Ausdrücken Mehrdeutigkeiten ergeben. So sind beispielsweise im Englischen ein *dataset* und ein *record* zwei sehr unterschiedliche Dinge, beide lassen sich jedoch auf Deutsch mit *Datensatz* übersetzen.

Datenelement (kurz: Element)

Im Kontext dieser Arbeit haben Daten in Datenmengen im Allgemeinen eine irgendwie geartete Struktur. Normalerweise lassen sie sich anhand irgendeines Kriteriums in *Elemente* unterteilen, die als getrennte Einheiten betrachtet werden können. (Dies schließt allerdings nicht aus, dass solche Elemente miteinander in Beziehung stehen – *Erde* und *Mars*[3] sind zwei separate Elemente. Sie können einzeln betrachtet werden. Dennoch besteht eine Nachbarschaftsbeziehung zwischen ihnen.)

Datensammlung

Als *Datensammlung* wird in dieser Arbeit eine von einer Organisation oder von Einzelpersonen zur Verfügung gestellte Datenmenge bezeichnet. In der Literatur wird hier oft von *dataset* gesprochen [ABK+07; GDH08; HHR+09]. Dies lässt sich aber einerseits nicht eindeutig übersetzen, andererseits ist es sehr allgemein und könnte genauso gut für Daten stehen, die bereits im Rahmen eines Filtervorgangs weiterverarbeitet wurden.

Ebenso werden Datensammlungen häufig über ihre physische Quelle definiert. Zum Beispiel kann ein SPARQL-Endpunkt oder ein Webservice angegeben sein, über den sie abgerufen werden können [GDH+09; DVF+10; KCL+15]. Eine Datensammlung als „Endpunkt“ zu bezeichnen, ist in dieser Arbeit allerdings nicht zielführend. Im Folgenden werden auch Datensammlungen verwendet, die nicht mittels Anfragen an eine bestimmte Quelle abgerufen werden können, und somit keinem konkreten Endpunkt entsprechen. Statt dessen können sie als Paket heruntergeladen werden. Das heruntergeladene Paket muss auf einem beliebigen eigenen Endpunkt verfügbar gemacht werden [Dea01].

Schema

Schema ist ein allgemeiner Begriff für die Definition von Datenstrukturen in einer Datensammlung. Ein Schema beschreibt Arten von Datenelementen in einer Datensammlung, ihre Attribute und die Zusammenhänge zwischen den Datenelementen. Je nach Kontext haben sich unterschiedliche Arten von Schemata herausgebildet. Datenbanken verwenden Datenbankschemata, XML-Dokumente können optional auf XML-Schemata basieren. Im Bereich des Semantic Web nehmen Ontologien die Funktion von Schemata ein. Dabei können jedoch prinzipiell auch Daten abgelegt werden, die nicht der Struktur aus der Ontologie entsprechen, oder mehrere Ontologien kombiniert werden.

[3] Hierbei sind Planeten unseres Sonnensystems gemeint.

Filtern und suchen

Die Begriffe *filtern* und *suchen* bezeichnen beide die Einschränkung einer Datenmenge mit dem Ziel, im aktuellen Kontext irrelevante Informationen auszuschließen. Dabei wird eher von einer *Filterung* gesprochen, wenn potenziell nach der Operation noch große Datenmengen übrig bleiben beziehungsweise nur eine geringe oder sogar überhaupt keine Reduktion der ursprünglichen Datenmenge stattfindet [EST08; JS10]. Eine *Suche* impliziert andererseits die Extraktion einer Datenmenge, die tendenziell so klein ist, dass sie direkt überschaubar ist (und somit aus wenigen oder einzelnen Elementen besteht) [Sch94; BWW+05].

In der Literatur wird keine klare Abgrenzung zwischen den Begriffen getroffen. Je nach Anwendungsfall können unterschiedlich große Datenmengen gewünscht sein. Zum Teil werden auch beide Begriffe im Zusammenhang mit denselben Konzepten verwendet [YS93; Huo08; Sei11]. Aus diesen Gründen wird in dieser Arbeit nicht zwischen *filtern* und *suchen* unterschieden.

2 Verwandte Arbeiten

In diesem Kapitel sollen zunächst grundlegende Konzepte erklärt werden, auf die im Verlauf der Arbeit wiederholt zurückgegriffen wird (Abschnitt 2.1). Ferner wird ein weitreichender Überblick über existierende Ansätze zur visuellen Repräsentation von Suchanfragen aus den Bereichen der Datenbanksysteme und des Semantic Web gegeben. Dabei wird nach einer kurzen Beschreibung über formale Anfragesprachen im Abschnitt 2.2 zunächst auf vorgestellte Visualisierungskonzepte eingegangen (Abschnitt 2.3). Anschließend werden Anwendungsfälle und Einsatzzwecke dieser Techniken diskutiert (Abschnitt 2.4). Die Analyse der existierenden Ansätze beschränkt sich auf die Abfrageeigenschaften. Repräsentationen von Änderungsanweisungen für die Datenmenge, wie sie in einzelnen Arbeiten enthalten sind [SBM+93; SCT91; DC96; BIJ01], werden hier nicht verfolgt.

Ebenso werden vorrangig die visuellen Aspekte beleuchtet. Das bedeutet, es wird auf Visualisierungen von Anfragen eingegangen sowie auf Interaktionen, die direkt in der Visualisierung stattfinden. Ansätze, welche natürlichsprachliche Formulierungen von Suchanfragen ermöglichen [Kap84; BE06], werden hier nicht betrachtet. Ebenso werden keine Konzepte untersucht, die die Eingabe über nichtvisuelle Kanäle unterstützen, beispielsweise per Spracheingabe, da sich derartige Eingaben auch auf visuelle Eingabeelemente neben der Visualisierung abbilden lassen sollten und somit aus Visualisierungssicht keine konzeptuellen Alleinstellungsmerkmale darstellen [KS90].

2.1 Grundlagen

Daten können in verschiedensten Formen vorliegen. Unstrukturierte Daten sind beispielsweise Fließtexte. Demgegenüber stehen strukturierte Daten, welche sich aus diskreten Elementen und einzeln abrufbaren Attributen zusammensetzen. Eine direkte Suche, beispielsweise eine Volltextsuche [Tun06], kann prinzipiell auf beliebigen Formaten stattfinden. Eine facettierte Suche hingegen benötigt strukturierte Daten. Diese strukturierten Daten lassen sich zum Teil aus unstrukturierten Daten extrahieren, wenn geeignete Umwandlungsregeln vorliegen beziehungsweise die Daten entsprechend angereichert werden. Als Beispiel wären hier Bildersammlungen zu nennen, aus deren Einträgen Metadaten (Größe, Farbtiefe und Ähnliches)

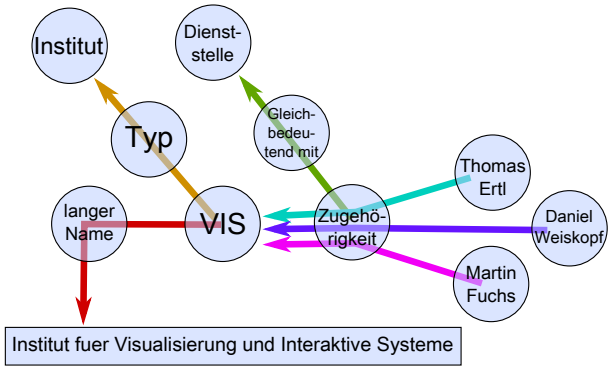


Abbildung 2.1: Ein beispielhafter RDF-Graph, welcher diverse Informationen über das Institut für Visualisierung und Interaktive Systeme in Form von Tripeln enthält. Derselbe Graph ist im Quelltextbeispiel 2.1 textuell dargestellt.

extrahiert werden können [YSL+03]. Auch das DBpedia-Projekt (siehe unten, Abschnitt 2.5.1) sammelt aus Wikipedia-Artikeln bestimmte Informationen, um sie in eine strukturierte Form zu bringen [LIJ+15].

Für die Speicherung von strukturierten Informationen sind diverse Formate verbreitet. Eine häufig gebrauchte Möglichkeit für diese Speicherung besteht in relationalen Datenbanken. Diese setzen im Allgemeinen ein festes Schema voraus, welches Tabellen mit bestimmten Spalten definiert sowie Bezüge zwischen diesen Tabellen definiert. Normalerweise ist es ohne Kenntnis des Schemas aber nicht möglich, Daten abzurufen.

Objektorientierte Datenbanken verwalten an Stelle von verknüpften Tabellenzeilen Objekte im Sinn der objektorientierten Programmierung. Durch Verknüpfungen zwischen Objekten entsteht ein Objektgraph. Eng damit verwandt ist das Modell RDF (Resource Description Framework), welches eine Struktur für Metadaten von strukturierten Informationen im Sinn des Semantic Web definiert [CDN+14]. Wie in Abbildung 2.1 gezeigt, wird die Graphstruktur dabei zu reinen Tripeln abstrahiert. Diese sind auch in der textuellen Repräsentation im Quelltextbeispiel 2.1 erkennbar. Die Bestandteile dieser Tripel werden mit den Begriffen Subjekt, Prädikat und Objekt im grammatikalischen Sinn bezeichnet. Alle Bestandteile werden entweder durch IRIs[1] [DS05] identifiziert oder, im Fall des Objekts eines Tripels, alternativ als Literal ausgedrückt. Ein Literal kann zum Beispiel eine Zeichenkette oder ein Zahlenwert sein.

Auf RDF aufbauende Standards wie RDFS (RDF Schema) [GB14] und OWL (Web Ontology Language) [DSB+04] erweitern das Metamodell. Durch sie können komplexe Klassenbeziehungen innerhalb der RDF-

[1]IRI: Internationalized Resource Identifier, eine internationalisierte Variante von URIs.

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .

ds:VIS a ds:Institute .
ds:VIS ds:longName "Institut fuer Visualisierung und
    Interaktive Systeme" .
ds:Thomas_Ertl ds:affiliation ds:VIS .
ds:Daniel_Weiskopf ds:affiliation ds:VIS .
ds:Martin_Fuchs ds:affiliation ds:VIS .
ds:affiliation owl:sameAs ds:Bureau .
```

Quelltextbeispiel 2.1: Der RDF-Graph aus Abbildung 2.1, ausgedrückt in der TURTLE-Notation [PCL14]. (Ein Ontologiepräfix `owl` ist definiert, während ein weiteres derartiges Kürzel `ds` als Identifikator eines domänenspezifischen Schemas angenommen wird.)

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix this: <http://example#> .

<http://example> a owl:Ontology ;
    owl:versionInfo "1.0" ;
    owl:versionIRI <http://example/1.0> .

this:Employee a owl:Class .
this:Company a owl:Class .

this:name a owl:DatatypeProperty ;
    rdfs:domain this:Company .

this:worksFor a owl:ObjectProperty ;
    rdfs:domain this:Employee ;
    rdfs:range this:Company .

this:superiorOf a owl:ObjectProperty ;
    rdfs:domain this:Employee ;
    rdfs:range this:Employee .
```

Quelltextbeispiel 2.2: Eine einfache OWL-Ontologie, ausgedrückt in der TURTLE-Notation [PCL14].

Graphen definiert werden. Auf diese Weise wird es möglich, wiederverwendbare Ontologien zu definieren. Eine solche Ontologie ist im Quelltextbeispiel 2.2 zu sehen und in Abbildung 2.2 visuell dargestellt.

Auf der Verwendung globaler (nicht Datensammlungs-spezifischer) IRIs basiert die Idee der *Linked Data*. Unabhängig voneinander aufgebaute und vorgehaltene Datensammlungen können übereinstimmende IRIs für diesel-

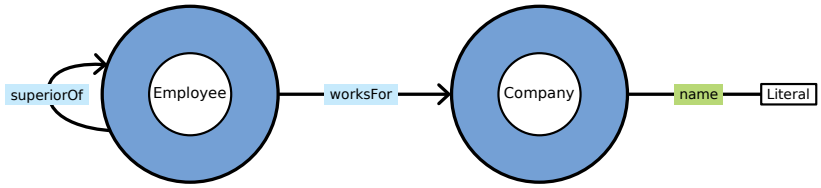


Abbildung 2.2: Die kleine Ontologie aus Quelltextbeispiel 2.2, visualisiert mit Version 1 der Notation VOWL [NL13a; NHL13; NL13b].

ben Konzepte verwenden. So können an sich unabhängige Datenquellen verknüpft und gemeinsam verwendet werden [BHB09].

Zu diesem Konzept kommt die Unabhängigkeit von einem festen Schema hinzu. Die Metainformationen, welche die Ontologie definieren, sind gemeinsam mit den Daten im selben Format abrufbar. Diese Metainformationen nehmen also keine konzeptuell getrennte Metaebene ein. Statt dessen bildet alles zusammen einen großen RDF-Graph. In ihm werden Informationen als der *TBox* („Terminological Box“, Metainformationen, Konzeptdefinitionen und Zusammenhänge zwischen Konzepten) oder der *ABox* („Assertional Box“, Datenelemente, die auf Grundlage der Konzepte definiert werden, sowie konkrete Zusammenhänge zwischen diesen Datenelementen) zugehörig bezeichnet.

Wegen dieser beiden Faktoren eignen sich RDF-basierte Daten gut für standardisierte Schnittstellen, welche domänen- und anwendungsfallunabhängig zum Abruf der Daten eingesetzt werden können.

2.2 Formale Anfragesprachen

Einleitend sollen hier einige weit verbreitete Beispiele textueller Anfragesprachen kurz umrissen werden. Dies dient als Grundlage für spätere Abschnitte, da visuelle Anfragekonzepte häufig auf Definitionen aus textuellen Anfragesprachen zurückgreifen.

2.2.1 SQL

SQL ist eine textuelle Anfragesprache für relationale Datenbanken [ISO99]. In der Sprache werden sowohl Einschränkungen für die Suche festgelegt als auch die Form der Ergebnisse angegeben. Bezüge auf konkrete Felder und Tabellen der Datenbank werden über deren Namen repräsentiert, welche aus einem separaten Datenbankschema zu entnehmen sind. Unbekannte beziehungsweise gesuchte Werte werden durch Platzhalter ausgedrückt, die auch mehrfach in der Anfrage verwendet werden können (Quelltextbeispiel 2.3).

```
SELECT P.firstName, P.lastName, P.age
FROM Person P, Company C, Person O
WHERE P.company = C.name
  AND C.owner = O.id
  AND P.lastName = O.lastName
  AND P.age < 60
```

Quelltextbeispiel 2.3: Eine einfache SQL-Anfrage. Sie sucht alle Mitarbeiter eines Unternehmens mit dem selben Nachnamen wie der Firmeneigentümer, die weniger als 60 Jahre alt sind.

Für die vorliegende Arbeit ist vorrangig relevant, dass die Anfrage grundlegend in mehrere Abschnitte unterteilt ist. Die Form der Ergebnisse wird als Aufzählung von Spaltennamen (hinter `SELECT`) für die Ergebnistabelle angegeben. Der Abschnitt nach `FROM` in der Anfrage gibt die verwendeten Tabellen an. Die Einschränkungen für die Abfrage befinden sich davon getrennt im mit `WHERE` beginnenden separaten Abschnitt. Letztendlich können noch einige weitere Abschnitte hinzugefügt werden. Diese beeinflussen wiederum durch Sortierung und Begrenzung der Ergebniszahl die Ergebnisliste.

2.2.2 SPARQL

Bei SPARQL handelt es sich um eine Anfragesprache, die für RDF-Graphen und die Suche von Daten im Semantic Web entwickelt wurde [The13]. Ihre im Quelltextbeispiel 2.4 gezeigte Syntax ist sowohl an SQL als auch an die RDF-Graphbeschreibungssprache Turtle [PCL14] angelehnt.

Das für die vorliegende Arbeit relevante Grundprinzip von SPARQL ist die Beschreibung eines Teilgraphen aus RDF-Tripeln. Einzelne Bestandteile dieser Tripel können in SPARQL durch Variablen ersetzt werden (im Quelltextbeispiel `?docTitle`, `?doc` etc.). Diese fungieren als Platzhalter für mögliche Ergebnisse. Suchergebnisse sind dann, vereinfacht ausgedrückt, sämtliche Ressourcen, die sich für eine der Variablen einsetzen lassen, ohne durch den restlichen Teilgraphen ausgedrückte Gegebenheiten im Graphen zu verletzen.

2.2.3 XPath

XPath ist eine Sprache zur Filterung von XML-Dokumenten [DC99]. Da XML-Dokumente hierarchische Strukturen darstellen, werden in XPath Pfade ausgedrückt, unter denen sich die gewünschten Informationen in der

```
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dc-terms: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT DISTINCT ?docTitle
WHERE {
  ?doc a swrc:InProceedings ;
       dc:title ?docTitle ;
       dc:creator ?author ;
       dc-terms:issued ?year .
  FILTER(?year >= "2012"^^xsd:gYear) .
  ?author foaf:homepage ?website .
  FILTER(strStarts(str(?website),
                   "http://www.vis.uni-stuttgart.de/")) .
}
LIMIT 20
```

Quelltextbeispiel 2.4: Eine SPARQL-Anfrage. Sie sucht in FACETED DBLP die Titel von 20 Konferenzbeiträgen, die seit 2012 von Mitarbeitern des Instituts für Visualisierung und Interaktive Systeme der Universität Stuttgart publiziert wurden.

Hierarchie finden lassen. Diese Pfade können auch flexible Teile enthalten, um beispielsweise eine variable Anzahl von Verschachtelungsebenen zu überbrücken. Ein kurzer XPATH-Ausdruck ist als Quelltextbeispiel 2.5 abgebildet.

```
/company:staff/company:employee[@yearsActive > 10]/company:award
```

Quelltextbeispiel 2.5: Ein beispielhafter XPATH-Ausdruck, welcher in einem XML-Dokument Auszeichnungen von Mitarbeitern sucht, die bereits seit mehr als zehn Jahren bei einem Unternehmen beschäftigt sind.

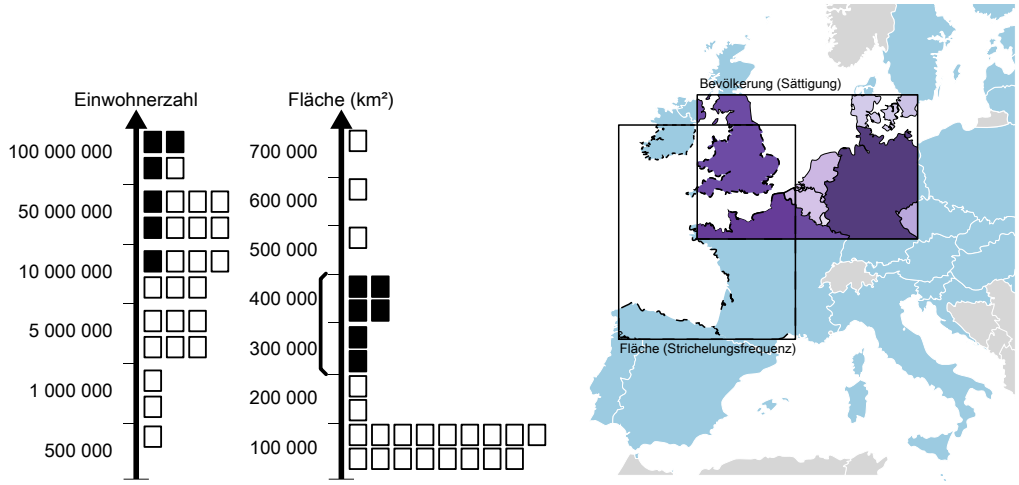
Während die Struktur von XML-Dokumenten auf Bäume beschränkt ist, lassen sich einige der Prinzipien zur Repräsentation von Pfaden auch auf Objektgraphen im Allgemeinen anwenden. Dabei werden Pfadmuster mit regulären Ausdrücken in Verbindung gebracht [MW95; GS02; Woo12]. Ähnliche Pfadmuster für Objektgraphen sind in Form von Property Paths auch in der Sprache SPARQL enthalten.

2.3 Visuelle Filtertechniken

Textuelle Anfragesprachen wie die eben gezeigten sind präzise. Allerdings erfordern sie Kenntnis von der jeweiligen Syntax, was bei untrainierten Benutzern nicht vorausgesetzt werden kann. Ebenso kann bei komplexeren Anfragen schnell der Überblick verloren gehen. Um diese Probleme zu umgehen, bietet sich eine visuelle Darstellung der Suchparameter an.

In vielen Visualisierungskonzepten, die unter den Stichworten *Filterung* oder *Suche* präsentiert wurden, muss die eigentliche Suche noch vom Benutzer durchgeführt werden. Durch die Visualisierung werden die Daten nur in einer bestimmten Weise dargestellt, die sich besonders für visuelle Vergleiche eignet oder die Muster besonders gut erkennen lässt [LRP95; KK96; JHS02]. Als Beispiel ist die Visualisierung Attribute Explorer [TSW+94] in Abbildung 2.3a dargestellt. Interaktion wird hierbei zum Teil durch eine mehrstufige Exploration der Daten ermöglicht, bei der bestimmte Merkmale in einer Ansicht erkannt werden und die betreffenden Datenelemente gezielt in einer modifizierten Ansicht betrachtet werden können. Häufig kommt dabei eine Linsenmetapher zum Einsatz, wie sie auch in Abbildung 2.3b gezeigt wird [BSP+93; RC94; KTW+13].

Alternativ werden ansonsten allenfalls bestimmte interessante Werte automatisch oder über nichtvisuelle Einstellungen definiert und ermittelt. Diese können dann in der Visualisierung der gesamten, ungefilterten Daten-



(a) Attribute Explorer [TSW+94] **(b)** Magic Lenses [BSP+93]

Abbildung 2.3: Beispiele für Visualisierungen zum Hervorheben einzelner Aspekte. In den gezeigten Beispielen werden Einwohnerzahl von Fläche von EU-Ländern im Jahr 2015 auf grafische Weise ausgedrückt.

menge hervorgehoben werden [KK94; Hea95; LTG+11; EW13]. Soll nicht die gesamte Datenmenge dargestellt werden, konzentrieren sich manche Visualisierungen auch auf das unmittelbare Umfeld einer fokussierten Ressource. Von dieser fokussierten Ressource aus kann der Benutzer zu anderen Elementen in der Datenmenge navigieren [CDR+99; ACK04; MG14; CDD+04]. Diese Vorgehensweise wird in einer Klassifikation nach Batini et al. als *Inside-Outside-Strategie* bezeichnet [BCC+91].

All diese Ansätze gehören der Kategorie der visuellen Anfragesysteme (VQS, visual query system) an [CCL+97]. Neben diesen Ansätzen, die Benutzern das selbständige Suchen in der Gesamtdatenmenge erleichtern, existieren jedoch auch einige Techniken, welche die Suchanfragen oder -parameter an sich visualisieren. Dies geschieht, indem ein VQS um eine visuelle Anfragesprache (VQL, visual query language) herum aufgebaut wird [CSA93].

Im Allgemeinen gehen VQLs darüber hinaus, lediglich die Eingabe der textuellen Filterbedingungen zu vereinfachen, wie dies von vielen Ansätzen unterstützt wird [STT91; DGJ+95; JNS00; BSS+; AMH10; HVG14]. Ebenso beschränken VQLs sich nicht auf Eingabeformulare, die im Prinzip die Syntax einer textuellen Anfragesprache nachbilden [KM89; BE06; ABK+07], wie es für VQSs aus der Gruppe der formularbasierten VQSs der Fall ist [CCL+97]. Als Beispiel wird in Abbildung 2.4 die logische Verknüpfung von Teilanfragen in VisualMOQL [OÖX+99] gezeigt.

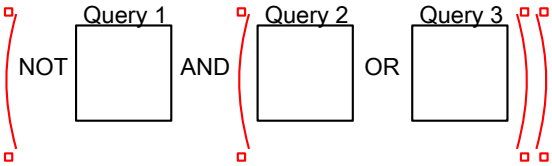


Abbildung 2.4: Logische Verknüpfung von Teilanfragen im „Query Canvas“ von VisualMOQL [OÖX+99]. Die visuelle Darstellung ähnelt stark einer formalen textuellen Repräsentation.

Gleichzeitig geben VQLs dennoch eine gewisse Struktur beziehungsweise einen Satz von Notationselementen vor. Es wird also keine reine Ähnlichkeitssuche auf Grundlage einer unbeschränkten Zeichnung vorgenommen [SF10].

2.3.1 Node-Link-basierte Ansätze

Ansätze, die auf Node-Link-Diagrammen aufbauen, setzen die Grundbestandteile solcher Diagramme auf unterschiedliche Arten ein. Die Knoten repräsentieren konkrete Einschränkungen für die gesuchten Elemente oder zeigen Daten an. Genau wie Knoten drücken die Kanten zum Teil selber

Einschränkungen aus, zu denen sich in der Datenmenge passende Elemente finden lassen. Zum Teil können die Kanten auch logische Zusammenhänge zwischen den als Knoten dargestellten Einschränkungen abbilden, die sich so nicht in der Datenmenge wiederfinden lassen. Im Folgenden werden unterschiedliche Ansätze besprochen, welche jeweils eines der beiden Modelle anwenden.

2.3.1.1 Node-Link-basierte Repräsentation von Objektgraphen

Werden Objektgraphen direkt mit Node-Link-Diagrammen repräsentiert, so wird der eigentliche Inhalt des Objektgraphen sichtbar. Knoten entsprechen dabei im Allgemeinen den Objekten des Objektgraphen und Kanten repräsentieren die Relationen zwischen diesen Objekten [WMP+05; Pie07; MC08]. Alternativ werden die Relationen mitunter als eigenständige Knoten visualisiert. Dadurch ergibt sich insgesamt ein bipartiter Graph [FTH06].

Um mit Hilfe einer derartigen Visualisierung bestimmte Objekte oder Teilgraphen zu finden, kann die Erkennung der gesuchten Graphbestandteile erleichtert werden. NodeTrix stellt genau wie die oben genannten Ansätze den gesamten Objektgraphen dar. Es hebt allerdings darin enthaltene Cluster mit einem hohen Grad an internen Verknüpfungen in Form von Adjazenzmatrizen (siehe auch Abschnitt 2.3.2) hervor [HFM07]. Jambalaya bleibt im Prinzip bei der Darstellung des kompletten Graphen. Der Ansatz erlaubt aber das Betrachten unterschiedlicher Abschnitte des Graphen auf unterschiedlichen Detailgraden zur selben Zeit [SMS+01]. Ferner umfasst die Darstellung in Konzepten wie PivotGraph ebenfalls den gesamten Graphen. Das Layout wird so angepasst, dass einzelne Aspekte im Vordergrund stehen [Wat06]. Die Suchanfragen selber werden jedoch überhaupt nicht visualisiert, sondern nur die Ergebnisse.

Die Anfragevisualisierung von Catarci et al. zeigt nur einen kleinen Ausschnitt aus dem Objektgraphen beziehungsweise dem Domänenmodell an, nämlich nur den direkten Umkreis um einen fokussierten Knoten [CDD+04]. Der von Sayers vorgestellte Ansatz [Say04], Techniken wie TGVizTab [Ala03], OntoSphere [BBP05], ASK-GraphView [AHK06] und PGV [DKS07] funktionieren in dieser Hinsicht auf dieselbe Weise. Derartige Visualisierungen erlauben zumeist das vom aktuellen Knoten ausgehende Weiternavigieren durch den Graphen. Die Hierarchieansicht aus Tabulator ist ebenso mit dieser Art der Inhaltsvisualisierung verwandt [BCC+06]. Diese Technik lässt sich allerdings auch unterstützend für andere Visualisierungen, wie den nachfolgend beschriebenen, nutzen [SGJ+13].

Repräsentation der Suchanfragen Für graphbasierte Datenstrukturen scheint eine Node-Link-basierte Repräsentation der eigentlichen Suchan-

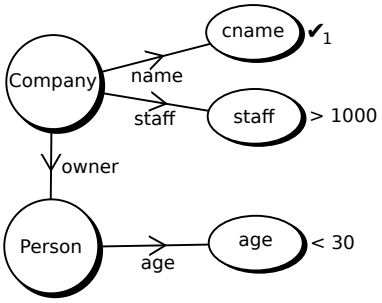
fragen intuitiv zu sein [SBM+93]. Aufbauend auf dem Graph-Modell von Catarci et al. [CSA93] wird dazu mindestens ein Knoten als Platzhalter für mögliche Suchergebnisse interpretiert, welche sich in den übrigen Graphkontext der Anfrage an Stelle des Platzhalters einfügen lassen. Dieses Funktionsprinzip ist direkt aus textuellen Anfragesprachen wie SPARQL (Abschnitt 2.2.2) übernommen.

Die in Abbildung 2.5a gezeigte Notation GQL geht beispielsweise von einer Kombination aus konstanten und variablen Knoten aus. Zwischen diesen Knoten werden Relationen durch Kanten ausgedrückt [PK95]. Einige weitere Konzepte verhalten sich diesbezüglich grundlegend genauso. Um die große Anzahl deutlich zu machen, sind diese im Folgenden aufgelistet – Erläuterungen zu etwaigen Unterschieden finden sich, thematisch sortiert, im Anschluss:

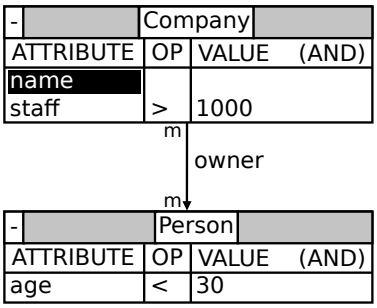
- GRAQULA [SBM+93], beispielhaft dargestellt in Abbildung 2.5b
- OQD [KM93], oben in Abbildung 1.1 sowie unten in Abbildung 2.8d gezeigt
- VQL [MK93], unten in Abbildung 2.9c gezeigt
- Cigales [CM94]
- MQuery [DC96]
- VOODOO [Feg99], beispielhaft dargestellt in Abbildung 2.5c
- XML-GL [CCD+99], unten in Abbildung 2.8b gezeigt
- qGraph [BIJ01]
- Teile von HyperFlow [DP05]
- GLOO [FH06], unten in Abbildung 2.9b gezeigt, und das darauf aufbauende OntoVQL [FH07]
- NITELIGHT [RSB+08; RS08; SRB+08], beispielhaft dargestellt in Abbildung 2.5d
- iSPARQL [Ope08]
- eine visuelle Sicherheitsrichtlinien-Anfragesprache [XSA08]
- LuposDate [GGS+09; GGS11]
- RDF-GL [HMF+10], beispielhaft dargestellt in Abbildung 2.5e

In einzelnen domänenspezifischen Konzepten sind nur geringe Variationen von Elementen vorgesehen. Kanten können beispielsweise ausschließlich Verbindungen zwischen Räumen in einem Gebäude beschreiben [LWL+13]. Ebenso können sie stets Beziehungen zwischen Molekülen repräsentieren, wie in GBLENDER [JBX+10] und seiner Erweiterung PRAGUE [JBC+12].

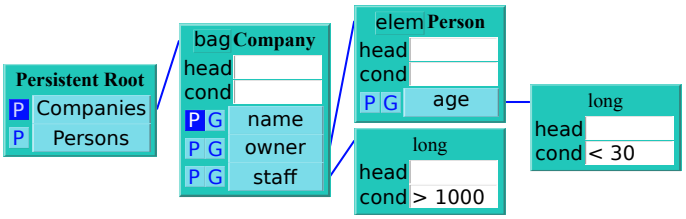
Die Anfragesprache GraphLog [CM90] und die darauf aufbauende Visualisierung Hy$^+$ [CM93] beinhalten zusätzlich eine Kantenart zum Ausdruck von transitiven Abschlüssen. Eine visuelle Anfragesprache namens G, welche die Definition derartiger transitiver Abschlüsse über bestimmte Kanten erlaubt, wurde zuvor bereits von Cruz et al. vorgeschlagen [CMW87].



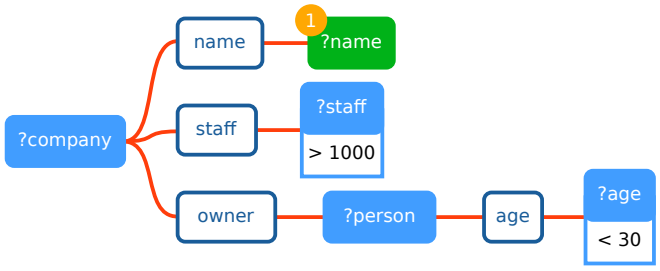
(a) GQL [PK95]



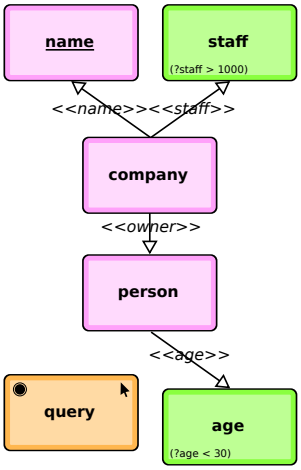
(b) GRAQULA [SBM+93]



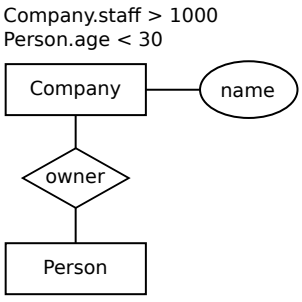
(c) VOODOO [Feg99]



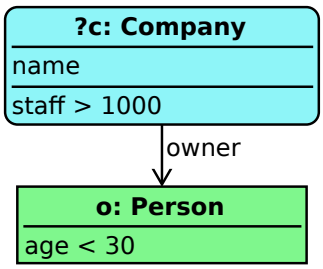
(d) NITELIGHT [RSB+08; RS08; SRB+08]



(e) RDF-GL [HMF+10]



(f) Notation von Campbell et al. [CEC87]



(g) VIZIQUER [BRZ09; ZB11]

Abbildung 2.5: Beispielhafte Auswahl von Objektgraph-basierten Visualisierungen für die Anfrage „Finde die Namen von Firmen mit mehr als eintausend Mitarbeitern, deren Besitzer unter dreißig Jahre alt ist.“

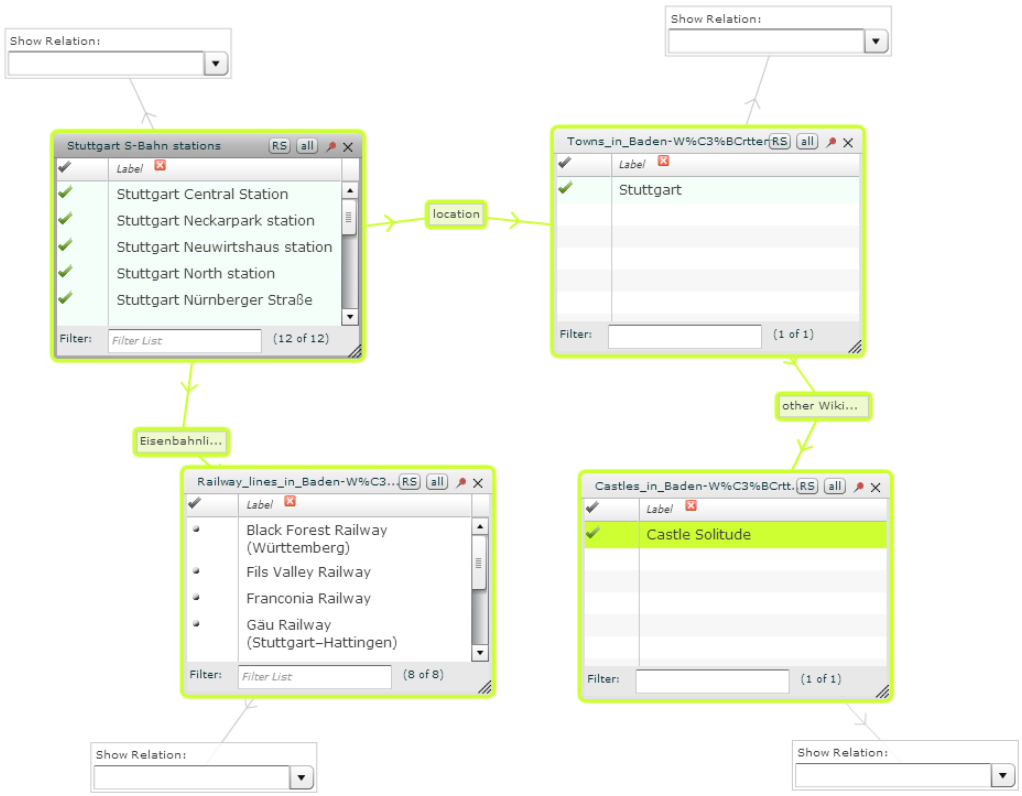


Abbildung 2.6: Screenshot von gFacet [HZL08], welcher die Abfrage von Eisenbahnlinien zeigt, die laut DBpedia (siehe unten, Abschnitt 2.5.1) thematische Bezüge zum Schloss Solitude haben.

Die (namenlose) Notation von Harth et al. fasst hingegen Tripel in Knoten zusammen, die bei übereinstimmenden Bestandteilen über Kanten verbunden werden [HKD06]. Sie ist unten in Abbildung 2.8c abgebildet. Optique-VQS stellt einen transformierten Graphen dar, in dem Knoten verdoppelt werden, um eine Baumstruktur zu erlangen [SGJ+13; SGJ+15]. Das in Abbildung 2.6 gezeigte gFacet stellt Beziehungen zwischen Datenelementen ebenso in Form eines Graphen dar. Es verwendet die Knoten des Graphen jedoch direkt zur Filterung nach Attributwerten [HZL08].

Node-Link-basierte Repräsentationen finden sich auch in Visualisierungskonzepten wieder, die direkt auf Schemadiagrammen aufbauen. QBD* geht beispielsweise direkt von der Darstellung eines Datenbankschemas aus [ACS90]. Dabei kam eine Benutzerstudie zu dem Schluss, dass diese Darstellungsform gegenüber einer textuellen SQL-Abfrage zur Vermeidung von Fehlern und zu einer höheren Eingabegeschwindigkeit beiträgt [CS95]. Campbell et al. haben eine weitere auf Schemadiagrammen aufbauende Anfragenotiation definiert [CEC87]. Sie ist in Abbildung 2.5f dargestellt. VCP nimmt ein baumförmiges Datenschema zur Grundlage und ermöglicht

das visuelle Eintragen von Umstrukturierungsanweisungen mit einzelnen Filtereinschränkungen in diese Baumstruktur [Koc06]. VKML zeigt das Entity-Relationship-Modell lediglich zur Auswahl von Attributen an. Währenddessen werden die eigentlichen Filtereinstellungen tabellarisch dargestellt [SCT91] (siehe Abschnitt 2.3.2). Vom Funktionsprinzip her ähnelt dies GFACET [HZL08]. Auch Erweiterungen gegenüber der herkömmlichen Entity-Relationship-Notation in Form von zusätzlichen Rahmen zur Abgrenzung von Anfragebestandteilen wurden vorgestellt [KG95]. Letztendlich erinnert SNAP visuell an Entity-Relationship-Diagramme, fügt aber einzelne Knoten- und Kantentypen hinzu [BH86].

Analog zu Entity-Relationship-Diagrammen können UML-Klassendiagramme als Grundlage für visuelle Suchanfragen dienen. Beispiele hierfür sind die Konzepte VTML [MC10] und das in Abbildung 2.5g gezeigte VIZIQUER [BRZ09; ZB11].

Eine leichte Abweichung zu den oben beschriebenen Konzepten ergibt sich in RELFINDER [HHL+09; HLS10]. An Stelle eines kompletten Anfragegraphen werden zwei oder mehr Datenelemente ausgewählt. RELFINDER sucht daraufhin sämtliche direkten und indirekten Verbindungen zwischen diesen Datenelementen in der Datensammlung und stellt den so ermittelten Subgraphen als Node-Link-Diagramm dar, wie in Abbildung 2.7 zu sehen ist.

Wertevergleiche In GRAQULA [SBM+93] können Knoten wie in Abbildung 2.8a gezeigt über Kanten verbunden werden, welche die Anwendung von Vergleichsoperatoren symbolisieren. Diese Vorgehensweise wurde auch von Chan beschrieben [Cha97] und wird ebenfalls in GQL unterstützt [PK95]. VOODOO erlaubt Ähnliches für die einzelnen Objektattribute, die in jedem Knoten dargestellt werden [Feg99]. PESTO wählt hingegen den Weg, Entitäten, deren Attribute verglichen werden, über zusätzliche Synchronisierungsknoten zu verbinden [CHM+96]. Die Auswahl an Operatoren ist dabei je nach Ausarbeitung und Fokus des Konzepts unterschiedlich. Neben den grundlegenden Operatoren zur Feststellung von Gleichheit und Ungleichheit sowie von Größenverhältnissen zwischen ordinalen Werten [MK93] werden teilweise Mengenoperationen [DP05] unterstützt. In einigen Konzepten werden derartige Operationen aber durch rein textuelle Terme dargestellt [SBM+93; MK93; CEC87]. Dies ist beispielsweise in Abbildung 2.5g anhand der VIZIQUER-Visualisierung [BRZ09; ZB11] zu sehen.

Manche Notationen beschränken sich darauf, Gleichheit durch die Verbindung über einen gemeinsamen Verbindungsknoten auszudrücken [MK93], wie beispielsweise das in Abbildung 2.8b gezeigte XML-GL [CCD+99]. Konzepte wie die in Abbildung 2.8c dargestellte

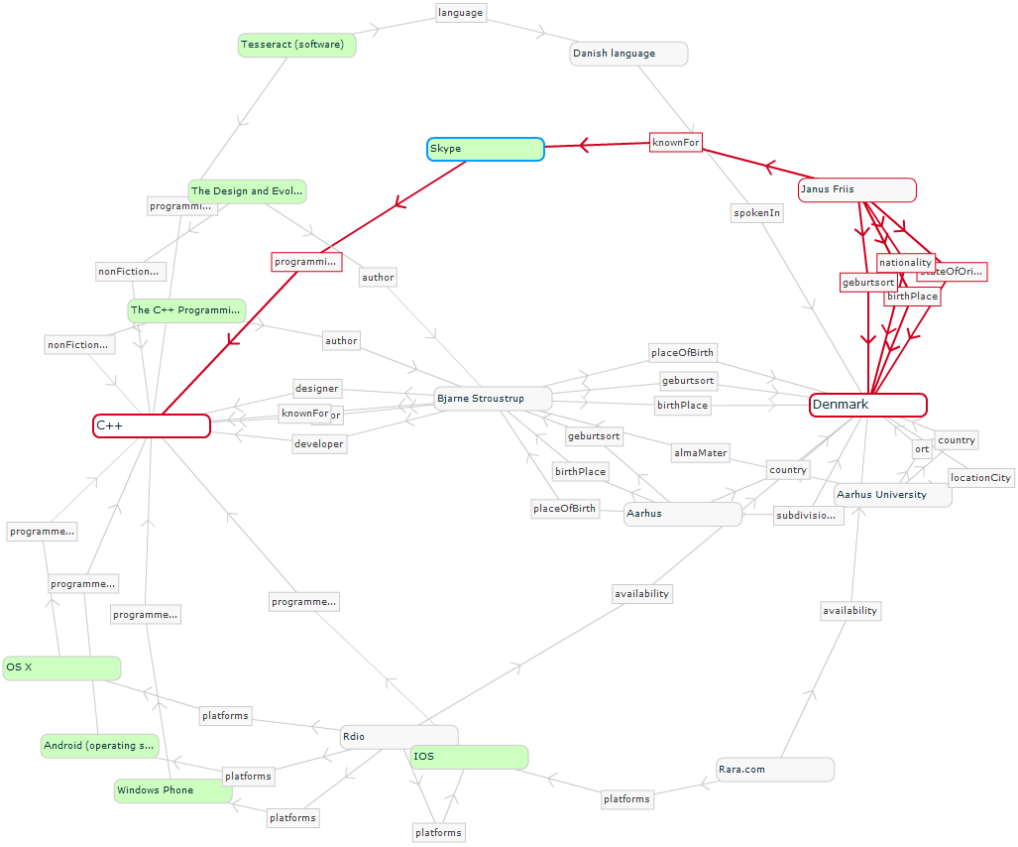
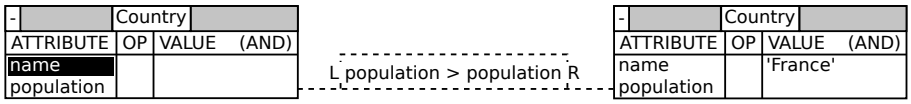


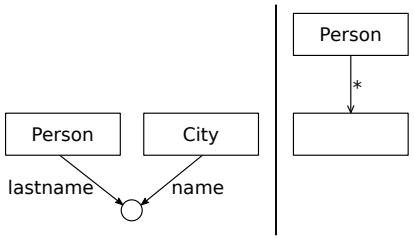
Abbildung 2.7: Screenshot von RelFinder [HHL+09; HLS10], wobei die in DBpedia (siehe unten, Abschnitt 2.5.1) bekannten Verbindungen zwischen der Programmiersprache C++ und Dänemark gesucht wurden. Die Verbindung über die Software Skype ist hervorgehoben.

Notation von Harth et al. stellen Gleichheit mittels einer Verbindungskante dar. Lediglich das in Abbildung 2.8d dargestellte OQD geht den Weg, die Gleichheit durch räumliche Überschneidung von Knoten als Venn-Diagramm darzustellen [KM93], wie auch in Abschnitt 2.3.3 beschrieben.

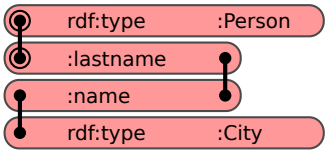
Negation Für die Negation können unterschiedlich mächtige Ansätze gefunden werden. GQL rahmt Subgraphen, die ausgeschlossen werden sollen, in Rechtecke ein [PK95]. Cigales [CM94], PESTO [CHM+96] und XML-GL [CCD+99] erlauben hingegen nur die Negation einzelner Knoten. In einzelnen Konzepten können zudem einzelne Relationen negiert werden [CM90; FH06]. Nur in einzelnen Fällen wird zwischen tatsächlicher und vermuteter Negation unterschieden [FH06]. Bei einer tatsächlichen Negation ist bekannt, dass die Negation einer Aussage gilt. Bei einer vermuteten Negation ist auf Grundlage einer Datensammlung lediglich unbekannt, ob



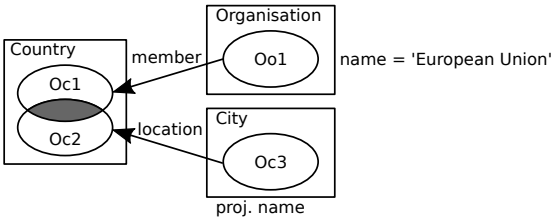
(a) „Finde die Namen von Ländern mit mehr Einwohnern als Frankreich.“ in GRAQULA [SBM+93].



(b) „Finde Personen, deren Nachname dem Namen einer Stadt entspricht.“ in XML-GL [CCD+99].



(c) Dieselbe Anfrage wie in Abbildung 2.8b in der Notation von Harth et al. [HKD06].



(d) „Finde Namen von Städten in Mitgliedsstaaten der EU.“ in OQD [KM93].

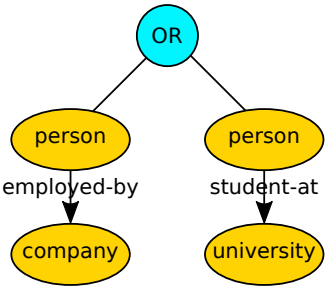
Abbildung 2.8: Repräsentation von Wertevergleichen in verschiedenen Visualisierungen.

eine Aussage zutrifft.

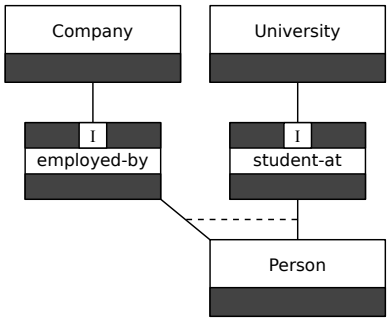
Disjunktionen Um Disjunktionen auszudrücken, werden zum Teil alternative Teilgraphen in der Anfragevisualisierung gezeigt. In NITELIGHT werden diese wie in Abbildung 2.9a gezeigt in Rechtecke eingerahmt [RSB+08]. Zusätzlich wird in GQL eine Negation der Disjunktion ermöglicht [PK95]. PESTO erlaubt die Definition alternativer Einschränkungen pro Knoten [CHM+96]. GLOO drückt Disjunktionen explizit über die in Abbildung 2.9b dargestellten Disjunktionsknoten aus. Diese können über spezielle Kanten mit anderen Knoten verbunden werden. Eine analoge Vorgehensweise zum Ausdruck von Konjunktionen ist ebenso erlaubt [FH06]. VQL verwendet die in Abbildung 2.9c abgebildeten Disjunktionskanten, welche alternative Kanten verbinden [MK93]. RDF-GL setzt unterschiedliche Arten von Disjunktionsknoten ein [HMF+10]. Die Darstellung mit Disjunktionskanten oder -knoten macht es erforderlich, dass die darüber verbundenen Graphen ansonsten strukturell disjunkt sind.

Aggregatfunktionen und berechnete Werte Aggregatfunktionen zur Filterung und andere berechnete Werte werden häufig nicht visuell aus-

(a) NITELIGHT [RSB+08; RS08; SRB+08]

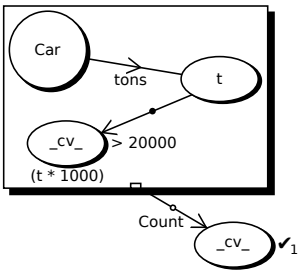


(b) GLOO [FH06]

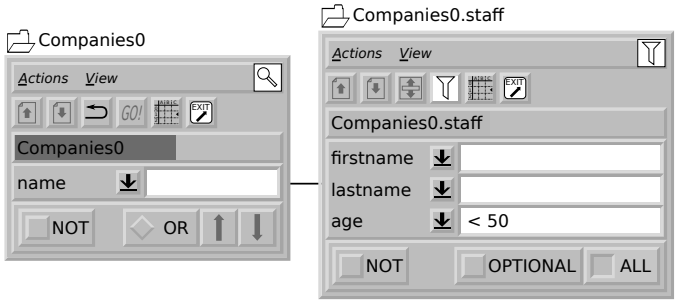


(c) VQL [MK93]

Abbildung 2.9: Disjunktive Anfragen in unterschiedlichen visuellen Notationen zur Suche nach Personen, die bei einem Unternehmen beschäftigt sind oder an einer Universität studieren.



(a) Zahl der Autos, die schwerer als 20000 kg sind, in GQL [PK95].



(b) Verwendung von PESTO [CHM+96], um Firmen zu finden, deren Mitarbeiter alle jünger als 50 Jahre sind.

Abbildung 2.10: Fortgeschrittene Funktionen in visuellen Objektgraph-basierten Anfragenotationen.

gedrückt. Statt dessen wird oft nur der jeweilige Funktionsname oder der mathematische Term dargestellt, der zur Berechnung verwendet wird [SBM+93; RSB+08; HMF+10]. Abbildung 2.10a zeigt dies beispielhaft für GQL [PK95]. Bestimmte Aggregatfunktionen, wie die Anzahl der erlaubten Ressourcen, die pro Ergebnis für einen bestimmten Platzhalter aus der Anfrage eingesetzt werden könnten, lassen sich auch auf andere Weise in die Visualisierung integrieren. Beispielsweise kann dies in Form von Kardinalitäten für grafische Elemente geschehen [SBM+93; BIJ01].

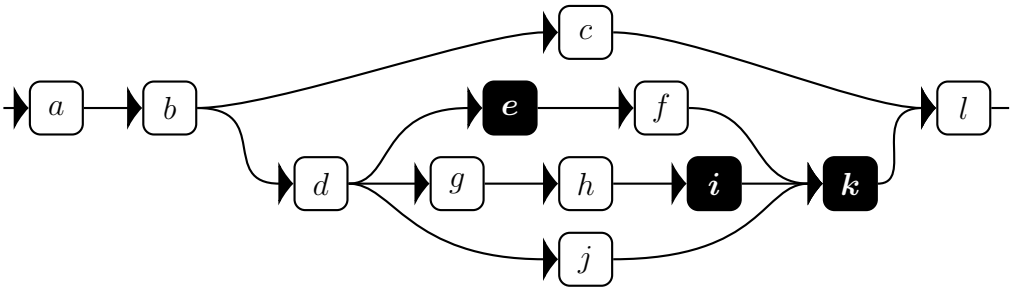
Vereinzelt findet sich zudem eine Unterstützung von Allquantoren. So erlaubt es das in Abbildung 2.10b gezeigte PESTO, bestimmte Knoten als Einschränkung für *alle* verbundenen Elemente desselben Typs festzulegen, statt nur die Existenz eines auf diese Weise verbundenen Elements zu fordern [CHM+96].

2.3.1.2 Flüsse, Rohre und Ströme

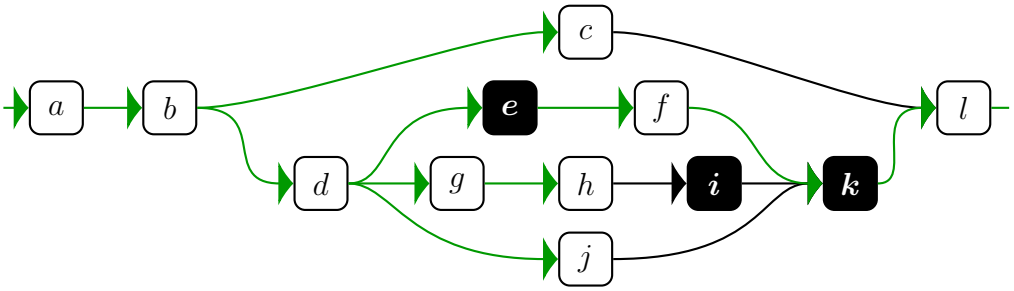
Ein alternativer Node-Link-basierter Ansatz besteht darin, statt Objektgraphmustern Filterkriterien und logische Verknüpfungen zwischen diesen darzustellen. Dazu wurde ursprünglich unter dem Namen Filter/Flow ein entsprechendes Konzept vorgestellt [Shn91; YS93; Shn94]. Die Kanten dienen zur logischen Verknüpfung der Filterkriterien. Dabei wird kein boolescher Ausdrucksbaum nachgebildet. Vielmehr wird über die Kanten ein Flussgraph aufgebaut. Jeder Pfad durch den Flussgraphen stellt eine mögliche Kombination von Filterkriterien dar. Alle Kriterien aus dieser Kombination müssen erfüllt werden, damit ein Datenelement Eingang in eine bestimmte Ergebnismenge am Ende jenes Pfades erhält. Analog zu elektrischen Schaltungen sind Filter in parallelen Flüssen somit zu Disjunktionen zusammengefügt. Filter, die sequenziell aufeinander folgen, bilden hingegen eine Konjunktion (Abbildung 2.11a). Benutzer können in dieser Darstellung auf einfache Weise nachverfolgen [YS93], welche Datenelemente die Filtereinschränkungen erfüllen und somit in eine Ergebnismenge am Ende des Graphen gelangen (Abbildung 2.11b).

Entsprechend dem Namen Filter/Flow wurde die Darstellung anhand einer Flussmetapher erklärt, bei der Daten wie Flüsse durch Filter fließen. Die Filter können Teile der Daten ausschließen. Dadurch nimmt die Dicke der Flüsse ab. Auf diese Weise entsteht ein ungefährer Eindruck vom Umfang der Ergebnismenge nach jedem Filter (Abbildung 2.11c). Benutzer können somit die Anfragen iterativ verfeinern, wenn die geeigneten Suchparameter noch nicht von Anfang an klar sind [Mar06]. Eine Alternative zur variablen Flussdicke stellt eine Anzeige der Größe der an jedem Knoten anliegenden Datenmenge durch einen „Ladebalken“ dar [Bos15, 24f.].

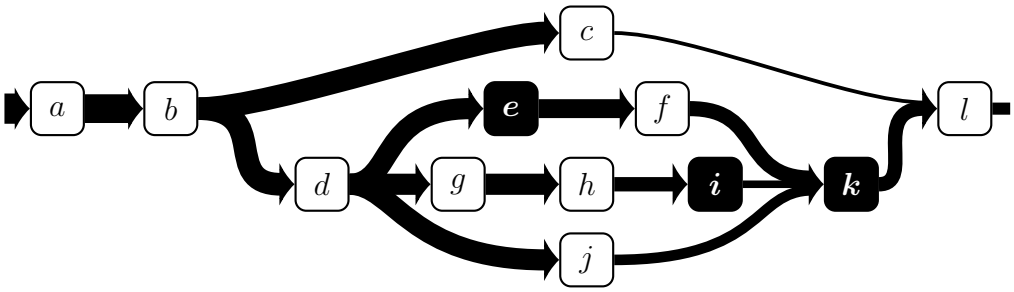
Mit diesem Modell sollte eine intuitive Verständlichkeit boolescher Verknüpfungen durch Benutzer erreicht werden, ohne dass diese in der Lage sein müssen, textuelle boolesche Ausdrücke verstehen oder formulieren zu können. Die bessere Verständlichkeit dieser grafischen Notation im Vergleich zu booleschen Ausdrücken in SQL-Syntax (siehe Abschnitt 2.2.1) wurde auch in einer Benutzerstudie belegt [YS93].



(a) Grundlegendes Funktionsprinzip eines Filter/Flow-Graphen: Die Knoten stellen Filter dar, sodass etwaige aus l austretende Datenelemente der Bedingung $a \wedge b \wedge (c \vee (d \wedge ((\neg e \wedge f) \vee (g \wedge h \wedge \neg i) \vee j) \wedge \neg k)) \wedge l$ genügen. Filter, welche die Negation einer Bedingung erfordern, sind farblich invertiert dargestellt.



(b) Filterung in dem Filter/Flow-Graphen aus Abbildung 2.11a: Im Beispiel wird ein Datenelement angenommen, welches genau die Bedingungen a, b, d, f, g, l (und somit auch $\neg c, \neg e, \neg h, \neg i, \neg j, \neg k$) erfüllt. Die Pfade, die dieses Datenelement durchlaufen kann, sind grün hervorgehoben.



(c) Die Dicke der Flüsse in einem Filter/Flow-Graphen vermittelt einen Eindruck vom Umfang der Datenmenge nach jedem Filter. Im abgebildeten Beispiel erfüllen beträchtliche Teile der Datenmenge nicht die Bedingungen c und j, während anhand der Bedingung g praktisch überhaupt keine Datenelemente aussortiert werden.

Abbildung 2.11: Grundlagen des Filter/Flow-Konzepts [Shn91; YS93; Shn94].

Eine Reihe von Arbeiten erweiterte das Filter/Flow-Konzept in unterschiedlicher Hinsicht[2]. Eine der häufigsten oberflächlichen Veränderungen in der grafischen Darstellung bestand darin, dass nicht wie im ursprünglichen Konzept sämtliche Flüsse in derselben Richtung verlaufen. Statt dessen können Knoten in mehreren Erweiterungen frei platziert werden [HSM+06; EST08; SHT+07; TIC09; JGZ+11]. Das Merkmal, die Flussbreite in Abhängigkeit von der Größe der Ergebnismenge zu variieren, wurde nicht immer aufgegriffen [SHT+07]. Teilweise wurde es durch explizite Zwischenergebnislisten ersetzt [MAG+04; HSM+06; JE13].

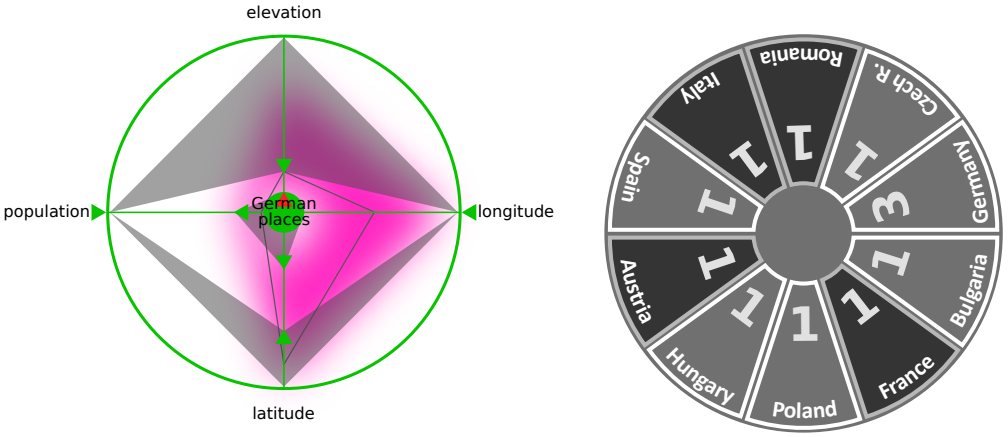
In den ursprünglichen Arbeiten zum Thema Filter/Flow wurde keine explizite Einschränkung gemacht, welche Filterfunktionen in Filterknoten zu finden sind. Die gezeigten Beispiele beschränkten sich auf Auswahllisten [YS93] und Zahlenbereiche [Shn94]. Erweiterungen wie DataMeadow [EST08] (Abbildung 2.12a) und FacetStreams [JGZ+11] (Abbildung 2.12b) verwendeten weitergehende Filtertypen. Manche davon waren für bestimmte Anwendungsgebiete spezialisiert, wie zum Beispiel Patentsuche [KBG+09] oder wie in Abbildung 2.13 gezeigt die Filterung nach geografischen Beziehungen [MAG+04]. Analog zu manchen Objektgraph-basierten Verfahren [HZL08] können daher für Flussgraphen zusätzliche Filterknoten für bestimmte Datentypen und Anwendungszwecke vorgesehen werden.

In der Verwendung für sogenannte Mashup-Systeme wie Yahoo! Pipes wurde der ursprüngliche Einsatzzweck des Filter/Flow-Konzepts teilweise abgewandelt. Zwar taucht auch bei Yahoo! Pipes die Filterung von Daten auf. Der Fokus liegt jedoch auf der Zusammenführung und Umstrukturierung von Daten aus unterschiedlichen Datenquellen [SHT+07]. Dies wird in den auf Yahoo! Pipes aufbauenden Erweiterungen beibehalten. Diese Erweiterungen extrahieren ihre Daten nicht mit spezialisierten Filtern aus diversen Webdiensten. Statt dessen erzeugen sie intern SPARQL-Anfragen (siehe Abschnitt 2.2.2). Diese Anfragen werden an standardisierte SPARQL-Endpunkte geschickt [MPP+07; JD08].

Das Konzept ExPlates verwendet Flüsse gleichermaßen zur Zusammenführung, Umwandlung und Visualisierung von Daten [JE13]. Die entfernt verwandte Technik VIQUEN bietet ebenso viele Möglichkeiten, die Eingangsdaten zu einem Ergebnisgraphen der gewünschten Struktur umzuformen [Del10].

Kaleidoquery bricht aus der ursprünglichen Flussmetapher aus. In diesem Konzept haben wie in Abbildung 2.14 gezeigt Einschränkungen für verschachtelte Objekte auch flussaufwärts beziehungsweise in parallelen Flussarmen eine Wirkung [MPG98]. Lark entfernt sich ebenso vom

[2]Dabei wurden mitunter alternative Bezeichnungen verwendet, sodass statt von Flüssen von Rohren [SHT+07; MPP+07] oder Strömen [JGZ+11] gesprochen wurde. Im weiteren Sinn kann die Funktionsweise von *Begriffsverbänden* [MGA+11] ebenfalls als Filter/Flow-basiert angesehen werden.



(a) Die Filterknoten in DataMeadow (hier: Filterung deutscher Ortschaften nach Einwohnerzahl, Höhe und Koordinaten) erlauben das gleichzeitige Filtern nach mehreren Attributen durch die Einschränkung von Wertebereichen in einem Starplot [EST08].

(b) Die Filterknoten in FacetStreams (hier: Auswahl von EU-Staaten mit Millionenstädten) sind für Touch-Bedienung auf horizontalen Bildschirmen ausgelegt [JGZ+11].

Abbildung 2.12: Filterknoten in diversen Erweiterungen des Filter/Flow-Konzepts.

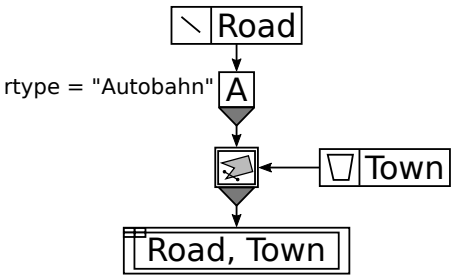


Abbildung 2.13: In der spatialen Filter/Flow-Variante nach Morris et al. [MAG+04] können Autobahnen ermittelt werden, welche durch Ortschaften hindurchgebaut wurden.

anfänglichen Konzept, da aufgespaltene Flüsse nicht wieder zusammengeführt werden [TIC09]. Andererseits können in Lark mehrere Ergebnisbereiche an den Graphen angefügt werden, wie in FindFlow [HSM+06] und ExPlates [JE13].

HyperFlow basiert ebenfalls auf Flussgraphen. Diese sind jedoch eher dazu da, den Ablauf eines Filter- und Verarbeitungsprozesses auszudrücken [DP05]. Bedingungen können als Knoten beziehungsweise verschachtelte Graphen in die Flüsse eingebunden sein. Dabei werden boolesche Verknüpfungen selber nicht als Flussgraphen, sondern als

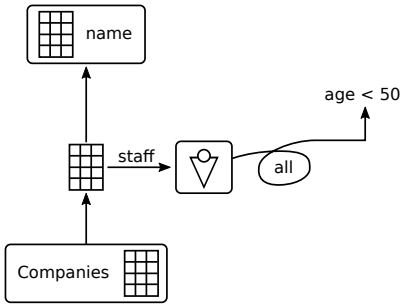


Abbildung 2.14: Eine ähnliche Anfrage wie in Abbildung 2.10b, welche Unternehmen sucht, deren gesamte Belegschaft unter 50 Jahre alt ist, hier dargestellt in KALEIDOQUERY [MPG98]. KALEIDOQUERY bricht teilweise aus der Flussmetapher aus, da die Bedingung age $<$ 50 Einfluss auf die Datenmenge am *name*-Knoten hat, welcher entlang der gerichteten Kanten nicht von der Bedingung aus erreichbar ist.

Ausdrucksbäume dargestellt. HYPERFLOW baut somit nicht auf dem FILTER/FLOW-Konzept auf. Ähnlich verhält es sich mit der Visualisierung von PROGRES, welche auf Flussgraphen basiert, aber die einzelnen Bestandteile von Bedingungen nicht weiter grafisch aufschlüsselt [Sch94].

Von diesen flussbasierten Ansätzen zu unterscheiden ist das Konzept DILIA. DILIA drückt zwar wie das FILTER/FLOW-Konzept logische Beziehungen durch Kanten aus, insgesamt stellt es allerdings keinen gerichteten Graphen dar [Sei11]. Vielmehr werden über die Kanten alle als Knoten aufgelisteten Kriterien zu paarweisen Konjunktionen zusammengefasst. Durch Interaktion können Disjunktionen und Konjunktionen mit negierten Kriterien erzeugt werden. Diese Disjunktionen und Konjunktionen werden dann aber lediglich textuell angezeigt. Analog zum FILTER/FLOW-Konzept wird auf den Konjunktionskanten eine Vorschau für die Größe der jeweiligen Ergebnismenge dargestellt.

2.3.2 Tabellenbasierte Ansätze

Eine vergleichsweise geringe Anzahl von Filtervisualisierungen geht von Tabellen als Grundlage aus. Ein Beispiel ist das auf Karnaugh-Veitch-Diagrammen basierende und in Abbildung 2.15 dargestellte KMVQL. Es generiert durch das Markieren von Tabellenzellen Filterausdrücke in disjunktiver Normalform [Huo08]. Auch Adjazenzmatrizen von Graphen können als Grundlage für tabellenbasierte Visualisierungen dienen [HFM07; BPL+11]. Dabei steht die Anfragevisualisierung selber allerdings mitunter im Hintergrund [EDG+08; GFM+11].

Einige frühe Konzepte verwenden eine tabellarische Darstellung, um di-

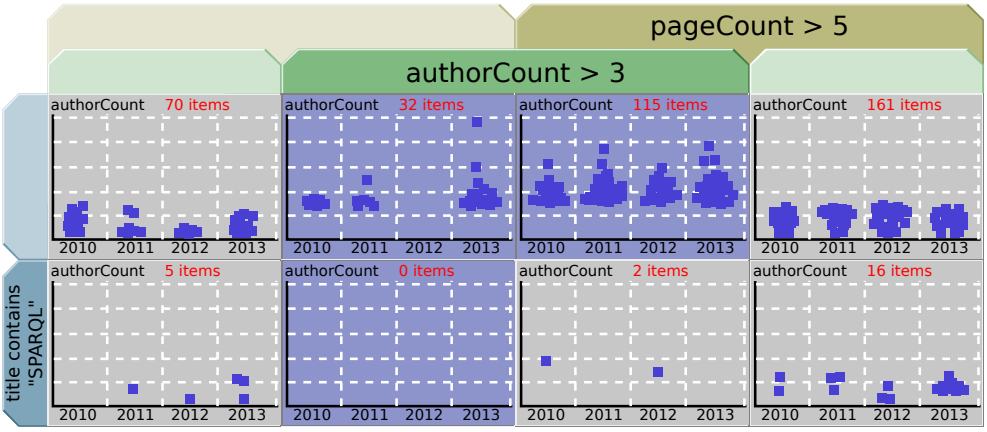


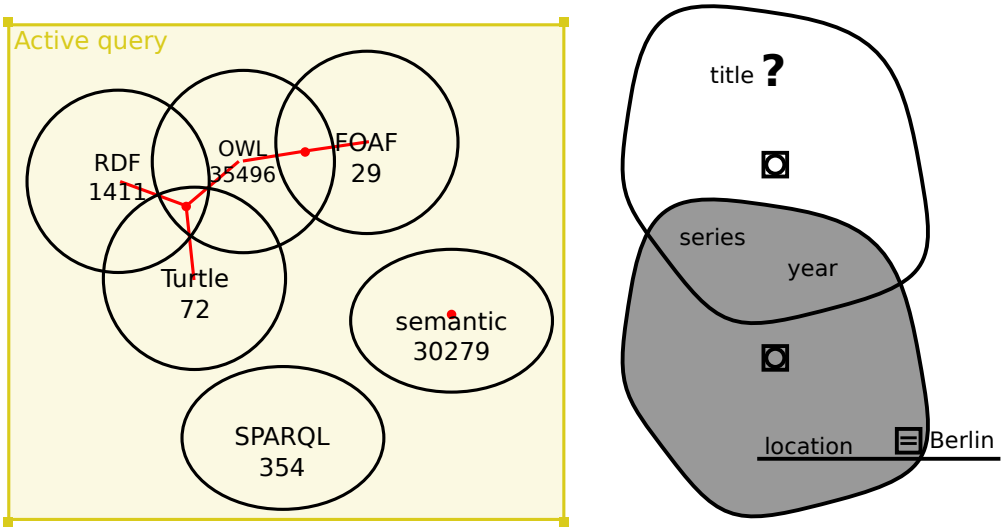
Abbildung 2.15: Suche nach Publikationen auf der Konferenz ESWC zwischen 2010 und 2013 in der Tabellenansicht von KMVQL [Huo08]. Über die Auswahl von Tabellenzellen wird hier der Ausdruck (authorCount $>$ 3) $\wedge$ $\neg$(contains(title, „SPARQL“) $\wedge$ (pageCount $>$ 5)) gebildet.

rekt die Eingabe von Beschränkungen für Tabellenspalten aus einer Datenbank zu ermöglichen. VKML erlaubt die Auswahl von Facetten der Datensammlung in einem Entity-Relationship-Diagramm. Filtereinschränkungen werden dann jedoch in einer Tabelle getätigt [SCT91]. HIQUEL scheint ebenfalls Tabellen zu enthalten, die nach einem ähnlichen Prinzip funktionieren [UZ83][3]. Gleichermaßen werden in GRAQULA Tabellen für die Darstellung einzelner Attributbeschränkungen verwendet [SBM+93], wie oben in den Abbildungen 2.5b und 2.8a zu sehen ist. In der näheren Vergangenheit wurde mit tFacet auf Tabellen aufgebaut. Dieses Konzept unterstützt eine Objektgraphstruktur durch die Verschachtelung und Unterteilung von Zellen [BH11].

2.3.3 Mengenorientierte Ansätze

Venn-Diagramme können eingesetzt werden, um Verknüpfungen von Filterkriterien in Form von Operationen auf den Ergebnismengen auszudrücken. VQuery ermöglicht das Hinzufügen von Kriterien als Schlüsselwörter, welche dann als elliptische Mengen dargestellt werden [Jon98]. Durch Übereinanderschieben und Markieren dieser Ellipsen und ihrer Schnittflächen können wie in Abbildung 2.16a dargestellt die booleschen Operatoren „und“, „oder“ und „nicht“ ausgedrückt werden. Ein Eindruck von der Größe der Ergebnismengen ist in VQuery jeweils nur durch die Anzeige einer Zahl gege-

[3] Laut einem Survey-Paper [Cha97]; die originale Publikation zu HIQUEL war nicht auffindbar.



(a) VQuery verwendet Venn-Diagramme, um Suchanfragen darzustellen [Jon98]. Die gezeigte Anfrage bezieht sich ausschließlich auf Bestandteile von Publikationstiteln und sucht somit Veröffentlichungen, in deren Titel die Wörter „RDF“, „OWL“ und „Turtle“, „OWL“ und „FOAF“ oder „semantic“ aber nicht „SPARQL“ vorkommen.

(b) In PICASSO dienen Venn-Diagramme zur Darstellung von Hypergraphen [KKS88]. Die gezeigte Anfrage ermittelt Titel von Publikationen, die auf beliebigen Konferenzen vorgestellt wurden, welche in Berlin stattfanden.

Abbildung 2.16: Beispiele mengenorientierter Anfragevisualisierungen.

ben. Prinzipiell können Größe und Position der Kreise in Venn-Diagrammen aber so gewählt werden, dass die Flächen proportional mit dem Umfang der dargestellten Datenmenge wachsen [HVA08]. Das Konzept OQD vereinigt die in Abschnitt 2.3.1.1 erläuterte Objektgraph-basierte Darstellung mit Venn-Diagrammen. Dazu werden, wie oben in Abbildung 2.8d gezeigt, Knoten eines Node-Link-Diagramms direkt als Mengen in Venn-Diagrammen genutzt [KM93].

Im Gegensatz zu den eben genannten Ansätzen können Venn-Diagramme ebenso auf Grundlage separat eingegebener Bedingungen so generiert werden, dass sich alle booleschen Kombinationen ergeben und der Benutzer die gewünschte(n) Kombination(en) auswählen kann [Mic82]. Es wird also eine ähnliche Idee wie im oben genannten KMVQL (siehe Abschnitt 2.3.2) verfolgt. Bei mehr als vier Bedingungen können die Venn-Diagramme jedoch geometrisch relativ komplex und damit unübersichtlich werden [Grü99].

Hypergraphen lassen sich als Mengen von Mengen darstellen. Jede der Mengen repräsentiert dabei eine Kante, welche die mit der Kante ver-

bundenen Knoten enthält. Aufbauend auf dieser Darstellungsform können ebenfalls Suchanfragen definiert werden. Im in Abbildung 2.16b gezeigten Konzept PICASSO drücken Schnittmengen nicht die Konjunktion mehrerer Filterkriterien aus, sondern einen Bezug zwischen mehreren Hyperkanten [KKS88].

2.3.4 Ansätze ohne feste Notation

Eine Reihe von Techniken verwendet keine Kombination vordefinierter Symbole zur Darstellung von Filterkriterien. Statt dessen werden grafische Primitive unterschiedlich positioniert, um entweder auf abstrakte Weise Kriterien auszudrücken oder direkt für Menschen erkennbare Grafiken bilden.

In InfoCrystal werden beispielsweise die logischen Verknüpfungen symbolisch ausgedrückt [Spo93]. Ansätze wie SageBrush stellen ein Beispiel dafür dar, wie Informationen auf abstrakte Weise in einer einfach konstruierbaren visuellen Notation integriert werden können [RKM+94]. SageBrush ist zwar selber nicht für Anfragen gedacht. Es beinhaltet aber als Editor für Informationsvisualisierung aus grafischen Primitiven ähnliche Konstruktionsaspekte.

Alternativ lassen sich die atomaren Filterkriterien grafisch darstellen [JS09]. Da hierbei konkrete Sachverhalte ausgedrückt werden müssen, sind derartige visuelle Repräsentationen aber durch ihr Vokabular oft auf einen bestimmten Anwendungsbereich begrenzt [VKS+12]. Diese Einschränkung lässt sich nur über die Definition eines umfangreichen visuellen Vokabulars wie Picsyms [Car85], MediaGlyphs [Ins02] oder iConji [Ove10] umgehen. Andernfalls ist ein hoher Grad an Abstraktion und eine vollständig benutzerseitige Anpassung der konkreten visuellen Symbole notwendig. Dieser Weg wird beispielsweise im DOODLE-Ansatz gewählt [Cru92]. Letztendlich können aber auch Zeitleisten (siehe Abschnitt 2.3.5) und Landkarten (siehe Abschnitt 2.3.6) als grafische Repräsentationen bestimmter Kriterien aufgefasst werden.

Ebenfalls zu dieser Kategorie zählen Konzepte, bei denen der grafische Aspekt sich auf die Position von Filterbestandteilen in Bezug auf die dargestellten Ergebnisse bezieht. Beispielsweise wird im Konzept Dust and Magnet wie in Abbildung 2.17 dargestellt eine Magnetmetapher angewendet. Dabei werden Datenelemente in Richtung von frei auf einer Fläche platzierbaren Filtern verschoben, wenn die pro Magnet konfigurierten Bedingungen von den Datenelementen erfüllt werden [SMS+05]. Dasselbe trifft auf davon abgeleitete Konzepte wie Magnet Mail [CL09], CloudMonster [CHB09], PhotoMagnets [CRB10] oder den Ansatz von Spritzer und Freitas [SF08] zu.

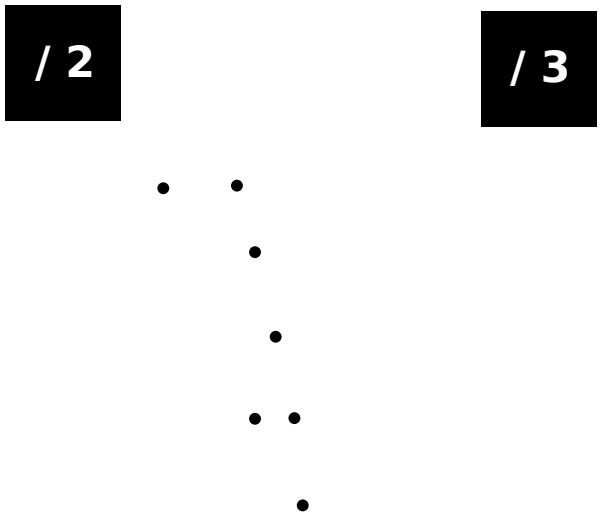


Abbildung 2.17: Anziehung der Zahlen von 1 bis 9 (als runde Staubpartikel) durch (quadratische) Magneten in Abhängigkeit des abgerundeten Quotienten aus der Division durch 2 beziehungsweise 3 im Konzept Dust and Magnet [SMS+05]. 1 (ganz unten) wird von gar keinem Magneten angezogen, die meisten Zahlen werden teilweise von beiden Magneten angezogen, jedoch stärker vom geteilt-durch-2-Magneten, und bei Zahlen wie 8 (oberer linker Partikel) überwiegt die Anziehungskraft durch diesen Magneten deutlich. Die Bedeutung der einzelnen Staubpartikel ist in der Visualisierung nicht direkt sichtbar, kann in Implementierungen aber interaktiv abgerufen werden.

2.3.5 Zeitbasierte Ansätze

Genau wie textuelle Anfragesprachen speziell für zeitbezogene Abfragen erweitert wurden (beispielsweise TQuel [Sno87]), wurden auch diverse Visualisierungen für Suchanfragen mit Zeitbezug vorgeschlagen. Ansätze, welche Daten lediglich übersichtlich darstellen und gegebenenfalls zusammenfassen, um eine manuelle Filterung zu erleichtern, konzentrieren sich bei temporalen Daten oft darauf, die Daten auf Grundlage einer Zeitleiste anzuordnen. Beispielsweise werden im Projekt Lifelines Ereignisfolgen anhand einzelner Ereignisse relativ zueinander ausgerichtet, wie in Abbildung 2.18 zu sehen ist. Daraufhin werden mögliche Übereinstimmungen sichtbar [WPQ+08]. CircleView stellt die Veränderungen mehrerer Skalare im Verlauf der Zeit auf kompakte Weise dar [KSS04]. FpMapViz gibt einen Überblick über algorithmisch identifizierte zyklisch wiederkehrende Muster [LJI11]. Letztendlich kann die allgemeine Granularität der zeitlichen Skala durch regelmäßige Muster angedeutet werden [CO12].

Mit EventFlow wurde eine Visualisierungstransformation definiert,

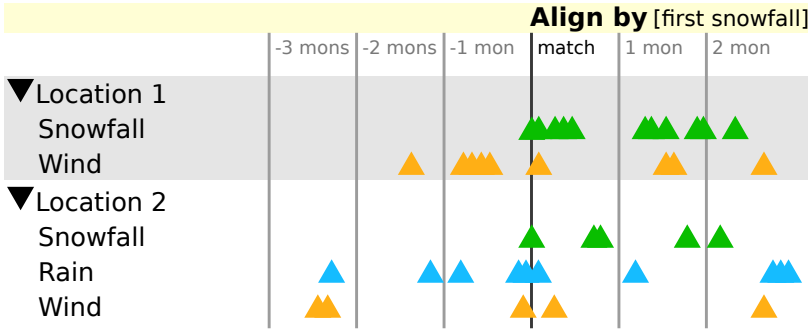


Abbildung 2.18: In LIFELINES können verschiedene Ereignisabfolgen anhand von Schlüsselereignissen zueinander ausgerichtet und auf diese Weise verglichen werden [WPQ+08]. Im gezeigten Beispiel werden (fiktive) Wetterdaten zweier Orte angezeigt.

welche Details in Ereignisketten zugunsten des groben Überblicks über die wichtigsten Phasen und Übereinstimmungen ausblendet [MLL+13]. Das Erkennen bestimmter Sachverhalte im Zusammenhang mit der zeitlichen Einordnung der Daten wird auf diese Weise unterstützt. Konkrete Anfragen werden dabei jedoch nicht visuell ausgedrückt.

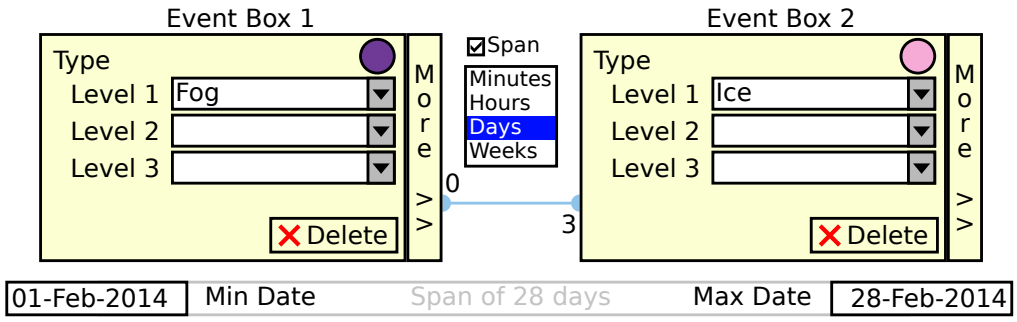
Neben diesen Ansätzen, die Benutzern das selbständige Suchen in der Gesamtdatenmenge erleichtern, existieren hier wiederum einige Techniken, welche die Suchanfragen oder -parameter an sich visualisieren. Die zwei grundlegenden erkennbaren Richtungen befassen sich einerseits mit Mustern in zeitabhängigen skalaren Werten und andererseits mit Abfolgen logischer Ereignisse.

Zeitabhängige Skalare Zeitabhängige Skalare können auf einfache Weise als Wert-Zeit-Diagramme dargestellt werden. Infolgedessen bauen Ansätze zur Suche in solchen Zeitreihen zum Teil auf Wert-Zeit-Diagrammen auf. Durch TIMEBOXES können in angezeigten Zeitreihen einzelne Ausschnitte markiert werden. Dies dient dazu, Zeitreihen mit übereinstimmenden Werteänderungen zu finden [HS01]. Dabei müssen die konkreten Zeitintervalle nicht exakt übereinstimmen [KHS02]. Auf diese Weise gefilterte Daten können dann nach anderen Kriterien (beispielsweise einer Vorreiter- oder Nachzüglerrolle gegenüber anderen Zeitreihen) weiter gefiltert werden [HS04]. Um mit der Filterung oder Suche zu beginnen, ist jedoch stets die Anzeige einiger oder aller Zeitreihen erforderlich. Es kann also keine Anfrage vorbereitet werden, wenn noch keine Daten vorliegen. QUERYSKETCH erlaubt hingegen das Skizzieren des gewünschten Kurvenverlaufs [Wat01]. Folglich können mit QUERYSKETCH auch Hypothesen zum Vorhandensein bestimmter Zeitreihen überprüft werden.

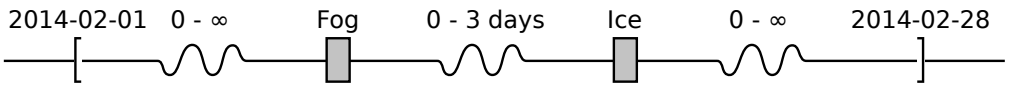
Ereignisabfolgen Werden statt skalaren Werten Ereignisse, Zustände oder andere potenziell komplexe logische Einheiten betrachtet, ist die einfachste Darstellungsform eine reine Aneinanderreihung [CCM92; KCH10; LWW+13; ZDF+15]. Diese Repräsentation kann für Anfragen genutzt werden, indem die gesuchten Ereignisse – oder deren Ausbleiben [MLM+13] – der Reihe nach angegeben werden. Dies ist beispielsweise in MQuery der Fall, wo sich Zeitabstände zwischen den gesuchten Ereignissen angeben lassen. Diese Abstände werden jedoch nur textuell dargestellt [DC96]. Gleiches gilt für DecisionFlow [GS14]. Entsprechend werden in Lifelines Ereignisbezeichnungen einfach aneinandergereiht. Eine Visualisierung der zeitlichen Zusammenhänge erfolgt erst für gefundene Ergebnisse [WPQ+08]. ActiviTree reiht aufeinanderfolgende Ereignisse ebenfalls aneinander. Dabei wird jeweils angedeutet, welche unterschiedlichen Ereignisse der gesuchten Sequenz in der aggregierten Ergebnismenge vorausgehen und folgen [VJC09].

Intervalle und Zeiträume Um Zeitabstände zwischen Ereignissen zu verdeutlichen, können diese als proportionale oder angedeutete Abstände zwischen den visuellen Ereigniselementen angezeigt werden. Bei der Ähnlichkeitssuche in Ereignissequenzen werden konkrete Zeitabstände zwischen Ereignissen proportional dargestellt, wobei gefundene Sequenzen in einem gewissen Rahmen von der Vorgabe abweichen dürfen [WPT+12]. Darstellungen wie die von Pattern Finder [FKS+06] stellen unmittelbar aufeinanderfolgende Ereignisse in direkter Nachbarschaft dar. Demgegenüber stehen zeitlich getrennte Ereignisse, die auch visuell wie in Abbildung 2.19a gezeigt durch einen Abstandhalter getrennt sind. Dieser Abstandhalter kann mit weiteren Einschränkungen versehen werden, die den genauen, Mindest- oder Maximalabstand zwischen den Ereignissen betreffen [CO12]. Gleichermaßen werden solche Einschränkungen in Practice Explorer verwendet [ENV+07], welcher in Abbildung 2.19b dargestellt ist. Um ungefähre Zeitabstände visuell auszudrücken, kann auf Ansätze wie PlanningLines zurückgegriffen werden. Im Vergleich mit textuellen Angaben zu ungefähren Zeitabständen haben sie sich als relativ schnell lesbar gezeigt [AMT+05]. An Stelle von beschreibenden Texten können die Ereignisse selbst als Grafiken ausgedrückt werden. Dies ist in QueryMarvel [JS09] und dem darauf aufbauenden VizPattern [JS10] der Fall, wie in Abbildung 2.19c sowie unten in Abbildung 2.20 zu sehen ist. Diese Techniken sehen zudem Disjunktionen und Konjunktionen von Ereignissen vor.

Neben einem direkten Anschluss und einem zeitlichen Abstand können zwischen Ereignissen ebenfalls weitere temporale Beziehungen bestehen. Als Beispiel sei die teilweise oder komplette Überschneidung von Ereignissen genannt. Diese Beziehungen lassen sich zu einer geringen Anzahl temporaler



(a) Pattern Finder [FKS+06]



(b) Practice Explorer [ENV+07]

(c) QueryMarvel [JS09]/VizPattern [JS10] – der erste Februar 2014 als Startdatum wird nicht in der Visualisierung gezeigt. Da die Grafiken stark domänenabhängig sind, wurden für das gezeigte Beispiel eigene Grafiken für „Nebel“ und „Eis“ gewählt.

Abbildung 2.19: „Finde Gegebenheiten im Februar 2014, bei denen maximal drei Tage nach Nebel Eis folgte.“ in mehreren zeitbasierten Anfragevisualisierungen.

Operatoren abstrahieren [All83]. Die Operatoren werden für die Darstellung in TVQL in einzelne Vergleiche zwischen Start- und Endzeitpunkten von Ereignispaaren aufgetrennt [HR95]. In einer Anfragesprache basierend auf dem temporalen erweiterten Entity-Relationship-Modell werden ähnliche Operatoren verwendet und als Relationen zwischen Zeiträumen visualisiert [KG95]. Ähnlich wurde in mindestens einem weiteren Ansatz vorgegangen [FSC97].

Ereignisse müssen nicht zwangsläufig als punktuell angenommen werden. Alternativ können sie auch als Phasen mit einer Dauer modelliert werden. Mit dieser Interpretation bietet sich eine Balkendarstellung an [FC00; VEC07; VJC09]. Anfänge und Enden der Balken können durch Verbindungselemente zueinander in Beziehung gesetzt werden [CC03; CO12].

Disjunktionen aus alternativen Ereignissequenzen können getrennt von der Zeitleistendarstellung angezeigt werden [CO12]. Ebenso lassen sich die

Disjunktionen direkt in die Grafik integrieren. Alternative Abläufe verlaufen ähnlich wie in FILTER/FLOW-Graphen (siehe Abschnitt 2.3.1.2) parallel zueinander [ENV+07; JS09; JS10; ZDF+15].

2.3.6 Räumliche und geometrische Filterung

Für visuelle Filteranfragen in Bezug auf geometrische oder geografische Sachverhalte wird häufig auf bestehende Anfragevisualisierungsparadigmen unter Einbeziehung Geometrie-bezogener Operatoren zurückgegriffen. CIGALES stellt Objektgraphmuster als Node-Link-Diagramm dar (siehe Abschnitt 2.3.1.1), bietet aber speziell ortsbezogene Operatoren an [CM94]. Morris et al. haben das FILTER/FLOW-Konzept (siehe Abschnitt 2.3.1.2) wie oben in Abbildung 2.13 gezeigt für räumliche Zusammenhänge angepasst [MAG+04].

Ein anderer Weg wird von Konzepten beschritten, welche die Anfrage direkt in die Visualisierung der geometrischen Bezüge einbetten. So wurde beispielsweise eine Technik vorgeschlagen, um Operatoren für geometrische Einschränkungen in ein dreidimensionales partielles Modell eines Gebäudes einzubinden [HB14].

2.3.7 Filterung in Objektgraphen

Um innerhalb einer Anfrage Bezüge zwischen bestimmten Objekten herzustellen, lassen sich zusammenfassend einige unterschiedliche Ansätze erkennen. Objekte können mit Namen versehen sein [DGJ+95; GGS11] oder wie in Abbildung 2.20 gezeigt durch individuelle Farben identifiziert werden [CO12; JS09]. Eine weitere Option besteht darin, für jedes Objekt einen separaten Kontext (beispielsweise eine Lebenslinie wie mit SOFTGRAPH [CCM92] oder bei UML-Sequenzdiagrammen [Obj15, S. 593ff.]) bereitzustellen.

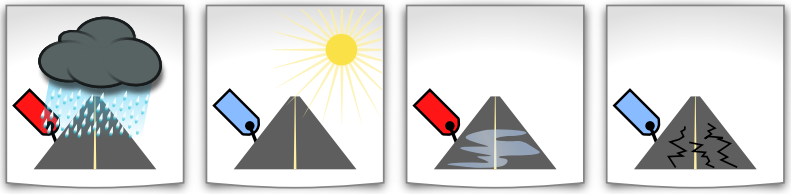


Abbildung 2.20: Eine QUERYMARVEL-Anfrage [JS09] nach einer Gegebenheit, bei der eine Straße nach einem Regen nass war und eine andere nach starker Sonneneinstrahlung Risse bekommen hat. Die unterschiedlichen Straßen werden durch farbige Markierungen identifiziert.

2.4 Anwendungsgebiete für visuelle Anfragen

In diesem Abschnitt soll ein kurzer Überblick über die Themengebiete gegeben werden, in deren Rahmen komplexe Anfragen und insbesondere Anfragevisualisierungen in der Literatur hauptsächlich verwendet werden. Nur tiefergehende Betrachtungen von Anwendungsfällen werden berücksichtigt. Beispiele, deren Szenarien nicht näher erläutert sind, werden nicht mitaufgenommen. Einige der gesammelten Themengebiete werden in nachfolgenden Kapiteln dieser Arbeit herangezogen, um Einsatzbeispiele der vorgestellten erweiterten und neuen Konzepte zu präsentieren.

2.4.1 Biologie

Komplexe Anfragen tauchen in diversen Teilbereichen der Biologie auf. Einerseits kann es um das Auffinden bekannter Sachverhalte innerhalb größerer Datenmengen gehen. Dies betrifft sowohl die makroskopische [Del10; MGA+11] als auch die mikroskopische [DP05; JBX+10; BWW+05] Ebene. Andererseits kann ein Zweck der Anfragen darin liegen, bislang unbekannte Eigenschaften oder Zusammenhänge zu entdecken [HVA08]. Ebenso werden häufig Veränderungen von Werten im Verlauf der Zeit berücksichtigt [HS04]. Meist ist dies im medizinischen Bereich im Hinblick auf den Krankheitsverlauf oder die Wirkung von Medikamenten bei Patienten von Interesse [DC96; FKS+06; JS09; RWW+11; MLM+13; GS14; PW14]. Einzelne der Arbeiten in diesem Bereich greifen tatsächliche Anfragen aus dem betreffenden Themengebiet auf [Del10].

2.4.2 Administrative Datenbanken

In Datenbanken zum Verwalten von Personen- und Organisationsdaten liegen häufig unterschiedliche Angaben zu jedem Datensatz vor. Dadurch kann eine facettierte Suche notwendig werden, wie in der Anfrage aus Abbildung 2.5 beispielhaft illustriert ist. Entsprechend greifen einige der Konzepte für Anfragevisualisierungen auf derartige Datenbanken zurück. Datenbanken von Einwohnern [EST08] oder registrierten Organisationen [HVG14] eines Landes bieten sich hierbei an. Ebenfalls kann eine Datensammlung Mitarbeiterdaten [Shn91; MPG98; DGJ+95], Studentendaten [EST08; CEC87] oder eine Mischung daraus [CHM+96; Feg99] enthalten. Gleichermaßen bieten allgemeine Informationen zum Zustand und der Wirtschaft eines Landes Gelegenheit für den Einsatz komplexer Suchanfragen [HMF+10]. Letztendlich werden auch die internen Organisationsstrukturen in Unternehmen betrachtet. Diese bringen Kunden mit Vorgängen in Zusammenhang, wie es beispielsweise bei Flugbuchungen [SS93], Buchbe-

stellungen [CCD+99] oder Kunden einer Bank und ihren Konten [KKS88] geschieht.

2.4.3 Produktauswahl

Die Auswahl eines Produkts hängt inhärent von verschiedenen Faktoren ab, die von jedem Kunden anders bewertet werden können. Folglich ergibt sich bei der Produktsuche oft die Notwendigkeit komplexer Anfragen. Beispielsweise trifft dies auf Digitalkameras [EST08], Autos [Huo08] und Hotels [JGZ+11] zu. Insbesondere bei der Wohnungssuche erfordern sowohl Metainformationen [Shn94; HSM+06] als auch Daten zum strukturellen Aufbau der Räume [LWL+13] komplexe Suchanfragen. Strukturell ähnlich zu betrachten ist die Suche nach Musikstücken [GDH+09; CHB09] und Filmen [AS94], sowie die Auswahl einer Bildungseinrichtung [Shn91]. Unter Zuhilfenahme des Zeitbezugs kann ebenso nach bestimmten Mustern in Aktienkursen gesucht werden [HS04; Wat01].

2.4.4 Wissenschaftliche Veröffentlichungen

Geht es um Anwendungsbeispiele in der wissenschaftlichen Literatur, so ist die Suche nach Publikationen selbst ein gewissermaßen naheliegendes Anwendungsgebiet. Die Anwendungsfälle basieren oft auf der Suche nach Metadaten [GDH+09; Sei11]. Aufgrund der Verfügbarkeit entsprechender Verzeichnisse wird hierbei oft auf reale Daten zurückgegriffen. Verzeichnisse wie DBLP [CFT+08] oder INSPEC [Spo93] eignen sich dafür.

2.4.5 Allgemeines Verhalten

Insbesondere im Zusammenhang mit der Zeit können bestimmte Muster im menschlichen Verhalten gesucht werden. Komplexe Suchanfragen werden dabei zum Teil im Zusammenhang mit Alltagsabläufen genannt [VEC07; PW14]. Dies bezieht thematisch begrenzte Situationen wie den Straßenverkehr mit ein [DVZ95]. Speziellere Abläufe, wie die Anmeldung zu einer Prüfung [JS10], können ebenso im Fokus komplexer Suchanfragen stehen.

2.4.6 Reiseplanung

Komplexe Anfragen können im Gebiet der Routen- und Reiseplanung auftreten. Unter Berücksichtigung diverser benutzerspezifischer Einschränkungen können beispielsweise Verbindungen mit öffentlichen Verkehrsmitteln gesucht werden. In diesem Bereich wurden bislang nur vereinzelt Vorschläge für visuelle Anfragen gezeigt [CM94]. Diverse Arbeiten sehen allerdings

komplexe Anfragen vor, in welchen eine bestimmte Ausstattung der Zwischenhaltestellen [OBM+13] oder ein bestimmtes Geschäft in der Nähe von Zwischenhalten [BSW+09; OBM+13] verlangt wird.

Dieses Anwendungsgebiet wurde auch im Projekt IP-KOM-ÖV berührt, aus dem Teile der Forschung für dieses Promotionsvorhaben finanziert wurden. Im Rahmen jenes Projekts wurde unter anderem ein semantisches Modell zur Unterstützung komplexer Verbindungsanfragen im öffentlichen Verkehr entwickelt. Mit diesem können beispielsweise Zwischenstopps für touristische Rundtouren über Attribute wie die Kategorie der Sehenswürdigkeiten eingeschränkt werden [KBS14].

2.4.7 Sonstige

Weitere Themengebiete, die im Rahmen von Anfragebeispielen vorgestellt werden, sind geschichtliche und archäologische Daten [Shn91; Shn94] sowie die technischen Zusammenhänge zwischen Bauteilen in einem Fahrzeug [MK93]. In eine ähnliche Richtung gehen auch Anfragen zum Aufbau und zu Anforderungen für Software-Produkte [MC10] sowie zu Sicherheitsrestriktionen in Rechnersystemen [XSA08].

Als weiteres Anwendungsgebiet werden mitunter umfangreiche oder unübersichtliche Dokumentsammlungen herangezogen. Dies schließt die Suche nach E-Mails [CL09], Fotos [CRB10] sowie nach Patenten [KBG+09] ein.

2.5 Datensammlungen

Für Beispiele in dieser Arbeit wird auf mehrere unterschiedliche Datensammlungen zurückgegriffen. Diese Datensammlungen wurden sowohl in Anlehnung an die in Abschnitt 2.4 beschriebenen Themengebiete als auch nach Verfügbarkeit ausgewählt. Einige der Datensammlungen werden im Verlauf der Arbeit immer wieder verwendet. Deshalb werden sie im Folgenden knapp beschrieben. Tabelle 2.1 gibt einen Überblick über die Eckdaten dieser Datensammlungen.

2.5.1 DBpedia

Bei DBpedia handelt es sich um eine RDF-Datensammlung. Für ihren Inhalt werden Informationen aus Wikipedia extrahiert und in RDF-Tripel überführt [ABK+07; LIJ+15]. Die Daten decken somit zahlreiche Themen-

[4] Es wurde jeweils auf die aktuelle, auf dbpedia.org verfügbare Version zugegriffen. Die Angabe zur Anzahl der Tripel bezieht sich auf DBpedia 2014 [DBp15].

Tabelle 2.1: Überblick über die im Verlauf dieser Arbeit wiederholt verwendeten Datensammlungen.

Datensammlung	Tripel	Stand
DBpedia	~3.000.000.000	verschiedene[4]
Faceted DBLP	~11.000.000	Februar 2014
Linked MDB	~6.000.000	Februar 2014
CIA World Factbook	~160.000	März 2014
Stack Exchange	~600.000.000	19. Mai 2014

gebiete ab. Die Struktur und Vollständigkeit sind jedoch relativ uneinheitlich.

2.5.2 Faceted DBLP

Faceted DBLP ist eine RDF-Version des Publikationsverzeichnisses DBLP [DB08]. Hierdurch wird das entsprechende Themengebiet aus den verwandten Arbeiten abgedeckt (Abschnitt 2.4.4). Publikationen sind mit ihren jeweiligen Fachzeitschriften beziehungsweise Konferenzen sowie mit ihren Autoren verknüpft.

2.5.3 Linked MDB

Die Datensammlung Linked MDB enthält eine aus unterschiedlichen Quellen zusammengesetzte Filmdatenbank [HC09]. Linked MDB bietet nur eine begrenzte Menge an Informationen pro Film, hauptsächlich Darsteller und Regisseur(e). Dennoch lassen sich mit dieser Datensammlung bereits einzelne Anfragen ähnlich wie die in Abschnitt 2.4.3 erwähnten Produktfilteranfragen modellieren.

2.5.4 CIA World Factbook

Das CIA World Factbook enthält politische, geografische und demografische Angaben zu Ländern der Erde. Diese Angaben ähneln zum Teil den in Abschnitt 2.4.2 erwähnten administrativen Daten. Eine auf dem DAML-Modell aufbauende RDF-Version dieser Daten steht zur Verfügung [Dea01].

2.5.5 Stack Exchange

Stack Exchange bezeichnet ein Netzwerk aus Frage-und-Antwort-Seiten. Benutzer können dort zielgerichtet Fragen stellen und beantworten [Sta10].

Das Netzwerk bietet über einhundert Frage-und-Antwort-Seiten zu unterschiedlichen Themen wie Programmierung (Stack Overflow), Hochschulwesen (Academia Stack Exchange) oder Reisen (Travel Stack Exchange) an. Benutzerkonten gelten netzwerkweit, wobei für jede einzelne Seite ein Punktestand geführt wird, der auch als Qualitätskontrolle dient. Stack Exchange veröffentlicht regelmäßig Schnappschüsse des gesamten Netzwerkinhalts (wobei personenbezogene Informationen anonymisiert werden), welche sich nach RDF überführen lassen.

Teil II

Neue und erweiterte Ansätze zu visuellen Anfragen

Vorbemerkungen

In diesem Teil der Arbeit werden verschiedene Ansätze zur visuellen Formulierung und Repräsentation von Filter- und Suchanfragen vorgestellt, die im Rahmen des Promotionsvorhabens entwickelt wurden. Einige davon stellen Erweiterungen bestehender Techniken dar (Filter/Flow-Erweiterungen in Abschnitt 3.1, Magnetic Querying in Abschnitt 3.3). Weitere Konzepte wurden grundlegend neuentwickelt und haben nur stellenweise Ähnlichkeit mit bestehenden Konzepten (QueryVOWL in Abschnitt 3.2, Filter Dials in Abschnitt 3.4, Aspect Grid in Abschnitt 4.1.1 und VESPa in Abschnitt 4.2).

Die Beschreibungen der Anfragekonzepte sind in zwei Gruppen gegliedert. Zunächst werden in Kapitel 3 Konzepte vorgestellt, die sich domänenunabhängig für die Suche in beliebigen Objektgraphen verwenden lassen. Sie dienen zur Filterung und Suche von beliebigen strukturierten Daten. Der Fokus dieser Visualisierungen liegt darauf, die Kombination unterschiedlicher Filterkriterien zu verdeutlichen. Zum Teil sind auch Möglichkeiten vorgesehen, einen vorläufigen Eindruck von der zu erwartenden Ergebnismenge nach Ausführung der Anfrage darzustellen. Kapitel 4 enthält anschließend Konzepte, welche auf bestimmte Einsatzgebiete oder Datentypen optimiert sind. Zwar sind derartige Ansätze nur begrenzt nutzbar, andererseits sind sie besonders intuitiv verwendbar, da Paradigmen zum Einsatz kommen, die den Nutzern bereits bekannt und speziell auf die jeweiligen Domänen oder Datentypen abgestimmt sind.

3 Allgemeine Ansätze

Zur Filterung beliebiger Daten bieten sich generische Visualisierungskonzepte an. Sie erlauben den Zugriff auch auf unbekannte Datensammlungen, ohne jeweils eine domänenspezifische Notation erlernen zu müssen. Generische Anfragevisualisierungen nutzen nicht die Besonderheiten bestimmter Datentypen oder Datenschemata, die nur in einzelnen Domänen zum Tragen kommen. Sie stützen sich allerdings auf strukturelle Eigenschaften der graphbasierten Datensammlungen, die gefiltert werden sollen. Im Folgenden werden mehrere dieser Konzepte vorgestellt, die unterschiedliche Gesichtspunkte der Anfragen in den Vordergrund rücken.

3.1 Filter/Flow

Das Filter/Flow-Konzept (s. Abschnitt 2.3.1.2) hat zum Ziel, boolesche Kombinationen von Filterkriterien darzustellen. In den verschiedenen vorgestellten Arbeiten wurden unterschiedliche Anpassungen für das Konzept vorgeschlagen. Einige davon sind rein ästhetischer Natur, während andere die Mächtigkeit des Konzepts erhöhen.

Im Rahmen dieses Promotionsvorhabens wurde das Erweiterte Filter/Flow-Konzept als Kombination aus vielen dieser bisherigen Anpassungen ausdefiniert (Abschnitt 3.1.2). Dabei wurden die folgenden Erweiterungen vorgenommen:

- Dem Konzept wurden neue Elemente hinzugefügt, um bessere Unterstützung für die Filterung in Objektgraphen zu bieten.
- Die Flussvisualisierung wurde ausgebaut, um verschiedene Datenmengen gleichzeitig zu filtern (Abschnitt 3.1.3).
- Eine Abbildung auf die Anfragesprache SPARQL wurde definiert, um das Konzept über SPARQL-Endpoints auf RDF-Daten anwendbar zu machen (Abschnitt 3.1.4).

> Die im Folgenden präsentierten Inhalte wurden zum Teil bereits anderweitig veröffentlicht. Insbesondere wird auf den im Folgenden aufgeführten Arbeiten, an denen der Autor maßgeblich mitwirkte, aufgebaut:
>
> HRE12 F. Haag, M. Raschke und T. Ertl. “Adaptable filter graphs – towards highly-configurable query visualizations”. In: *INFORMATIK 2012: Was bewegt uns in der/die Zukunft?* Bd. 208. LNI. Gesellschaft für Informatik e.V. (GI), 2012, S. 1059–1073

HLE12 F. Haag, S. Lohmann und T. Ertl. “Simplifying filter/flow graphs by subgraph substitution”. In: *Proc. Visual Languages and Human-Centric Computing (VL/HCC ’12)*. IEEE, 2012, S. 145–148. DOI: 10.1109/VLHCC.2012.6344501

HLE13b F. Haag, S. Lohmann und T. Ertl. “Evaluating the readability of Extended Filter/Flow graphs”. In: *Proc. 2013 Graphics Interface Conference (GI ’13)*. CIPS, 2013, S. 33–36

HLE14 F. Haag, S. Lohmann und T. Ertl. “SparqlFilterFlow: SPARQL query composition for everyone”. In: *The Semantic Web: ESWC 2014 Satellite Events*. Bd. 8798. LNCS. Springer, 2014, S. 362–367. DOI: 10.1007/978-3-319-11955-7_49

HLB+14 F. Haag, S. Lohmann, S. Bold und T. Ertl. “Visual SPARQL querying based on Extended Filter/Flow graphs”. In: *Proc. Advanced Visual Interfaces (AVI ’14)*. ACM, 2014, S. 305–312. DOI: 10.1145/2598153.2598185

HLE13a F. Haag, S. Lohmann und T. Ertl. “A Flexible Architecture for Filter/Flow-Based Visual Querying”. Graphics Interface (GI ’13) Poster Session (unveröffentlicht). 2013

Zusätzlich wird inhaltlich auf die folgenden studentischen Arbeiten, an deren Betreuung der Autor beteiligt war, zurückgegriffen:

Fer12 T. Ferber. “Visualisierung von Filter/Flow-Graphen auf mobilen Endgeräten”. Betreuer: Florian Haag. Diplomarbeit. Universität Stuttgart, 2012

Bol13 S. Bold. “Visuelle Filterung von Daten des Semantic Web”. Betreuer: Florian Haag, Steffen Lohmann. Diplomarbeit. Universität Stuttgart, 2013

3.1.1 Reinterpretation des ursprünglichen Konzepts

Für das ursprüngliche Filter/Flow-Konzept wie im Abschnitt 2.3.1.2 beschrieben (FFK) gilt in den ursprünglichen Arbeiten [YS93; Shn94] die Konvention, dass der Anfragegraph an genau einer Stelle beginnt und an genau einer Stelle endet. Die Datenquelle ist somit implizit dem Beginn des Filtergraphen vorgeschaltet. Die Zusammenführung von Daten aus unterschiedlichen Quellen ist damit nicht vorgesehen. Gleichermaßen ergibt sich eine endgültige Ergebnismenge am Ende des Graphen. Die Darstellung mehrerer paralleler Filteroperationen, die in unterschiedliche Ergebnismengen münden, ist auf diese Weise nicht möglich.

An Stelle eines festen Start- und Endpunkts des Filtergraphen kann man sich Beginn und Ende auch als zwei zusätzliche Knoten vorstellen, wobei der erste Knoten keine eingehenden Flüsse und der letzte Knoten keine abge-

henden Flüsse hat. In Analogie zu modularen Systemen wie AVS [UFK+89] oder dem IRIS Explorer [Fou95] repräsentiert der erste Knoten dann die Datenquelle, der letzte dient als Ergebnisanzeige.

Mit dieser Interpretation muss die FFK-Struktur eines azyklischen, gerichteten Graphen mit Flüssen und Filterknoten nicht geändert werden, um auch mehrere Start- und Endpunkte im Filtergraphen vorzusehen. Indem man Flüsse an mehreren Datenquellenknoten beginnen lässt, nutzt man das FFK also dahingehend aus, dass auch Daten aus mehreren Quellen zusammengeführt (wie in Yahoo! Pipes [SHT+07], siehe Seite 27) und parallele Filtervorgänge repräsentiert werden können. Wird nicht erzwungen, dass der Filtergraph zum Ende hin zu einem Fluss vereinigt wird, sondern an mehreren Stellen endet, an denen jeweils Ergebnisknoten angefügt werden, können die parallelen Filtervorgänge auch zu unterschiedlichen Ergebnismengen führen (wie in den ab Seite 27 beschriebenen Konzepten FindFlow [HSM+06] der Lark [TIC09]). Zu einer gegebenen Problemstellung können dann auch mehrere alternative Anfragen entworfen und ihre Ergebnisse verglichen werden. So können beispielsweise Fehler bei der Anfrageerstellung erkannt werden, die durch unachtsames Abändern existierender Anfragen entstanden sind [AP10]. In ExPlates waren beide Eigenschaften – die Zusammenführung mehrerer Datenquellen sowie die Einordnung in mehrere Ergebnismengen – bereits enthalten [JE13].

Da wie oben erklärt Datenquellen- und Ergebnisanzeigeknoten vorhanden sind, ist zu überlegen, ob diese Sonderfälle darstellen, die nur an bestimmten Stellen im Filtergraphen auftreten dürfen. Für reine Datenquellenknoten ist es schwer vorstellbar, dass sie eingehende Daten verarbeiten könnten – die eingehende Datenmenge müsste dazu mit der aus dem Datenquellenknoten stammenden Datenmenge verbunden werden, was den Datenquellenknoten zu einem komplexen Verbindungsknoten machen würde. Sie sollten deshalb immer den Beginn von Flüssen in einem Filter/Flow-Graphen darstellen. Ergebnisanzeigeknoten lassen sich hingegen intuitiv zu herkömmlichen Filterknoten abstrahieren, indem man sie als Filterknoten, deren Filterfunktion bezogen auf die eingehende Datenmenge die Identitätsfunktion ist, interpretiert. Auf diese Weise lassen sich Ergebnisanzeigeknoten außer am Ende auch in der Mitte von Flüssen einbinden, da die aktuelle Datenmenge einfach unverändert an die stromabwärts liegenden Knoten weitergeleitet wird. Dadurch entsteht die Möglichkeit, neben den finalen Ergebnissen auch Zwischenergebnisse darzustellen, wie dies von Morris et al. [MAG+04] sowie in FindFlow [HSM+06] vorgesehen ist. Ähnlich wie in Lark [TIC09] – im weiteren Sinn auch wie in Anwendungen wie Tabulator [BCC+06] oder NITELIGHT [RS08] – können auch verschiedene Ergebnisvisualisierungen durch unterschiedliche Ergebnisanzeigeknoten unterstützt werden [TIC09].

3.1.2 Erweitertes Filter/Flow-Konzept

Das Erweiterte Filter/Flow-Konzept (EFFK) baut auf der in Abschnitt 3.1.1 beschriebenen Interpretation auf und vereint viele der im Abschnitt 2.3.1.2 vorgestellten Erweiterungen zu einem integrierten Ansatz. Während in bisherigen Arbeiten die Erweiterungen zum Teil nur implizit oder nur in Bezug auf einzelne, bestimmte Filterknoten eingeführt wurden, werden die angepassten Merkmale für das EFFK explizit definiert. Die folgenden Erweiterungen gegenüber dem FFK wurden definiert und sind im Folgenden beschrieben:

- freie Platzierbarkeit von Knoten (Abschnitt 3.1.2.2)
- mehrere Rezeptoren und Emitter pro Knoten (Abschnitt 3.1.2.1)
- paralleles Filtern mehrerer separater Datenmengen (Abschnitt 3.1.3)

Die dadurch angestrebten Vorteile sind

- eine platzsparende Darstellung von FFK-Graphen,
- die Reduktion der Knoten- und Kantenzahl und
- das Zusammenfassen verwandter Knoten zu Filterknoten mit mehreren Emittern zur Vermeidung von Redundanz

unter Beibehaltung der Lesbarkeit des ursprünglichen Konzepts.

3.1.2.1 Aufbau von Filterknoten

Der Aufbau eines Knotens im EFFK ist in Abbildung 3.1 erläutert. Der Körper eines angezeigten Knotens enthält Filtereinstellungen oder die Anzeige von Zwischenergebnissen. Da beliebig viele Knotentypen definiert werden können, die abhängig von ihrer Funktion jeweils unterschiedliche Ein- und Ausgaben unterstützen, richtet sich der Inhalt des Knotenkörpers vollkommen nach dem jeweiligen Knotentyp.

Rezeptoren dienen als Verbindungselemente zu ankommenden Flüssen, während *Emitter* als Verbindungselemente für abgehende Flüsse fungieren. Zusammengefasst werden Rezeptoren und Emitter als *Konnektoren* bezeichnet. Abhängig vom Knotentyp ist die Anzahl der Konnektoren fest oder variabel. Ebenso können je nach Knotentyp auch konnektorspezifische Filtereinstellungen vorgenommen werden.

Dadurch, dass Knoten mehrere Rezeptoren haben können, wird die Mächtigkeit des FFK erhöht, da nun Filterfunktionen mit mehreren Parametern möglich sind (Knoten v_8 in Abbildung 3.2). Beispielsweise kann eine *join*-Funktion analog zu Datenbankabfragen realisiert werden. Dazu nehmen zwei Rezeptoren i_1 und i_2 zwei unterschiedliche Datenmengen entgegen. Der Knotenkörper enthält eine Konfigurationsmöglichkeit für ein Attribut x der an i_1 anliegenden Elemente und ein Attribut y der an i_2 anliegenden Elemente. Die *join*-Operation soll über diese Attribute ausgeführt

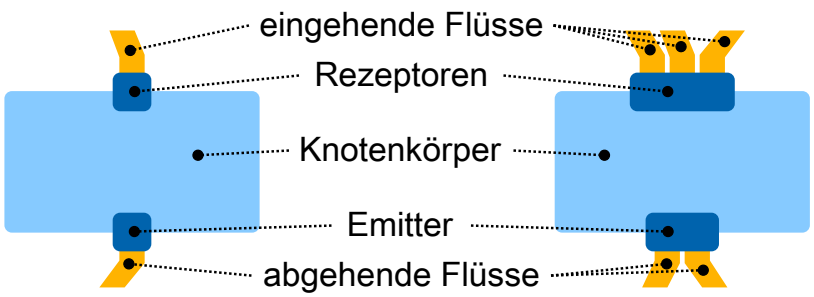


Abbildung 3.1: Aufbau zweier Knoten im EFFK. Als Flussrichtung wird in dieser und den folgenden Abbildungen implizit von oben nach unten angenommen.

werden. Ein Emitter kann dann die Menge der Datenelemente ausgeben, die an i_1 anliegen, und für die mindestens ein Element an i_2 anliegt, sodass $x = y$ gilt. Diese Erweiterung, unterschiedliche Flüsse mit unterschiedlichen Funktionen als Eingabe für Knoten zu verwenden, war bereits bei manchen Knoten in YAHOO! PIPES gegeben [SHT+07], wurde dort aber nicht explizit auf generische Weise als Teil des Konzepts definiert.

Das Verbinden mehrerer Flüsse mit einem einzelnen Rezeptor hat nicht denselben Effekt, da auf diese Weise die Vereinigungsmenge aller ankommenden Datenmengen gebildet wird, die als ein einzelner Parameter in den Knoten eingeht (Knoten v_7 in Abbildung 3.2). Dies ist an jedem Rezeptor möglich; es sind keine speziellen Vereinigungsknoten erforderlich, wie dies im von Bosch beschriebenen Konzept der Fall ist [Bos15, 24f.]. Andere (nicht kommutative) Mengenoperationen wie eine Schnittmenge lassen sich wiederum durch mehrere Rezeptoren ausdrücken, indem die an einem Rezeptor anliegende Datenmenge mit der Datenmenge von den anderen Rezeptoren geschnitten wird.

Durch die Verwendung mehrerer Emitter können Filterfunktionen, die im FFK durch mehrere Knoten ausgedrückt werden mussten, zu einem Knoten zusammengefasst werden (Abbildung 3.3). Dies ist dann der Fall, wenn mehrere Knoten jeweils unterschiedliche Variationen von im Grunde derselben Filterfunktion ausführen, beispielsweise das negierte und das nicht negierte Ergebnis einer Vergleichsoperation (Abbildung 3.3a). Im EFFK können mehrere Emitter – ein negierter und ein nicht negierter – im selben Knoten bereitgestellt werden, was die Gesamtzahl der notwendigen Knoten reduziert (Abbildung 3.3b).

3.1.2.2 Räumliches Layout

An Stelle der linearen Anordnung, bei welcher der FFK-Graph immer in dieselbe Richtung erweitert wird [YS93; MPG98; MAG+04], geht das

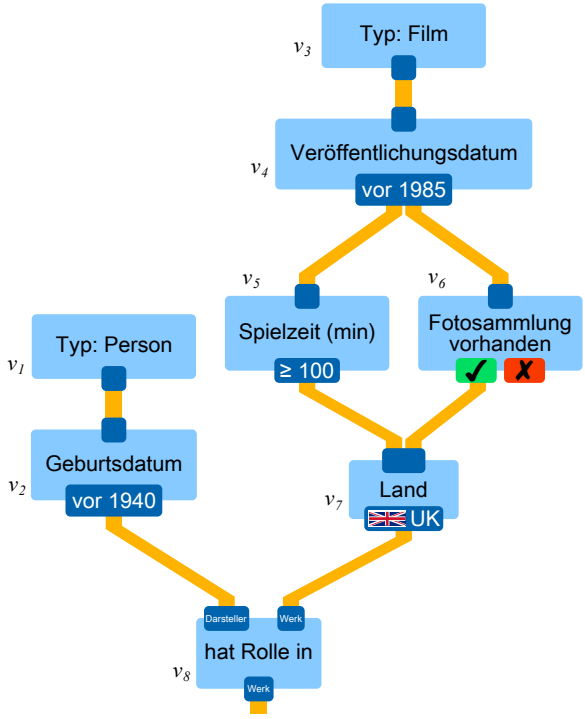
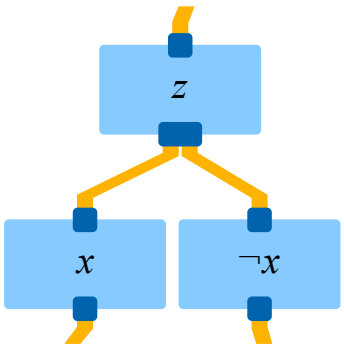


Abbildung 3.2: Ein EFFK-Graph. Der Knoten v_8 kann durch das Vorhandensein mehrerer Rezeptoren eine Filterfunktion mit zwei Parametern unterstützen: Der linke Knoten nimmt den ersten Parameter in Form der Menge von Darstellern, die in einem der zu filternden Werke mitgespielt haben müssen, entgegen. Der rechte Knoten erwartet die Menge der zu filternden Werke als zweiten Parameter. Im Gegensatz dazu werden zwei Flüsse in denselben Rezeptor von v_7 geleitet. Dadurch entsteht eine Vereinigungsmenge; die beiden von v_5 und v_6 gefilterten Datenmengen werden von v_7 nicht getrennt voneinander verarbeitet.

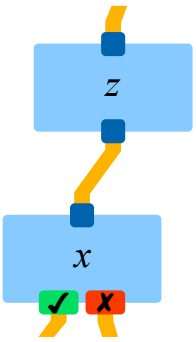
EFFK von frei platzierbaren Filterknoten aus. Diese frei platzierbaren Knoten können vom Benutzer beliebig angeordnet werden, wobei der Verlauf der Flüsse entsprechend angepasst werden kann, um die Knoten zu verbinden, wie dies in FindFlow [HSM+06] und einigen weiteren verwandten Arbeiten [EST08; SHT+07; TIC09; JGZ+11] der Fall ist. Auf diese Weise erhalten Benutzer die Möglichkeit, die Filterknoten passend zu ihrem mentalen Modell anzuordnen und zu gruppieren.

3.1.2.3 Strukturelle Mächtigkeit

Jeder FFK-Graph lässt sich verlustfrei in einen EFFK-Graphen transformieren.



(a) Hat jeder Knoten nur einen Emitter, so muss die Bedingung x zweimal ausgedrückt werden (einmal davon negiert).



(b) Bei Knoten mit mehreren Emittern genügt es, die gemeinsame Bedingung x einmal auszudrücken und die Unterschiede (Ergebnis *wahr* oder *falsch*) über die Emitter zu unterscheiden.

Abbildung 3.3: Die Unterstützung mehrerer Emitter an einem Knoten erlaubt eine Reduktion der Anzahl an Filterknoten. In den hier gezeigten Beispielen wird jeweils ein Fluss so aufgeteilt, dass einerseits die Bedingung $z \wedge x$ und andererseits die Bedingung $z \wedge \neg x$ erfüllt ist.

Ein FFK-Graph ist ein zusammenhängender, gerichteter Graph

$$G = (V, E)$$
$$E \subseteq V \times V$$

wobei V die Menge der Knoten und E die Menge der Kanten darstellt.

Im FFK [YS93] existiert zudem die Beschränkung auf nur einen Start- und Endknoten (siehe Abschnitt 3.1.1). Dazu muss

$$\begin{aligned}
&\exists v_s \in V \nexists v_0 \in V : \exists (v_0, v_s) \in E \\
&\qquad\qquad \wedge (\forall v_1 \in V : v_1 \neq v_s \Leftrightarrow (\exists v_2 \in V : (v_2, v_1) \in E)) \\
&\exists v_e \in V \nexists v_0 \in V : \exists (v_e, v_0) \in E \\
&\qquad\qquad \wedge (\forall v_1 \in V : v_1 \neq v_e \Leftrightarrow (\exists v_2 \in V : (v_1, v_2) \in E))
\end{aligned}$$

gelten. Für das EFFK entfallen diese Einschränkungen einfach, da hier beliebig viele Start- und Endknoten erlaubt sind.

Ein EFFK-Graph $\dot{G}$ ist definiert als

$$\dot{G} = (\dot{V}, \dot{E}, I, O)$$

mit Knoten $\dot{V}$ und Kanten $\dot{E}$

$$\dot{V} \subseteq \mathcal{P}(I) \times \mathcal{P}(O)$$
$$\dot{E} \subseteq O \times I$$

wobei I die Menge der Rezeptoren, O die Menge der Emitter und $\mathcal{P}(I)$ beziehungsweise $\mathcal{P}(O)$ die Potenzmenge davon darstellt. Um nun einen FFK-Graphen G in einen EFFK-Graphen $\dot{G}$ zu überführen, muss $\dot{G}$ so gefüllt werden, dass

$$\begin{aligned} I &= \{i_n | v_n \in V\} \\ O &= \{o_n | v_n \in V\} \\ \dot{V} &= \{(\{i_n\}, \{o_n\}) | v_n \in V\} \\ \dot{E} &= \{(o_n, i_m) | (v_n, v_m) \in E\} \end{aligned}$$

gilt. Für jeden Knoten v_n aus V wird also ein Rezeptor $i_n \in I$ und ein Emitter $o_n \in O$ benötigt, sodass ein Knoten $\dot{v}_n := (\{i_n\}, \{o_n\})$ definiert werden kann. $\dot{G}$ ist dann äquivalent zu G. Das EFFK ist somit abwärtskompatibel zum herkömmlichen FFK; jede mit dem FFK dargestellte Anfrage kann auch mit dem EFFK ausgedrückt werden.

3.1.2.4 Auswertung von Anfragen

Grundlegend geht die Filter/Flow-Metapher davon aus, dass über die Flüsse Datenmengen – Teilmengen der kompletten Datensammlung D – transportiert werden. Rezeptoren nehmen Datenmengen entgegen und Emitter geben Datenmengen zurück. Jede Datenmenge aus einem Emitter ergibt sich aus den flussaufwärts liegenden Datenmengen. Ein beliebiges Element aus einer Datenmenge ergibt sich jedoch nicht aus beliebigen Elementen der flussaufwärts liegenden Datenmengen, sondern aus bestimmten Elementen jener Datenmengen. Für jedes zurückgegebene Datenelement können also bestimmte Datenelemente genannt werden, die von flussaufwärts liegenden Emittern zurückgegeben werden. Eine solche Zuordnung jedes Emitters im Anfragegraphen zu einem Datenelement aus der bereitgestellten Datensammlung wird nun als *Belegung* bezeichnet.

Eine Belegung a kann über eine Funktion

$$\text{valueOf}_a : \mathcal{P}(D) \times O \to D \cup \{\varepsilon\}$$

definiert werden. Jeder von einem Emitter abgehende Fluss transportiert das Element aus dem Emitter weiter zum Rezeptor am Ende des Flusses. Wird ein Datenelement von einem Filterknoten mit dem Emitter o_α ausgefiltert, also nicht von o_α zurückgegeben, so gibt der Emitter ε zurück.

Die Auswertung eines Anfragegraphen kann man sich nun so vorstellen, dass sämtliche möglichen Belegungen ermittelt werden. Für jeden Emitter ist dabei jeweils nur der flussaufwärts erreichbare Subgraph relevant. Gibt ein Emitter also die Datenmenge $D_\alpha \subseteq D$ zurück, so geht jedes Datenelement $d \in D_\alpha$ aus mindestens einer Belegung des Anfragegraphen hervor.

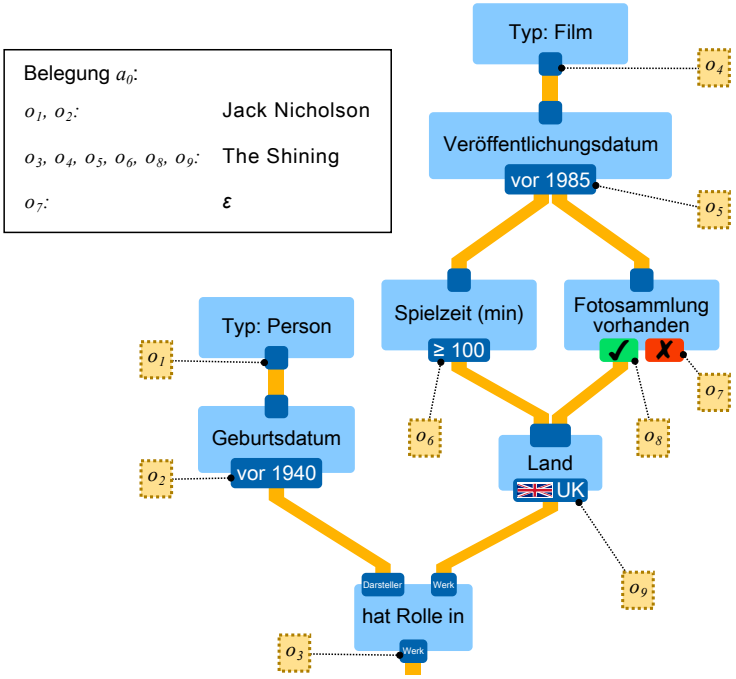


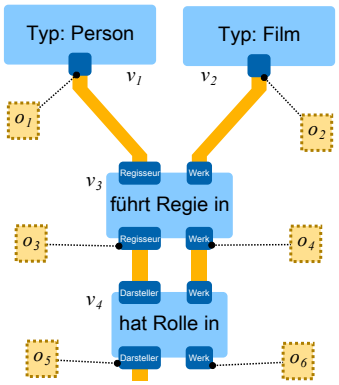
Abbildung 3.4: Der EFFK-Graph aus Abbildung 3.2, der auf einer Filmdatenbank ausgeführt wird: Eine mögliche Belegung, welche den Film „The Shining“ zur Ergebnismenge hinzufügt, ist eingetragen. (Die Belegung ist nur zur Erklärung des Konzepts eingetragen und nicht Teil der Visualisierung.)

Ein Anfragegraph mit einer solchen Belegung und einem entsprechenden Datenelement ist in Abbildung 3.4 dargestellt. Die gesamte Ergebnismenge an einem Emitter ist dann die Menge aller Werte des Emitters aus allen Belegungen außer ε.

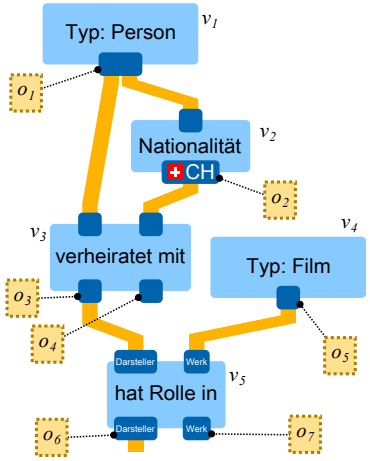
Bei der Entwicklung des EFFK wurden nun zwei Vorgehensweisen zur Auswertung der Anfragegraphen verglichen. Beide werden anhand von Belegungen erklärt. Die ursprüngliche Filter/Flow-Metapher, bei der über die Flüsse Datenmengen transportiert werden, wird von den Vorgehensweisen in unterschiedlichem Ausmaß unterstützt.

Die erste Vorgehensweise (Vorgehensweise A) ermittelt Belegungen direkt auf Grundlage des Anfragegraphen in seiner ursprünglichen Form. Dies funktioniert problemlos in dem in Abbildung 3.4 gezeigten Beispiel. Auch der in Abbildung 3.5a dargestellte Anfragegraph mit zwei parallelen Kanten, die unterschiedliche Konnektoren derselben Knoten verbinden, lässt sich auf diese Weise wie erwartet auswerten.

Die Auswertung des Graphen aus Abbildung 3.5b funktioniert jedoch nicht wie ewartet. Das besondere Merkmal ist hier, dass der Knoten v_3 zwei Rezeptoren hat, welche (indirekt) beide mit demselben Emitter verbunden



(a) o_5 liefert die Menge der Personen zurück, die in mindestens einem Film gleichzeitig als Darsteller und als Regisseur tätig waren. In jeder Belegung, in der o_5 nicht ε zugeordnet ist, liegen an den Emittern von v_3 und v_4 jeweils die Person aus o_1 und der Film aus o_2 an.



(b) o_6 sollte alle Darsteller von Filmen zurückliefern, die mit Schweizer Staatsbürgern verheiratet sind. Dem Emitter o_1 kann aber in jeder Belegung nur ein Element zugeordnet sein, welches auch von o_2 weitertransportiert wird. An v_3 liegt somit unweigerlich zweimal dasselbe Datenelement an. Da niemand mit sich selber verheiratet ist, ist o_3 und o_6 somit in jeder Belegung ε zugeordnet.

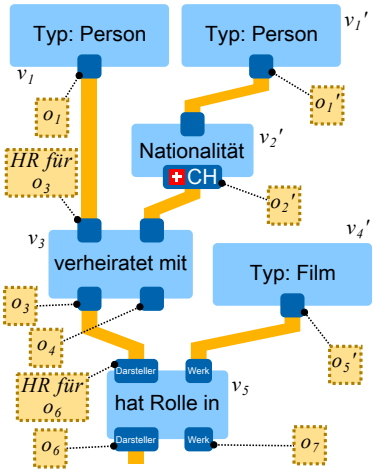
Abbildung 3.5: Anwenden von Belegungen auf den unveränderten Graphen (Vorgehensweise A).

sind. Die in v_3 gefilterte Relation *verheiratet mit* nimmt zwei Personen, also zwei Elemente aus derselben Datenmenge, entgegen. Daher ist es durchaus intuitiv und somit wünschenswert, dass man beide Rezeptoren mit derselben Datenquelle (v_1) verbinden kann. Durch die Vorgehensweise A schlägt die Auswertung jedoch fehl, da dem Emitter o_1 in jeder Belegung a genau ein Datenelement zugeordnet ist. Entweder, dieses Datenelement erfüllt nun die Einschränkung von v_2 (dann gilt $\text{valueOf}_a(D, o_2) = \text{valueOf}_a(D, o_1)$) oder es erfüllt sie nicht (dann gilt $\text{valueOf}_a(D, o_2) = \varepsilon$). An den Rezeptoren von v_3 liegt dann also entweder zweimal dasselbe Datenelement an oder einmal ε. In beiden Fällen ist o_3, und damit auch o_6, in jeder Belegung ε zugeordnet – die Ergebnismenge an o_6 ist entgegen den Erwartungen leer.

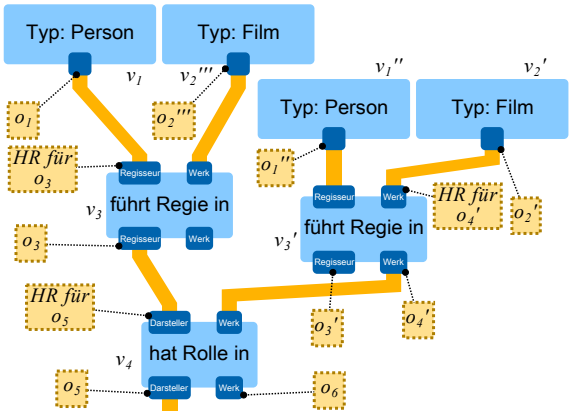
Zur Lösung dieses Problems kann eine alternative Vorgehensweise zur Auswertung des Anfragegraphen herangezogen werden (Vorgehensweise B): Bevor die Belegungen ermittelt werden, wird der Graph so umstrukturiert, dass keine parallelen Verbindungen zwischen Knoten mehr existieren. Diese Umstrukturierung findet nicht im angezeigten Anfragegraphen statt, sondern lediglich im internen Abbild der Graphstruktur im Speicher.

Für die Umstrukturierung wird festgelegt, dass für jeden Knoten mit

mindestens einem Rezeptor pro Emitter ein *Hauptrezeptor* festgelegt werden kann. Die flussaufwärts von jeglichen anderen Rezeptoren aus erreichbaren Subgraphen werden jeweils pro Rezeptor dupliziert. Somit haben die duplizierten Subgraphen garantiert keine Verbindung zu dem über den Hauptrezeptor erreichbaren Subgraphen. Auf diese Weise wird der Anfragegraph aus Abbildung 3.5b dahingehend transformiert, dass sich der in Abbildung 3.6a gezeigte Graph ergibt. Dabei sind jeweils nur die relevanten Hauptrezeptoren markiert. Die eingetragene Belegung kann dann zum erwarteten Ergebnis führen, da innerhalb einer Belegung a auch $\text{valueOf}_a(D, o_1) \neq \text{valueOf}_a(D, o_1')$ gelten kann. Somit kann v_3 an seinen Rezeptoren unterschiedliche Datenelemente erhalten.



(a) Umstrukturierter Graph aus Abbildung 3.5b: Die Auswertung funktioniert nun wie erwartet, da v_3 innerhalb einer Belegung von o_1 und o_1' (und somit von o_2') unterschiedliche Datenelemente erhalten kann.



(b) Umstrukturierter Graph aus Abbildung 3.5a: Nach der Umstrukturierung ist nicht mehr sichergestellt, dass die Personen in der Ergebnismenge von o_5 jeweils im selben Film Regie geführt und mitgespielt haben: In einer Belegung können o_2' und o_2''' unterschiedliche Datenelemente zugewiesen sein.

Abbildung 3.6: Anwenden von Belegungen auf den umstrukturierten Graphen (Vorgehensweise B): Subgraphen, die flussaufwärts an Rezeptoren außer dem Hauptrezeptor (HR) anliegen, werden vor der Auswertung dupliziert.

Vorgehensweise B kann nun auf den in Abbildung 3.5a gezeigten Graphen angewendet werden. Dabei fällt auf, dass dieser nicht mehr wie oben gezeigt funktioniert (Abbildung 3.6b). Nach der Umstrukturierung ist nicht mehr sichergestellt, dass an den *Werk*-Rezeptoren von v_3 und v_4 in jeder Belegung derselbe Film anliegt. Die Personen in der Ergebnismenge an o_5 haben also alle definitiv bei einem Film Regie geführt und definitiv in einem Film (zu dem auch ein Regisseur bekannt ist) mitgespielt. Dies muss

aber nicht notwendigerweise beim selben Film geschehen sein. Genau diese Einschränkung war aber in der ursprünglichen Anfrage aus Abbildung 3.5a beabsichtigt.

Für das EFFK wurde trotz des eben besprochenen Problems die Vorgehensweise *B* gewählt, da sie besser mit der ursprünglichen Flussmetapher zusammenpasst. Der Metapher nach repräsentieren die Kanten im Anfragegraphen Datenmengen [Shn91; JGZ+11], keine einzelnen Datenelemente. Somit ist es naheliegender, davon auszugehen, dass im Allgemeinen über mehrere Flüsse aus demselben Emitter jeweils die gesamte Datenmenge zur Verfügung steht und nicht nur ein für all diese Flüsse übereinstimmendes Element.

3.1.2.5 Berücksichtigung von Objekten

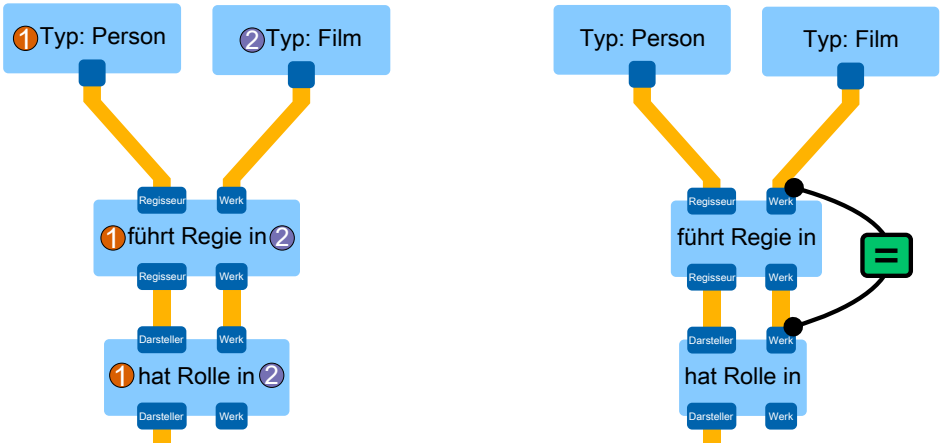
Wie oben beschrieben, ist der Anfragegraph aus Abbildung 3.5a wegen der Duplizierung der Subgraphen gleichbedeutend mit dem in Abbildung 3.6b dargestellten. Beim Filtern von Objektgraphen kann es aber tatsächlich vorkommen, dass Datenelemente mit gemeinsamen anderen Datenelementen verknüpft sind. Somit könnte durchaus ein Interesse daran bestehen, in einer Anfrage an mehreren Stellen auf dasselbe Datenelement zu verweisen. Um in der Visualisierung deutlich zu machen, dass mehrere Knoten sich auf dasselbe Datenelement beziehen, bestehen vier grundlegende Möglichkeiten:

Auslagern eines Subgraphen Die Knoten, die sich auf dasselbe Datenelement beziehen, können in einen von der restlichen Anfrage abgetrennten Subgraphen ausgelagert werden. Dies ist jedoch nicht sehr flexibel, da innerhalb dieser Subgraphen wiederum nur auf ein bestimmtes Datenelement zugegriffen werden kann. Zudem wird dabei die Flussmetapher durchbrochen, wenn der Subgraph nicht als Zwischenstück in den Hauptfluss miteingebettet ist.

Visuelle Übereinstimmung Die Knoten können mit einem gemeinsamen visuellen Merkmal versehen werden, das das gemeinsame Datenelement repräsentiert. Im einfachsten Fall ist dies ein textueller Hinweis, der wie der Name einer Variablen ausdrückt, dass an mehreren Stellen auf denselben Wert zugegriffen wird [RSB+08]. Alternativ kann dies auch eine eindeutige farbliche Markierung sein [JS09; Bol13]. Eine direkte Verbindung zwischen den Knoten, der man visuell folgen kann, besteht dadurch allerdings nicht (siehe Abbildung 3.7a).

Visuelle Verbindung der Elemente Knoten, die sich auf dasselbe Datenelement beziehen, können durch zusätzliche Kanten miteinander verbunden werden [MPG98]. Diese Kanten sind kein Teil des Flusses und stellen somit auch keinen Datenstrom dar, was wiederum von

der Flussmetapher abweicht. Zudem werden auf diese Weise „quer“ (nicht entlang von Flüssen, potenziell zwischen parallelen und somit eigentlich unverbundenen Teilgraphen) verlaufende Kanten zur Visualisierung hinzugefügt. Diese können die Darstellung unübersichtlicher machen, wie in Abbildung 3.7b angedeutet ist.



(a) Durch sich wiederholende Markierungen (hier: Farben und Zahlen) kann ein Bezug zwischen verschiedenen Knoten hergestellt werden. Eine direkte visuelle Verbindung zwischen den Knoten, die sich entlang der Flüsse verfolgen lässt, besteht so allerdings nicht.

(b) Knoten, welche auf dasselbe Datenelement zugreifen, können durch direkte Kanten verbunden werden, um die Gleichheit der herangezogenen Elemente anzuzeigen. Diese Verbindungen verlaufen jedoch nicht zwangsläufig entlang der Flüsse und brechen somit aus der Filter/Flow-Metapher aus.

Abbildung 3.7: Lösungsvorschläge zum Problem aus Abbildung 3.6b.

Visuelle Verbindung über die Flüsse Die Information, dass über mehrere Flüsse dasselbe Datenelement ausgeliefert werden soll, lässt sich auch in die Flüsse selbst einbetten. Hierzu wird ein spezieller *Synchronisierungsknoten* benötigt. Dieser wird bei der Auswertung nach Vorgehensweise B gesondert behandelt, indem er nie dupliziert wird. Um das besondere Verhalten zu verdeutlichen, bietet es sich an, den Synchronisierungsknoten und seinen Emitter visuell klar von den anderen Knoten zu unterscheiden. Der genannte Anfragegraph kann mit Hilfe dieses speziellen Knotens wie in Abbildung 3.8 gezeigt abgewandelt werden. Auf diese Weise wird erfolgreich die Mengeninterpretation der Flüsse beibehalten. Das EFFK bleibt mächtig genug, um Anfragen wie „Finde Personen, die im selben Film mitgespielt und Regie geführt haben.“ auszudrücken. Die Flussmetapher wird nicht wie in anderen Lösungen [MPG98] durchbrochen, da Knoten hier weiterhin

ausschließlich auf flussabwärts liegende Subgraphen Auswirkungen haben.

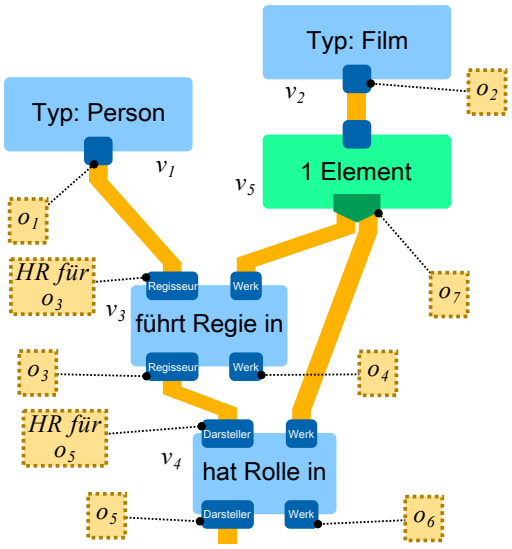


Abbildung 3.8: Korrigierter Anfragegraph aus Abbildung 3.5a: Der mit „1 Element“ beschriftete Synchronisierungsknoten wird bei der internen Umstrukturierung des Graphen nach Vorgehensweise B selbst dann nicht dupliziert, wenn er nicht an einen Hauptrezeptor (HR) angeschlossen ist. Dadurch kann erzwungen werden, dass in jeder Belegung an v_3 und an v_4 jeweils derselbe Film anliegt.

3.1.2.6 Subgraph-Ersetzung

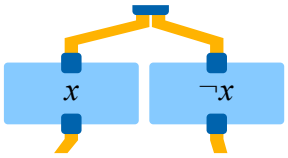
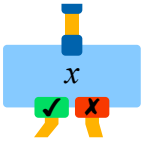
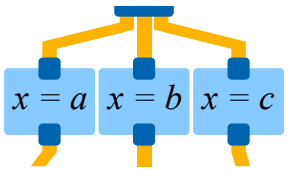
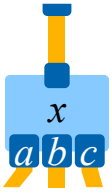
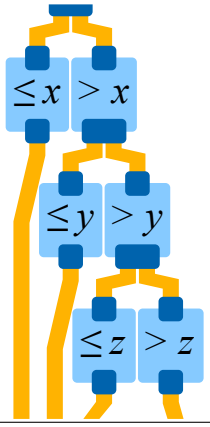
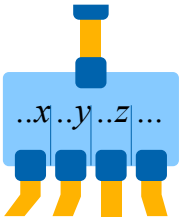
Die Eigenschaft des EFFK, dass jeder Filterknoten mehrere Emitter besitzen kann, kann dazu verwendet werden, den Graphen insgesamt kompakter zu machen. Dazu wird die Anzahl der Knoten reduziert. Dies ist möglich, indem ein Subgraph aus mehreren miteinander verbundenen Knoten durch einen neuen Knoten ersetzt wird. Jeder ursprünglich separate Knoten, von dem aus Flüsse abgehen, die aus dem zu ersetzenden Subgraphen hinausführen, wird dabei durch einen zusätzlichen Emitter am neuen Knoten repräsentiert. Die Emitter bilden dann jeweils die Filterfunktion des ursprünglichen Knotens nach. Dieses Konzept war oben bereits in Abbildung 3.3 angedeutet.

Bevor eine derartige Ersetzung durchgeführt wird, kann es notwendig sein, Subgraphen zu vertauschen. Dies ist zumindest in dem Fall möglich, in dem ein Subgraph insgesamt nur über maximal einen Rezeptor und maximal einen Emitter mit dem Rest des Filtergraphen verbunden ist. Abhängig von

den konkreten Filterfunktionen sind auch andere Verschiebungen möglich, ohne die Gesamtbedeutung des Filtergraphen zu ändern.

Die konkreten Ersetzungsmöglichkeiten hängen von der Zusammensetzung des Subgraphen und den definierten Filterknoten ab. Sie können durch Abbildungen von Subgraphmustern auf Filterknoten, die die Funktionen mehrerer anderer Knoten in sich vereinen, definiert werden. Einige Beispiele solcher Abbildungen sind in Tabelle 3.1 dargestellt.

Tabelle 3.1: Diese Tabelle zeigt mögliche Ersetzungen von FILTER/FLOW-Subgraphen durch kompaktere Knoten (angelehnt an Tabelle 1 aus Referenz [HLE12]).

Erläuterung der Ersetzung	Ursprünglicher Subgraph	Kompakterer Ersatz
Ein zusätzlicher Emitter kann das Gegenteil einer Filterbedingung ausdrücken (siehe auch Abbildung 3.3).		
Soll dieselbe binäre Operation mit mehreren unterschiedlichen rechten Operanden durchgeführt werden, lässt sich dies zu einem Knoten zusammenfassen.		
Datenmengen können auch in mehrere Kategorien aufgeteilt und entsprechend getrennt weitergeleitet werden. Die hierzu benötigten Filter lassen sich zu einem Knoten mit einem Emitter pro Wertebereich des Unterscheidungsmerkmals für die Kategorisierung kombinieren.		

3.1.3 Flüsse

Im FFK (und ebenso im EFFK) wird im Allgemeinen davon ausgegangen, dass eine einzelne Datenmenge den Graphen durchläuft und schrittweise

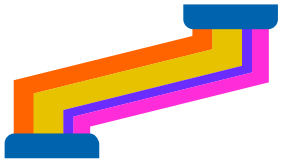


Abbildung 3.9: Im EFFK können Flüsse in parallele Bahnen aufgeteilt werden, die jeweils separate Datenmengen repräsentieren.

eingeschränkt wird. Entsprechend repräsentiert jeder Fluss die Datenmenge an der jeweiligen Stelle im Graphen, und Flüsse, die im selben Knoten zusammenlaufen, werden stets vereinigt.

Neben den Neuerungen des EFFK wird nun von einer oder mehreren Datenmengen ausgegangen, die im Filtergraphen existieren. Jeder Fluss enthält eine oder mehrere dieser Datenmengen, da Flüsse nun dahingehend erweitert werden, dass ein Fluss in mehrere parallele Bahnen unterteilt werden kann. Jede der Bahnen repräsentiert eine der im Fluss enthaltenen Datenmengen (Abbildung 3.9). Wie im FFK kann die Breite jedes Flusses dazu verwendet werden, einen Hinweis auf den Inhalt der Datenmenge, beispielsweise auf die Anzahl der Elemente, zu vermitteln. Im Fall von Flüssen mit parallelen Bahnen kann jede der Bahnen eine individuelle Breite haben. Somit können Benutzer auch bei mehreren parallel verlaufenden Datenmengen nachvollziehen, durch welche Knoten welche der Datenmengen erkennbar eingeschränkt werden. Mit dieser Möglichkeit ähnelt das EFFK dem Konzept der Parallel Sets, wo dasselbe mit parallelen Koordinaten möglich ist [KBH06].

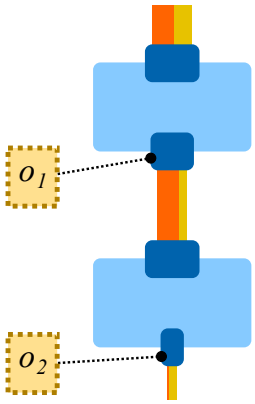
Erreicht ein in mehrere Bahnen unterteilter Fluss einen Filterknoten mit nur einem Rezeptor und nur einem Emitter, wird der Fluss vom Filterknoten im Normalfall an den Emitter weitergeleitet. Dabei werden die Datenmengen der einzelnen Bahnen im Fluss gefiltert (Abbildung 3.10a). Entsprechend werden in Ergebnisanzeigeknoten die unterschiedlichen Zwischenergebnismengen separat dargestellt (Abbildung 3.10b).

Laufen mehrere Flüsse im selben Rezeptor zusammen, sind mehrere Verfahren denkbar:

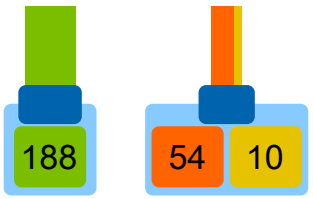
V_1: Die Vereinigungsmenge aus allen enthaltenen Datenmengen wird als neue Datenmenge weiterverarbeitet.

V_2: Datenmengen werden separat verarbeitet, wobei für Datenmengen, die in mehreren Flüssen enthalten sind, die Vereinigungsmenge betrachtet wird.

Die konkrete gewählte Strategie ist vom Filtertyp abhängig. Im Normalfall wird V_2 für den Hauptrezeptor verwendet, da es die Idee des Hauptrezeptors



(a) Parallele Bahnen in einem Fluss werden separat gefiltert. Im gezeigten Beispiel mit zwei parallel gefilterten Datenmengen entfernt o_1 vorrangig Elemente aus der rechten Datenmenge, o_2 dafür sehr viele aus der linken.



(b) Ergebnisanzeigeknoten, welche die genaue Anzahl der Elemente in Datenmengen anzeigen, sind an Flüsse angeschlossen. Im Vergleich wird sichtbar, dass bei Anschluss an einen Fluss mit mehreren Bahnen (rechts) die Anzahl der Elemente separat für jede Ergebnismenge angezeigt wird.

Abbildung 3.10: Die Bahnen in einem Fluss werden durch Filterknoten hindurch aufrechterhalten und unabhängig voneinander gefiltert.

ist, die eingehenden Datenmengen zu filtern und weiterzuleiten. Währenddessen setzen etwaige andere Rezeptoren V_1 ein. Ansonsten müsste eine Strategie zur Kombination der unterschiedlichen Datenmengen an den verschiedenen Rezeptoren gefunden werden. Wie in Tabelle 3.2 gezeigt, gäbe es hierfür zahlreiche Interpretationsmöglichkeiten, die für Benutzer unüberschaubar sein könnten. Daher wird für Rezeptoren, die nicht der Hauptrezeptor sind, gewissermaßen die einfachste Interpretation gewählt, indem alle anliegenden Datenmengen vor der Verarbeitung vereinigt werden.

Um Filtergraphen mit unterteilten Flüssen zu ermöglichen, wurde eine neue Kategorie von Spezialknoten mit strukturellen Funktionen definiert. Die Knoten dieser Kategorie lassen sich wie folgt unterteilen:

Flüsse zusammenführen Knoten dieser Art führen mehrere zuvor getrennte Flüsse zu einem einzelnen Fluss zusammen. Konkret funktioniert dies so, dass eine variable Anzahl von Rezeptoren angeboten wird. Jeder Rezeptor leitet die Vereinigung aus allen ankommenden Datenmengen an eine Bahn des Ergebnisflusses weiter. Auf diese Weise lassen sich unterschiedliche Flüsse zu einem unterteilten Fluss kombinieren, sodass die vormals separaten Datenmengen den Filtergraphen gemeinsam weiter durchlaufen können.

Bahnen trennen Dies stellt die Umkehr der Operation aus dem vorigen Knoten dar. Jede Bahn der am einzigen Rezeptor eingehenden Flüsse

Tabelle 3.2: Mögliche Interpretationen, wenn Flüsse mit mehreren Datenmengen (d_x) an mehreren Rezeptoren (i_y) anliegen. Der Rezeptor i_1 ist hier der Hauptrezeptor des Emitters o_1.

i_1	i_2	o_1
d_1, d_2	d_3, d_4	$d_{1,3}, d_{2,4}$
d_1, d_2	d_3, d_4	$d_{1,3}, d_{1,4}, d_{2,3}, d_{2,4}$
d_1, d_2	d_3	$d_{1,3}$
d_1, d_2	d_3	$d_{1,3}, d_{2,3}$
d_1, d_2	d_3, d_4, d_5	$d_{1,3}, d_{2,4}$
d_1, d_2	d_3, d_4, d_5	$d_{1,3}, d_{1,4}, d_{1,5}, d_{2,3}, d_{2,4}, d_{2,5}$
d_1, d_2	d_3, d_2	$d_{1,3}, d_2$
d_1, d_2	d_2, d_3	$d_{1,3}, d_2$
d_1, d_2	d_2, d_3	$d_{1,2}, d_{2,3}$
...		

wird an einen anderen Emitter weitergeleitet. Die Bahnen der Flüsse werden also aufgetrennt und können anschließend unabhängig voneinander weitergefiltert werden.

Datenmenge aufspalten Diese Art von Knoten dienen dazu, Datenmengen basierend auf beliebigen Kriterien neu zu gruppieren beziehungsweise Flüsse neu zu unterteilen. Im Allgemeinen wird dazu ein Selektor festgelegt, der die Elemente der eingehenden Datenmengen in unterschiedliche Gruppen unterteilt, welche dann jeweils eine Bahn im abgehenden Fluss bilden. Da sich je nach Datenmenge und Anwendungsfall unterschiedlichste Gruppierungskriterien definieren lassen, ist diese Gruppe von Knoten auch nicht begrenzt. Je nach Einsatzzweck können beliebige Knoten zum Aufteilen von Datenmengen vorgesehen werden.

Im Zusammenhang mit dem Hauptrezeptor ist hierbei zu beachten, dass für jede Datenmenge ein anderer Rezeptor als Hauptrezeptor fungieren kann. Auf diese Weise kann der Knoten zum Zusammenführen von Flüssen verwirklicht werden – mit jedem Rezeptor wird eine Datenmenge assoziiert, aus dem einzigen Emitter tritt ein Fluss mit einer Bahn für jede der Rezeptordatenmengen aus, und für jede dieser abgehenden Datenmengen gilt der jeweilige Rezeptor als Hauptrezeptor (Abbildung 3.11). Ein Beispiel einer vollständigen Anfrage mit parallelen Bahnen ist in Abbildung 3.12 dargestellt.

Bei der Auswertung eines Anfragegraphen können die unterschiedlichen Datenmengen an sich vollständig ausgeblendet werden: Für jede Datenmenge an einem Emitter spielt eine (nicht notwendigerweise echte) Teilmenge

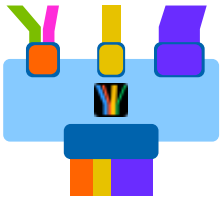


Abbildung 3.11: Der Knoten zum Zusammenführen von Flüssen zu parallelen Bahnen in einem Fluss wandelt die bei jedem Rezeptor eingehende Datenmenge jeweils in eine separate Bahn um. Zur Verdeutlichung sind die Rezeptoren in den selben Farben wie die resultierenden Bahnen markiert.

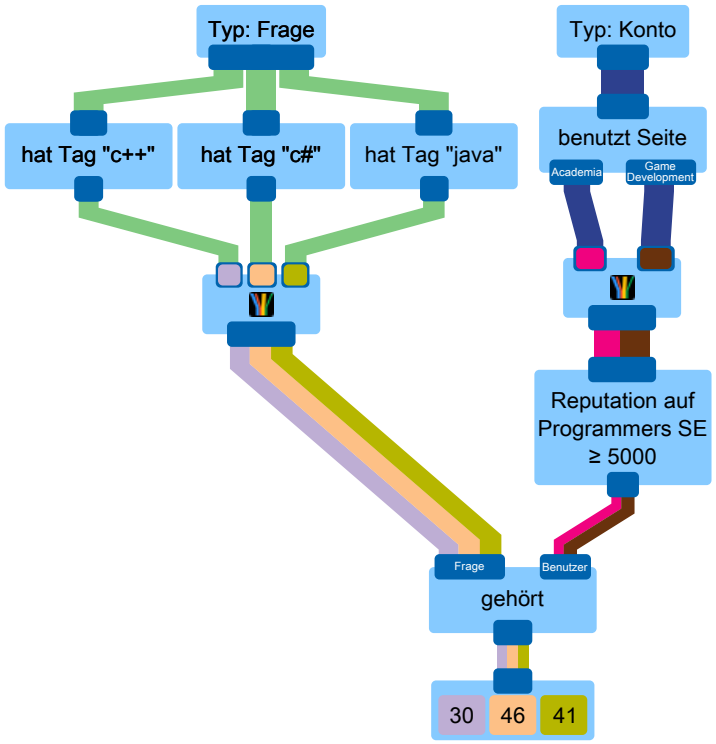


Abbildung 3.12: Ein Anfragegraph mit parallel ausgewerteten Datenmengen: Auf der Frage-und-Antwort-Seite Programmers Stack Exchange wird separat nach Fragen gesucht, die als einer von drei Programmiersprachen zugehörig gekennzeichnet sind, und die von einem Autor mit mindestens 5000 Punkten in Programmers Stack Exchange sowie Benutzerkonten auf den Seiten Academia Stack Exchange und Game Development Stack Exchange desselben Frage-und-Antwort-Netzwerks verfasst wurden. Als Ergebnis wird die Anzahl der jeweils gefundenen Fragen gezeigt. Die Fluss- und Bahnbreiten im gesamten Graphen sind proportional zum gerundeten Logarithmus aus der Anzahl der Zwischenergebnisse.

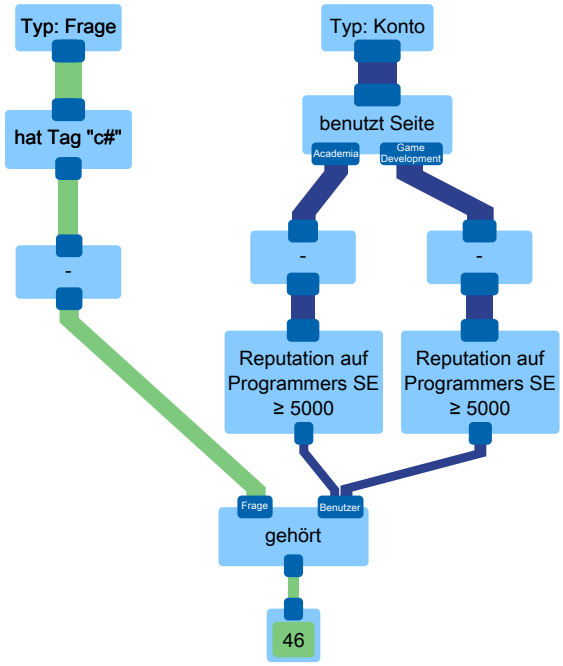


Abbildung 3.13: Dieser Graph wurde aus dem Anfragegraphen in Abbildung 3.12 extrahiert. Es sind lediglich die Graphbestandteile erhalten, die zur Auswertung der Datenmenge, die Fragen mit dem Stichwort „c#“ enthält, notwendig sind.

der eingehenden Datenmengen eine Rolle. Dies gilt transitiv für den gesamten Graphen. Darum lässt sich für jede gegebene Datenmenge an einem gegebenen Emitter ein Subgraph aus dem kompletten Anfragegraphen extrahieren. Dieser Subgraph enthält nur die Teile des Anfragegraphen, die für die betrachtete Datenmenge am betrachteten Emitter relevant sind.

Für Rezeptoren, die nach dem oben beschriebenen Verfahren V_1 behandelt werden, werden die von eingehenden Flüssen mit mehreren Datenmengen erreichbaren Subgraphen vervielfältigt. Auf diese Weise findet die Vereinigung der Flüsse am betrachteten Rezeptor statt. Jeder Fluss beherbergt dabei nur eine Datenmenge.

Für Rezeptoren, die nach Verfahren V_2 behandelt werden, passiert dasselbe, wobei aber Datenmengen, die keinen Einfluss auf die resultierende Datenmenge haben, entfernt werden. Abbildung 3.13 zeigt ein einfaches Beispiel dieses Konzepts. Nach der Extraktion eines solchen Graphen ohne parallele Bahnen kann zur Auswertung wie gehabt Vorgehensweise B aus Abschnitt 3.1.2.4 angewandt werden.

3.1.4 Abbildung auf SPARQL

SparqlFilterFlow ist eine Abbildung des EFFK auf die Anfragesprache SPARQL (siehe Abschnitt 2.2.2). Dabei werden lediglich die Filtereinschränkungen für die SPARQL-Anfrage visualisiert. Sonstige mit der Anfrage in Zusammenhang stehende Informationen sind nicht Teil der Visualisierung. Unter anderem erfolgt keine visuelle Repräsentation der Art der Anfrage oder der als Ergebnisspalten projizierten Variablen. Diese Anfragedetails werden von den Ergebnisknoten individuell festgelegt und wirken sich somit nicht auf die eigentlichen Anfrageeinschränkungen aus.

Die für SparqlFilterFlow definierten Knoten lassen sich in drei Gruppen unterteilen, die im Folgenden beschrieben werden.

3.1.4.1 Datenfilterknoten

Die Gruppe der grundlegenden Filter führt Vergleiche von Attributwerten in Datenelementen durch. Filter dieser Gruppe entfernen Datenelemente aus der Ergebnismenge, welche die Filterkriterien nicht erfüllen. Diese Vergleiche beziehen sich auf Attributwerte, die in den Datenelementen gespeichert sind. Dafür wird meist ein Eigenschaftspfad angegeben, der relativ zum Datenelement ausgewertet wird. Filter in dieser Gruppe schließen Vergleichsknoten für ordinale Werte (wie beispielsweise Zahlen oder Datumsangaben, mit Operatoren wie *gleich* oder *kleiner*, vergleiche Abbildung 3.14), Zeichenketten und IRIs ein. Neben den eigentlichen Werten können sich die Filter auch auf andere Eigenschaften von Literalen beziehen. Dies betrifft beispielsweise die Sprachauszeichnung oder die Anzahl der im Literal enthaltenen Zeichen. Die konkrete Darstellung der Filterknoten richtet sich nach den zu filterenden Daten. Je nach Anwendungsfall können dabei gleichermaßen spezielle Filter vorgesehen werden, ähnlich wie dies in anderen Filtervisualisierungen der Fall ist [Shn94; SGJ+13].

3.1.4.2 Strukturfilterknoten

Mit der zweiten Gruppe von Filtern können nicht nur explizit im Graph gespeicherte Daten analysiert werden, sondern die Graphstruktur selbst. Daher macht sich diese Gruppe von Filtern eine inhärente Eigenschaft von Linked Data und insbesondere SPARQL zunutze, die in anderen Datenstrukturen wie relationalen Datenbanken gewöhnlich nicht verfügbar ist. In Linked Data existieren keine separaten Schemainformationen, die benötigt werden, um Daten aus einem RDF-Graphen abzurufen. Die eigentlichen Schemainformationen – welche Eigenschaften sind für ein gegebenes Element hinterlegt, wie viele Werte sind für jede der Eigenschaften gespeichert – sind implizit in den Ergebnissen zu beliebigen SPARQL-Anfragen ent-

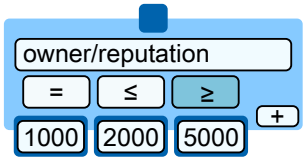


Abbildung 3.14: Ein Filter für ordinale Werte aus SparqlFilterFlow, der für den Vergleich der Punkte eines Frage- oder Antwortschreibers („owner“) im Netzwerk Stack Exchange über die Operation *größer oder gleich* mit den Werten *1000*, *2000* und *5000* konfiguriert ist. Im Gegensatz zu den meisten anderen Abbildungen in diesem Kapitel ist diese Abbildung nur wenig von der tatsächlichen Darstellung in einer Implementierung abstrahiert. Auf diese Weise wird der Umgang mit diesem Knotentyp in SparqlFilterFlow verdeutlicht. Daher sind in diesem Fall auch Interaktionselemente wie Eingabefelder und Schaltflächen erkennbar.

halten. So können auf dieselbe Weise, wie Kriterien für Attributwerte im RDF-Graphen aufgestellt werden können, ebenfalls Kriterien für das Schema definiert werden.

Filterknoten, die die Existenz eines Werts für eine gegebene Eigenschaft überprüfen, gehören dieser Gruppe an. Weitere denkbare Filtertypen, die sich in diese Gruppe einordnen ließen, prüfen, aus welchen Ontologien Eigenschaften eines gegebenen Datenelements entstammen. Ebenso können sie Einschränkungen auf die Anzahl der Werte einer gegebenen Eigenschaft anwenden.

3.1.4.3 Steuerungsknoten

Die dritte Knotengruppe führt im engeren Sinn eigentlich keine Filterfunktion aus. Stattdessen hilft sie dabei, die Flüsse des Anfragegraphen umzulenken und umzustrukturieren. Reine Layout-bezogene Hilfsknoten, welche Flüsse vereinigen und somit die Anzahl langer Kanten reduzieren können, bevor die Flüsse an ihrem endgültigen Zielknoten ankommen, gehören dieser Gruppe an. Ebenso zählen die in Abschnitt 3.1.3 erwähnten Knoten zum Aufspalten von Flüssen dazu.

3.1.4.4 Einschränkungen

Ob sich eine gegebene Anfrage mit dem EFFK ausdrücken lässt, hängt zunächst davon ab, ob geeignete Filterknoten zur Verfügung stehen, und weniger von der Mächtigkeit des Konzepts an sich. So können beispielsweise mit dem *Regie*-Knoten aus Abbildung 3.8 neben Regisseuren nur dann auch Werke abgerufen werden, wenn der Knoten einen entsprechenden Emitter anbietet.

Sehr umfangreiche Anfragen führen auch im EFFK zu einem großen Anfragegraphen. Das EFFK verbessert hier durch die geringere Knoten- und Kantenzahl die Situation gegenüber dem ursprünglichen Modell. Zudem ist es nicht unbedingt notwendig, einen Filter/Flow-Anfragegraphen als Ganzes zu verstehen. Stattdessen können Benutzer die einzelnen Pfade verfolgen. Auf diese Weise können sie selbst bei einem sehr großen Graphen noch nachvollziehen, warum bestimmte Datenelemente auf eine bestimmte Weise gefiltert werden. Soll die Anfragegraphgröße dennoch verringert werden, kann in Umsetzungen des EFFK vorgesehen werden, Subgraphen manuell zu einzelnen benutzerdefinierten Filterfunktionsknoten zusammenzufassen. Diese zuletztgenannte Komprimierung war bereits für das FFK vorgesehen [YS93].

3.1.5 Evaluation der kompakten Darstellung

Um sicherzustellen, dass die kompaktere Darstellung keine Nachteile für die Lesbarkeit der Filter/Flow-Graphen mit sich bringt, wurde eine Benutzerstudie durchgeführt. Die Filter/Flow-Visualisierung könnte insbesondere dann hilfreich sein, wenn die Anfrage zu komplex ist, als dass man sie in einfacher Textform nachvollziehen könnte. Daher wurden in der Studie nur FFK-Graphen mit mindestens 32 Knoten und 41 Kanten und entsprechende EFFK-Graphen mit mindestens 16 Knoten und 30 Kanten verwendet.

In der Studie sollten Filtergraphen, die mit der in Abschnitt 3.1.1 beschriebenen Interpretation des FFK dargestellt waren, mit Repräsentationen derselben Graphen im EFFK verglichen werden. Dazu wurden einige Filtergraphen und Datenelemente gezeigt. Studienteilnehmer sollten in den Filtergraphen ablesen, welche Datenelemente in welche Ergebnismengen gelangen können. Dieser Ansatz der Evaluation findet sich in verwandten Arbeiten wieder, wo für eine gegebene Filtervisualisierung entschieden werden musste, welche Datenelemente die dargestellten Einschränkungen erfüllen [YS93; MC10; JGZ+11].

Die folgenden Aspekte wurden in dieser Studie evaluiert:

- Kombination mehrerer Knoten zu einem Knoten (siehe Abschnitt 3.1.2.6)
- mehrere Emitter an einem Knoten (siehe Abschnitt 3.1.2.1)
- unterschiedliche Filtereinstellungen abhängig vom Emitter (siehe Abschnitt 3.1.2.1)

Da EFFK-Graphen mit FFK-Graphen mit demselben Aussagegehalt verglichen werden sollten, wurde insbesondere auf die folgenden Aspekte *nicht* eingegangen:

- mehrere Rezeptoren an einem Knoten (siehe Abschnitt 3.1.2.1), da dies gegenüber dem ursprünglichen Modell die Mächtigkeit erhöht und sich somit nicht äquivalent in beiden Modelle ausdrücken lässt
- parallele Bahnen in Flüssen (siehe Abschnitt 3.1.3), da auch hierzu kein Gegenstück im FFK existiert
- Synchronisierungsknoten (siehe Abschnitt 3.1.2.5), da im FFK mit nur einem Rezeptor und nur einem Emitter pro Knoten kein Anwendungsfall dafür auftritt

Um die Aufgabeninhalte allgemein verständlich zu halten, wurden in der Studie einfache Szenarien beschrieben, wie das Finden geeigneter Autos oder Wohnungen. Filter/Flow-Graphen könnten auch in komplexen Anwendungsgebieten im Zusammenhang mit technischen oder wissenschaftlichen Datenbanken nützlich sein. Dennoch wurden in der Evaluation allgemeinverständliche Domänen verwendet, um Verzerrungen der Studienergebnisse durch inhaltliche Missverständnisse zu vermeiden. Diese Vorgehensweise wird häufig zur Auswertung visueller Anfragetechniken gewählt, für die zum Teil Datensammlungen mit Hotels [JGZ+11], Grundstücken [HSM+06], Beschäftigten eines Unternehmens [YS93] oder Digitalkameras [EST08] herangezogen werden.

3.1.5.1 Teilnehmer

Die Studie wurde mit 28 Teilnehmern (21 männlich, 7 weiblich) im Alter von 19 bis 30 Jahren ($M = 24,5$ Jahre) durchgeführt, die auf dem Campus einer technisch ausgerichteten Universität rekrutiert wurden. Alle Teilnehmer studierten oder arbeiteten zum Zeitpunkt der Studie in einem technischen oder naturwissenschaftlichen Fachgebiet, wie beispielsweise Luft- und Raumfahrttechnik, Umweltschutztechnik, Architektur, Informatik, Mathematik oder Physik. Diese Auswahl erlaubte es, zu beobachten, wie Menschen aus unterschiedlichen Fachgebieten mit Filter/Flow-Graphen zurechtkommen. In allen einbezogenen Fachgebieten muss mit großen Datenmengen gearbeitet werden und Menschen, die in diesen Gebieten arbeiten, könnten komplexe Suchanfragen benötigen, um sich in diesen Datenmengen zurechtzufinden.

22 der Teilnehmer gaben an, zumindest über grundlegende Programmierkenntnisse zu verfügen. Drei weitere besaßen Vorwissen zu Boolescher Logik oder elektrischen Schaltungsdiagrammen, welche eine gewisse Verwandtschaft zur Filter/Flow-Visualisierung aufweisen. Da heutzutage praktisch alle technologieorientierten Studiengänge eine grundlegende Informatikausbildung beinhalten, entsprach dieses häufige Vorkommen von geringem Vorwissen zur Programmierung und zur Booleschen Logik den Erwartungen und wurde als repräsentativ für die Zielgruppe angesehen.

3.1.5.2 Materialien und Aufbau

Für die Studie wurden sechs Anfragen an fiktive Datensammlungen vorbereitet. Jede der Anfragen enthielt zwischen zwei und vier Ergebnismengen. Jede der Anfragen wurde als Graph nach dem FFK sowie als Graph nach dem EFFK dargestellt. Die Darstellung wurde dabei stets durch eine prototypische Implementierung erzeugt (siehe Abschnitt 3.1.7). Aus diesem Grund enthielten einige der Knoten auch sichtbare Steuerelemente aus der interaktiven Benutzerschnittstelle des Prototypen. Dies war ein realistischer Aspekt der in der Studie verwendeten Grafiken. Filter/Flow-Graphen könnten im praktischen Einsatz ebenfalls in interaktive Benutzerschnittstellen eingebettet sein. Entsprechend sind in vergleichbaren Studien in den gezeigten Visualisierungen Steuerelemente der Benutzerschnittstelle sichtbar [TIC09; Sei11]. Zudem existierte zu jeder Anfrage eine kurze Beschreibung des jeweiligen fiktiven Alltagsszenarios.

Die in der Studie verwendeten FFK-Graphen entsprachen dem in Tabelle 3.3 dargestellten Mengengerüst. Die unterschiedlichen Graphgrößen resultieren aus der kompakteren Darstellungsweise des EFFK, während die inhaltliche Komplexität der Graphen jeweils übereinstimmte. Die Graphen wurden auf einem 22-Zoll-TFT-Bildschirm mit einer Bildschirmauflösung von 1920 × 1200 Pixeln im Vollbildmodus angezeigt. Ein Beispiel übereinstimmender Graphen, die in der Studie verwendet wurden, ist in den Abbildungen 3.15 und 3.16 dargestellt.

Tabelle 3.3: Größenordnung der Graphen in der Benutzerstudie zum EFFK. Für das FFK und das EFFK gelten wegen der kompakteren Darstellungsweise des letzteren Modells jeweils unterschiedliche Zahlen.

	FFK		**EFFK**	
	Knoten	**Kanten**	**Knoten**	**Kanten**
Minimum	32	41	16	30
Maximum	53	100	29	44
Mittelwert	39,7	66	21,2	34,7

Für jede Anfrage wurden zehn Datenelemente vorbereitet. Für jedes Datenelement wurde jedem in der Anfrage vorkommenden Attribut ein Wert zugewiesen. Diese Werte wurden teilweise willkürlich gewählt, teilweise im Hinblick darauf, dass jeder Pfad durch den Anfragegraphen von mindestens einem Datenelement abgedeckt wird. Die steckbriefartigen Beschreibungen dieser Datenelemente wurden auf Papierkarten gedruckt. Ein Beispiel zweier solcher Karten befindet sich in Abbildung 3.17.

Jedem Teilnehmer wurden sowohl FFK- als auch EFFK-Graphen gezeigt. Auf diese Weise war es möglich, um einen Vergleich der zwei Modelle zu bitten. Jeder Studienteilnehmer bekam den FFK-Graphen für drei der

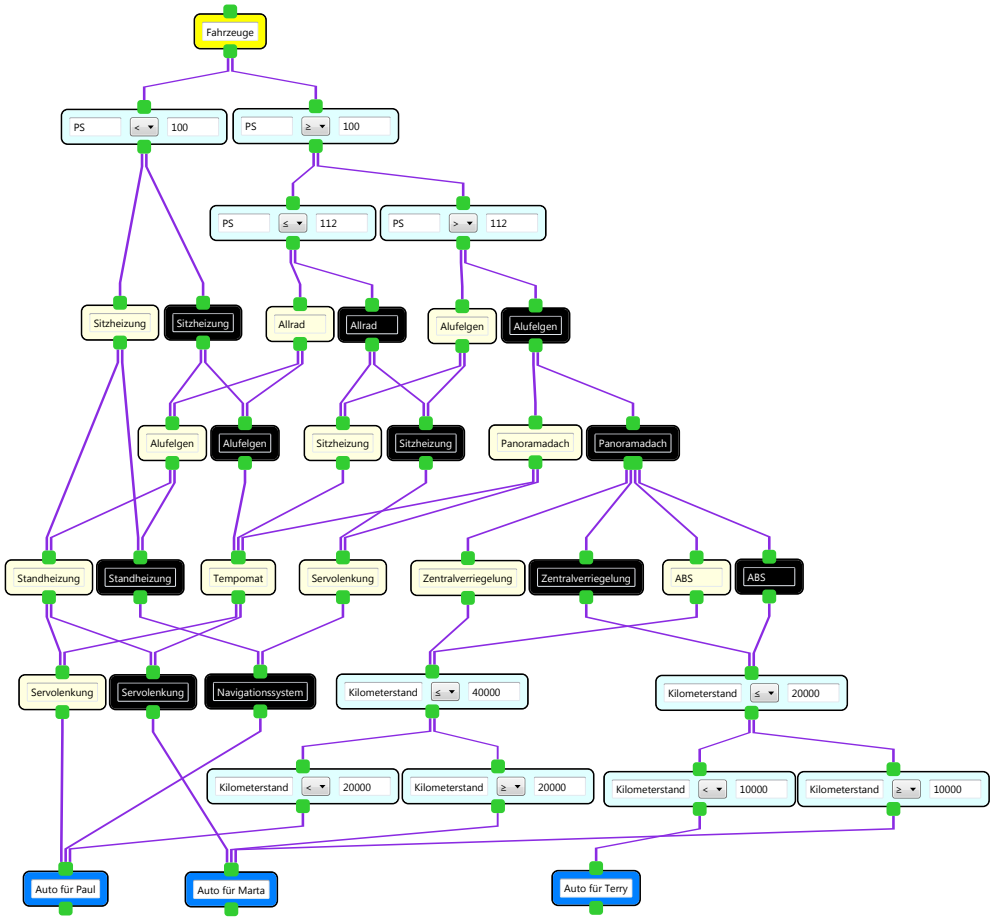


Abbildung 3.15: FFK-Graph aus der Benutzerstudie (siehe Abschnitt 3.1.5). Diese Grafik wurde als Begleitmaterial zusammen mit der Veröffentlichung zur Benutzerstudie [HLE13b] eingereicht.

sechs Anfragen zu sehen und den EFFK-Graphen für die anderen drei. Kein Teilnehmer erhielt jedoch beide Darstellungsformen derselben Anfrage. Die Teilnehmer wurden dazu einer von zwei Gruppen zugeteilt. Dadurch wurden auch Carryover-Effekte durch das Erinnern an Anfragen vermieden.

Zudem wurde die Reihenfolge ausbalanciert, in der die Teilnehmer die zwei Modelle zu sehen bekamen. Eine Untergruppe der Teilnehmer begann mit den drei FFK-Graphen, die andere mit den drei EFFK-Graphen. Jeder Teilnehmer musste insgesamt 30 Fragen beantworten, wobei jede der Fragen eine der oben genannten Datenelementkarten einbezog. Die Reihenfolge der Anfragen und der Datenelementkarten war vom Zufall bestimmt. Dasselbe gilt für die Auswahl der (jeweils fünf aus insgesamt 10) Datenelementkarten für jede der Anfragen.

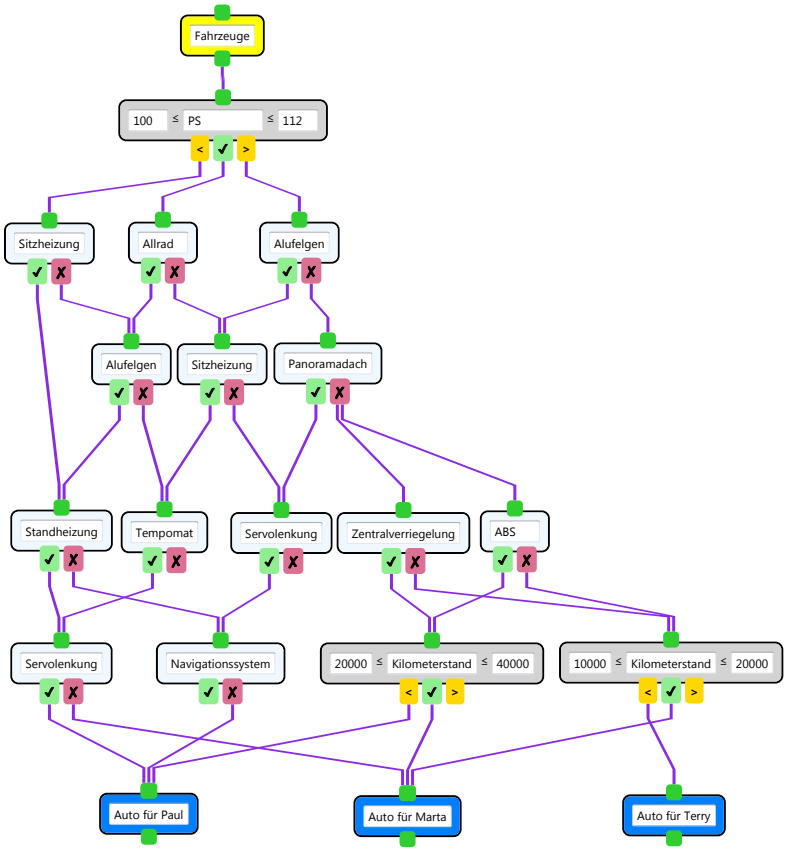


Abbildung 3.16: EFFK-Graph aus der Benutzerstudie (siehe Abschnitt 3.1.5), welcher dieselbe Filterung repräsentiert wie der in Abbildung 3.15 dargestellte FFK-Graph. Diese Grafik wurde als Begleitmaterial zusammen mit der Veröffentlichung zur Benutzerstudie [HLE13b] eingereicht.

Zusammenfassend betrachtet wurde ein 2 × 4 „Mixed Design“[1] angewandt. Der Modelltyp (FFK oder EFFK) fungierte als unabhängige Variable, die nach dem Within-Subjects-Prinzip[2] untersucht wurde. Die vier Fälle, die sich aus der Balance durch die Reihenfolge der Modelle und der Anfragen ergaben, wurden nach dem Between-Subjects-Prinzip[3] getestet.

[1] Das bedeutet, das Within-Subjects-Prinzip und das Between-Subjects-Prinzip (siehe jeweils folgende Fußnoten) wurden kombiniert.

[2] Beim Within-Subjects-Prinzip kommt jeder Teilnehmer mit allen Fällen in Bezug auf eine Studienvariable in Berührung. Der Vergleich zwischen Beobachtungen bei unterschiedlichen Fällen schließt also auch den Vergleich zwischen Beobachtungen ein und desselben Teilnehmers ein.

[3] Beim Between-Subjects-Prinzip erlebt jeder Teilnehmer während der Studie nur einen der Fälle in Bezug auf eine Variable. Der Vergleich zwischen Beobachtungen bei unterschiedlichen Fällen ist also immer auch ein Vergleich zwischen Beobachtungen unterschiedlicher Teilnehmer.

Nummer: 1/1c-VehicleFeatures

Fahrzeug: VW Polo Classic 75

- **ABS:** Ja
- **Allrad:** Nein
- **Alufelgen:** Nein
- **Einparkhilfe:** Nein
- **Kilometerstand:** 14293
- **Kurvenlicht:** Nein
- **Navigationssystem:** Nein
- **Nebelscheinwerfer:** Ja
- **Panoramadach:** Ja
- **PS:** 75
- **Servolenkung:** Ja
- **Sitzheizung:** Nein
- **Standheizung:** Ja
- **Tempomat:** Nein
- **Zentralverriegelung:** Ja

(a) Dieses Datenelement gelangt in keine der Ergebnismengen und kommt somit für keinen der Kunden in Frage.

Nummer: 3/1c-VehicleFeatures

Fahrzeug: Honda Civic Aero Deck

- **ABS:** Nein
- **Allrad:** Nein
- **Alufelgen:** Nein
- **Einparkhilfe:** Ja
- **Kilometerstand:** 15783
- **Kurvenlicht:** Nein
- **Navigationssystem:** Ja
- **Nebelscheinwerfer:** Nein
- **Panoramadach:** Nein
- **PS:** 114
- **Servolenkung:** Ja
- **Sitzheizung:** Ja
- **Standheizung:** Nein
- **Tempomat:** Nein
- **Zentralverriegelung:** Nein

(b) Dieses Datenelement gelangt in zwei der drei Ergebnismengen, da es für „Paul“ und für „Marta“ in Betracht kommt.

Abbildung 3.17: Zwei Datenelementkarten aus der Benutzerstudie (siehe Abschnitt 3.1.5), passend zu den Grafiken aus den Abbildungen 3.15 und 3.16. Das Szenario, das gegenüber den Studienteilnehmern erläutert wurde, besteht darin, dass drei Personen mit unterschiedlichen Präferenzen ein Auto aussuchen möchten.

Die Teilnehmer wurden nach dem Zufallsprinzip einem der vier Fälle zugeteilt. Die abhängigen Variablen waren die Zeit und die Korrektheit der Anfragen sowie die vom Benutzer geäußerte Vorliebe für eines der Modelle.

3.1.5.3 Durchführung

Die Studie wurde immer nur für einen Teilnehmer gleichzeitig durchgeführt. Zunächst wurde eine gedruckte Einleitung zu den generellen Aspekten des Filter/Flow-Konzepts bereitgestellt. Diese Einleitung verwendete einen „neutralen“[4] Filter/Flow-Graphen. Er bestand nur aus einfachen Knoten (keine Negationen, keine duplizierten oder anderweitig spezielle Filterknoten). Die Teilnehmer wurden aufgefordert, bei Unklarheiten nachzufragen. Vier einleitende Beispiele wurden präsentiert. Auf diese Weise konnten sich die Teilnehmer an die Durchführung der Aufgaben gewöhnen. Für diese Vorbereitung wurde ein weiterer neutraler Filter/Flow-Graph auf dem Bildschirm angezeigt.

Für jeden Teilnehmer teilte sich die Studie in zwei Abschnitte auf, einen für FFK und einen für EFFK-Graphen. Zu Beginn jedes solchen Abschnitts erhielten die Teilnehmer eine Beschreibung der Besonderheiten der jeweiligen Visualisierung. Anschließend mussten die Teilnehmer je fünf Aufgaben für jeden der drei gezeigten Anfragegraphen lösen. Für jede Aufgabe erhielten sie eine Datenelementkarte. Nachdem sie einen Moment lang einen Überblick über die auf der Karte aufgelisteten Attribute und Werte gewonnen hatten, wurde ein Anfragegraph auf dem Bildschirm gezeigt. Die Teilnehmer mussten darin ablesen, welche Ergebnismengen das Datenelement beim Durchlaufen des Graphen erreichen könnte (siehe auch Abbildung 3.17 für zwei Beispiele inklusive Antworten). Antworten wurden manuell notiert, während die benötigte Zeit automatisch aufgezeichnet wurde. Dazu drückten die Teilnehmer nach dem Beantworten der Frage eine Taste zum Beenden der Aufgabe. Diese Messung fand mit dem für die Studie entwickelten Werkzeug stattfand, das auch die Graphen anzeigte. Somit entsprach die aufgezeichnete Zeitspanne exakt der Zeitspanne, während derer der Graph auf dem Bildschirm sichtbar war. Die Teilnehmer wurden über die Zeitmessung informiert. Sie wurden jedoch darauf hingewiesen, dass sie ihre Priorität auf Korrektheit und nicht auf Schnelligkeit legen sollten. So sollten willkürliche Antworten durch Raten vermieden werden.

Am Ende jeder Aufgabe wurde ein leerer Bildschirm gezeigt, um zu verhindern, dass sich die Teilnehmer den Graph auswendig merkten. Gleichzeitig bot dies eine Gelegenheit für eine Pause. Die ersten zwei Aufgaben dienten jeweils zur Übung, die anderen drei flossen in die Auswertung mit

[4] das heißt, ohne Merkmale, die ihn eindeutig als FFK- oder EFFK-Graphen identifiziert hätten

Tabelle 3.4: Aggregierte Ergebnisse der Benutzerstudie zum EFFK: Anzahl der Teilnehmer, Mittelwerte und Standardabweichungen („SD“) für Korrektheit und Antwortzeit in Sekunden, sowie die Anzahl der Teilnehmer, die das jeweilige Modell bevorzugen.

		Korrektheit		Antwortzeit (s)		
Modell	**Anzahl**	**Mittel**	**SD**	**Mittel**	**SD**	**bevorzugt**
FFK	28	0,987	0,027	30,1	6,7	9
EFFK	28	0,967	0,059	26,0	6,6	19

ein. Nach Beendigung aller Aufgaben mussten die Teilnehmer die beiden Visualisierungsmodelle in einem Fragebogen vergleichen. Darin wurden sie aufgefordert, eine Vorliebe für eines der Modelle zu äußern und die von ihnen wahrgenommenen Vor- und Nachteile jedes der Modelle aufzuzählen. Zusätzlich wurden die Teilnehmer noch gebeten, einige demografische Daten anzugeben.

Beide Graphtypen wurden während der gesamten Benutzerstudie neutral als „Graphtyp A“ und „Graphtyp B“ bezeichnet. Insgesamt nahm die Studie ungefähr 50 bis 60 Minuten pro Teilnehmer in Anspruch. Jeder Teilnehmer, der mit der Studie anfing, erhielt eine finanzielle Aufwandsentschädigung.

In einem Graphen mit n Ergebnismengen und $m \leqslant n$ vom Datenelement erreichten Ergebnismengen mussten Teilnehmer genau die m erreichten Ergebnismengen nennen. Dies ist gleichbedeutend mit n booleschen Aussagen der Form „erreicht“/„nicht erreicht“. Unter diesen n booleschen Aussagen ging jede korrekte Aussage als $\frac{1}{n}$ in die summierte Punktzahl der Aufgabe ein.

3.1.5.4 Ergebnisse

Die ermittelten Punktzahlen wurden mit einer Mixed ANOVA verglichen. Die Werte waren für alle vier Fälle ähnlich. Tabelle 3.4 zeigt die Mittelwerte und die Standardverteilung von Korrektheit und Antwortzeiten bei jedem der Modelle. Die Reihenfolge der Bestandteile der Studie schien keinen Einfluss zu haben.

Die Aufgaben konnten in beiden Graphmodellen gut gelöst werden; es wurden insgesamt wenig Fehler gemacht. Dies war bei der Studienplanung wegen der hohen Komplexität der Anfragegraphen nicht vorgesehen worden.

In Bezug auf die Korrektheit der Antworten gab es zwischen den beiden Modellen keine signifikanten Unterschiede ($F(1, 24) = 2,63; p = 0,118$). Die Unterschiede in der Bearbeitungszeit waren hingegen signifikant ($F(1, 24) = 19,25, p < 0.001$). Die Teilnehmer konnten die Aufgaben mit dem EFFK im

Durchschnitt vier Sekunden (oder 14%) schneller lösen als mit dem FFK.

Auch subjektiv wurde das EFFK von den Teilnehmern tendenziell bevorzugt, da zwei Drittel dieses Modell im Fragebogen als besser bewerteten. Die Teilnehmer nannten Eigenschaften wie „klarer“ (16×), „schneller“ (8×) und „kompakter“ (6×) als Begründungen. Das FFK erhielt dementsprechend einige negative Kommentare, die sich unter den Stichworten „zu viele Kanten“ (8×) und „Durcheinander“ (5×) zusammenfassen lassen.

Kritische Kommentare zum EFFK betrafen hauptsächlich die visuelle Darstellung. Vier Teilnehmer fanden, dass die komplexeren Filterknoten zum Teil zu viel Information auf zu kleinem Raum enthielten. Zwei weitere gaben an, dass insbesondere der Vergleichsoperator leicht zu übersehen ist.

Insgesamt verstanden die Teilnehmer jedoch schnell, wie man mit dem EFFK-Graphen umgehen muss. Sie nannten auch keine ernsthaften Schwierigkeiten, die beim Lesen der erweiterten Visualisierung aufgetreten wären. Zusammenfassend lässt sich daher festhalten, dass die EFFK-Graphen nicht schlechter lesbar sind als entsprechende FFK-Graphen. Durch die kompaktere, platzsparendere und zum Teil einfachere Darstellung werden sie von Benutzern tendenziell bevorzugt.

3.1.6 Evaluation der Abbildung auf SPARQL

Im Rahmen einer Diplomarbeit wurde von einem Studenten eine qualitative Benutzerstudie zu einer abgewandelten Version von SparqlFilterFlow durchgeführt [Bol13]. Dazu wurde eine vom Studenten erstellte webbasierte Implementierung des Konzepts eingesetzt, welche die grundlegenden Features unterstützte. Abweichend vom oben beschriebenen Konzept werden Knoten, die sich auf dasselbe Objekt beziehen, farblich gekennzeichnet (siehe Abschnitt 3.1.2.5). Dabei können auch Objekte aus unterschiedlichen, parallel (nicht sequenziell) zueinander verlaufenden Flüssen verwendet werden.

Im Rahmen der Studie mussten die Teilnehmer jeweils vier abgebildete Filter/Flow-Graphen lesen und die entsprechenden Anfragen in natürlicher Sprache formulieren. Ebenso mussten sie jeweils vier Anfragen, die in natürlicher Sprache gegeben waren, mit Hilfe des Prototyps, in dem das Vokabular aus DBpedia [ABK+07] zur Verfügung stand, visuell ausdrücken. Ein Beispiel für eine solche Anfrage lautete: „Gesucht wird eine Programmiersprache (*Programming Language*), die durch ‚Java (programming language)‘ oder ‚BASIC‘ beeinflusst (*influenced by*) wurde.“

Die Studie wurde mit insgesamt 16 Teilnehmern durchgeführt. Es wurden jeweils nur bei der letzten, kompliziertesten Verständnis- und Konstruktionsaufgabe fehlerhafte Lösungen genannt. Insgesamt zeigten sich die Studienteilnehmer sehr zufrieden mit dem Konzept. Auf einer elfstufigen

```
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
PREFIX journal: <http://dblp.l3s.de/d2r/resource/journals/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?author
WHERE {
  ?author a foaf:Agent .
  ?doc swrc:journal journal:tvcg ;
       dc:creator ?author .
  ?otherDoc dc:title ?title ;
            dc:creator ?author .
  FILTER(contains(?title, "visual")
         && contains(?title, "query")) .
}
```

Quelltextbeispiel 3.1: Die SPARQL-Anfrage, zu der in Abbildung 3.18 ein Ausschnitt der Ergebnisliste gezeigt wird.

Skala, auf der die Studienteilnehmer ihre Zufriedenheit mit den Gesichtspunkten der Lesbarkeit von Anfragen, der Erstellbarkeit von Anfragen und dem visuellen Datenfilterungskonzept allgemein angeben sollten, lagen alle Durchschnittswerte im oberen Viertel.

3.1.7 Implementierung

Neben der in Abschnitt 3.1.6 kurz erwähnten webbasierten Umsetzung wurde das EFFK mit C# und WPF als Anwendung für das Microsoft .NET Framework [Mic15] implementiert. Konzeptuell wurden die oben beschriebenen Features inklusive der in Abschnitt 3.1.3 beschriebenen Flüsse mit mehreren Datenmengen verwirklicht. Für die Datenanbindung wurde in dieser Implementierung die in Abschnitt 3.1.4 vorgestellte Filter/Flow-Interpretation SparqlFilterFlow mittels der Bibliothek dotNetRDF [VZA+15] umgesetzt.

Die Implementierung stellt die Anfragegraphen visuell dar und erlaubt die Bearbeitung mittels Drag-und-Drop. Filtereinstellungen können direkt in den Filterknoten vorgenommen werden. Zusätzliche Knoten können über eine Werkzeugleiste eingefügt werden, und neue Flüsse lassen sich durch das Verbinden von Emittern und Rezeptoren per Drag-und-Drop hinzufügen. Abbildung 3.18 zeigt die prototypische Anwendung. Quelltextbeispiel 3.1 repräsentiert die SPARQL-Anfrage für die in der Abbildung gezeigte Ergebnisliste.

Als Datenquelle für die Implementierung kann ein beliebiger SPARQL-Endpoint gewählt werden.

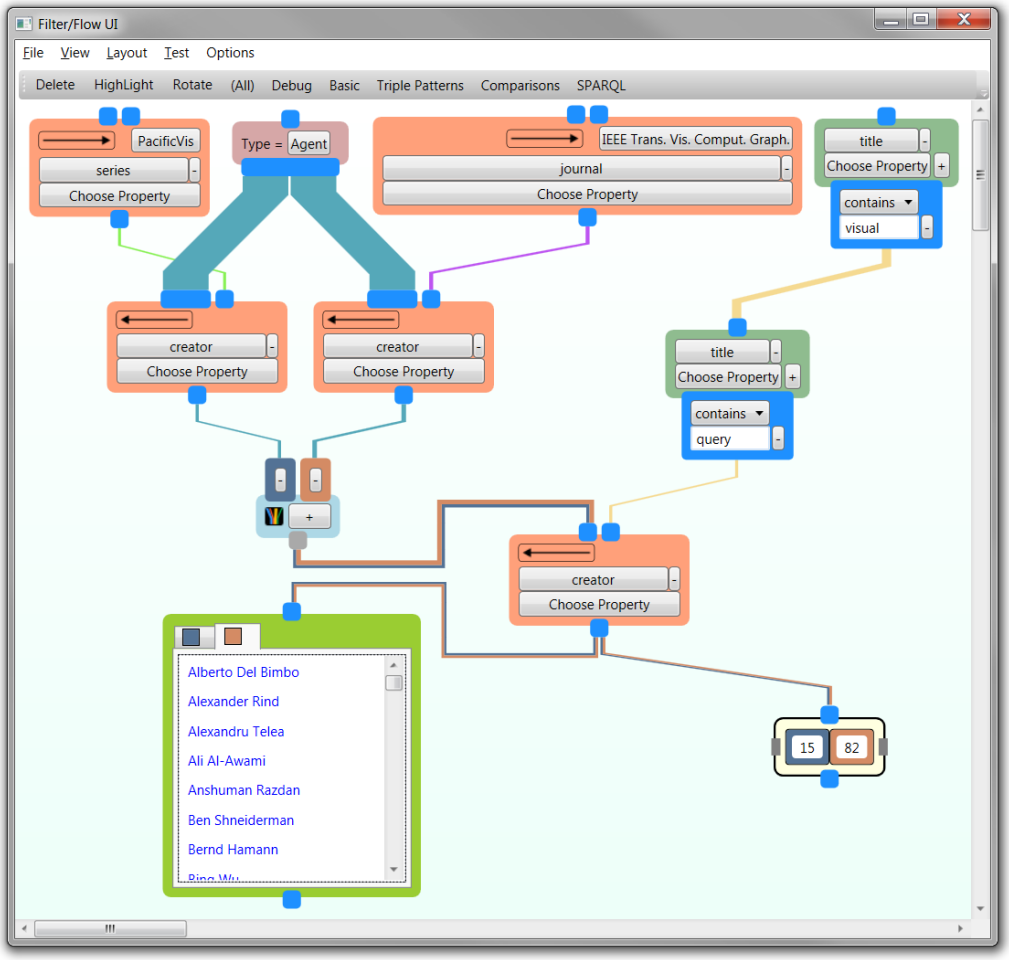


Abbildung 3.18: Die Desktop-basierte prototypische Implementierung des EFFK-Systems. In diesem Screenshot ist ein auf FACETED DBLP [DB08] ausgeführter Anfragegraph zu sehen, welcher Autoren sucht, die einerseits mindestens eine Veröffentlichung auf der Konferenz PACIFICVIS oder in der Fachzeitschrift TVCG haben und andererseits mindestens eine Veröffentlichung, deren Titel die Wörter „visual“ und „query“ enthält. Die Ergebnisse werden je nach Veröffentlichung auf der Konferenz oder in der Fachzeitschrift separat ermittelt, da die betreffenden Mengen nicht vereinigt werden, sondern in parallelen Bahnen in die Ergebnisknoten geleitet werden. Die beiden Ergebnismengen sind jedoch nicht notwendigerweise disjunkt, falls Autoren die Bedingungen beider Anfragen erfüllen. Die Größen der zwei Ergebnismengen werden dargestellt. Ebenso wird ein Ausschnitt der Ergebnisliste der TVCG-Autoren angezeigt, da im Ergebnislistenknoten (unten rechts) der Reiter für die TVCG-Autoren-Anfrage aktiv ist. Der Quelltext der entsprechenden SPARQL-Anfrage ist als Quelltextbeispiel 3.1 aufgeführt.

3.1.7.1 Graphauswertung

Zur Auswertung eines FILTER/FLOW-Graphen müssen die Verknüpfungen zwischen den Filterknoten analysiert werden, um herauszufinden, welche Filterkriterien in welchen Kombinationen zusammen erfüllt werden müssen. Die naive Vorgehensweise dazu, welche vermutlich ähnlich wie das Verständnis menschlicher Benutzer von FILTER/FLOW-Graphen funktioniert, betrachtet sämtliche alternativen Pfade, die vom Anfang zum Ende des Graphen führen. Das Gesamtergebnis des Ausdrucks ist dann eine Disjunktion aus den Kombinationen der Filterkriterien jedes solchen Pfads. Diese Vorgehensweise ist allerdings nicht besonders effizient, da Filterknoten, die mehreren Pfaden angehören, mehrfach ausgewertet werden müssen. Selbst bei zwischengespeicherten Auswertungsergebnissen für jeden Knoten (beziehungsweise jeden Emitter, siehe Abschnitt 3.1.2.1) lässt sich auf diese Weise der Filtergraph nicht in einen kompakten booleschen Ausdruck umformen, sondern nur in die unter Umständen unübersichtlichere disjunktive Normalform (DNF), wie in Abbildung 3.19 veranschaulicht.

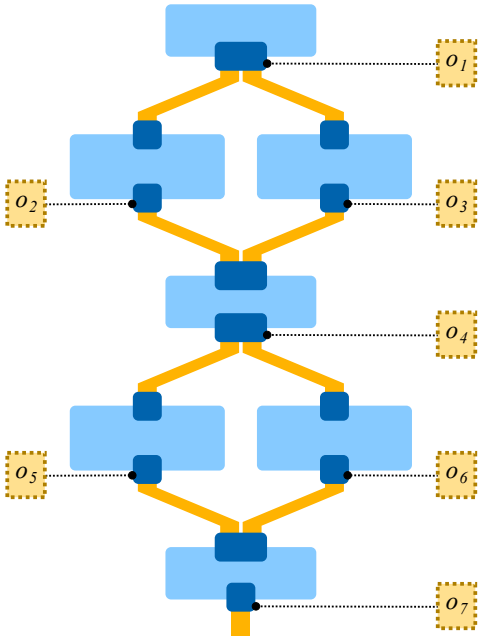


Abbildung 3.19: Dieser FILTER/FLOW-Graph lässt sich intuitiv in den relativ kompakten Ausdruck $o_1 \wedge (o_2 \vee o_3) \wedge o_4 \wedge (o_5 \vee o_6) \wedge o_7$ umformen (wobei der von einem Emitter repräsentierte Term durch den Namen des jeweiligen Emitters ausgedrückt wird). Mit der naiven Vorgehensweise, jeden Pfad separat zu betrachten, entsteht ein ungleich längerer Ausdruck in DNF: $(o_1 \wedge o_2 \wedge o_4 \wedge o_5 \wedge o_7) \vee (o_1 \wedge o_3 \wedge o_4 \wedge o_5 \wedge o_7) \vee (o_1 \wedge o_2 \wedge o_4 \wedge o_6 \wedge o_7) \vee (o_1 \wedge o_3 \wedge o_4 \wedge o_6 \wedge o_7)$

Da in der Literatur zum FILTER/FLOW-Konzept keine konkreten Angaben zur Vorgehensweise zu finden waren, wurde für die Implementierung ein neuer Algorithmus entworfen. Die Auswertung geschieht jeweils für eine bestimmte Datenmenge in ihrem Zustand, wie sie aus einem bestimmten Emitter austritt. Wie am Ende von Abschnitt 3.1.3 erläutert, lässt sich zu jeder solchen Datenmenge an einem gegebenen Emitter jedoch ein äquivalenter Anfragegraph erzeugen, der vollständig ohne Flüsse mit mehreren Datenmengen auskommt. Um die Lesbarkeit des Algorithmus zu erhöhen, wird in diesem Dokument die Variante ohne Berücksichtigung der Datenmengen gezeigt.

Der Algorithmus benötigt als Eingabe einen Emitter o_0 (von dem aus flussaufwärts der Graph durchlaufen wird) sowie einen Grenzknoten *limit* (an dem die Traversierung des Graphen gestoppt wird). Ist kein Grenzknoten angegeben, so wird die Traversierung bis zu dem oder den Ursprungsknoten des Graphen fortgesetzt. Der Vorgang ist als Algorithmus 3.1 dargestellt.

Der Algorithmus betrachtet beim Durchlaufen des Graphen jeden Filterknoten. Eingehende Flüsse an Rezeptoren, die nicht der Hauptrezeptor sind, werden durch rekursive Aufrufe von *getTree* zu Ausdrucksbäumen zusammengefasst. Sie werden der Reihe nach in der Liste *args* abgelegt (Zeile 7ff.). Der aktuell betrachtete Emitter wird dann zu einem Knoten für den Ausdrucksbaum umgewandelt, wobei die in *args* gesammelten Ausdrucksbäume als Parameter verwendet werden (Zeile 19). Die Traversierung wird jeweils über den Hauptrezeptor fortgesetzt (Zeile 20ff.). Bei allen Rezeptoren wird die Situation, dass mehrere Flüsse eingehen, durch eine zusätzliche Funktion *getParallel* behandelt (Zeilen 16 und 29).

Die Funktion *getParallel* (Algorithmus 3.2) ermittelt die Ausdrucksbäume für mehrere parallel verlaufende Teilgraphen und fasst diese in einer Disjunktion zusammen. Potenziell wird auch ein Baum zu einem weiter flussaufwärts liegenden Subgraph ermittelt und angehängt, bis hin zu einem etwaigen Grenzknoten. *getParallel* benötigt als Parameter lediglich eine Menge von Emittern $\tilde{O}$ sowie optional einen Grenzknoten *limit*. Zudem greift die Funktion intern auf eine weitere Funktion *maxDist* zurück, welche für einen gegebenen Emitter $\mathring{o}$ die maximale Anzahl an Emittern bestimmt, die durchlaufen werden müssen, um von $\mathring{o}$ aus einen beliebigen Emitter aus $\tilde{O}$ erreichen zu können. Dabei werden nur solche Emitter aus $\tilde{O}$ berücksichtigt, die von $\mathring{o}$ aus erreichbar sind. Gibt es keine solchen Emitter, so ist *maxDist* undefiniert.

Der Emitter, an dem die parallelen Graphpfade wieder zusammenlaufen, wird über die Funktion *ncp* („nearest common predecessor“, nächster gemeinsamer Vorgänger) bestimmt. In *getParallel* werden Gruppen aus Emittern gebildet, welche jeweils denselben nächsten gemeinsamen Vorgänger haben. Beginnend bei der Gruppe, die den nächstgelegenen gemeinsamen

```
1: function getTree(o_0, limit)
2:     args, chain: leere Listen
3:     o_current := o_0
4:     n_current := Knoten von o_0
5:     while o_current ≠ ε and o_current ≠ limit and |n_current.Inbound| > 0 do
6:         r_main := Hauptrezeptor von n_current
7:         for all r ∈ Rezeptoren von n_current \ {r_main} do
8:             n_flows := |bei r eingehende Flüsse|
9:             if n_flows = 0 then
10:                args.append(ε)
11:            else if n_flows = 1 then
12:                e_0 := einziger bei r eingehender Fluss
13:                args.append(getTree(Emitter von e_0, ε))
14:            else
15:                Õ := alle mit r verbundenen Emitter
16:                args.append(getParallel(Õ, ε))
17:            end if
18:        end for
19:        chain.prepend(createTreeNode(o_current, args))
20:        if r_main ≠ ε then
21:            n_flows := |bei r_main eingehende Datenmengen|
22:            if n_flows = 0 then
23:                o_current := ε
24:            else if n_flows = 1 then
25:                o_current := Emitter des bei r_main eingehenden Flusses
26:            else
27:                Õ := alle mit r_main verbundenen Emitter
28:                o_c := ncp(Õ)
29:                chain.prepend(getParallel(Õ, o_c))
30:                o_current := o_c
31:            end if
32:        end if
33:    end while
34:    if o_current ≠ ε and o_current ≠ limit then
35:        chain.prepend(createTreeNode(o_current, args))
36:    end if
37:    return Konjunktion aus Elementen von chain sowie true
38: end function
```

Algorithmus 3.1: Funktion *getTree* zur Extraktion eines booleschen Ausdrucksbaums aus einem EFFK-Graphen.

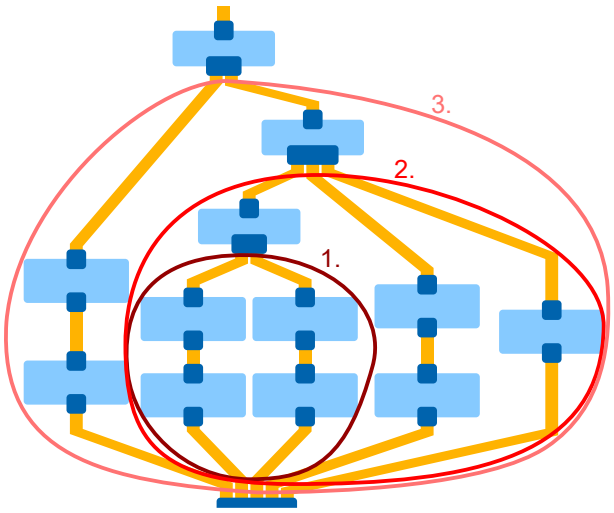


Abbildung 3.20: Parallele Subgraphen, die dem selben Rezeptor vorausgehen, werden zu Gruppen mit demselben nähesten gemeinsamen flussaufwärts liegenden Element zusammengefasst. Die Gruppen, die am frühesten wieder zu einem gemeinsamen Pfad zusammentreffen, werden als erstes in einer Disjunktion gruppiert.

Vorgänger von allen Gruppen hat, wird dann ein Disjunktionsbaumknoten für die parallelen Subgraphen aus dieser Gruppe erzeugt. Der nächste gemeinsame Vorgänger der betreffenden parallelen Subgraphen wird dann in die noch zu verarbeitenden Gruppen von parallelen Subgraphen miteinbezogen, da der Disjunktionssubgraph insgesamt parallel zu einem der anderen betrachteten Subgraphen verlaufen könnte. Dieser Vorgang wird so lange wiederholt, bis alle Gruppen verarbeitet sind und somit ein gemeinsamer Disjunktionsbaumknoten übrigbleibt (Abbildung 3.20).

Die Funktion *ncp* ermittelt zu einer Menge $\hat{O} \subseteq O$ aus Emittern den nächstgelegenen gemeinsamen Vorgänger als den flussaufwärts nächstgelegenen Emitter, von dem aus alle Emitter aus $\hat{O}$ erreichbar sind. Ist kein solcher Emitter vorhanden, wird ε zurückgegeben (das heißt im konkreten Programm `null`). Sofern $\hat{O}$ mehr als einen Emitter enthält, greift *ncp* einen beliebigen Emitter aus $\hat{O}$ heraus. Von diesem aus gesehen werden so lange eindeutige Vorgänger gesucht, bis ein Vorgänger gefunden wurde, der auch einen eindeutigen Vorgänger für alle anderen Emitter aus $\hat{O}$ darstellt, oder bis der Anfang des Anfragegraphen erreicht ist. Ein Vorgänger ist dabei *eindeutig*, wenn man beim Zurückverfolgen der Flüsse zwangsläufig auf ihn trifft und er somit nicht durch einen parallelen Subgraphen umgangen werden kann. Die Berechnungsvorschrift ist als Algorithmus 3.3 abgebildet.

```
1:  function getParallel(Õ, limit)
2:      last := ε
3:      attached := ∅
4:      groups := {(o_p, Ô)|Ô ⊆ Õ ∧ o_p = ncp(Ô)
5:          ∧∄Ǒ ⊆ Õ, Ǒ ∩ Ô ≠ ∅ : ncp(Ǒ) ist Nachfolger von o_p}
6:      groupsL := Liste mit Elementen aus groups
7:      while |groupsL| > 0 do
8:          groupsL aufsteigend nach maxDist(o_p) sortieren
9:          erstes Element (o_p, Ô) aus groupsL entnehmen
10:         parallel: leere Liste
11:         for all ô ∈ Ô do
12:             add := t_α für (ô, t_α) ∈ attached, sonst ε
13:             subTree := Konjunktion aus getTree(ô, o_p), add und true
14:             parallel.append(subTree)
15:         end for
16:         last := Disjunktion aus allen Knoten in parallel und false
17:         if o_p ≠ ε then
18:             attached := attached ∪ {last}
19:         end if
20:         for all (o_ξ, O_ξ) ∈ groupsL do
21:             if O_ξ ∩ Ô ≠ ∅ then
22:                 O_ξ := O_ξ\(O_ξ ∩ Ô)
23:                 if O_ξ = ∅ then
24:                     (o_ξ, O_ξ) aus groupsL entfernen
25:                 else if o_p ≠ limit and o_p ≠ ε then
26:                     O_ξ := O_ξ ∪ {o_p}
27:                 end if
28:             end if
29:         end for
30:     end while
31:     return Konjunktion aus dem letzten Wert von o_p, last und true
32: end function
```

Algorithmus 3.2: Function *getParallel* zum Umwandeln mehrerer parallel verlaufender Subgraphen eines EFFK-Graphen zu einer Disjunktion aus Ausdrucksbäumen.

```
1: function NCP(Ô)
2:     if |Ô| ⩽ 0 then
3:         return FEHLER
4:     else if |Ô| = 1 then
5:         return direkter Vorgänger-Emitter des Elements aus Ô
6:     else
7:         o_c := ein Element aus Ô
8:         Ô' := Ô\o_c
9:         while o_c ≠ ε do
10:            if Ô' enthält nur eindeutige Nachfolger von o_c then
11:                return o_c
12:            end if
13:            r_c,main := Hauptrezeptor des Knotens von o_c
14:            F_c := bei r_c,main eingehende Flüsse
15:            if |F_c| ⩽ 0 then
16:                o_c := ε
17:            else if |F_c| = 1 then
18:                o_c := direkter Vorgänger-Emitter des Knotens aus o_c
19:            else
20:                o_c := ncp(alle Emitter Flüsse aus F_c)
21:            end if
22:        end while
23:        return ε
24:    end if
25: end function
```

Algorithmus 3.3: Function *ncp* („nearest common predecessor“, nächster gemeinsamer Vorgänger), welche in einem EFFK-Graphen den nächstgelegenen Emitter sucht, der ein gemeinsamer Vorgänger einer gegebenen Menge von Emittern ist.

3.1.7.2 Architektur

Die Architektur der Implementierung wurde so gewählt, dass die gesamte Anwendung für verschiedene Einsatzzwecke flexibel erweiterbar ist. Insbesondere wurde darauf geachtet, dass unterschiedliche Arten von Datenquellen angeschlossen, zusätzliche Filterknoten verwendet und unterschiedliche Frontend-Technologien eingesetzt werden können.

Der durch einen Filter/Flow-Graphen dargestellte Ausdruck wird zunächst durch den Algorithmus aus dem vorangegangenen Abschnitt erkannt. Das Ergebnis wird dann durch einen abstrakten Ausdrucksbaum repräsentiert. Die objektorientierte Repräsentation der Baumknoten wird über das *Factory-Pattern* erzeugt, wie in Abbildung 3.21 dargestellt. Von

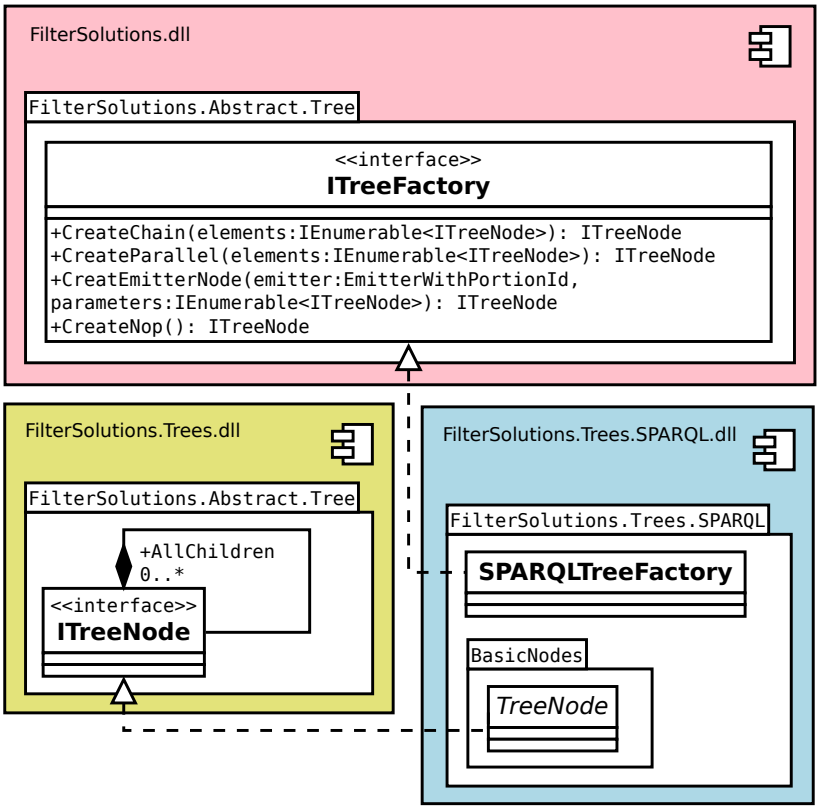


Abbildung 3.21: UML-Klassendiagramm der Factory für Ausdrucksbäume in der Filter/Flow-Implementierung. Das Modul *FilterSolutions.Trees.SPARQL.dll* könnte komplett ersetzt werden, um Anfragen in der Syntax einer anderen Abfragesprache als SPARQL zu erzeugen.

den erzeugten Baumknoten können so letztendlich unterschiedliche Darstellungen des Anfrageausdrucks erzeugt werden – beispielsweise SQL-Code oder SPARQL-Code.

Sämtliche Filter/Flow-Knoten basieren auf einer Basisklasse, zu der je nach Anwendungsfall Kindklassen erzeugt werden können. Grundlegende, „abstrakte“ Operationen, wie das einfache Vergleichen von Werten, sind vordefiniert. Daneben kann der Prototyp so aber auch auf relativ unkomplizierte Weise an neue Anwendungsfälle angepasst werden. Für Anfragen im Bereich des öffentlichen Verkehrs können beispielsweise Knoten zur Filterung aller grundlegenden Reisedaten wie Abfahrts- oder Ankunftszeit sowie Start- und Zielort angeboten werden. Dies trägt zur Vereinfachung der Graphstruktur gemäß dem Prinzip der Subgraphersetzung (siehe Abschnitt 3.1.2.6) bei.

Um das Konzept auf unterschiedlichen Arten von Endgeräten testen zu können, wurden Datenmodell und visuelle Darstellung in zwei Schichten

getrennt (siehe Abbildung 3.22). Die Visualisierungsschicht stellt dabei für jeden Knotentyp aus dem Datenmodell eine passende Visualisierung zur Verfügung. Die geeignete Visualisierung wird über eine zur Laufzeit ermittelte Zuordnungstabelle erzeugt. Neu hinzugefügte Filtertypen im Datenmodell benötigen somit ebenso ein passendes Visualisierungsmodul in der Visualisierungsschicht.

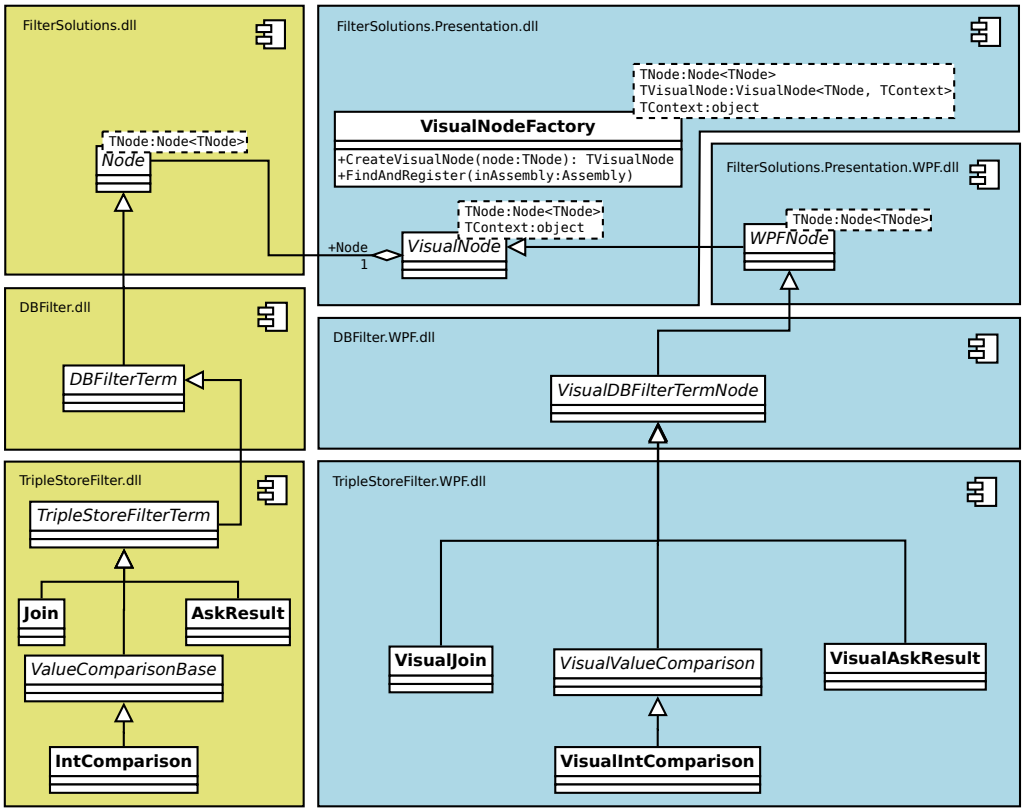


Abbildung 3.22: UML-Klassendiagramm der Knotenklassen in der Filter/Flow-Implementierung. Durch die Unterteilung in mehrere Module mit einer Datenhaltungsschicht (links) und einer davon getrennten Visualisierungsschicht (rechts) lässt sich der Prototyp für andere Arten von Frontends anpassen. (Pakete beziehungsweise Namensräume sind in diesem Diagramm der Übersichtlichkeit halber nicht berücksichtigt. Ebenso ist nur eine kleine, repräsentative Auswahl von Knotentypen zu sehen.)

Die Visualisierungsschicht ist auf diese Weise komplett austauschbar. So konnten neben der WPF-basierten Implementierung für Desktop-Rechner weitere Visualisierungsschichten umgesetzt werden. Dies wurde gezeigt, indem eine zusätzliche Implementierung für Desktop-Systeme, basierend auf Gtk# (dargestellt in Abbildung 3.23), sowie eine WPF-basierte Implementierung für den berührungsempfindlichen Microsoft-Surface-Tisch

teilweise erstellt wurden. Zudem wurde durch den modularen Aufbau eine Diplomarbeit möglich, die unter Verwendung von Mono for Android eine Visualisierungsschicht für mobile Android-Geräte entwickelte [Fer12].

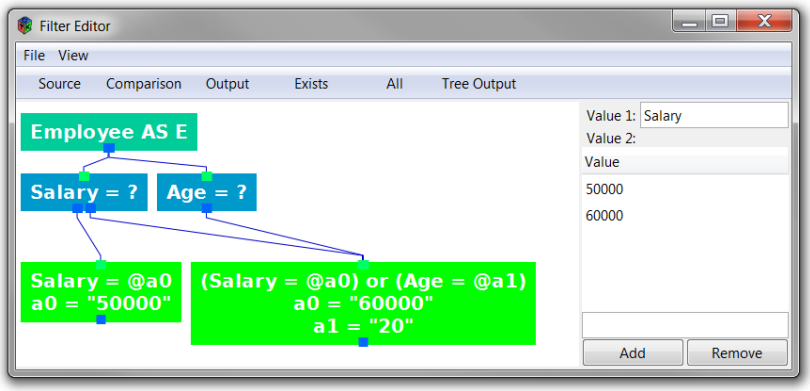


Abbildung 3.23: Eine frühe Version der prototypischen EFFK-Implementierung, für die ein Gtk#-Frontend angefertigt wurde. In der untersten Reihe der Knoten werden jeweils aus dem Anfragegraph konstruierte Bedingungen angezeigt, wie sie direkt in die WHERE-Klausel einer SQL-Anfrage eingesetzt werden könnten.

3.1.7.3 Benutzbarkeit der Implementierung

Die Implementierung mit dem WPF-basierten Frontend und dem SPARQL-basierten Backend stellt eine Reihe grundlegender Filterknoten für Wertevergleiche zur Verfügung. Ebenso sind Knoten zur Anzeige der Größe von Ergebnismengen sowie von Elementen aus Ergebnismengen vorhanden, die zu Anfragegraphen hinzugefügt werden können. An strukturellen Knoten steht zum Nachweis der Umsetzbarkeit der parallelen Bahnen (siehe Abschnitt 3.1.3) der Zusammenführungsknoten aus Abbildung 3.11 bereit.

Die Anwendung lässt sich mit einem Zeigegerät bedienen und kann über einen Einstellungsdialog dahingehend konfiguriert werden, dass auf einen benutzerdefinierten SPARQL-Endpunkt zugegriffen wird. Die Generierung der SPARQL-Anfragen ruft auch bei relativ komplexen Anfragegraphen (ca. 50 Knoten) keine erkennbare Verzögerung hervor und die Anfragen zur Ermittlung von Ergebnismengen oder deren Größe werden bei Änderungen im Graph jeweils umgehend versandt. Die Wartezeit, bis die entsprechenden Antworten vom Server eintreffen und die Visualisierung aktualisiert wird, hängt vom aktuellen Zustand des Servers ab und variiert auch bei kleinen Anfragen (2 Knoten) zwischen unter einer Sekunde und mehreren Minuten. Bei komplexeren Anfragen besteht eine Tendenz zu längeren Wartezeiten (mindestens einige Sekunden), wobei die Wartezeit aber auch stärker von

der Server-Software des aktuell verwendeten Endpunkts und der konkreten Anfragestruktur abhängt – so scheint beispielsweise der für DBpedia verwendete Triple Store Virtuoso mehr Zeit für die Auswertung von Anfragen zu benötigen, wenn diese mit `UNION` verknüpfte alternative Subgraphen enthalten.

3.1.8 Fazit

Durch die integrierten Verbesserungen des EFFK gegenüber dem FFK ist nun klar definiert, wie Filter/Flow-Graphen in kompakter Weise dargestellt werden können. Ebenso ist es nun möglich, mittels der parallelen Bahnen mehrere ähnliche Anfragen gleichzeitig durchzuführen und die Veränderungen der Ergebnismengen im Verlauf der Filterung miteinander zu vergleichen. Die Abbildung auf SPARQL ermöglicht die Anwendung der kombinierten Verbesserungen für Anfragen an Datensammlungen, die über eine einheitliche Schnittstelle angesprochen werden können, was die praktische Einsetzbarkeit des Konzepts gewährleistet. Durch die vorliegende Implementierung können diese Neuerungen bereits praktisch ausprobiert werden.

3.2 QueryVOWL

Die Graphdarstellung im Filter/Flow-Konzept wurde dazu eingesetzt, die logische Verknüpfung von Filterkriterien auszudrücken. Im Gegensatz dazu stellen Objektgraph-basierte Verfahren (siehe Abschnitt 2.3.1.1) die Zusammenhänge zwischen den gesuchten Datensätzen als Graph dar. Benutzer definieren also eine Schablone für das Graph-Muster, das sie finden möchten.

Im Rahmen dieses Promotionsvorhabens wurde eine Objektgraph-basierte Anfragevisualisierung entwickelt, die sich auf die Anfragesprache SPARQL abbilden lässt. Dadurch gibt sie Benutzern ohne SPARQL-Kenntnisse die Möglichkeit, Anfragen im Zusammenhang mit komplexen Objektstrukturen an RDF-Datenbanken zu richten. Im Gegensatz zu anderen Konzepten, welche sich dicht an der SPARQL-Syntax orientieren [SRB+08; GGS11; Del10], war es nicht das Ziel, eine visuelle Abbildung der SPARQL-Syntax zu definieren. Stattdessen sollten die zugrundeliegenden Anfragekonzepte wiedergegeben werden. Gleichzeitig sollte eine möglichst breite Auswahl an Anfragevariationen möglich sein.

Um die Einarbeitungszeit in die Visualisierung zu senken, wurde auf der Notation VOWL aufgebaut. Diese Notation hat sich bereits im Bereich der Ontologievisualisierung bewährt. Dazu wurden die visuellen Elemente aus VOWL übernommen und teilweise angepasst. Anschließend wurden sie

über eine Reihe neu hinzugefügter Definitionen auf SPARQL-Fragmente abgebildet. Die Absicht hinter dieser Vorgehensweise war, zumindest Benutzern, die bereits mit VOWL vertraut sind, den Lernaufwand für die neue Anfragevisualisierung – QueryVOWL – zu verringern.

3.2.1 VOWL, visuelle Notation für OWL-Ontologien

Im Rahmen des Promotionsvorhabens wurden Beiträge zur Entwicklung von VOWL 1 und insbesondere zur Weiterentwicklung von VOWL 1 zu VOWL 2 beigesteuert. Daher wird an dieser Stelle die Ontologievisualisierung VOWL mit Fokus auf die zweite Version vorgestellt. Auf diese Weise soll sowohl der wissenschaftliche Beitrag im Bereich der Ontologievisualisierung beschrieben als auch eine Grundlage für das anschließend vorgestellte QueryVOWL-Konzept gelegt werden.

> Die im Folgenden präsentierten Inhalte wurden zum Teil bereits anderweitig veröffentlicht. Insbesondere wird auf den im Folgenden aufgeführten Arbeiten, an denen der Autor maßgeblich mitwirkte, aufgebaut:
>
> NHL13 S. Negru, F. Haag und S. Lohmann. "Towards a unified visual notation for OWL ontologies: insights from a comparative user study". In: *Proc. Semantic Systems (I-SEMANTICS '13)*. ACM, 2013, S. 73–80. DOI: 10.1145/2506182.2506192
>
> LNH+14 S. Lohmann, S. Negru, F. Haag und T. Ertl. "VOWL 2: user-oriented visualization of ontologies". In: *Proc. Knowledge Engineering and Knowledge Management (EKAW '14)*. Bd. 8876. LNCS. Springer, 2014, S. 266–281. DOI: 10.1007/978-3-319-13704-9_21
>
> LNH+16 S. Lohmann, S. Negru, F. Haag und T. Ertl. "Visualizing ontologies with VOWL". In: *Semantic Web Journal* (voraussichtlich 2016). Angenommen am 21.08.2015.
>
> LHN15 S. Lohmann, F. Haag und S. Negru. "Towards a visual notation for OWL: a short summary of VOWL". In: *OWLED '15*. 2015

3.2.1.1 Grundlegende Idee

VOWL wurde von Negru et al. [NL13a; NHL13; NL13b] als Vorschlag einer einheitlichen visuellen Notation für OWL-Ontologien entworfen. Dies sollte eine Lücke füllen, da für häufig verwendete Modellierungskonzepte gewöhnlich standardisierte Visualisierungen definiert sind. Schemata relationaler Datenbanken lassen sich als ER-Diagramme darstellen [Che76], für die grafische Repräsentation objektorientierter Klassenstrukturen steht UML zur Verfügung [Obj15, S. 192ff.], aber für die Ontologiebeschreibungsspra-

che OWL [W3C12] existiert noch keine entsprechende einheitliche visuelle Darstellung.

VOWL legt das Hauptaugenmerk auf die TBox (siehe Seite 12), welche die in der Ontologie vorhandenen Konzepte und die möglichen Beziehungen zwischen ihnen ausdrückt. Informationen aus der ABox, welche unter Verwendung der Konzepte konkrete Individuen beschreibt, werden allenfalls stark aggregiert dargestellt. So kann beispielsweise der Radius von Klassenknoten angepasst werden, um die ungefähre Anzahl der zur Klasse gehörenden Individuen anzudeuten.

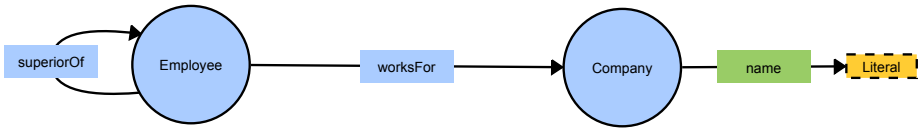
Zur Definition der ersten VOWL-Version (oben dargestellt in Abbildung 2.2) wurden Schritt für Schritt grafische Elemente für die zentralen Bestandteile des Modells OWL in deren Spezifikation [DSB+04] entworfen. Diese visuelle Notation wurde für VOWL 2 angepasst und erweitert. Dadurch werden zusätzliche Konzepte aus OWL 2 [W3C12] unterstützt und Rückmeldungen von Benutzern aufgegriffen.

3.2.1.2 Ontologiedarstellung in VOWL

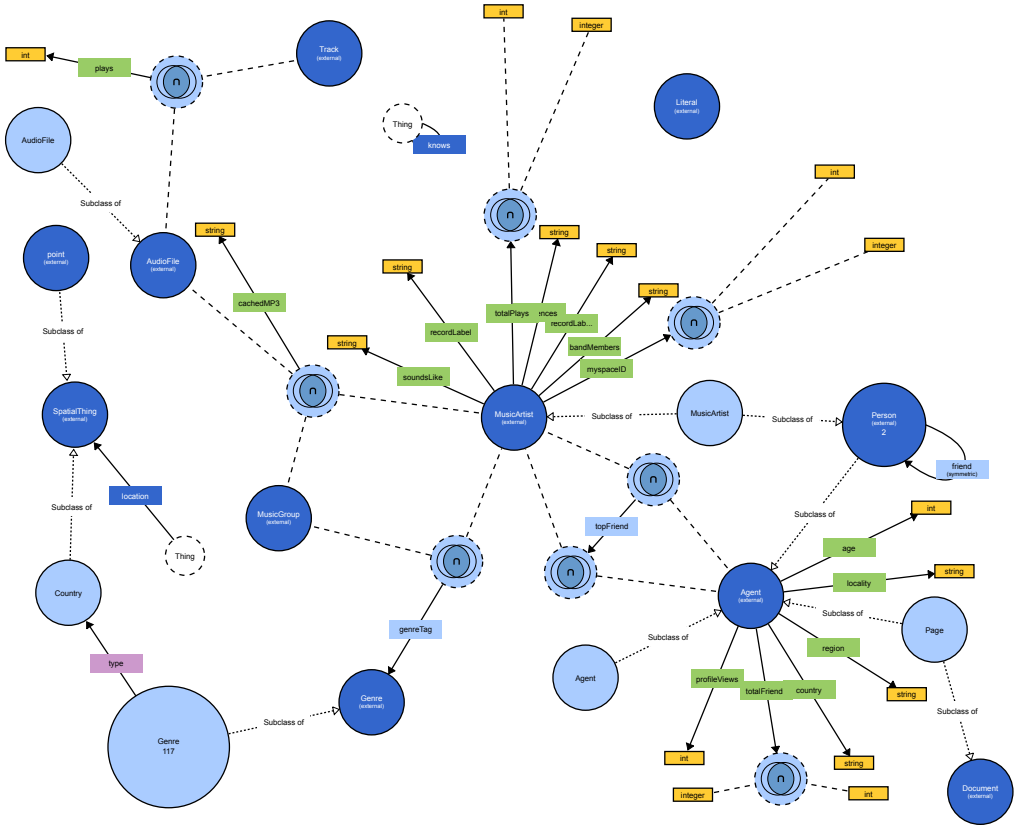
Ontologien werden in VOWL als Node-Link-Diagramm dargestellt (Abbildung 3.24). Dieses Diagramm wird in VOWL standardmäßig kräftebasiert ausgerichtet. Die Knoten repräsentieren dabei Klassen und Datentypen; die meisten Kanten stellen Eigenschaften dar. Zudem existieren noch einige besondere Kanten, die beispielsweise Inklusionsbeziehungen ausdrücken. Auch Vererbungsbeziehungen sind durch spezielle Kanten ausgedrückt, welche sich visuell an Implementierungskanten aus UML orientieren [Obj15, S. 39f.]. Durch die Node-Link-Darstellung soll erreicht werden, dass sich Verbindungen über mehrere Klassen hinweg besser nachvollziehen lassen als beispielsweise mit einer Adjazenzmatrixdarstellung [KEC06].

VOWL verwendet Farben zur Hervorhebung bestimmter Attribute der Klassen, Datentypen und Eigenschaften. Als *deprecated* („nicht mehr zu verwenden“) markierte Klassen und Eigenschaften werden beispielsweise mit einer besonderen Farbe dargestellt. Ebenso erhalten aus anderen Ontologien importierte Klassen, Datentypen und Eigenschaften eine spezielle Farbgebung. Alle zum Verständnis der Ontologie essentiellen Informationen sind jedoch zusätzlich in textueller Form vorhanden. Somit wäre der Aufbau einer Ontologie auch bei rein zweifarbiger Darstellung der Visualisierung noch erkennbar.

VOWL bildet nicht automatisch jeden Bestandteil einer Ontologie mit genau einem grafischen Element ab. Vielmehr wurden für die Visualisierung diverse Eigenarten von OWL-Ontologien ausgenutzt, um die Anzahl und Länge der Kanten zu verringern und somit einem unübersichtlichen Knotengewirr vorzubeugen:



(a) Eine einfache in VOWL 2 dargestellte Ontologie, die oben im Quelltextbeispiel 2.2 sowie in Abbildung 2.2 in der ersten Version von VOWL gezeigt wurde.



(b) Ontologie MySpO [Jac09] dargestellt mit VOWL 2.

Abbildung 3.24: Zwei Beispiele für die Ontologiedarstellung mit VOWL 2.

- Mengen von Klassen, die als äquivalent zueinander definiert sind, werden zu einem einzelnen Knoten zusammengefasst. Da jegliche Eigenschaften, die für eine der äquivalenten Klassen definiert sind, genauso für alle anderen gelten, drückt der gemeinsame Knoten genau die gewünschte Bedeutung aus. Entsprechend werden Mengen von zueinander äquivalenten Eigenschaften durch nur eine einzelne Kante repräsentiert.
- Ontologiebestandteile, die wegen ihres hohen Abstraktionsgrades oft keine zentrale spezifische Bedeutung für eine Ontologie haben, aber gleichzeitig an vielen Stellen der Ontologie benutzt werden, werden in

VOWL *multipliziert*, das heißt, mehrfach dargestellt. Auf diese Weise kann eine Kopie jedes solchen Knotens möglichst dicht an der Klasse, mit der sie verbunden ist, platziert werden. Dies ermöglicht kurze Kanten und verhindert unnötige Kantenkreuzungen. Diese Multiplikation betrifft beispielsweise die globale Basisklasse *owl:Thing* sowie Datentypen für primitive Werte.

VOWL definiert eine Reihe von Interaktionen in der Visualisierung. Diese dienen zum Zugriff auf Informationen, die nicht so wichtig sind, dass sie ständig in der Visualisierung angezeigt werden müssten, die aber dennoch abrufbar sein sollten. Zu diesem Zweck können Elemente der Visualisierung vom Benutzer interaktiv markiert werden. Elemente, die mit den markierten Elementen in einem bestimmten Zusammenhang stehen, werden daraufhin ebenfalls (farblich) hervorgehoben.

Als Beispiel wären Paare zueinander inverser Eigenschaften zu nennen. Diese werden in VOWL als Linie mit Pfeilspitzen an beiden Enden sowie zwei Beschriftungen dargestellt. Wird eine der Beschriftungen markiert, so wird die entsprechende Pfeilspitze hervorgehoben, um die Richtung anzuzeigen, zu der die markierte Beschriftung gehört.

Ferner können Implementierungen beliebige Zusatzinformationen zum markierten Element außerhalb der eigentlichen Visualisierung ausgeben. Somit können auch Informationen zur Ontologie, die in VOWL gar nicht auftauchen, als auch modellspezifisch verknüpfte Daten aus externen Quellen hinzugezogen werden.

3.2.1.3 Abstrakte Definitionen

VOWL ist durchgehend modular definiert. Dies betrifft nicht nur die einzelnen Visualisierungselemente, die je nach Aufbau der Ontologie unterschiedlich kombiniert werden können. Auch die Grundbestandteile, aus denen sich diese Visualisierungselemente zusammensetzen, sind für sich genommen mit einer abstrakten Bedeutung versehen. Auf diese Weise wurde eine einheitliche und flexibel anpassbare grafische Darstellung erreicht.

Diese abstrakte Definition findet sich insbesondere in der Farbgebung wieder: In der Spezifikation werden keine konkrete Farben für die Visualisierungsbestandteile festgelegt. Stattdessen wurde für VOWL ein abstraktes Farbschema (in Form einer Liste von funktionsbezogenen Farbnamen) definiert, auf das in der Spezifikation der visuellen Elemente verwiesen wird. Für jede der definierten abstrakten Farben wird eine konkrete Farbe als Teil eines Standardschemas vorgeschlagen. Implementierungen und Nutzern von VOWL steht es jedoch frei, andere Farbschemata einzusetzen. Auf diese Weise können dynamisch spezielle Einschränkungen in Abhängigkeit vom Einsatzkontext und den Zielnutzern berücksichtigt werden. Beispielsweise

kann ein Farbschema gewählt werden, das farbfehlsichtigen Nutzern hilft. Ebenso kann eine VOWL-Darstellung an die allgemeine Farbgebung eines größeren Systems angeglichen werden.

3.2.1.4 Benutzerstudien

Neben einer qualitativen Benutzerstudie basierend auf der ersten Version von VOWL [NHL13] und einer qualitativen Benutzerstudie, die sich auf das VOWL-Protégé-Plugin konzentrierte [LNB14], wurde VOWL 2 in einer weiteren qualitativen Benutzerstudie mit den alternativen Ontologievisualisierungen SOVA [BJK+10] und GrOWL [KWV07] verglichen. Dabei sollten jeweils Fragen zu dargestellten Ontologien beantwortet werden, die darüber Auskunft geben, inwieweit die zentralen Bestandteile und bestimmte Zusammenhänge in den Ontologien erkannt werden.

In der letztgenannten Studie, an der der Autor dieser Arbeit maßgeblich beteiligt war, wurden insgesamt drei Ontologien gezeigt:

- Die Ontologie Friend of a Friend (FOAF) [BM14] wurde in der Version 0.99 als Beispiel einer kleineren Ontologie verwendet.
- Als Beispiel einer größeren Ontologie fungierte die Personas Ontology (PersonasOnto) [Neg14] in der Version 1.5.
- Als kleine Trainingsontologie zum Erklären der visuellen Notationen vor den eigentlichen Aufgaben wurde die Modular Unified Tagging Ontology (MUTO) [LDA11] in der Version 1.0 eingesetzt.

Die Visualisierungen SOVA und GrOWL wurden gewählt, weil sie durch ihre Node-Link-Darstellung für ähnliche Fragestellungen wie VOWL geeignet sind und sich somit zum Vergleich eignen. Darüber hinaus richten sie sich mit ihrer weitgehend grafischen Darstellung mit nur wenigen formalen Elementen auch an Benutzer, die keine Vorkenntnisse zur formalen Definition von Ontologien haben. Als Implementierung von SOVA wurde während der Studie ein Protégé-Plugin verwendet. Für GrOWL kam eine eigenständige Java-Anwendung zum Einsatz und für VOWL eine webbasierte Implementierung [LLM+15]. Zusätzlich erhielten die Teilnehmer der Studie eine gedruckte Übersicht mit kurzen Erklärungen zu den Bestandteilen der jeweiligen Visualisierungen.

Für die Studie wurden 18 Aufgaben vorbereitet. Dabei wurden jeweils sechs Fragen zu jeder Visualisierung gestellt. Von diesen bezogen sich jeweils drei auf FOAF und drei auf PersonasOnto. Es gab keine feste Zuordnung zwischen Sechsergruppen aus Fragen und Visualisierungen. Die Zuordnung wurde stattdessen gleichmäßig von Teilnehmer zu Teilnehmer variiert, um alle Kombinationen abzudecken. Entsprechend erhielt jeder Teilnehmer sechs Fragen zu jeder der drei Visualisierungen. So konnten Teilnehmer die Visualisierungen untereinander vergleichen.

Die Aufgaben beruhten auf der Beantwortung von Fragen wie „Welche Klassen scheinen für die Ontologie von zentraler Bedeutung zu sein?“, „Welche Klassen wurden aus anderen Ontologien importiert?“ oder „Welche Dateneigenschaften hat die Klasse *Image*?“. Durch diese Aufgaben wurden die Teilnehmer dazu gebracht, sich mit den Darstellungen der Ontologien auseinanderzusetzen und die Beziehungen zwischen deren Elementen nachzuvollziehen. Während die Antworten zwar auf Korrektheit überprüft wurden, lag das hauptsächliche Interesse darin, zu erfahren, wie die Teilnehmer mit den Visualisierungen umgehen. Daher wurden Teilnehmer auch stets dazu aufgefordert, „laut mitzudenken“, während sie die Visualisierungen auf dem Bildschirm betrachteten und je nach Bedarf mit ihnen interagierten. Zusätzlich erhielten die Teilnehmer noch einen Fragebogen zu ihren Vorkenntnissen und den während der Studie erkannten Schwierigkeiten.

Für die Studie wurden Teilnehmer ausgewählt, die in Zukunft mit Ontologien in Kontakt kommen könnten, jedoch nicht notwendigerweise mit formalen Ontologiebeschreibungssprachen. Daher wurden sechs Forscher aus unterschiedlichen Feldern der Informationstechnologie (jedoch nicht Semantic-Web-Technologien) eingeladen. Keiner davon brachte tiefergehende Vorkenntnisse im Bereich Ontologien mit. Die verwendeten Ontologien waren den Teilnehmern inhaltlich unbekannt.

Die Aufgaben wurden von den Teilnehmern zumeist korrekt gelöst (84%). Zwei Teilnehmer gaben bei dem Versuch auf, in GrOWL einzelne der Aufgaben zu lösen. Die Implementierung verfügte über keine Suchfunktion für bestimmte Ontologieelemente, was das gezielte Auffinden sehr mühselig machte. Im Allgemeinen wurde von den Teilnehmern VOWL bevorzugt. Als besondere Vorzüge wurden dabei die relativ klare Darstellung mit wenigen verwirrend übereinanderliegenden Kanten genannt, wie auch die Beschriftungen, die die Visualisierung selbsterklärend machen. Ebenso wurde die Kreisform der Klassenknoten positiv hervorgehoben, welche das radiale und dennoch verdeckungsfreie Anliegen von Pfeilen ermöglicht. Letztendlich wurde es als vorteilhaft empfunden, dass einzelne Notationselemente an die visuelle Darstellung aus anderen bereits bekannten Visualisierungen wie UML angelehnt waren.

3.2.1.5 Interviews mit erfahrenen Benutzern

Zusätzlich zu den Benutzerstudien wurde fünf Benutzern, die bereits mit Ontologien gearbeitet hatten, die Möglichkeit gegeben, VOWL auszuprobieren. Dieser Schritt der Evaluation war eher als „Interview“ denn als Studie mit festgelegten Aufgaben gedacht. Dementsprechend sollten die Teilnehmer keine festgelegten Aufgaben lösen. Sie erhielten lediglich einige Fragen, um diverse Eigenarten der Visualisierung in den Fokus zu rücken (zum Beispiel: „Werden Verknüpfungen zwischen Elementen durch das interakti-

ve Highlighting hinreichend klar dargestellt?“), wenn die Teilnehmer nicht ohnehin bereits von selbst darauf gestoßen waren.

Die Teilnehmer konnten wiederum selbständig die webbasierte VOWL-Implementierung verwenden. Dieses Mal stand neben FOAF, Personas-Onto, MUTO noch die Ontologie Semantically-Interlinked Online Communities (SIOC) [BB10] zur Verfügung. Gruppen aus jeweils zwei Teilnehmern betrachteten und kommentierten die Visualisierung gleichzeitig (und ein Teilnehmer tat dies alleine). Auf diese Weise ergaben sich hier auch Diskussionen zwischen mehreren Nutzern. Insgesamt wurden drei separate Interview-Runden durchgeführt.

In jeder der Interview-Runden sagten Teilnehmer aus, dass Klassen und Eigenschaften auf Anhieb als solche zu erkennen seien. Speziellere Konstrukte wie Subproperties und Mengen aus äquivalenten Klassen waren nicht immer von Anfang an klar. Nach kurzer Erklärung wurden sie jedoch als verständlich eingestuft. Bei Vereinigungsmengen- und Schnittmengenknoten herrschte in einem Fall Uneinigkeit darüber, ob eine zusätzliche explizite Beschriftung (als „union“ beziehungsweise „intersection“) nötig sein könnte. Die Standardfarbcodierung wurde von zwei Teilnehmern anfangs als unklar eingestuft. Diese und weitere Teilnehmer konnten sich im Verlauf des Interviews jedoch die Bedeutung einiger Farben noch selbständig erschließen.

Das kräftebasierte Layout wurde generell als positiv bewertet. Die Möglichkeit, alternative Layouts anzuwenden, um beispielsweise Hierarchien baumförmig darzustellen, wurde allerdings als Wunsch genannt. Gleichzeitig wurde die Möglichkeit, Knoten an benutzerdefinierten Orten befestigen zu können, von zwei Teilnehmern explizit als nützlich hervorgehoben. Ebenso befand ein Teilnehmer die Option zum Ändern des Abstands der Elemente als hilfreich. Ein weiterer bemerkte, dass die Darstellung bei größeren Ontologien sehr viele Elemente und zum Teil überlappende Texte beinhaltet.

3.2.1.6 Überprüfung von VOWL mittels OntoViBe

Bei OntoViBe (Ontology Visualization Benchmark) handelt es sich um eine Sammlung von Ontologien, die so gestaltet sind, dass sie die meisten der in OWL 2 unterstützten Konzepte und Kombinationen daraus enthalten [HLN+14; HLN+15]. Der Zweck dieser Ontologien besteht darin, Ontologievisualisierungen auf Vollständigkeit und Robustheit hin zu testen.

Wird diese Ontologie mit VOWL dargestellt (Abb. 3.25), ist erkennbar, dass die generelle Struktur der Ontologie sichtbar ist. Klassen und Literale sind klar unterscheidbar, Domains und Ranges von Eigenschaften können abgelesen werden und die Vererbungshierarchie zwischen Klassen wird dargestellt. Insbesondere ist beispielsweise im Bereich ① der Abbildung zu sehen, dass Eigenschaften, deren Domain und Range übereinstimmt, korrekt

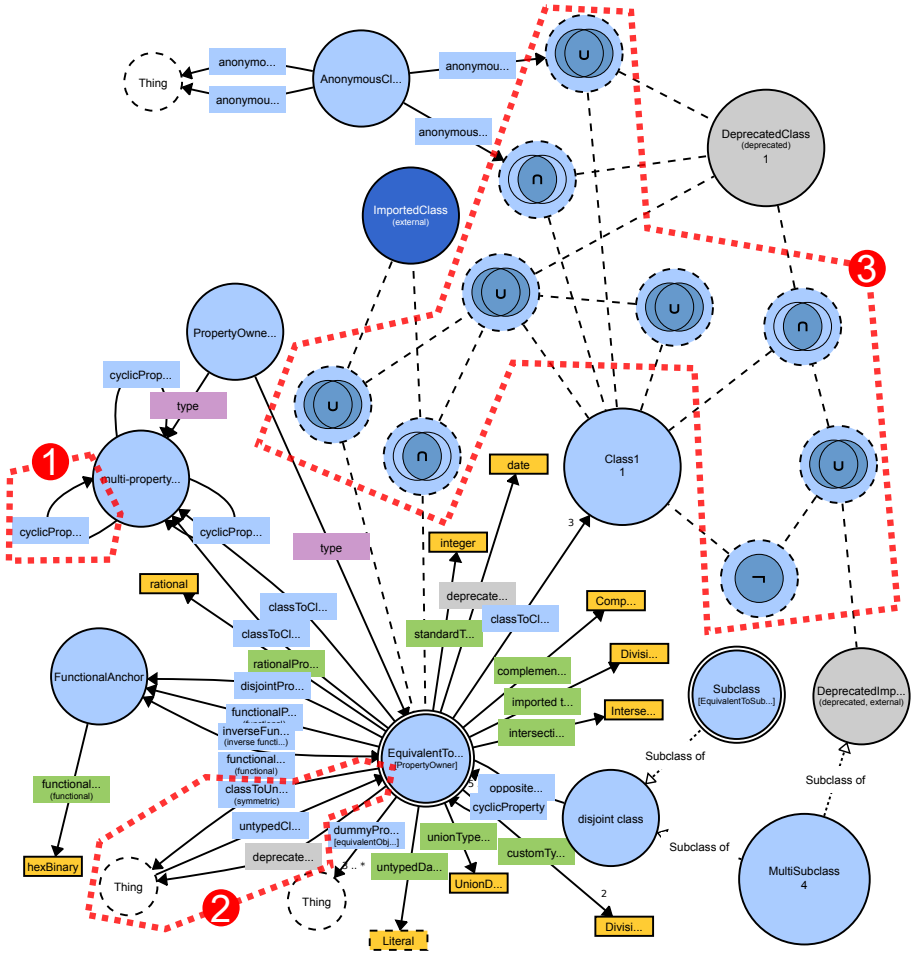


Abbildung 3.25: Der größte zusammenhängende Subgraph der Benchmark-Ontologie OntoViBe [HLN+14; HLN+15], wie sie mit VOWL dargestellt wird. (Die gepunkteten Polygonrahmen sind nicht Teil der Visualisierung und wurden für Erklärungen im Text nachträglich hinzugefügt.)

dargestellt werden. Im Fall mehrerer Eigenschaften mit übereinstimmender Domain und Range werden die entsprechenden Kanten ebenfalls so geführt, dass alle Eigenschaften zu sehen sind, wie dies im Bereich ② der Fall ist. Dahingegen zeigen sich auch noch einzelne Schwächen in der aktuellen Version von VOWL, die in zukünftigen Versionen behoben werden müssen: Einige der Klassen für Vereinigungs- und Schnittmengen in Bereich ③ sind anonym, andere werden durch IRIs identifiziert. In der VOWL-Darstellung sind diese Identifikationen jedoch noch nicht zu sehen. Ebenso ist die Verbindungsrichtung zwischen einigen dieser Klassen noch nicht eindeutig erkennbar.

3.2.2 Entwicklung von QueryVOWL

QueryVOWL wurde im Rahmen dieses Promotionsvorhabens konzeptuell entworfen und im Rahmen einer Diplomarbeit weiter ausdefiniert sowie prototypisch implementiert. Konzept und Prototyp wurden anschließend auch im Rahmen eines Workshops und der Demo-Session einer Konferenz vorgestellt.

> Die im Folgenden präsentierten Inhalte wurden zum Teil bereits anderweitig veröffentlicht. Insbesondere wird auf den im Folgenden aufgeführten Arbeiten, an denen der Autor maßgeblich mitwirkte, aufgebaut:
>
> HLS+ F. Haag, S. Lohmann, S. Siek und T. Ertl. "Visual querying of linked data with QueryVOWL". In: *Joint Proceedings of SumPre 2015 and HSWI 2014-15*. Noch nicht erschienen. CEUR-WS
>
> HLS+15b F. Haag, S. Lohmann, S. Siek und T. Ertl. "QueryVOWL: visual composition of SPARQL queries". In: *The Semantic Web: ESWC 2015 Satellite Events*. Bd. 9341. LNCS. Springer, 2015. DOI: 10.1007/978-3-319-25639-9_12
>
> HLS+15a F. Haag, S. Lohmann, S. Siek und T. Ertl. "QueryVOWL: a visual query notation for linked data". In: *The Semantic Web: ESWC 2015 Satellite Events*. Bd. 9341. LNCS. Springer, 2015. DOI: 10.1007/978-3-319-25639-9_51
>
> Zusätzlich wird inhaltlich auf die folgenden studentischen Arbeiten, an deren Betreuung der Autor beteiligt war, zurückgegriffen:
>
> Sie14 S. Siek. "VOWL-basierte Visualisierung für SPARQL-Anfragen". Betreuer: Florian Haag, Steffen Lohmann. Diplomarbeit. Universität Stuttgart, 2014

3.2.3 Grundlegender Aufbau

QueryVOWL funktioniert wie andere in Abschnitt 2.3.1.1 beschriebene Objektgraph-basierte Anfragesprachen, indem ein Graphmuster in Form eines Node-Link-Diagramms aufgebaut wird, welches zum Teil aus Platzhaltern besteht. Die Filterung geschieht dann dahingehend, dass in der Datenmenge alle Elemente gesucht werden, die sich im gegebenen Graphmuster für die Platzhalter einsetzen lassen. Jedes Ergebnis stellt eine mögliche Belegung für das Graphmuster dar, das jeden der Platzhalter auf eine IRI oder einen primitiven Wert aus dem RDF-Graphen abbildet.

Um den Lernaufwand zu minimieren, wurden viele der visuellen Merkmale direkt aus VOWL übernommen (Abbildung 3.26). Einzelne der visuellen Elemente wurden, wo nötig, erweitert, um schwächere Einschränkungen (beispielsweise Eigenschaften ohne festgelegte Richtung) auszudrücken.

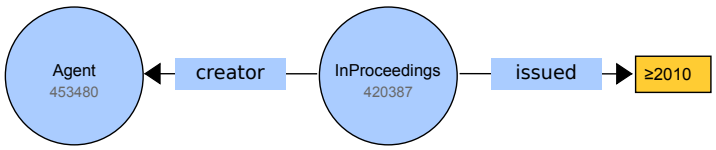
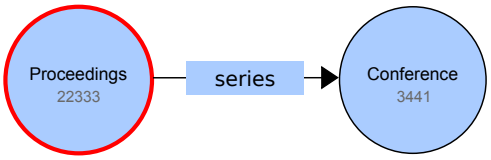
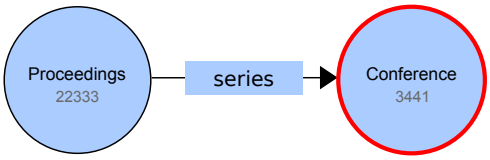


Abbildung 3.26: Die an VOWL angelehnte Notation von QueryVOWL stellt ein Graphmuster in einer Node-Link-Ansicht dar. Das gezeigte Graphmuster entspricht allen Subgraphen in Faceted DBLP [DB08], welche eine Konferenzpublikation, die frühestens 2010 erschienen ist, mit einem ihrer Autoren enthalten.



(a) Diese Anfrage ruft Tagungsbände von Konferenzen ab, die zu einer Konferenzserie gehören.



(b) Diese Anfrage ruft Konferenzserien ab, in deren Rahmen Tagungsbände veröffentlicht wurden.

Abbildung 3.27: Die Markierung eines Elements in einer QueryVOWL-Anfrage stellt ein Teil der Anfrage dar, wie anhand dieser zwei auf Faceted DBLP [DB08] basierenden Anfragen erkennbar wird.

Außerdem wurden an einigen Stellen interaktive Elemente hinzugefügt, um den Filtergraphen zu bearbeiten.

Die bereits aus VOWL bekannte Markierung wurde auch in QueryVOWL übernommen und stellt hier einen Teil der Anfrage dar. Semantisch gesehen ist das markierte Element dasjenige, dessen Wert eigentlich „gesucht“ wird (Abb. 3.27).

3.2.4 Visuelle Elemente

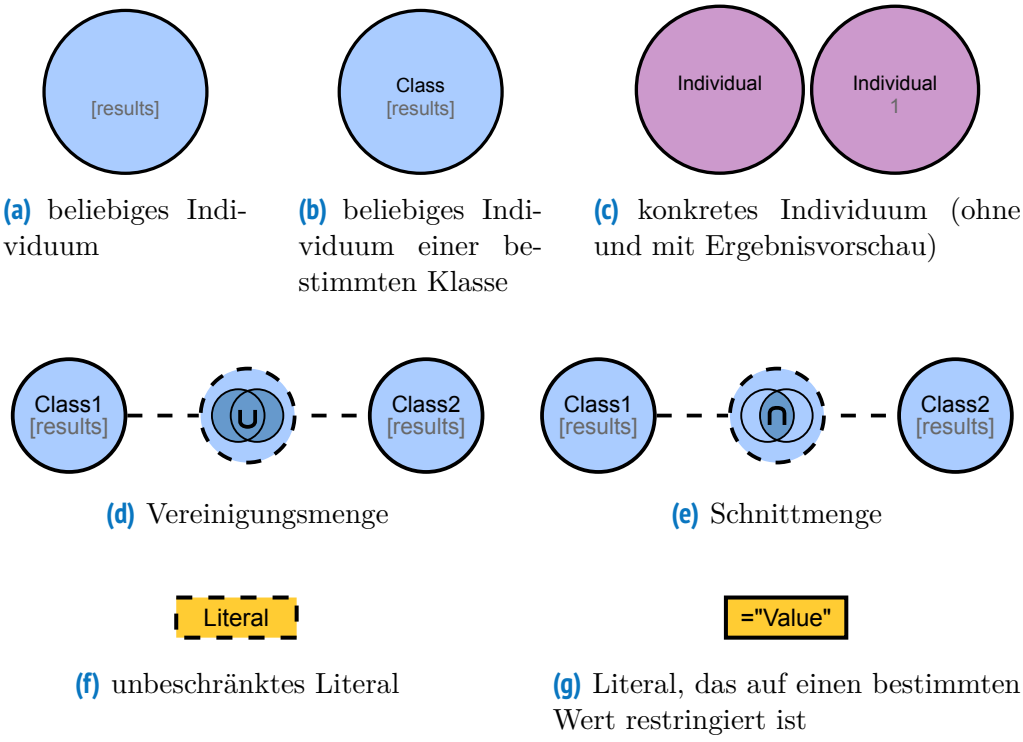
Die folgenden Abschnitte beschreiben die visuellen Elemente, die für QueryVOWL definiert sind. Ihre Bedeutung ist durch eine Abbildungsvorschrift auf ein SPARQL-Fragment definiert. Diese wird jeweils durch ein Beispiel verdeutlicht.

3.2.4.1 Klassen

Die VOWL-Klassennotation, die bereits im in Abschnitt 3.2.1.5 beschriebenen Interview zu VOWL als verständlich eingestuft wurde, wird eingesetzt, um Platzhalter für gesuchte Individuen zu repräsentieren. In einem Ergebnis wird jeder Klassenknoten auf genau ein passendes Individuum

aus der Datenmenge abgebildet. Ein passendes Individuum erfüllt dabei die Einschränkungen, die über den Klassenknoten selber und das umliegende Graphmuster definiert sind. Die Menge aller auf diese Weise passender Individuen stellt somit implizit eine Klasse dar. Dies entspricht der visuellen Notation in Form eines VOWL-Klassenknotens.

Enthält der Klassenknoten keine Beschriftung (Abbildung 3.28a), so sind keine Einschränkung bezüglich des Typs der Individuen vorgegeben, analog zu den „unknown concept“-Knoten aus OntoVQL [FH07] (siehe Seite 18). Andernfalls sind die Individuen auf Instanzen der Klasse beschränkt, deren Namen im Klassenknoten angezeigt wird (Abbildung 3.28b). Dabei tritt als wichtiger Unterschied zu VOWL der Umstand auf, dass in QueryVOWL zwei oder mehr Klassenknoten existieren können, welche dieselbe Klasse bezeichnen, da sie im Anfrageergebnis als Platzhalter für unterschiedliche Individuen fungieren. Dieses Merkmal wurde in einem Vergleich von VQLs bereits als typisch für einige der Anfragekonzepte eingestuft [Cha97]. Es wird lediglich von OQD (siehe Seite 18) umgangen, indem die eigentlichen (beliebig oft vorkommenden) Platzhalterknoten in Rechtecke eingerahmt werden, sodass für jede Klasse genau ein Rechteck existiert [KM93]. Dies berücksichtigt allerdings nicht die in RDF-Graphen mögliche Mehrfachvererbung.



(a) beliebiges Individuum

(b) beliebiges Individuum einer bestimmten Klasse

(c) konkretes Individuum (ohne und mit Ergebnisvorschau)

(d) Vereinigungsmenge

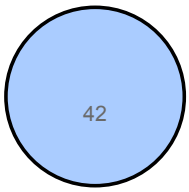
(e) Schnittmenge

(f) unbeschränktes Literal

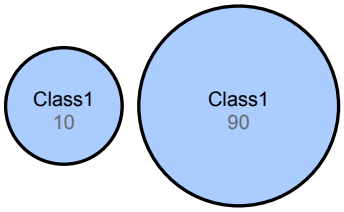
(g) Literal, das auf einen bestimmten Wert restringiert ist

Abbildung 3.28: Knotentypen in QueryVOWL

Die Anzahl der insgesamt passenden Individuen wird ebenfalls im Klassenknoten angezeigt (Abbildung 3.29a). Optional kann diese auch, analog zu VOWL, auf den Radius der Klassenknoten abgebildet werden (Abbildung 3.29b).



(a) Anzeige der Ergebnisanzahl



(b) Abbildung der Ergebnisanzahl auf den Radius

Abbildung 3.29: Knotenmarkierungen in QueryVOWL

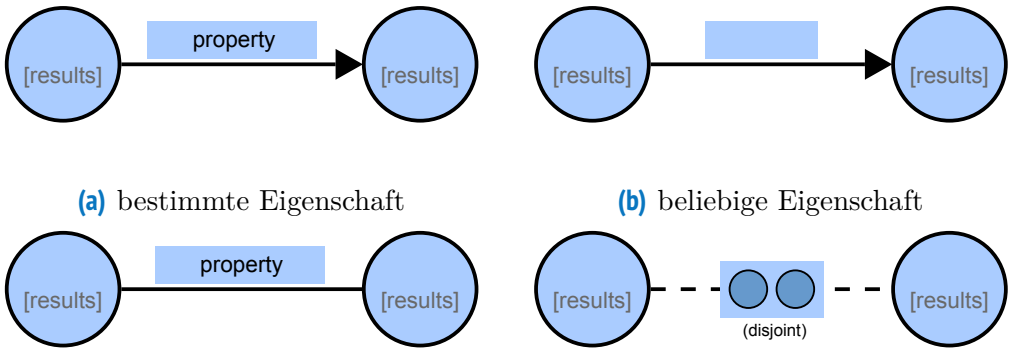
3.2.4.2 Individuen

Bestimmte Individuen, die bereits in der Anfrage als konstant festgelegt sind, können in QueryVOWL als Individuenknoten ausgedrückt werden (Abbildung 3.28c). Ihre Beschriftung zeigt das Label des Individuums. Zur Unterscheidung von den Klassenknoten wurde für Individuenknoten die Farbgebung von RDFS-Knoten aus VOWL verwendet. Konstante Individuenknoten lassen sich auch als Repräsentationen von Mengen, die genau das festgelegte Individuum enthalten, interpretieren. Deshalb kann optional in einem Individuenknoten die Anzahl *1* angezeigt werden.

3.2.4.3 Eigenschaften

Wie in VOWL können QueryVOWL-Knoten durch Eigenschaften verbunden werden (Abbildung 3.30a). Auf diese Weise wird gefordert, dass die gefundenen Ressourcen durch die entsprechenden Eigenschaften verbunden sind.

Soll nicht festgelegt werden, welche Eigenschaft konkret gefunden werden muss, so kann die Beschriftung der Eigenschaft im QueryVOWL-Graphen entfallen (Abbildung 3.30b). Die Eigenschaft wird auf diese Weise selber zum Platzhalter und jedes Ergebnis enthält auch eine konkrete Eigenschafts-IRI, auf welche die Eigenschaftskante abgebildet wird. Auf diese Weise lässt sich folglich ebenfalls ermitteln, wie zwei Knoten in der Datensammlung miteinander verbunden sind. In QueryVOWL ist dies allerdings im Gegensatz zu manchen anderen Arbeiten [HHL+09; HLS10; ZB11] nur für direkte Verbindungen vorgesehen.



(a) bestimmte Eigenschaft (b) beliebige Eigenschaft

(c) Eigenschaft in beliebiger Richtung (d) Disjunktivität

Abbildung 3.30: Kantentypen in QueryVOWL

3.2.4.4 Ungerichtete Eigenschaften

Um für eine Eigenschaft offen zu lassen, in welcher Richtung diese verlaufen muss, kann wie in qGraph [BIJ01] (siehe Seite 18) die Pfeilspitze an der Eigenschaftskante entfernt werden (Abbildung 3.30c). Dies ist sowohl für bestimmte Eigenschaften wie auch für die unbeschrifteten Eigenschaften-Platzhalter möglich.

3.2.4.5 Literale

Die VOWL-Darstellung für Literale wurde übernommen, um primitive Werte zu repräsentieren (Abbildung 3.28f). Sollen diese Werte eingegrenzt werden, so kann der gewünschte Wert in dem Literalknoten dargestellt werden (Abbildung 3.28g). Dazu können ähnlich wie in anderen Konzepten [MK93; DC96; RSB+08] auch Vergleichsoperatoren verwendet werden. So kann man beispielswerte Datumswerte durch ein frühestmögliches Datum oder Zahlenwerte durch einen Maximalwert einschränken.

3.2.4.6 Disjunktivität

Sofern sonstige Einschränkung im Graphmuster dem nicht entgegenstehen, können in einem Ergebnis mehrere Klassenknoten auf dasselbe Individuum abgebildet werden, wie dies auch für qGraph beschrieben ist [BIJ01]. Um zu erzwingen, dass sie auf unterschiedliche Individuen abgebildet werden, können die Klassenknoten paarweise mit einer speziellen Kante verbunden werden. Als grafische Darstellung wurde die VOWL-Repräsentation von Disjunktivität gewählt (Abbildung 3.30d). Dieselbe Vorgehensweise lässt sich auf Literalknoten anwenden.

3.2.4.7 Vereinigungsmengen

Ähnlich wie Klassenknoten werden Vereinigungsmengenknoten auf eine IRI pro Ergebnis abgebildet (Abbildung 3.28d). Die IRI muss jedoch der Disjunktion aus allen Einschränkungen der mit dem Vereinigungsmengenknoten verbundenen Klassen- oder Individuenknoten entsprechen. Die Menge aller für den Vereinigungsmengenknoten passenden Individuen ist also die Vereinigungsmenge aus allen für die verbundenen Knoten passenden Individuen. Diese Darstellung einer Subgraphdisjunktion hat somit Gemeinsamkeiten mit der Darstellungsweise aus NITELIGHT [RSB+08] oder GLOO [FH06] (ab Seite 18 beschrieben).

3.2.4.8 Schnittmengen

Analog zur Vereinigungsmenge können in QueryVOWL auch Schnittmengenknoten verwendet werden (Abbildung 3.28e). Die Menge aller für den Schnittmengenknoten passenden Individuen ist die Schnittmenge aller für die verbundenen Knoten passenden Individuen.

3.2.5 Interaktion

Um zu belegen, dass sich die definierte Anfragesprache als interaktives Konzept aufbauen lässt, wurden im Rahmen der Diplomarbeit zum Thema einige Interaktionselemente vorgeschlagen [Sie14]. Diese betrafen einerseits die visuellen Elemente von QueryVOWL selber und andererseits die Oberfläche, in die der Anfragegraph eingebettet wird. Einige Beispiele dafür sind in Abbildung 3.31 dargestellt.

Um eine einfache Interaktionsmöglichkeit direkt in der Visualisierung bereitzustellen, wurden die visuellen QueryVOWL-Elemente mit anklickbaren Hotspots versehen. Alle Elemente erhielten dabei einen Hotspot zum Entfernen des Elements. Klassenknoten und Eigenschaftsknoten wurden zudem mit Hotspots zum Entfernen ihrer internen Einschränkung – der festgelegten Klasse oder Eigenschaft – ausgestattet. Ferner wurden elementtypspezifische Funktionen wie Hotspots zum Anfügen von Eigenschaftskanten an Klassenknoten oder zum Umkehren der Richtung einer Eigenschaftskante definiert. Letztendlich wurden auch aufklappbare Auswahllisten zum Ersetzen einer Klasse in einem Klassenknoten beziehungsweise einer Eigenschaft in einer Eigenschaftskante zur Visualisierung hinzugefügt.

Für die QueryVOWL-Visualisierung wurde vorgesehen, dass sie mit einer Seitenleiste und einer Ergebnisliste zusammen eingefügt wird. Die Seitenleiste ist so definiert, dass dort diverse Editieroptionen vorhanden sind, wie eine Auswahl zum Ersetzen des aktuell ausgewählten Elements oder zum Hinzufügen neuer Knoten, die mit dem ausgewählten Element

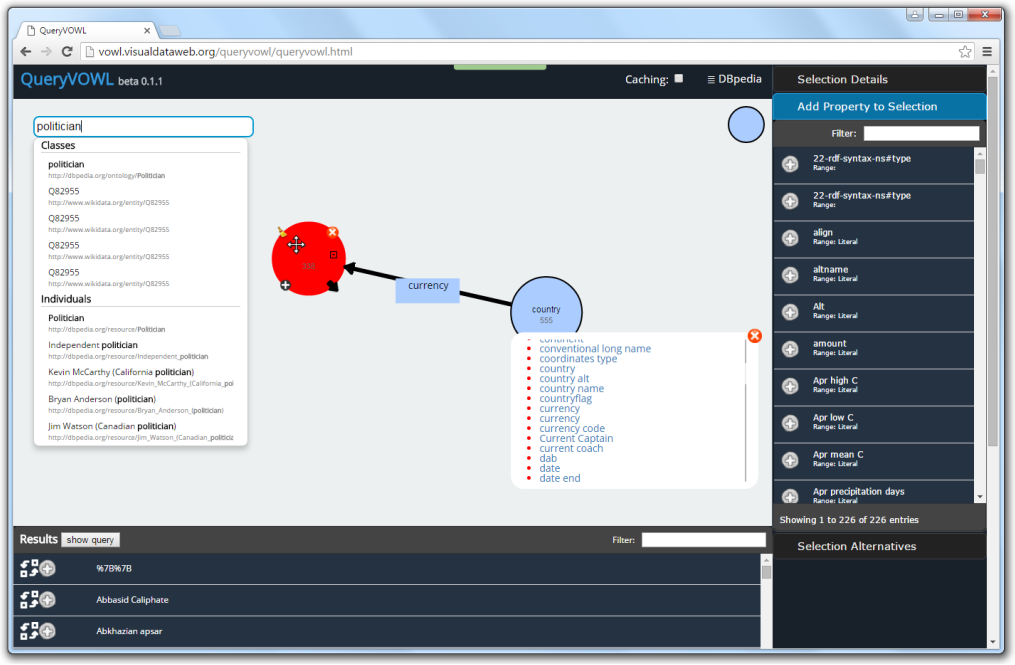


Abbildung 3.31: Im Vergleich zu VOWL wurden einige der Elemente für QueryVOWL um interaktive Bestandteile erweitert, um Benutzern das Erstellen und Erweitern von Anfragen zu ermöglichen. Im Screenshot des Prototypen, der im Rahmen der Diplomarbeit zum Thema [Sie14] entstand und die Interaktionsmöglichkeiten umsetzt, erkennt man, dass neben den Editiermöglichkeiten in der Seitenleiste per Volltextsuche in der Datensammlung und dem dazugehörigen Schema neue Elemente zur Anfrage hinzugefügt werden können. Ein Popup stellt Eigenschaften dar, die mit dem *country*-Knoten verknüpft sind. Zudem befindet sich der Mauszeiger über einem der Klassenknoten, weshalb für diesen Knoten Symbole an den anklickbaren Hotspots gezeigt werden.

verbunden sind. Die Ergebnisliste ist dazu gedacht, alle IRIs oder primitiven Werte, die für das markierte Element eingesetzt werden können, aufzulisten.

3.2.6 Einschränkungen

Mit QueryVOWL können komplette alternative Subgraphen auf relativ einfache Weise ausgedrückt werden, indem sie durch einen Vereinigungsmengenknoten verbunden werden. Ebenso könnten Disjunktionen in den Filtern innerhalb von Knoten vorgesehen werden, beispielsweise in Klassenknoten (Abschnitt 3.2.4.1) oder Literalknoten (Abschnitt 3.2.4.5). Einige andere Muster für logische Verknüpfungen werden aber nicht unterstützt. So ist beispielsweise bislang keine visuelle Notation vorgesehen, um

die Existenz eines Attributs mit einem bestimmten Wert oder alternativ die Existenz eines anderen Attributs mit einem anderen bestimmten Wert zu fordern. Stattdessen müsste der Klassenknoten, mit dem die Attributknoten verbunden wären, dupliziert werden, um die beiden Varianten separat auszudrücken. Beide Versionen des Klasseknotens müssten dann mit einem Vereinigungsmengenknoten verbunden werden.

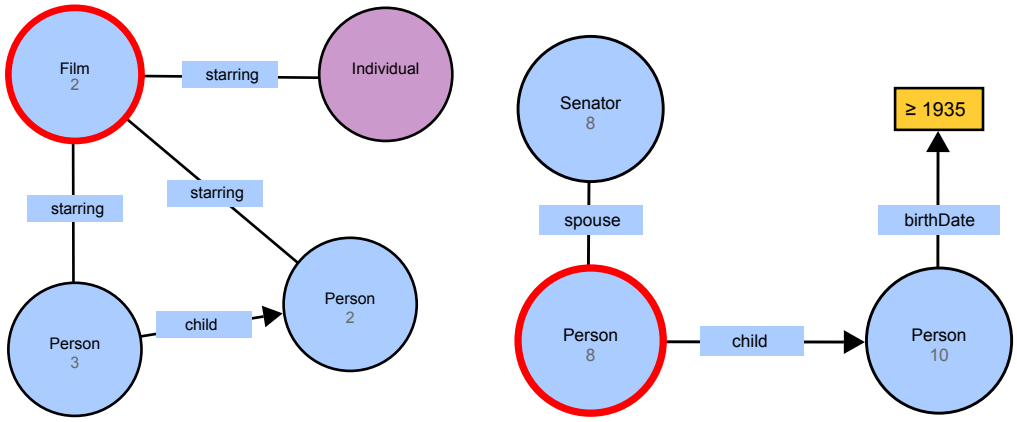
Eine weitere Funktionalität, die momentan noch nicht in der visuellen Notation enthalten ist, stellen Kardinalitäten für Eigenschaften dar. Durch einen entsprechenden Aufbau des Anfragegraphen lässt sich mit QueryVOWL im aktuellen Zustand ausdrücken, dass ein Datenelement eine bestimmte Anzahl von Werten für eine bestimmte Eigenschaft haben soll. Dazu müssen entsprechend viele Knoten erzeugt und mittels der entsprechenden Eigenschaftskanten mit dem Klassenknoten für das Datenelement verbunden werden. Ebenso müssen die neuen Knoten über Disjunktivitätskanten untereinander verbunden werden. Für eine hohe Anzahl von Eigenschaftswerten wird der Graph durch diese Vorgehensweise aber unnötig komplex. Zudem ist es auf diese Weise nur möglich, eine genaue Anzahl von Eigenschaftswerten einzufordern; eine Mindest- oder Maximalzahl kann nicht ausgedrückt werden. Da auch die Ontologievisualisierung VOWL noch weiterentwickelt wird und die akkurate Repräsentation von Kardinalitäten eine der noch offenen Fragen darstellt, besteht die Aussicht darauf, dass zukünftige Versionen von QueryVOWL auf diesen Erweiterungen von VOWL aufsetzen können.

Letztendlich basiert QueryVOWL auf SPARQL, weshalb der Funktionsumfang durch SPARQL begrenzt ist. Dadurch fehlt beispielsweise die Möglichkeit, Verbindungen unterschiedlicher Länge zwischen Klassen auf Grundlage regulärer Pfadausdrücke abzurufen [Woo12].

3.2.7 Evaluation

Im Rahmen der Diplomarbeit zum Thema wurde von dem Studenten eine kleine Benutzerstudie mit sechs Teilnehmern aus unterschiedlichen nicht informationstechnischen Berufsgruppen durchgeführt [Sie14]. Die Teilnehmer mussten dabei sieben in natürlicher Sprache formulierte Fragen mit QueryVOWL darstellen (Abbildung 3.32) sowie die Bedeutung einer gezeigten QueryVOWL-Anfrage in natürlicher Sprache ausdrücken (Abbildung 3.33).

In der Studie wurde ein ebenfalls im Rahmen der Diplomarbeit entwickelter webbasierter Prototyp von QueryVOWL eingesetzt. Der Prototyp unterstützt nur grundlegende QueryVOWL-Elemente, nämlich Klassen, Individuen, Eigenschaften und Literale. Die Anzahl der Knoten, die eingefügt werden können, ist programmseitig nicht begrenzt, jedoch sind



(a) „In wie vielen Filmen spielt Bruce Willis die Hauptrolle?“, „Gibt es darunter einen Film, in der eine Person gemeinsam mit ihrem Kind in der Hauptrolle vertreten ist?“

(b) „Wie viele Personen sind Ehepartner eines Senators?“, „Wie viele davon haben ab 1935 geborene Kinder?“

Abbildung 3.32: Zwei der sieben Kompositionsaufgaben der Benutzerstudie aus der Diplomarbeit zu QueryVOWL [Sie14]. Die Studienteilnehmer erhielten schrittweise Teile der natürlichsprachlichen Anfragen (in den Bildunterschriften paraphrasiert) und mussten in der webbasierten Implementierung nach und nach die visuellen Anfragen erzeugen.

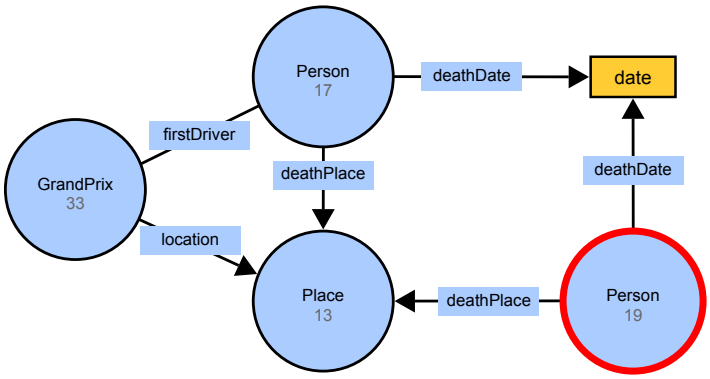


Abbildung 3.33: Die Verständnisaufgabe der Benutzerstudie zu QueryVOWL aus der Diplomarbeit zum Thema [Sie14]. Studienteilnehmer erhielten die QueryVOWL-Darstellung und mussten in ihren eigenen Worten beschreiben, dass mit der Anfrage Personen gesucht werden, die Sterbedatum und -ort mit dem ersten Fahrer eines Grand Prix teilen.

bei den meisten Datensammlungen nur begrenzt komplexe Anfragen möglich, da sich die Ergebnismenge schon mit wenigen (ca. 5) Knoten auf wenige Ergebnisse reduzieren lässt. Die Verarbeitungsgeschwindigkeit hängt hauptsächlich vom SPARQL-Endpunkt auf dem Server ab, wie in Abschnitt 3.1.7.3 bereits in Bezug auf den EFFK-Prototyp beschrieben wurde. Der Prototyp ist in Abbildung 3.34 dargestellt. Auf diese Weise kamen in der Studie die grundlegenden Bestandteile von QueryVOWL zum Einsatz.

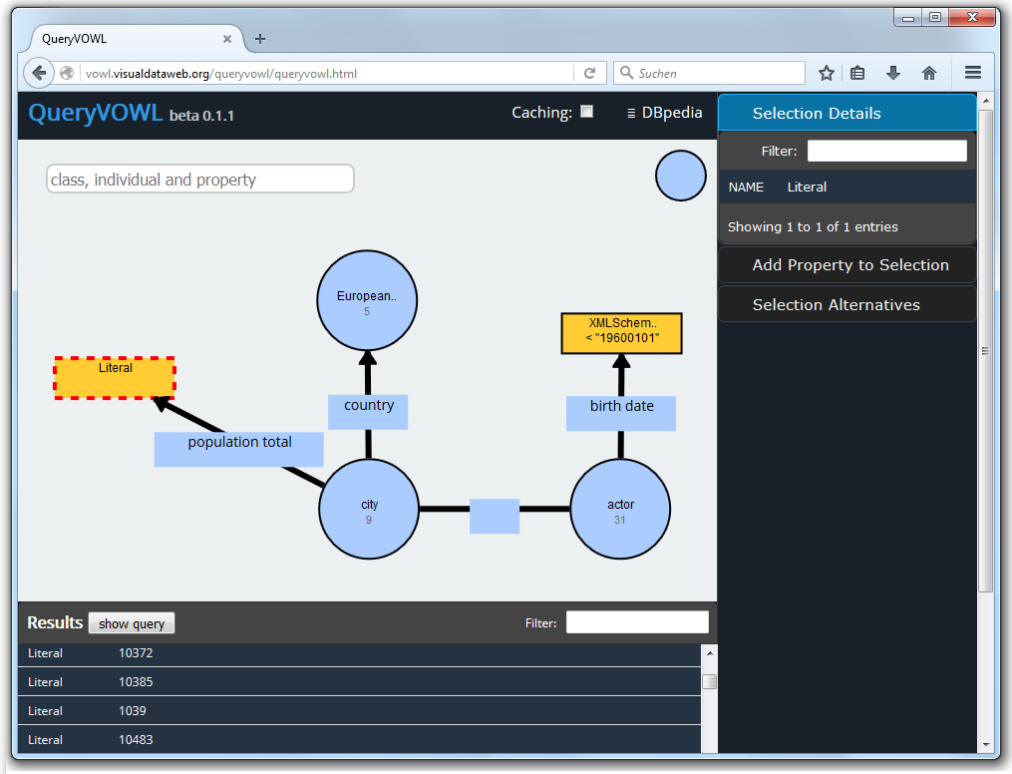


Abbildung 3.34: Die Benutzeroberfläche des Prototyps, der im Rahmen der Diplomarbeit zum Thema [Sie14] entwickelt wurde. Gezeigt wird die Anfrage „Ermittle die Einwohnerzahlen von europäischen Städten, die mit einem vor 1960 geborenen Schauspieler im Zusammenhang stehen.“ an DBpedia [ABK+07].

Die Teilnehmer fanden das Konzept insgesamt nachvollziehbar und konnten die Aufgaben weitgehend lösen. Dabei hielten sie sich allerdings eng an die zuvor vorgestellten Funktionen und experimentierten nicht bereitwillig mit der interaktiven Implementierung.

Schwierigkeiten stellte dabei zunächst die Umsetzung der in natürlicher Sprache formulierten Frage auf das Datenmodell dar, da zeitweise nicht klar war, ob ein Begriff eine Klasse oder eine Eigenschaft bezeichnete. Zum

Beispiel war nicht allen umgehend klar, dass ein *Kind* eine *Person* ist, die in einer *Kind*-Beziehung zu einer anderen Person steht, nicht ein Individuum aus der Klasse *Kind*.

Zudem blieb den Teilnehmern die Unterscheidung zwischen Klassen und Individuen unklar. Ein weiterer erkannter offener Punkt war die Richtung von Relationen, die teilweise mehrdeutig war beziehungsweise nicht korrekt identifiziert wurde – in welche Richtung muss beispielsweise eine *Kind*-Relation zeigen, auf die Eltern oder auf die Kinder? Dies könnte möglicherweise mit expliziteren Beschriftungen der Art *hat Kind* und *ist Kind von* (ähnlich, wie dies beispielsweise von der Weboberfläche Pubby gelöst wird [CB11]) verbessert werden.

Des Weiteren fiel es den Teilnehmern auch schwer, ein Graphmuster aufzustellen, das denselben Wert für eine Datentypeigenschaft zweier Individuen verlangt. Bei Objekteigenschaften stellte dies keine Schwierigkeit dar und die Teilnehmer bauten die Verbindung über einen gemeinsamen verbindenden Klassenknoten auf. Im Gegensatz dazu gingen sie bei Datentypeigenschaften davon aus, dass zwei Literalknoten, die über eine „Vergleichskante“ verbunden sind, eingesetzt werden müssten. Dies wurde möglicherweise von der konkreten Implementierung beeinflusst, die bei Literalknoten den Vergleich über einen Vergleichsoperator stark in den Vordergrund rückt.

3.2.8 Implementierung

Neben dem webbasierten Prototyp aus der oben genannten Diplomarbeit wurde zu QueryVOWL auch eine WPF-basierte Implementierung in C# angefertigt (Abbildung 3.36 unten). Bei ihrer Entwicklung wurde weniger Wert auf Benutzerfreundlichkeit der interaktiven Oberfläche gelegt, dafür mehr Wert auf die Vollständigkeit der Sprache. Zudem dient die WPF-Implementierung als Muster für eine objektorientierte Umsetzung der QueryVOWL-Notation.

Wie in Abbildung 3.35 dargestellt, verfügen alle Knoten unter anderem über eine Methode `DoAddRestrictions`. Diese Methode fügt die Einschränkungen beziehungsweise Graphmuster zur entstehenden SPARQL-Anfrage hinzu, wenn diese generiert wird. Für die meisten Knotentypen ist die Implementierung dieser Methode trivial – Klassenknoten mit Klassenbeschränkung fügen beispielsweise das entsprechende Tripel mit der `rdf:type`-Einschränkung zum Graphen hinzu. Daher bleibt für Anfragengraphen, die nur n solche Knoten und m Eigenschaftskanten enthalten, die Zeitkomplexität für die Erzeugung der SPARQL-Anfrage auch im Bereich von $\mathcal{O}(n \cdot m)$.

Für die Knoten, die Mengenoperationen darstellen, ist jedoch mehr Rechenaufwand notwendig, da die Teilnahme einer Klasse an einer Mengen-

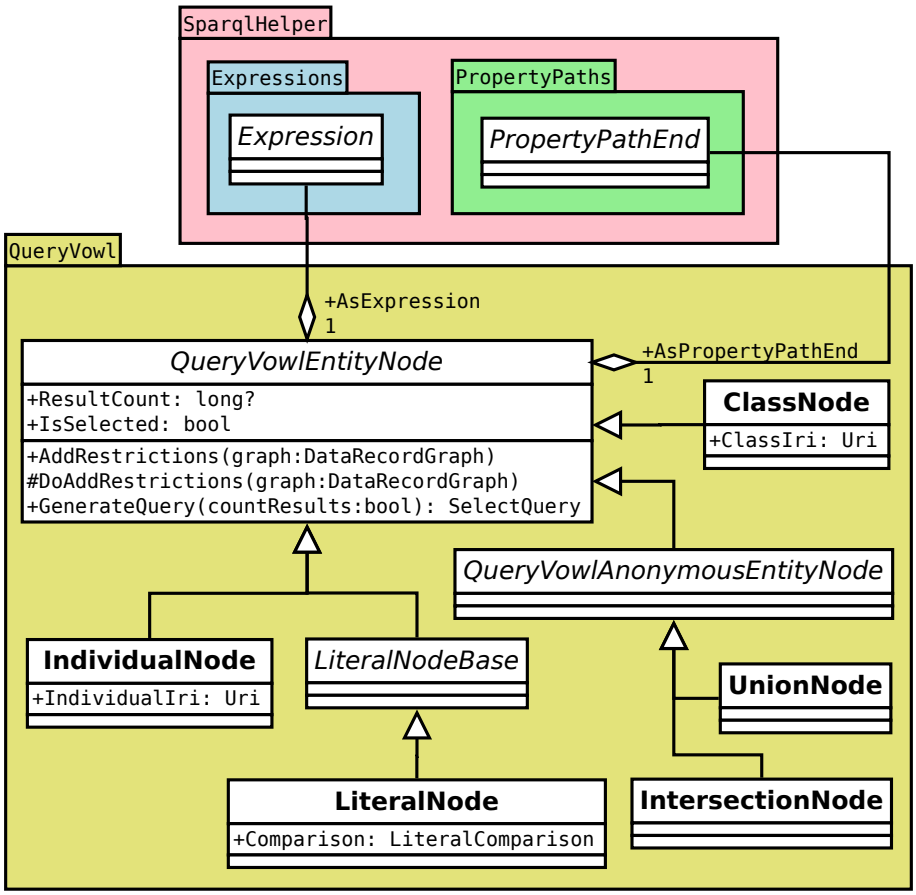


Abbildung 3.35: Die Hierarchie der Knotenklassen in der prototypischen C#-QueryVOWL-Implementierung.

operation keine Rückwirkungen auf die Klasse an sich haben darf: Die Ergebnismenge eines Klassenknotens, der auf Individuen einer Klasse *Fisch* restringiert ist, sollte sämtliche in einer Datensammlung aufgelisteten Fische enthalten, und die eines Knotens, der auf Individuen der Klasse *Hund* eingeschränkt ist, sollte alle Hunde enthalten. Dennoch können diese zwei Knoten durch einen Schnittmengenknoten verbunden sein, dessen Ergebnismenge in den meisten Datensammlungen leer sein wird.

Die Ergebnismengen der Klassenknoten und der Knoten, welche anonyme, auf Mengenoperationen basierende Klassen darstellen, müssen entkoppelt werden. Dazu müssen die Einschränkungen der Vereinigungsmengenknoten von den in die Vereinigungsmenge miteinbezogenen Knoten für eine SPARQL-Variable, welche die anonyme Vereinigungsmengenklasse eindeutig repräsentiert, dupliziert werden. Die Einschränkungen von Schnittmengenknoten müssen sogar durch Kopien der Subgraphen ausgedrückt werden, die eigentlich mit den in die Mengenoperation einbezogenen Knoten

verbunden sind. Diese Subgraphkopien werden dann wiederum mit der Variable verknüpft, welche die Schnittmengenklasse eindeutig repräsentiert. Algorithmus 3.4 zeigt das Vorgehen für Vereinigungsknoten (siehe Abschnitt 3.2.4.7), Algorithmus 3.5 das für Schnittmengenknoten (siehe Abschnitt 3.2.4.8).

Die Desktop-Implementierung belegt die Umsetzbarkeit aller definierten Sprachelemente von QueryVOWL. Da sie nicht auf Benutzerfreundlichkeit ausgelegt ist, sind die meisten Einstellungsmöglichkeiten nur über Tastenkombinationen erreichbar. Ebenso besteht keine Möglichkeit, direkt im Programm die Ergebnisliste für den markierten Knoten abzurufen. Die vom SPARQL-Endpunkt erfragte Ergebnisanzahl wird jedoch in der Visualisierung angezeigt und die gesendete Anfrage kann in Debug-Ausgaben begutachtet werden. Wie in zuvor genannten Implementierungen hängt die Berechnungsgeschwindigkeit der Ergebnisse stark vom Server ab. Bei diesem Prototyp fiel auf, dass insbesondere das Einbeziehen von Disjunktivitätskanten (siehe Abschnitt 3.2.4.6) die Ergebnisfindung durch den Triple Store Apache Jena Fuseki stark verlangsamt.

3.2.9 Fazit

Mit QueryVOWL steht eine objektgraph-basierte Anfragesprache zur Verfügung. Im Gegensatz zu vielen ähnlichen existierenden Ansätzen ist sie an eine bestehende Ontologievisualisierung angelehnt. Dadurch wird in der Summe der Lernaufwand für Benutzer beider Notationen reduziert. Gleichermaßen bietet sich die gemeinsame Verwendung der beiden Konzepte an, beispielsweise durch die Auswahl von QueryVOWL-Anfrageelementen aus einer VOWL-Ontologievisualisierung. Die Definition der Notation anhand von SPARQL-Anfragebestandteilen erlaubt eine themenunabhängige Verwendung von QueryVOWL für beliebige standardkonforme SPARQL-Endpunkte.

Require: x sei die eindeutige Variable, die den Knoten repräsentiert

```
1:  function doAddRestrictions(graph, node)
2:      connected := in die Vereinigungsmenge miteinbezogene Knoten
3:      if |connected| = 1 then
4:          n_0 := einziges Element aus connected
5:          alle Einschränkungen von n_0 in graph zu x hinzufügen
6:      else if |connected| > 1 then
7:          G := ∅
8:          for all n ∈ connected do
9:              g := mit n verbundener Subgraph; n durch x repräsentieren
10:             G := G ∪ g
11:         end for
12:         SPARQL-UNION aus allen Graphen in G zu graph hinzufügen
13:     end if
14: end function
```

Algorithmus 3.4: Einschränkungen eines Vereinigungsmengenknotens aus QueryVOWL.

Require: x sei die eindeutige Variable, die den Knoten repräsentiert

```
1:  function doAddRestrictions(graph, node)
2:      connected := in die Vereinigungsmenge miteinbezogene Knoten
3:      if |connected| = 1 then
4:          n_0 := einziges Element aus connected
5:          alle Einschränkungen von n_0 in graph zu x hinzufügen
6:      else if |connected| > 1 then
7:          for all n ∈ connected do
8:              g' := Kopie des mit n verbundenen Subgraphen; für jeden
9:                    Graphbestandteil γ wird ein Element γ' erzeugt
10:             jegliche Vorkommen von n' in g' durch x ersetzen
11:         end for
12:     end if
13: end function
```

Algorithmus 3.5: Einschränkungen eines Schnittmengenknotens aus QueryVOWL.

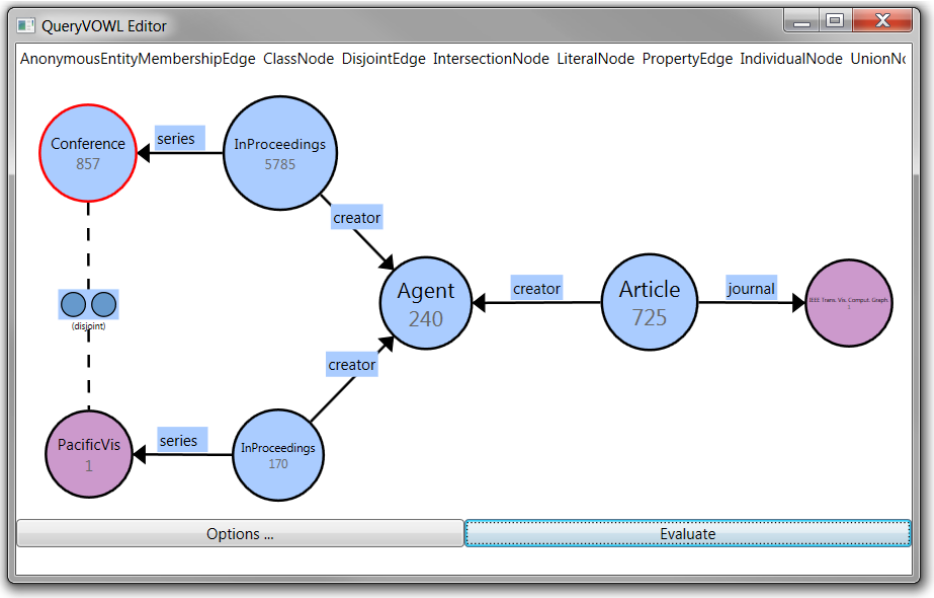


Abbildung 3.36: Der Desktop-Prototyp zum Konzept QueryVOWL. Die dargestellte Anfrage ermittelt eine Liste von Konferenzen, auf denen auch Autoren von Veröffentlichungen in der Fachzeitschrift TVCG und auf der Konferenz PacificVis publiziert haben. Die entsprechende SPARQL-Anfrage ist in Quelltextbeispiel 3.2 zu sehen.

```
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
PREFIX journal: <http://dblp.l3s.de/d2r/resource/journals/>
PREFIX conf: <http://dblp.l3s.de/d2r/resource/conferences/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?conference
WHERE {
  ?conference a swrc:Conference .
  ?author a foaf:Agent .
  ?tvcgDoc a swrc:Article ;
           dc:creator ?author ;
           swrc:journal journal:tvcg .
  ?apvisDoc a swrc:InProceedings ;
            dc:creator ?author ;
            swrc:series conf:apvis .
  ?otherDoc a swrc:InProceedings ;
            dc:creator ?author ;
            swrc:series ?conference .
  FILTER(?conference != conf:apvis) .
}
```

Quelltextbeispiel 3.2: Die SPARQL-Anfrage, die in Abbildung 3.36 mit der QueryVOWL-Notation gezeigt wird.

3.3 Magnetic Querying

Das Konzept Dust & Magnet diente dazu, Cluster ähnlicher Elemente in Datensammlungen zu erkennen [SMS+05] (siehe auch Seite 32 im Abschnitt 2.3.4). Während die einzelnen Datenelemente als „Staubpartikel“ zwischen Magneten platziert wurden, erlaubte die Visualisierung keine genaueren Aussagen dazu, welches Element von welchen Magneten genau angezogen wird. Zudem wurde von rein skalaren Attributen ausgegangen; eine Objektstruktur war nicht vorgesehen.

Aus diesem Grund wurde Dust & Magnet im Rahmen dieses Promotionsvorhabens zum Konzept Magnetic Querying[5] weiterentwickelt. Dazu wurden die folgenden Erweiterungen in das Konzept eingebracht:

- Verknüpfung von Magneten zur Auswertung verschachtelter Objekte in Datenelementen.
- Visuelle Markierungen an den „Staubpartikeln“ zur Anzeige, durch welche Magneten momentan Anziehungskräfte ausgeübt werden.
- Verknüpfung von Magneten zur Definition von Anziehungskräften, die sich aus Kombinationen von Kriterien ergeben.

Die Erweiterungen sollen die Flexibilität des Ansatzes erhöhen. Ebenso soll die Unterstützung für die Zuordnung zu Clustern basierend auf komplexen Kombinationen von Kriterien verbessert werden.

Wie in verwandten Arbeiten werden die Magneten kreisförmig dargestellt [SF08]. Jeder Magnet erhält eine individuelle zufällig gewählte Farbe[6]. Die Filter- beziehungsweise Anziehungsfunktion des Magneten wird durch einen kurzen Text knapp umrissen. Einige solche Magneten sind beispielhaft in Abbildung 3.37 dargestellt.

3.3.1 Abbildung auf Objektgraphen

Datenelemente können komplexe Objekte sein, die von unterschiedlichen Magneten basierend auf verschiedenen Attributen angezogen werden. In dieser Situation kann im Konzept Dust & Magnet nicht erreicht werden, dass mehrere Magneten ihre Anziehungskraft jeweils nur entfalten, wenn

[5] Der Name Magnetic Querying wurde in Anlehnung an den Titel der Arbeit, auf der dieses Konzept aufbaut – Dust & Magnet – gewählt. Während dies Benutzer mit physikalischen Vorkenntnissen verwirren könnte, schienen in der Benutzerstudie der ursprünglichen Arbeit keine Unklarheiten bezüglich der Magnetmetapher aufzutreten. Daher wurde die Magnetmetapher beibehalten.

[6] In konkreten Implementierungen kann diese Farbe auch anwendungsfall- oder benutzerabhängig gewählt werden. So kann die Visualisierung beispielsweise in ein vorhandenes Farbschema eingefügt werden. Ebenso können Unterscheidungsschwierigkeiten bei Farbfehlsichtigkeit vermieden werden.

Abbildung 3.37: Die Magneten in MAGNETIC QUERYING haben zufällig gewählte Farben und umreißen über einen kurzen Text knapp die Filter- beziehungsweise Anziehungsfunktion. Diese Abbildung zeigt Magneten zum Objektvergleich, zur Zählung von Eigenschaftswerten, zur Substring-Suche und zur Auswertung des Editierabstands nach dem Levenshtein-Algorithmus zwischen einem Eigenschaftswert und einem vorgegebenen Text.

sie sich auf dasselbe in einem Datenelement verschachtelte Objekt beziehen. Beispielsweise könnten für eine Menge von Dokumenten zwei Magneten jeweils einen bestimmten Vor- und einen bestimmten Nachnamen eines Autors fordern. Mit dem ursprünglichen Modell kann nicht sichergestellt werden, dass dieser Vor- und dieser Nachname beim *selben* Autor gefunden werden muss.

Formal ausgedrückt bedeutet dies: Sei P die Menge aller für Datenelemente verfügbarer Eigenschaften. Seien

$$E = \{e_1, \ldots, e_m\}$$
$$\mathrm{p}_x : E \to \mathcal{P}(E) \qquad \text{mit } x \in P$$

die Menge aller Datenelemente E und eine Funktion p_x, die zu einem Datenelement alle Werte der Eigenschaft x (weitere Datenelemente) zurückgibt, in der Definition als Potenzmenge von E, also $\mathcal{P}(E)$ ausgedrückt. Seien außerdem

$$M = \{m_1, \ldots, m_n\}$$
$$\mathrm{data} : M \times E \to \mathcal{P}(E)$$

die Menge M aller Magneten und eine Funktion data, die für einen Magneten die Menge der Datenelemente ermittelt, die für die Berechnung der Anziehungskraft des Magneten auf ein bestimmtes Datenelement herangezogen werden. Hat ein Datenelement e_1 für eine Eigenschaft a nun mehrere Werte e_2 und e_3, also

$$\{e_2, e_3\} \subseteq \mathrm{p}_a(e_1)$$

und ziehen zwei Magneten m_1 und m_2 das Datenelement e_1 basierend auf Werten von Eigenschaften b und c dieser Eigenschaftswerte e_2 und e_3 (also verschachtelter Eigenschaftswerte) an, das heißt

$$\mathrm{data}(m_1, e_1) = \mathrm{p}_b(e_2)$$
$$\mathrm{data}(m_2, e_1) = \mathrm{p}_c(e_3)$$

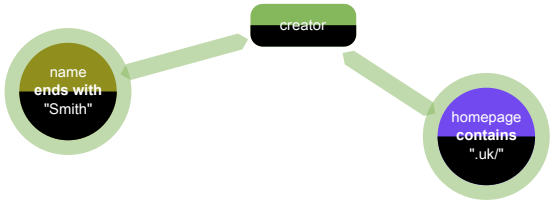


Abbildung 3.38: Hierarchieknoten können mit Magneten verknüpft werden, damit sich die Anziehungskraft der Magneten basierend auf Eigenschaftswerten desselben verschachtelten Datenelements berechnet. Im gezeigten Beispiel wird erreicht, dass nur eine Anziehungskraft besteht, wenn derselbe Autor einer Publikation (Attribut *creator*) den Namen „Smith“ hat und die Top-Level-Domain „.uk“ in seiner Website-Adresse auftaucht.

so kann mit dem herkömmlichen Konzept Dust & Magnet nicht sichergestellt werden, dass e_2 und e_3 für die betrachteten verschachtelten Eigenschaftswerte übereinstimmen. Dies kann wie beim oben genannten Beispiel mit dem Vor- und Nachnamen eines Autors wünschenswert sein, wenn unterschiedliche Magneten sich auf Eigenschaften desselben verschachtelten Objekts beziehen sollen.

Daher wurde für Magnetic Querying ein neuer *Hierarchieknoten* eingeführt. Er ist in Abbildung 3.38 dargestellt. Der Hierarchieknoten bildet gewissermaßen einen Teil der gesuchten Objektgraphstruktur nach. Dies ähnelt den Objektgraph-basierten Konzepten aus Abschnitt 2.3.1.1. Die Menge aller Hierarchieknoten H wird dabei von den Definitionen

$$\begin{aligned} \mathrm{c} &: M \cup H \to H \cup \{\varepsilon\} \\ \mathrm{prop} &: H \to P \\ \mathrm{v} &: H \times E \to E : (h, e_1) \mapsto \begin{cases} e_2 \in \mathrm{p}_{\mathrm{prop}(h)}(e_1) & \text{wenn } \mathrm{c}(h) = \varepsilon \\ e_2 \in \mathrm{p}_{\mathrm{prop}(h)}(\mathrm{v}(\mathrm{c}(h), e_1)) & \text{sonst} \end{cases} \end{aligned}$$

unterstützt.

Die Funktion c drückt eine Verknüpfung zu einem Hierarchieknoten aus, die von einem Magneten oder einem anderen Hierarchieknoten ausgehen kann. Solche Verknüpfungen dürfen keine Zyklen bilden, können aber transitiv aneinander angeschlossen sein. prop repräsentiert eine Eigenschaft, die einem Hierarchieknoten zugeordnet ist. Die Funktion v ermittelt für ein Datenelement e_1 ein von einem Hierarchieknoten h zurückgegebenes (nichtdeterministisch bestimmtes) Datenelement e_2, wobei eine etwaige Kette verknüpfter Hierarchieknoten transitiv miteinbezogen wird. Die Anziehungskraft der Magneten, die an h angeschlossen sind, wird letztendlich auf Grundlage des zurückgegebenen Datenelements e_2 berechnet. So lange sich

die Datenmenge und die Konfiguration des Hierarchieknotens nicht ändern, bleibt die Funktion v für jede Kombination aus Argumenten konstant.

Werden nun mehrere Magneten m_1 und m_2 mit demselben Hierarchieknoten verbunden, dem die Eigenschaft a zugeordnet ist

$$\begin{aligned} \mathrm{c}(m_1) &= \mathrm{c}(m_2) \\ \mathrm{prop}(\mathrm{c}(m_1)) &= a \end{aligned}$$

so kann die data-Funktion auf Grundlage dieses Hierarchieknotens definiert werden als

$$\begin{aligned} \mathrm{data}(m_1, e_1) &= \mathrm{p}_b(\mathrm{v}(\mathrm{c}(m_1), e_1)) \\ \mathrm{data}(m_2, e_1) &= \mathrm{p}_c(\mathrm{v}(\mathrm{c}(m_2), e_1)) \end{aligned}$$

Auf diese Weise wird nun garantiert, dass m_1 und m_2 beide von demselben Element, das über die Eigenschaft a mit e_1 und e_2 verknüpft ist, ausgehen.

3.3.2 Erweiterung der ursprünglichen Visualisierung

Bei Einsatz mehrerer Magneten kann es mitunter problematisch sein, nachzuvollziehen, welche Magneten welche Anziehungskraft auf die „Staubpartikel" ausüben. Dies wurde bereits für das Konzept Magnet Mail erkannt. Dort wird die Anziehung durch den selektierten Magneten mittels einer farblichen Hervorhebung aller entsprechenden Partikel visualisiert [CL09]. Die Stärke der Anziehung wird durch eine unterschiedlich starke Einfärbung ausgedrückt.

Ohne Selektion eines Magneten ist es dennoch auch in Magnet Mail nicht möglich, die Anziehungskraft eines Magneten auf einen Blick zu erkennen. Insbesondere kann nicht direkt abgelesen werden, ob ein Partikel von mehreren Magneten und unterschiedlich stark angezogen wird. Daher wurde die Darstellung dieser Partikel für Magnetic Querying erweitert. Vom Zentrum jedes Partikels aus wird wie in PhotoMagnets [CRB10] durch kurze Linien angedeutet, aus welchen Richtungen der Partikel angezogen wird (Abbildung 3.39). Die Länge der Linie drückt die Anziehungskraft aus, während die Farbe der Farbe des anziehenden Magneten entspricht. Abbildung 3.40 zeigt die erweiterten Staubpartikel im praktischen Einsatz.

3.3.3 Kombination von Anziehungskräften

Um die Anziehungskraft mehrerer Magneten mit unterschiedlichen Anziehungskriterien in einem Punkt zu konzentrieren, ohne die Magneten übereinanderschieben zu müssen, wurden neue Verknüpfungsmagneten definiert (Abbildung 3.41). Eine beliebige Anzahl anderer Knoten kann mit diesen

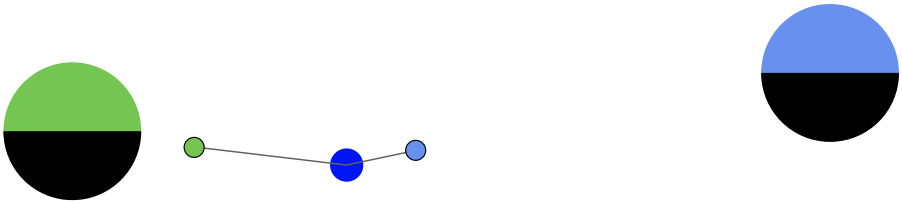


Abbildung 3.39: Die „Staubpartikel“ (Mitte) im Ansatz Magnetic Querying zeigen durch kurze Linien an, von welchen Magneten sie jeweils wie stark angezogen werden. Richtung der Linien und Farbe der Endpunkte weisen auf den Magneten hin. Die Länge der Linien zeigt die Stärke der Anziehungskraft an.

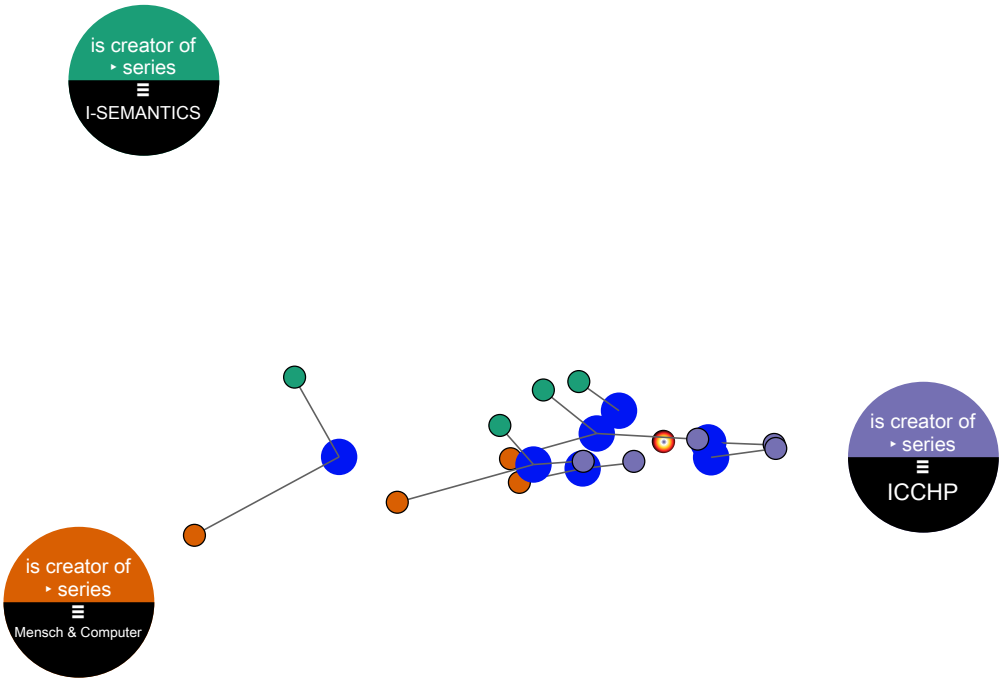


Abbildung 3.40: Auf Grundlage von Faceted DBLP werden die Koautoren des Autors (dieses Kriterium ist außerhalb der Visualisierung konfiguriert) als Staubpartikel abhängig von der Anzahl ihrer Veröffentlichungen auf den Konferenzen *I-SEMANTICS*, *Mensch & Computer* sowie *ICCHP* platziert. Dabei werden maximal 10 Publikationen pro Konferenz als Maximum angenommen. Anhand der relativ langen Richtungslinien ist beispielsweise erkennbar, dass zwei Personen vergleichsweise viele Veröffentlichungen auf der Konferenz M&C getätigt haben – in einer interaktiven Umsetzung von Magnetic Querying wäre per Markierung erkennbar, dass es sich dabei um Steffen Lohmann und Thomas Schlegel handelt.

(a) Konjunktionsmagnet **(b)** Disjunktionsmagnet

Abbildung 3.41: Verknüpfungsmagneten in Magnetic Querying, welche die Anziehungskraft einzelner Magneten auf verschiedene Weise kombinieren.

Verknüpfungsmagneten verbunden werden. Die Verknüpfungsmagneten berechnen dann eine kombinierte Anziehungskraft aus den einzelnen Anziehungskräften der verbundenen Magneten.

Um unterschiedliche Kombinationen aus Anziehungskräften zuzulassen, können verschiedene Arten von Verknüpfungsmagneten vorgesehen werden:

Konjunktion Konjunktionsmagneten (Abbildung 3.41a) ahmen eine Konjunktion aus den Filterkriterien nach. Die volle Anziehungskraft wird nur dann erreicht, wenn alle verbundenen Magneten die maximale Anziehungskraft ausüben. Ist diese Bedingung nicht erfüllt, erlischt die Anziehungskraft allerdings nicht vollkommen. Stattdessen wird die Summe aus den Anziehungskräften der verbundenen Magneten addiert. Diese Summe wird durch die Anzahl der verbundenen Magneten dividiert, um die anteilige Anziehungskraft zu berechnen.

Disjunktion Im Gegensatz zu Konjunktionsmagneten entfalten Disjunktionsmagneten (Abbildung 3.41b) ihre vollständige Anziehungskraft bereits, wenn mindestens einer der verbundenen Magneten Elemente mit voller Stärke anzieht. Dazu wird unter allen verbundenen Magneten derjenige ermittelt, der die stärkste Anziehungskraft auf ein gegebenes Datenelement ausübt. Diese Anziehungskraft wird dann für den Disjunktionsmagneten insgesamt angenommen.

An solche Verknüpfungsmagneten können beliebige Magneten angeschlossen werden, auch weitere Verknüpfungsmagneten. Lediglich zyklische Verbindungen sind ausgeschlossen. Auf diese Weise ergibt sich eine Struktur, die einem Ausdrucksbaum gleicht. Für die visuelle Darstellung des Sachverhalts, dass Magneten in eine Konjunktion oder eine Disjunktion einbezogen sind, wurde eine Kantenform gewählt, die an UML-Kompositionskanten angelehnt ist [Obj15, S. 181ff.].

Die kombinierte Anziehungskraft mehrerer Magneten, die mit einem Verknüpfungsmagneten verbunden sind, geht von der Position des Verknüpfungsmagneten aus. Neben ihrer eigenen direkten Anziehungskraft können Magneten also auch eine indirekte Wirkung über Verknüpfungsmagneten haben. Für die Unterscheidung, ob die verbundenen Magneten auch für sich genommen eine zusätzliche Anziehungskraft ausüben, lassen sich die in Tabelle 3.5 aufgezeigten Fälle identifizieren.

Tabelle 3.5: Unterscheidbare Fälle für die Kombination aus direkter (d) und indirekter (i) Anziehung durch Magneten in Magnetic Querying.

		nicht verbunden			
		d, i	d	i	—
verbunden	d, i	A	B	C	D
	d	E	F	G	H
	i	I	J	K	L
	—	M	N	O	P

Für Magneten, die nicht mit Verknüpfungsmagneten verbunden sind, sind die Spalten d, i und d sowie i und — in Tabelle 3.5 jeweils gleichbedeutend, da die indirekte Anziehungskraft nicht zum Tragen kommt. Für die Fälle E bis H, in denen die Verbindung zu einem Verknüpfungsmagneten keine Auswirkung hat, wird keine praktische Verwendung gesehen. Gleiches gilt für die Fälle M und N, in denen die Verbindung zu einem Verknüpfungsmagneten den Magneten komplett wirkungslos macht. Ebenso wurde auch das Verhalten in den Fällen C und D als zu speziell eingeschätzt. In diesen Fällen übt der unverbundene Magnet keine eigene Anziehungskraft aus, entfaltet diese aber, wenn er an einen Verknüpfungsmagneten angeschlossen wird.

Die übrigen Fälle sind zum Teil gleichbedeutend und lassen sich zu vier Zuständen zusammenfassen. Diese Zustände sind in Tabelle 3.6 beschrieben. Jedem Zustand wurde ein Symbol zugeordnet, welches beim Magneten dargestellt wird. In Abbildung 3.42 wird beispielhaft ein Konjunktionsmagnet in Aktion gezeigt.

3.3.4 Implementierung

Magnetic Querying wurde mit C# und WPF als Anwendung für das Microsoft .NET Framework [Mic15] prototypisch implementiert. Ein Screenshot der Anwendung ist in Abbildung 3.43 zu sehen.

Die prototypische Anwendung ruft Daten von einem SPARQL-Endpunkt ab, der in den Optionen konfiguriert werden kann. Dabei werden alle Informationen angefordert, die für die aktuelle Grundauswahl an

Tabelle 3.6: Die Zustände der Magneten für Verknüpfungen mit den entsprechenden Zustandssymbolen. Zu jedem Zustand sind auch die Fälle aus Tabelle 3.5 notiert.

Symbol	Erläuterung	Fälle
	Partikel werden vom Magneten angezogen. Zusätzlich wird die Anziehungskraft auch miteinbezogen, falls der Magnet an einen Verknüpfungsmagneten angeschlossen ist.	A/B
	Der Magnet übt seine Anziehungskraft auf Partikel nur aus, falls er nicht an einen Verknüpfungsmagneten angeschlossen ist. Andernfalls wird die Anziehungskraft des Magneten nur für die Gesamtanziehung des Verknüpfungsmagneten berücksichtigt.	I/J
	Der Magnet übt selber überhaupt keine Anziehungskraft auf Partikel aus. Ist er an einen Verknüpfungsmagneten angeschlossen, wird die Anziehungskraft allerdings für die Gesamtanziehung des Verknüpfungsmagneten berücksichtigt.	K/L
	Der Magnet übt keine Anziehungskraft aus und wird auch durch Verknüpfungsmagneten nicht berücksichtigt.	O/P

Datenelementen und die aktuellen Magneten notwendig sind. Die eigentliche Berechnung der Anziehungskräfte erfolgt lokal, wodurch in diese Berechnung auch Funktionen wie eine Editierdistanz zwischen Werten miteinbezogen werden können, die vom SPARQL-Endpunkt nicht unterstützt werden.

Bei der Änderung von Einstellungen werden die Daten erneut vom Server abgerufen. Die Anzeige inklusive der Position der Magneten wird in festen Zeitintervallen (standardmäßig alle 5 Sekunden) aktualisiert. Eine kontinuierliche Aktualisierung ist ohne gesonderte Optimierungen der Darstellung nicht möglich, da die Staubpartikel und ihre Einzelteile als WPF-Geometrieobjekte in der Benutzerschnittstelle existieren und bereits beim Zeichnen von 500 Staubpartikeln eine kleine (unter einer Sekunde), aber erkennbare Verzögerung auftritt.

Die Implementierung diente als Machbarkeitsnachweis für die konzeptuellen Erweiterungen wie Hierarchieknoten, die Kombination von Anziehungskräften und die Darstellung der Anziehungskraft auf jeden Partikel (Abbildung 3.44). Daher wurden andere Features, wie die Animation aus dem ursprünglichen Konzept [SMS+05], in diesem Prototyp nicht umgesetzt.

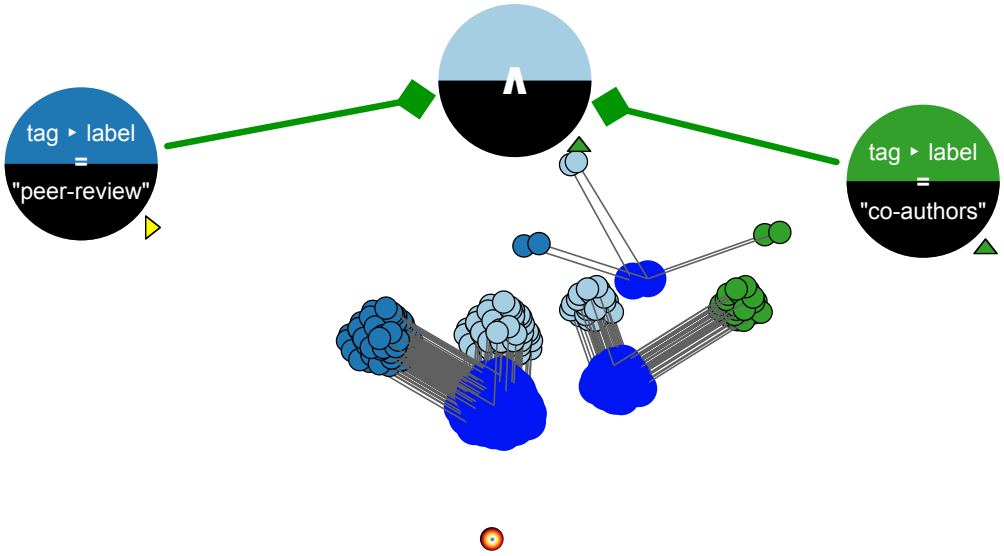


Abbildung 3.42: Einsatz eines Konjunktionsmagneten zur Anziehung von Fragen aus Academia Stack Exchange nach thematischen Tags. Es wird erkennbar, dass sich nur zwei Fragen mit dem Thema *Peer-Review* (Tag „peer-review“) in Bezug auf Koautoren (Tag „co-authors“) befassen („Who can write a ‘Review Article’?“ und „How to offer a reviewer to be co-author?“). Die theoretische Anziehungskraft durch die einzelnen Magneten wird durch die Richtungsanzeiger an den Partikeln verdeutlicht. Der linke Magnet hat wegen seines aktuellen Zustands (kleines gelbes Dreieck, vergleiche Tabelle 3.6) jedoch keine direkte Auswirkung auf die Position von Partikeln.

3.3.5 Fazit

Magnetic Querying erweitert das Konzept Dust & Magnet [SMS+05] sowohl visuell als auch funktional. Die primäre visuelle Erweiterung besteht in der Anzeige der auf jeden Staubpartikel einwirkenden Anziehungskräfte.

Funktional ist Magnetic Querying durch die in Abbildung 3.44 verwendeten neuen Magnet-/Knotentypen mächtiger als das ursprüngliche Konzept. Konjunktions- und Disjunktionsmagneten erlauben nun die Kombination der Anziehungskräfte mehrerer Magneten. Hierarchieknoten erweitern das Konzept dazu, dass es jetzt auch für die Analyse von Objektgraphen geeignet ist, da man mit mehreren Magneten auf dasselbe verschachtelte Datenelement eines gegebenen Staubpartikels Bezug nehmen kann.

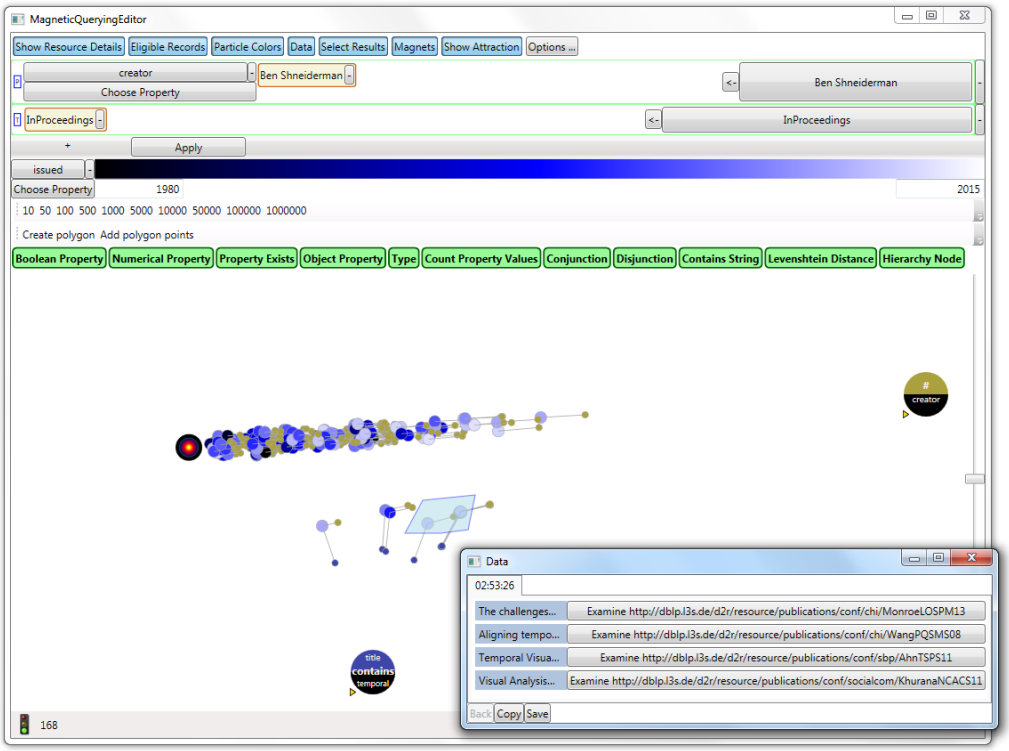


Abbildung 3.43: In diesem Screenshot der Implementierung von MAGNETIC QUERYING werden die 168 in FACETED DBLP [DB08] gelisteten Konferenzpublikationen von Ben Shneiderman als Staubpartikel dargestellt und von zwei Magneten angezogen. Der rechte Magnet zieht Publikationen um so stärker an, je mehr Autoren an ihnen mitgewirkt haben. Der untere Magnet übt eine Anziehungskraft auf alle Publikationen aus, deren Titel das Wort „temporal“ enthält. Je dunkler der Blauton der Staubpartikel ist, desto älter sind die jeweiligen Publikationen. Vier (teilweise übereinanderliegende) Staubpartikel sind in einem Selektionspolygon eingerahmt, weshalb die Titel und IRIs der entsprechenden Publikationen in der Ergebnisliste im *Data*-Fenster angezeigt werden.

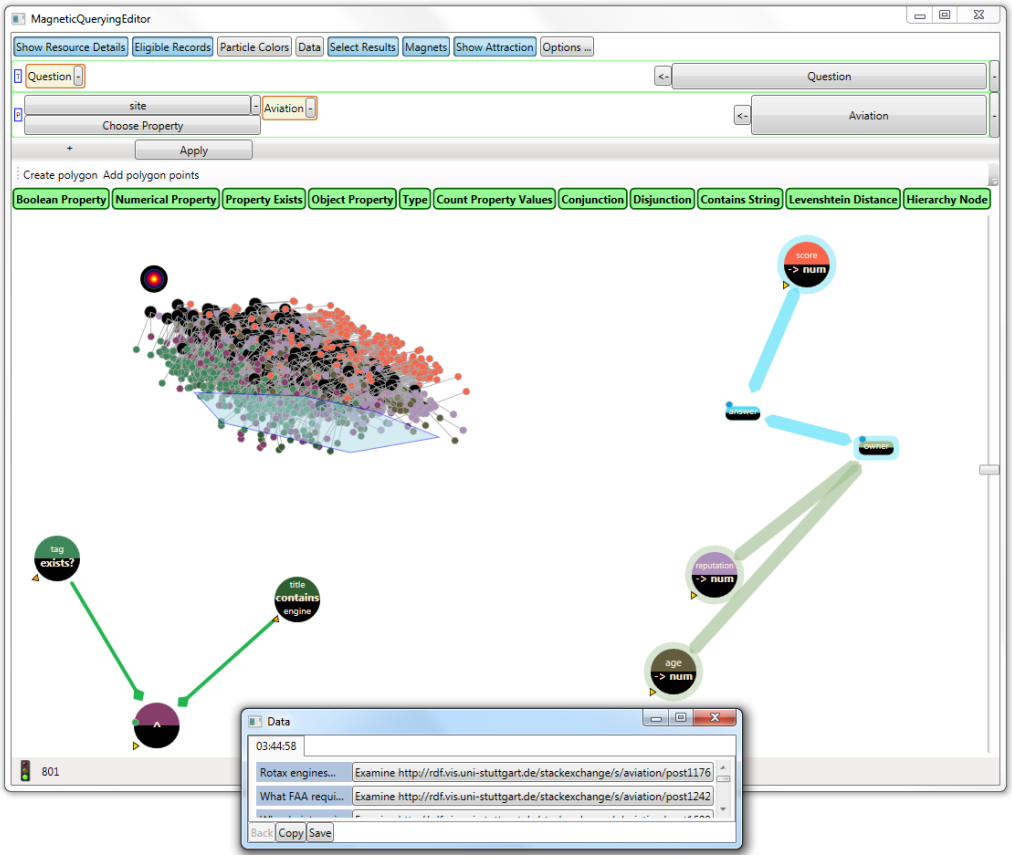


Abbildung 3.44: Dieser Screenshot der Implementierung von Magnetic Querying zeigt als Staubpartikel 801 Fragen von der Frage-und-Antwort-Seite Aviation Stack Exchange. Auf der rechten Seite wird über Hierarchieknoten bewirkt, dass die Magneten Fragen um so stärker anziehen, wenn sie (mindestens) eine hoch bewertete Antwort von einem Benutzer haben, der selber über einen hohen Punktestand verfügt und für sich ein hohes Alter angegeben hat. In der Ergbnisliste sind einige Fragen zu sehen, die von dem Konjunktionsmagneten unten links stark angezogen werden, da sie mit mindestens einem Tag versehen sind und das Wort „engine“ in ihrem Titel haben.

3.4 Filter Dials

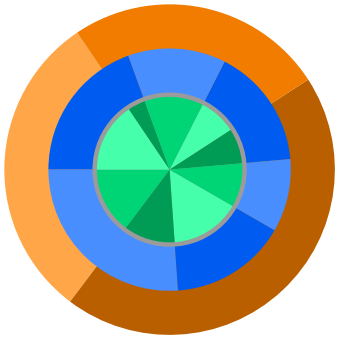
Das Konzept Filter Dials wurde mit dem Ziel entwickelt, schnell einen kompakten Überblick über die Datenmengen zu bekommen, die aus der Kombination diverser Filterkriterien resultieren. Der Fokus liegt dabei nicht darauf, eine vollständige Abdeckung der gesamten Datenmenge [Huo08] zu erhalten. Genausowenig wird angestrebt, eine eindeutige Einteilung von Datenelementen in Gruppen zu erreichen. Vielmehr soll es möglich sein, eine Reihe von Filterkriterien zu bestimmen, auf deren Grundlage dann jeweils mehrere Teilmengen gebildet werden. Diese Teilmengen werden miteinander geschnitten. Dadurch kann man ermitteln, ob die so erreichte Kombination von Filterkriterien zu einer leeren Datenmenge führt oder nicht. Auf diese Weise soll es Benutzern ermöglicht werden, sich einen Eindruck von der Zusammensetzung einer Datensammlung zu verschaffen.

> Die im Folgenden präsentierten Inhalte wurden zum Teil bereits anderweitig veröffentlicht. Insbesondere wird auf den im Folgenden aufgeführten Arbeiten, an denen der Autor maßgeblich mitwirkte, aufgebaut:
>
> HE14 F. Haag und T. Ertl. "Filter Dials – combine filter criteria, see how much data is available". In: *Proc. Advanced Visual Interfaces (AVI '14)*. ACM, 2014, S. 369–370. DOI: 10.1145/2598153.2600038
>
> Zusätzlich wird inhaltlich auf die folgenden studentischen Arbeiten, an deren Betreuung der Autor beteiligt war, zurückgegriffen:
>
> Raj15 J. Rajamani. "Filter (Dials | ...)" Betreuer: Florian Haag. Masterarbeit. Universität Stuttgart, 2015

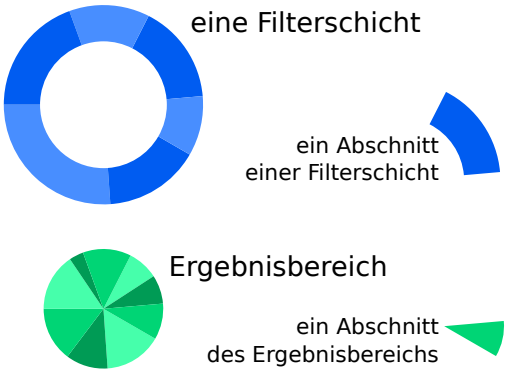
3.4.1 Einzelne Filter Dials

Ein einzelnes Filter Dial besteht aus einer beliebigen Anzahl von Filterschichten und einem Ergebnisbereich (Abbildung 3.45). Für die Filter Dials wurde ein radiales Layout gewählt. Einerseits erlaubt dies eine unbegrenzte Verschiebung der Filterschichten zueinander durch Rotation. Andererseits verdeutlicht es die Filtermetapher. Die Filterschichten umschließen den Ergebnisbereich. Somit lässt sich der Bereich außerhalb eines Filter Dials als die vollständige, ungefilterte Datenmenge begreifen. Nur Teilmengen davon können durch die Filterschichten hindurch in den Ergebnisbereich vordringen.

Jede Filterschicht betrachtet einen bestimmten Aspekt der Datenelemente, beispielsweise ein bestimmtes Attribut. Eine Filterschicht enthält einen oder mehrere Abschnitte. Jeder dieser Abschnitte repräsentiert einen für den betrachteten Aspekt akzeptierten Wert oder Wertebereich. Werden Daten abhängig von ihrem Erzeugungsjahr gefiltert, könnten eine entspre-



(a) Ein komplettes Filter Dial (normalerweise mit Beschriftungen).



(b) Bestandteile eines Filter Dials.

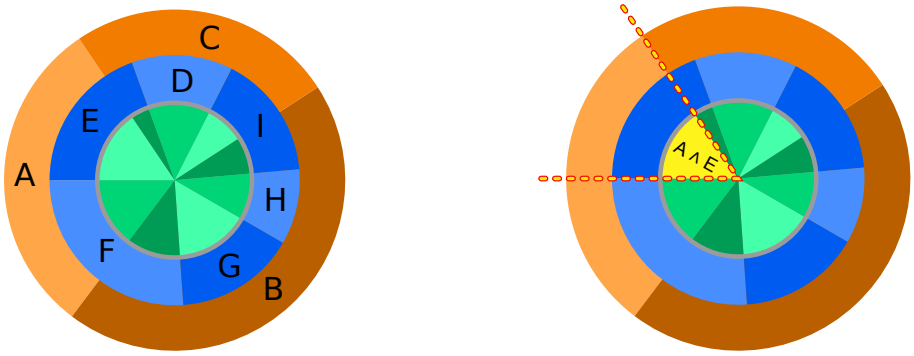
Abbildung 3.45: Aufbau eines Filter Dials.

chende Filterschicht beispielsweise in die Abschnitte *1979*, *1982–1986* und *1991* unterteilt werden.

Es besteht keine Notwendigkeit, dass die Abschnitte zusammen alle möglichen Werte abdecken. Ebenso besteht keine Einschränkung, die Wertebereiche von Abschnitten überschneidungsfrei zu halten. Vielmehr ist die Idee, dass für jeden vom Benutzer als interessant empfundenen Wert oder Wertebereich ein Abschnitt erzeugt wird.

Der Ergebnisbereich in der Mitte eines Filter Dials vermittelt einen ersten Eindruck von den gefilterten Ergebnismengen. Abhängig von den Abschnitten der Filterschichten ist auch er in mehrere Abschnitte unterteilt. Dazu werden die Grenzen zwischen sämtlichen Abschnitten in sämtlichen Filterschichten in einer gedachten Linie bis zum Mittelpunkt des Filter Dials verlängert. Auf diese Weise wird der kreisförmige Ergebnisbereich in mehrere Kreissektoren zerteilt. Diese stellen die Abschnitte des Ergebnisbereichs dar. Jeder dieser Ergebnisabschnitte entspricht der Datenmenge, die sich aus der Konjunktion aller Filterschichtabschnitte an der jeweiligen Stelle ergibt. Der in Abbildung 3.46b gelb hervorgehobene Abschnitt des Ergebnisbereichs zwischen den eingetragenen Schnittlinien entspricht somit beispielsweise der Konjunktion aus den Filterkriterien A und E aus Abbildung 3.46a.

Im konkreten Einsatz wird also für jede Filterschicht ein bestimmter Filter beziehungsweise ein bestimmtes Attribut ausgewählt. Dieses Attribut wird einmal für die gesamte Schicht angezeigt. Für jeden Abschnitt jeder Filterschicht wird ein Wert oder ein Wertebereich, der für das Attribut gefordert wird, gewählt und dargestellt. Ein konkretes Beispiel dazu wird in Abbildung 3.47 gezeigt.



(a) Ein FILTER DIAL, wobei die einzelnen Abschnitte der Filterschichten mit Buchstaben markiert sind. Da jeder Abschnitt eine Filterbedingung repräsentiert, stehen die Buchstaben in dieser Abbildung gleichzeitig für die Bedingungen.

(b) Das FILTER DIAL aus Abbildung 3.46a, in dem nun ein Abschnitt im Ergebnisbereich hervorgehoben ist. Gestrichelte Linien deuten an, wie die Größe des Ergebnisbereichsaschnitts ermittelt wurde. Der Abschnitt repräsentiert die Datenmenge, die sich aus der Konjunktion der Bedingungen A und E aus Abbildung 3.46a ergibt.

Abbildung 3.46: Unterteilung des Ergebnisbereichs in mehrere Abschnitte.

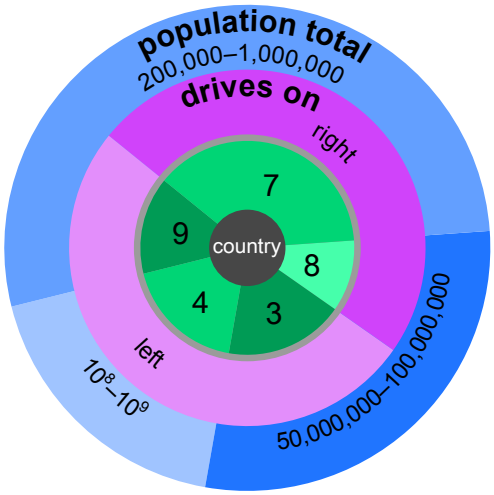


Abbildung 3.47: Dieses FILTER DIAL wird auf DBPEDIA [ABK+07] ausgeführt, um Länder mit bestimmten Eigenschaften auszuwählen. In diesem Fall wird nach der Einwohnerzahl und der Straßenseite, auf welcher gefahren wird, gefiltert. Dabei werden beispielsweise vier Länder („country“) mit Linksverkehr („drives on“ = „left“) gefunden, die zwischen einhundert Millionen und einer Milliarde Einwohner („population total“) haben (Pakistan, Indonesien, Japan und Bangladesch).

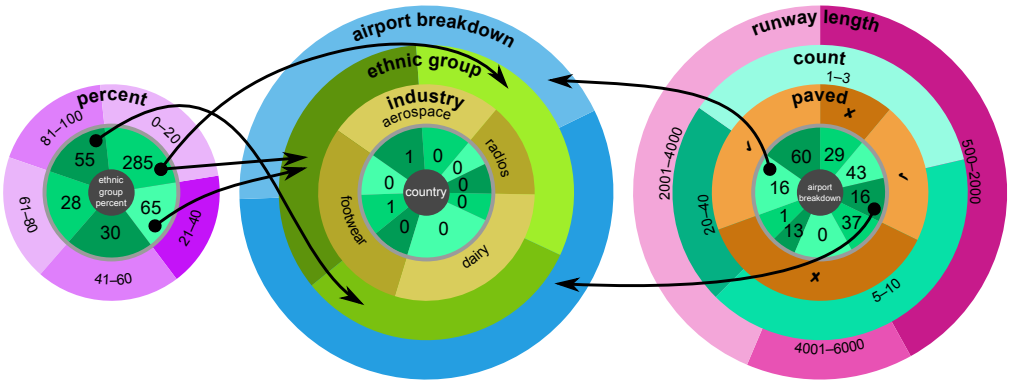


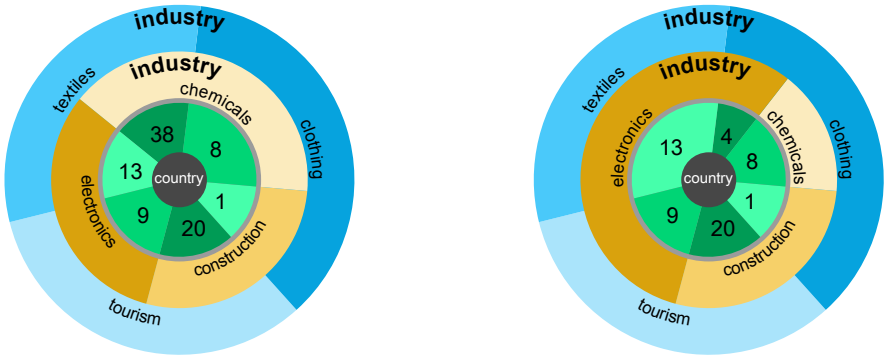
Abbildung 3.48: Mehrere Filter Dials lassen sich verketten. Dabei fungieren Ergebnismengen aus einem Filter Dial als Eingabe für Filterschichtabschnitte eines anderen Filter Dials. In dieser Anfrage werden (im mittleren Filter Dial) Länder ermittelt, in denen bestimmte Industriezweige existieren, die über Flughäfen mit bestimmten (im rechten Filter Dial festgelegten) Eigenschaften verfügen und in denen ethnische Gruppen mit bestimmten Anteilen an der Gesamtbevölkerung (im rechten Filter Dial ermittelt) beheimatet sind.

3.4.2 Verkettung mehrerer Filter Dials

Mehrere Filter Dials können unabhängig voneinander eingesetzt werden. So können nebeneinander unterschiedliche Daten aus derselben Datensammlung extrahiert werden. Sie können jedoch auch verbunden werden. Dies geschieht, indem eine oder mehrere Ergebnismengen aus einem Filter Dial als Eingabe für Filterschichtabschnitte eines anderen Filter Dials benutzt werden (Abbildung 3.48).

Abschnitte von Filterschichten ohne externen Dateneingang ermöglichen es, Attribute von Datenelementen mit konstanten, manuell aufgelisteten Werten zu vergleichen. Durch die Verknüpfung mehrerer Filter Dials wird erreicht, dass die Attribute mit Elementen aus einer anderen, ebenfalls über ein Filter Dial ermittelten Datenmenge abgeglichen werden. Dies entspricht einer *join*-Operation in relationalen Datenbanken.

Als Nebeneffekt dieser Verkettungsmöglichkeit findet in Anfragen, die aus mehreren verketteten Filter Dials bestehen, eine klare visuelle Unterteilung konzeptuell getrennter Teile der Anfrage statt. Dies könnte Benutzern helfen, einen besseren Überblick über die Anfrage zu erlangen. Ebenso kommen kollaborative Szenarien ähnlich der Vorgehensweise, die in Zusammenhang mit LARK [TIC09] vorgestellt wurden, in Betracht.



(a) Vor der Verschiebung: Unter anderem sind die Industriezweige *Chemikalien* („chemicals“) und *Textilien* („textiles“) kombiniert.

(b) Nach der Verschiebung: Die Industriezweige *Elektronik* („electronics“) und *Bekleidung* („clothing“) sind kombiniert.

Abbildung 3.49: Die Trennlinien zwischen Abschnitten einer Filterschicht können verschoben werden, um neue Kombinationen von Kriterien herbeizuführen.

3.4.3 Interaktion

Ein Filter Dial wird aufgebaut, indem Filterschichten und Filterschichtabschnitte hinzugefügt oder entfernt und konfiguriert werden. Davon abgesehen sieht das Konzept Filter Dials jedoch zwei Interaktionen vor, die direkt zur Exploration einer Datensammlung beitragen. Einerseits kann die Grenze zwischen zwei benachbarten Abschnitten einer Filterschicht verschoben werden (Abbildung 3.49), andererseits kann die Filterschicht als Ganzes rotiert werden (Abbildung 3.50). Beide Interaktionen sind Drag-und-Drop-basiert und eignen sich somit sowohl für Desktop-Systeme mit Zeigegeräten als auch für Touch-Geräte. Während eine dieser Verschiebungen ausgeführt wird, aktualisiert sich gemäß dem Paradigma der dynamischen Anfragen [Shn94] laufend der Ergebnisbereich, indem die Anzahl der jeweiligen Ergebniselemente angepasst und bei Bedarf neue Ergebnisbereiche erzeugt oder bestehende Ergebnisbereiche entfernt werden.

3.4.4 Einschränkungen

Das Konzept der Filter Dials verschafft einen Überblick über bestimmte Kombinationen von Filterkriterien. Eine Auswahl von Attributen und zulässigen Wertebereichen muss aber dennoch durch den Benutzer erfolgen. Ohne eine solche Auswahl helfen auch Filter Dials nicht dabei, einen Überblick zu erlangen.

Durch die Verschiebungen von Filterschichtabschnitten und ganzen Fil-

(a) Vor der Rotation: Unter anderem ist die *chemische* Industrie („chemicals“) mit der Ressource *Kupfer* („copper“) kombiniert, *Bergbau* („mining“) mit *Erdgas* („natural gas“) und *Lebensmittelverarbeitung* („food processing“) mit *Wasserkraft* („hydropower“).

(b) Nach der Rotation: Die *chemische* Industrie ist mit der Ressource *Wasserkraft* kombiniert, *Bergbau* mit *Kupfer* und *Lebensmittelverarbeitung* mit *Erdgas*.

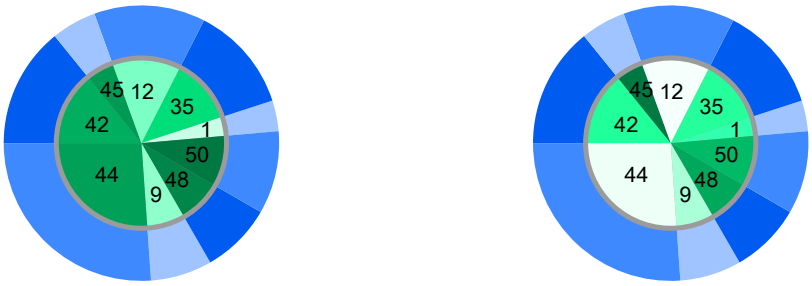
Abbildung 3.50: Filterschichten können als Ganzes rotiert werden, um rund um ein Filter Dial neue Kombinationen von Kriterien zu erzeugen.

terschichten (Abschnitt 3.4.3) können die Kombinationen der unterschiedlichen Filterkriterien variiert werden. Bei einer Drehung der gesamten Filterschicht verschieben sich allerdings alle Filterschichtabschnitte. Hat ein Benutzer also bereits eine Kombination aus Filterkriterien gefunden, die weiterverwendet werden soll, so verschwindet der entsprechende Abschnitt des Ergebnisbereichs beim Weiterdrehen der Filterschicht unter Umständen wieder.

Letztendlich ist der kreisförmige Aufbau der Filter Dials ebenfalls nicht notwendigerweise optimal. Er ist platzsparend und lässt eine Filtermetapher zu, nach der Elemente aus einer ungefilterten Datensammlung durch die Filterringe in den zentralen Ergebnisbereich vordringen. Allerdings kann die damit einhergehende Anordnung von Text auf einer Kreislinie einige Beschriftungen schwerer lesbar machen. Zudem steht für den Ergebnisbereich und insbesondere jeden einzelnen Abschnitt darin relativ wenig Platz zur Verfügung. So bleibt neben dem Umfang jeder Ergebnismenge kaum Platz für weitere Informationen.

3.4.5 Der Ergebnisbereich

Im ursprünglichen Entwurf und in den vorangegangenen Beispielen wurden die Abschnitte des Ergebnisbereichs ausschließlich dazu verwendet, um die Anzahl der Elemente in der Ergebnismenge darzustellen. Die Farben dienten



(a) Je größer die Ergebnismenge, desto dunkler wird der Farbton.

(b) Je mehr Ergebniselemente im selben Raum Platz finden müssen, desto dunkler wird der Farbton.

Abbildung 3.51: Alternative Farbcodierungen für den Ergebnisbereich

lediglich zur Unterscheidung der Abschnitte.

Aus Diskussionen bei der Vorstellung des Konzepts ergab sich auch eine Reihe weiterer Darstellungsmöglichkeiten für den Ergebnisbereich, die im Folgenden kurz erläutert werden.

3.4.5.1 Farbskala gemäß Ergebnisanzahl

Die Abschnitte des Ergebnisbereichs können abhängig vom Umfang der Ergebnismenge anhand einer diskreten oder kontinuierlichen Farbskala ausgefüllt werden (Abbildung 3.51a). Eher noch als die Zahlen (die natürlich trotzdem angezeigt werden können) würde dies einen schnellen Überblick über die unterschiedlichen Ergebnismengen geben. Ein Vorteil dabei wäre, dass die Farbskala nicht monoton sein müsste. Sie könnte beispielsweise zu beiden Enden hin zur selben Farbe tendieren. Ein – vom Awendungsfall abhängig zu wählender – bevorzugter mittlerer Bereich könnte dann in einer anderen Farbe hervorgehoben werden. Benachbarte Abschnitte mit ähnlicher Ergebnismengengröße hätten jedoch dieselbe Farbe und ließen sich so möglicherweise nicht ohne Weiteres unterscheiden.

3.4.5.2 Farbskala gemäß Ergebnisanzahl und Größe

Die Größe der Abschnitte im Ergebnisbereich ergibt sich allein aus der Platzierung der Grenzen zwischen Filterschichtabschnitten und hängt nicht mit der Ergebnisanzahl zusammen. Daher kam auch die Idee auf, über die Farbe das Verhältnis zwischen der Größe des Abschnitts und dem Umfang der Datenmenge auszudrücken (Abbildung 3.51b). Die Grenzen zwischen den Filterschichtabschnitten können dahingehend verschoben werden, dass alle Abschnitte im Ergebnisbereich ungefähr dieselbe Farbe haben. In dieser Situation kann man davon ausgehen, dass die relative Größe der Abschnitte

ungefähr mit der relativen Größe der Ergebnismengen übereinstimmt. Sind andererseits alle Abschnitte im Ergebnisbereich ungefähr gleich groß, so geben die Farben den ungefähren Umfang der Ergebnismenge entlang einer Skala an.

3.4.6 Implementierung

Vom Konzept Filter Dials existiert lediglich eine rudimentäre Implementierung. Sie unterstützt nur fünf Filtertypen und zeigt nur die ringförmige Geometrie der Filter Dials ohne Beschriftungen an. Dieser Prototyp wurde mit C# und WPF für das Microsoft .NET Framework erstellt und dient als einfacher Beleg der Umsetzbarkeit.

Filterschichten sowie Abschnitte darin können interaktiv hinzugefügt und entfernt werden. Die Trennlinien zwischen den Abschnitten lassen sich wie in Abbildung 3.49 gezeigt mit der Maus verschieben, während das Rotieren ganzer Filterschichten wie in Abbildung 3.50 nicht unterstützt ist. Der Ergebnisbereich wird bei den beschriebenen Änderungen automatisch in die entsprechende Anzahl von Abschnitten aufgeteilt. Auf Knopfdruck wird auch eine Auswertung über eine Anbindung an einen SPARQL-Endpunkt vorgenommen. Da innerhalb eines Filter Dials nur Konjunktionen zur Anwendung kommen, geschieht die Bestimmung des Umfangs der Ergebnismengen auf dem Server meist innerhalb weniger Sekunden. Im Fall von Disjunktionen durch die in Abschnitt 3.4.2 beschriebenen Verkettungen verlängert sich die Antwortzeit zum Teil, sodass für eine benutzungsbereite Implementierung zumindest ein Indikator für ausstehende Werte vorgesehen werden sollte.

3.4.7 Fazit

Mittels Filter Dials kann ein schneller Überblick über den Umfang unterschiedlicher Teilmengen einer Datensammlung gewonnen werden. Während sich bestehende Konzepte meist auf eine einzelne Kombination von Kriterien oder auf den Überblick über alle potenziell möglichen Kombinationen von Kriterien konzentrierten, erlaubt es das Konzept Filter Dials, gezielt bestimmte Kombinationen auszuwählen und auszuwerten. Durch die Verknüpfbarkeit mehrerer Filter Dials können dabei auch mehrstufige komplexe Anfragen erzeugt werden. Da die Visualisierung kompakt ist, könnte sie auch die Filterung auf kleinen Bildschirmen unterstützen.

4 Domänenspezifische Ansätze

In diesem Kapitel werden mehrere Ansätze für die Visualisierung von Suchanfragen beschrieben, die auf bestimmte Datentypen oder Anwendungsfälle ausgerichtet sind. Die speziellen Eigenschaften dieser Datentypen und Anwendungsfälle werden in die Visualisierungskonzepte miteinbezogen. So können die Suchanfragen auf intuitive Weise grafisch dargestellt werden.

4.1 Aspect Grid

Bei Aspect Grid handelt es sich um einen Ansatz, mit dessen Hilfe multivariate Datenelemente übersichtlich dargestellt und gefiltert werden können. Dabei ist auch vorgesehen, dass sich über die Quelldaten, aus denen strukturierte Informationen zu jedem Datenelement extrahiert wurden, eine Rückkopplungsschleife ergibt. Über diese kann die Suche verfeinert werden.

> Die im Folgenden präsentierten Inhalte wurden zum Teil bereits anderweitig veröffentlicht. Insbesondere wird auf den im Folgenden aufgeführten Arbeiten, an denen der Autor maßgeblich mitwirkte, aufgebaut:
>
> HHJ+14 F. Haag, Q. Han, M. John und T. Ertl. "Aspect Grid: a visualization for iteratively refining aspect-based queries on document collections". In: *INFORMATIK 2014: Big Data – Komplexität meistern*. Bd. 232. LNI. Gesellschaft für Informatik e.V. (GI), 2014, S. 655–660
>
> Zusätzlich wird inhaltlich auf die folgenden studentischen Arbeiten, an deren Betreuung der Autor beteiligt war, zurückgegriffen:
>
> Nau15 A. Naumov. "Facettiertes Suchinterface für die explorative Suche in gewichteten Kundenrezensionen". Betreuer: Florian Haag, Qi Han, Markus John. Bachelorarbeit. Universität Stuttgart, 2015

4.1.1 Das Konzept

Im Rahmen dieses Promotionsvorhabens wurde das Konzept Aspect Grid entworfen. Es handelt sich dabei um eine tabellenbasierte Darstellungsform multivariater Datenelemente mit interaktiver Filtermöglichkeit. Das Konzept beinhaltet eine aggregierte Übersicht über die Datenelemente. Sie basiert auf den aktuellen Filtereinstellungen und bezieht eine inkrementelle Verfeinerung der Suchparameter mit ein. Konkret wurde das Konzept für

die Auswahl von Produkten auf Grundlage von Kundenrezensionen entwickelt. Wie in Abschnitt 4.1.3 beschrieben wird, lässt es sich jedoch auch abstrahieren und auf andere Anwendungsfälle anwenden.

Im Zusammenhang mit Kundenrezensionen werden in Aspect Grid zu einer Sammlung von Textdokumenten aus einer Domäne zunächst domänenspezifische Aspekte ausgewählt, zu denen sich in vielen der Dokumente Aussagen finden lassen. Dies kann manuell erfolgen. Es können jedoch dabei auch Verfahren zur automatischen Erkennung wichtiger Aspekte [BE10] zum Einsatz kommen. Die Dokumente können daraufhin auf Aussagen zu den verschiedenen Aspekten hin untersucht werden. So kann beispielsweise nach Bewertungen auf Grundlage der Aspekte gesucht werden, indem Techniken der Sentiment-Analyse [Liu12] miteinbezogen werden.

Jeder Aspektbewertung wird eine Farbe zugewiesen. In der folgenden Erklärung wird zwischen drei diskreten Stufen unterschieden, nämlich *negativ* (rot), *neutral* (gelb) und *positiv* (grün). Wird zu einem Aspekt keine Aussage getroffen, ist keine Bewertung bekannt und es wird die Farbe grau verwendet.

Eine Tabelle stellt, wie in Abbildung 4.1 erkennbar ist, den zentralen Teil im Ansatz Aspect Grid dar. Die Spalten dieser Tabelle entsprechen Aspekten, jede Zeile repräsentiert ein Textdokument. Auf diese Weise können die betrachteten Aspekte von Reviews auf einen Blick wahrgenommen werden. Die Zellen zeigen die Bewertung des jeweiligen Aspekts in jedem Textdokument sowohl durch ihre Beschriftung als auch durch ihre Farbe an.

Die Dokumente können nun basierend auf den Aspektbewertungen gefiltert werden. Hierzu können im Spaltenkopf Einschränkungen für jeden Aspekt festgelegt werden. Werden insgesamt mehrere Einschränkungen angegeben, so werden diese mittels *all*- und *any*-Verknüpfungen in Form eines booleschen Ausdrucksbaums zu Konjunktionen beziehungsweise Disjunktionen zusammengefügt (Abbildung 4.2).

Durch die Filterung werden diejenigen Dokumente, die nicht den Filterkriterien entsprechen, minimiert. Das bedeutet, dass sie in verkleinerten Zeilen dargestellt werden. Zudem werden alle auf diese Weise verkleinert aufgelisteten Dokumente, welche dieselben Aspektbewertungen haben, zu einer einzelnen Zeile gruppiert. Diejenigen Dokumente, welche die Filterkriterien erfüllen, behalten wie gehabt ihre eigenen Tabellenzeilen voller Höhe.

Da Benutzer nicht zwangsläufig von Anfang an die gewünschten Filterkriterien finden, können Tabellenzeilen markiert werden. Auf diese Weise können die eigentlichen geschriebenen Rezension angezeigt werden. In diesen Texten können Benutzer Textabschnitte markieren, die sie interessant finden. Aspekte und/oder Bewertungen, welche in den markierten Textabschnitten genannt werden, können dann in die Filterkriterien mitaufgenom-

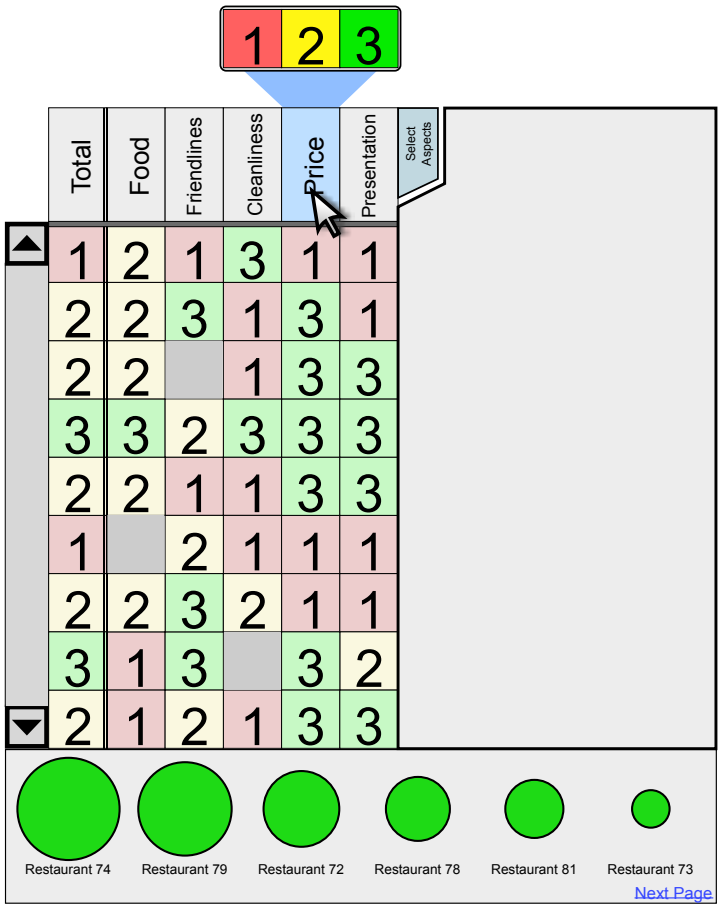


Abbildung 4.1: Der grundlegende Aufbau der Visualisierung ASPECT GRID am Beispiel von Restaurant-Rezensionen (übernommen von Abbildung 1 aus Referenz [HHJ+14]; © Gesellschaft für Informatik e.V.): Jede Spalte in der zentralen Tabelle stellt einen Aspekt dar, der in Rezensionen erwähnt wird. Jede Zeile entspricht einer Rezension. Pro Spalte können Filter eingerichtet werden. Da dies in der gezeigten Abbildung noch nicht geschehen ist, sind die Restaurants im unteren Bereich der Abbildung allein nach der Anzahl ihrer Rezensionen (die über den Radius der Kreisdiagramme angedeutet wird) sortiert.

men werden. Auf diese Weise wird eine Rückkopplungsschleife ermöglicht. Über diese können Benutzer iterativ die Suche verfeinern, indem sie aus gefundenen Rezensionen besonders wichtige Aussagen mit in die Filterkriterien aufnehmen.

Handelt es sich bei den Textdokumenten um Kundenrezensionen, so soll basierend auf diesen Rezensionen normalerweise ein Produkt gewählt werden. Die Rezensionen werden daher für jedes Produkt aggregiert. So kann für jedes Produkt ein Kreisdiagramm gezeigt werden. Darin ist dar-

Abbildung 4.2: Filtereinschränkungen für die Spalten in ASPECT GRID führen dazu, dass Rezensionen (hier aus einem Datensatz mit Restaurantbewertungen [PGP+14]), welche die Einschränkungen nicht erfüllen, verkleinert dargestellt werden (Abbildung übernommen aus Abbildung 1 in Referenz [HHJ+14]; © Gesellschaft für Informatik e.V.). Die zu filternden Aspekte können direkt in den Spaltenköpfen, aber auch durch Selektion von Abschnitten in den Rezensionstexten ausgewählt werden. Ebenso lassen sie sich durch boolesche Operatoren verknüpfen.

gestellt, welcher Anteil an Kundenrezensionen den aktuellen Filterkriterien entsprechen. Da in Abbildung 4.1 noch keine Filterkriterien gewählt sind, entsprechen noch einhundert Prozent der Rezensionen den Filterkriterien. In Abbildung 4.2 wird jedoch das Mengenverhältnis dargestellt, da nun zwei Mengen in den Kreisdiagrammen zu sehen sind. Um Benutzern auch ein Gefühl dafür zu geben, ob für ein Produkt insgesamt eher viele oder eher wenige Kundenrezensionen verfügbar sind, wird der Radius der Kreis-

diagramme anhand der Gesamtzahl an Kundenrezensionen für das Produkt skaliert. Zudem werden diese Kreisdiagramme zunächst nach dem Anteil an Kundenrezensionen, welche die Filterkriterien erfüllen, und zweitrangig nach der Gesamtzahl an Rezensionen für das Produkt geordnet. Das Produkt mit dem höchsten Anteil an Kundenrezensionen, welche den Filterkriterien entsprechen, steht also ganz vorne. Falls es mehrere derartige Produkte gibt, wird dasjenige mit den meisten Kundenrezensionen an den Anfang gestellt, wie dies bereits in Abbildung 4.1 der Fall war.

4.1.1.1 Einsatzbeispiel

Die Verwendung von Aspect Grid soll im Folgenden an einem fiktiven Beispielszenario verdeutlicht werden. Dabei geht es darum, dass eine Benutzerin ein Restaurant auswählen möchte, um einen Freund zu treffen.

Zunächst wird eine Datensammlung mit Kundenrezensionen zu den geographisch in Frage kommenden Restaurants in der Visualisierung Aspect Grid geladen. Idealerweise geschieht dies automatisch durch eine Suchmaschine, die auf Grundlage der anfänglichen Suchbegriffe erkannt hat, dass die Benutzerin Restaurants auswählen möchte. Die Benutzerin sieht sich einige Zeilen in der Tabelle von Aspect Grid an, wie sie in Abbildung 4.1 gezeigt wird. Sie erkennt, dass einige der Rezensionen für sie keine Rolle spielen. Ihr persönlich ist wichtig, dass das Essen bei den anderen Kunden einen positiven Eindruck hinterlassen hat. Alternativ gibt sie sich mit einem neutralen Eindruck vom Essen zufrieden, wenn zumindest der Preis positiv bewertet wurde, nach dem Prinzip „Das Essen sollte sehr gut sein, ansonsten sollte es zumindest nicht zu viel kosten." Sie fügt diese drei Filter hinzu und verbindet sie mit den passenden Verknüpfungsoperatoren. Diese Bedingungen sind in Abbildung 4.2 sind als Kreise über der Tabelle zu sehen.

Während die Benutzerin einige der auf diese Weise gefilterte Rezensionen markiert und liest, fällt ihr eine Aussage über das Aussehen des Essens auf. Getreu dem Prinzip „Das Auge isst mit." markiert sie die betreffende Aussage im Text der Rezension. Daraufhin wird wie in Abbildung 4.2 gezeigt eine positive Bewertung für den Aspekt *Presentation* automatisch als neuer Filter vorgeschlagen. Die Benutzerin bestätigt das Hinzufügen des neuen Filters.

Der untere Bereich von Aspect Grid zeigt nun eine kurze Liste von Restaurants. Diese sind auf Grundlage der Rezensionen nach der Vorliebe der Benutzerin angeordnet. Aus dieser Liste wählt sie nun eines der ersten Restaurants aus, basierend auf zusätzlichen Faktoren, die sich nicht automatisch aus Kundenrezensionen ermitteln lassen.

4.1.2 Gewichtung

Während oben Filterkriterien durch Ausdrucksbäume verbunden wurden, ist auch ein alternatives Modell für die Auswahl von Rezensionen nach Aspekten denkbar. Im Folgenden wird beschrieben, wie an Stelle der Ausdrucksbäume Gewichtungen für die Aspekte eingesetzt werden können.

Die zur Verfügung stehenden Textanalysetechniken konzentrieren sich oft auf eine Sentiment-Analyse. Es werden also häufig Bewertungen extrahiert. Sind Bewertungen von Aspekten bekannt, so möchten Benutzer in den meisten Fällen ohnehin die besten Bewertungen, wenn auch möglicherweise nur für einzelne der Aspekte, finden. Nur selten werden Benutzer gezielt nach schlechten Bewertungen bestimmter Aspekte suchen. Daher bietet es sich an, an Stelle einer binären Filterung eine Gewichtungs- und Sortierfunktion als Grundlage der Suchvisualisierung zu verwenden.

Um dies zu erreichen, wird der boolesche Ausdrucksbaum ersetzt durch eine Möglichkeit, die einzelnen Aspekte benutzerspezifisch zu gewichten. Zu jedem Dokument wird dann auf Grundlage der gewichteten und betrachteten Aspekte eine Gesamtbewertung berechnet. Mittels dieser werden die Dokumente sortiert. Die Rückkopplungsschleife, dass sich in den einzelnen Dokumenten Textbestandteile auswählen lassen und die darin beschriebenen Aspekte in die Suche einbezogen werden, existiert weiterhin. Zudem können die im Text erwähnten Aspekte auch direkt stärker oder schwächer gewichtet werden.

Die Produkte werden wiederum nach den aggregierten Rezensionen sortiert. Die Rezensionen werden nun nicht mehr davon abhängig unterschieden, ob sie den Filterkriterien entsprechen oder nicht. Stattdessen wird die Gesamtbewertung der Rezensionen berücksichtigt. Auf diese Weise unterstützt die Gewichtungs-basierte Variante von Aspect Grid den Benutzer ebenfalls dabei, eine Produktauswahl basierend auf ihren persönlichen Auswahlkriterien durchzuführen.

Diese Variante von Aspect Grid wurde im Rahmen einer Bachelor-Arbeit konkret ausdefiniert und prototypisch implementiert [Nau15]. Der Prototyp ist in Abbildung 4.3 zu sehen. Dabei wurde auch eine kleine Benutzerstudie mit sieben Teilnehmern durchgeführt. Die Teilnehmer empfanden laut eigener Aussage die Steuerung der Gewichtung pro Aspekt hilfreich, wenn es darum ging, Produkte zu finden, die insbesondere in bestimmten, vorgegebenen Aspekten eine gute Bewertung hatten. Die meisten der Teilnehmer konnten sich zahlreiche Produktfelder vorstellen, für die sie Aspect Grid gerne einsetzen würden. Als Beispiele wurden Kinofilme oder Lebensmittel genannt. Zwei Teilnehmer merkten jedoch auch an, die prototypische Umsetzung von Aspect Grid biete zu viele verschiedene Interaktionsmöglichkeiten und rege sie daher nicht zur Verwendung an.

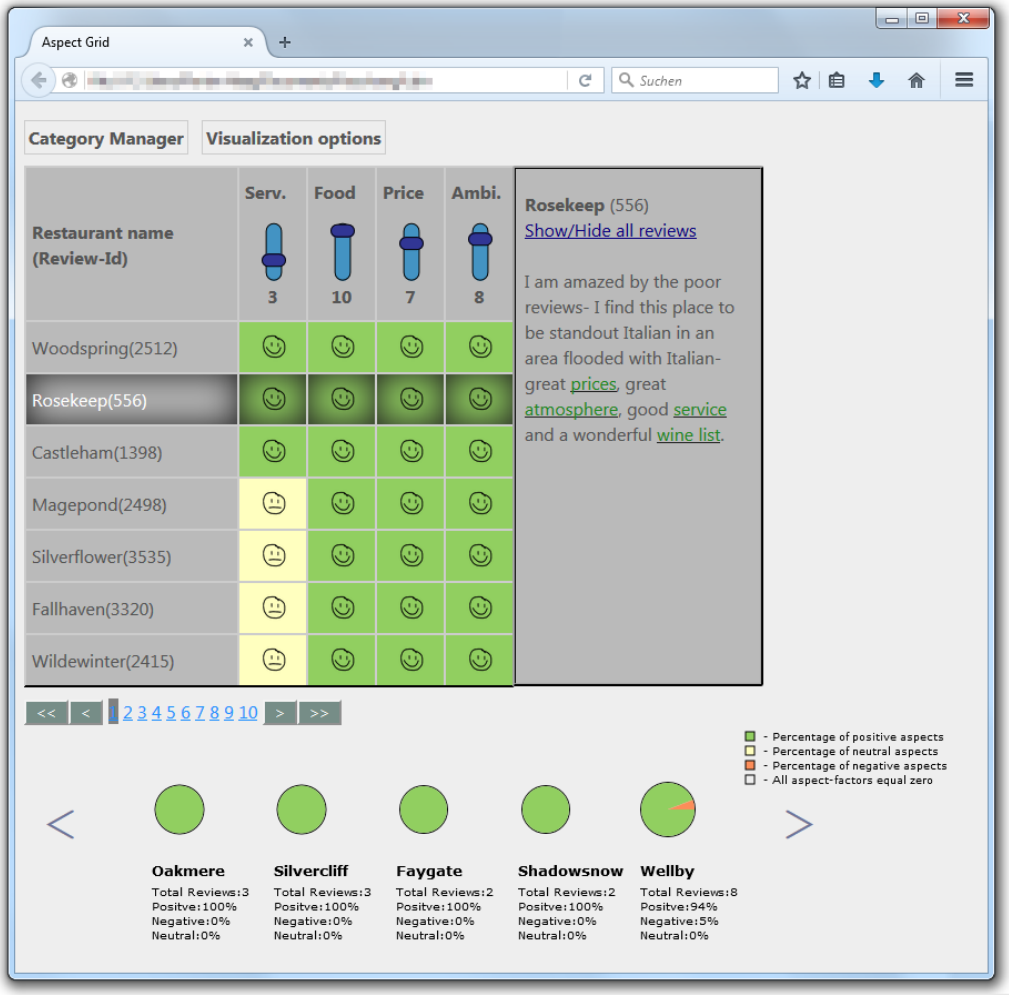


Abbildung 4.3: In der gewichteten Variante von Aspect Grid können die einzelnen Aspekte unterschiedlich gewichtet werden, was die Sortierung der Rezensionen in der Tabelle beeinflusst. Der im Rahmen einer Bachelor-Arbeit [Nau15] entstandene webbasierte Prototyp zeigt hier Daten aus demselben Datensatz wie in Abbildung 4.2 [PGP+14]. Mit den gezeigten Einstellungen wir am meisten Wert gelegt auf die Qualität der Speisen. Auch positive Bewertungen von Umgebung und Preisen werden stark gewichtet. Eine gute Bewertung der Service-Qualität wird dagegen nicht gefordert, um Rezensionen in der Tabelle nach oben zu bringen.

4.1.3 Verallgemeinerung

Während oben vorrangig auf die Verwendung von Aspect Grid in Bezug auf Produkte und ihre Kundenrezensionen eingegangen wurde, lassen sich die Anwendungsfälle auch verallgemeinern. Grundlegend ist der Ansatz Aspect Grid anwendbar auf jede Datensammlung, in der Datenelemente

durch Mengen aus Unterelementen beschrieben werden, zu denen sich eine begrenzte Anzahl an Attributen ermitteln lässt.

Sind die Unterelemente Textdokumente, so können die Attribute mittels Textanalysetechniken extrahiert werden. Andererseits sind auch bereits vorstrukturierte Daten zur Anzeige geeignet. Drücken die Attribute der Unterelemente reine Bewertungsstufen aus, so ist die gewichtete Variante von Aspect Grid (Abschnitt 4.1.2) vorzuziehen. Andernfalls kann die ursprüngliche Variante (Abschnitt 4.1.1) eingesetzt werden.

4.1.4 Fazit

VESPa dient dazu, eine schnelle facettierte Filterung und Sortierung von Datenelementen mit einer kleinen Anzahl von Attributen durchzuführen. Zudem unterstützt es die iterative Eingrenzung der Ergebnisse. Damit ist es für Benutzer nun möglich, direkt aus den Suchergebnissen weitere Einschränkungen zur Suche hinzuzufügen.

4.2 VESPa

VESPa (Visual Event Sequence Patterns) ist ein Visualisierungskonzept, das dazu dient, Muster von Ereignisabfolgen zu modellieren. Konkrete Vorkommnisse solcher Ereignisabfolgen können dann in Datensammlungen gesucht werden. Ebenso kann die Visualisierung dazu verwendet werden, die abstrakte Idee einer Ereignisabfolge visuell auszudrücken, ohne dass ein konkretes Beispiel bekannt ist. Der Fokus liegt bei VESPa auf logischen Ereignissen sowie mehreren Akteuren, die im Lauf der Ereignisabfolge miteinander interagieren können. Das Konzept wurde im Rahmen der Mini-Challenge 2 aus der VAST Challenge 2014, „The Kronos Incident“ [Vis14], entwickelt. Dabei wurde es mit der entsprechenden Datensammlung getestet, lässt sich jedoch auch auf andere Szenarien anwenden.

> Die im Folgenden präsentierten Inhalte wurden zum Teil bereits anderweitig veröffentlicht. Insbesondere wird auf den im Folgenden aufgeführten Arbeiten, an denen der Autor maßgeblich mitwirkte, aufgebaut:
>
> KHH+15 R. Krüger, D. Herr, F. Haag und T. Ertl. “Inspector-Gadget: integrating data preprocessing and orchestration in the visual analysis loop”. In: *Proc. EuroVis Workshop on Visual Analytics (EuroVA ’15)*. The Eurographics Association, 2015. DOI: 10.2312/eurova.20151096
>
> HKE16 F. Haag, R. Krüger und T. Ertl. “VESPa: a pattern-based visual query language for event sequences”. In: *Proc. Information Visualization Theory and Applications (IVAPP ’16)*. Noch nicht erschienen. 2016

4.2.1 Motivation

Bewegungen von Personen in der realen Welt lassen sich zu spatio-temporalen Ereignissen abstrahieren. Der Tagesablauf einer Person lässt sich somit als Ereignissequenz betrachten. Liegen Bewegungsdaten zahlreicher Personen vor, kann es für diverse Anwendungsfälle relevant sein, bestimmte Bewegungsmuster zu finden. Beispielsweise kann der Besuch von Geschäften durch Kunden ausgewertet [LS93], das Verhalten von Tieren analysiert [RH04; SLC+08] oder der Taxiverkehr innerhalb einer Stadt untersucht [YZL+09] werden.

Entsprechend sollten in der genannten VAST Challenge die Ortsveränderungen von Personen untersucht und miteinander in Verbindung gebracht werden. Auf diesem Weg sollten Hinweise für die Aufklärung eines (fiktiven) Verbrechens gefunden werden. Dazu schien es hilfreich, bestimmte Bewegungs- beziehungsweise Ereignisabfolgen in der zur Verfügung gestellten Datensammlung von Bewegungsdaten aus dem Alltag fiktiver Mitarbeiter eines Unternehmens zu suchen.

In diesen Bewegungsdaten sollten ungewöhnliche Vorgänge erkannt werden, indem Ortsveränderungen, Aufenthalte und Treffen der Mitarbeiter an bestimmten Orten betrachtet und gesucht werden. Orte müssen dazu nicht geometrisch exakt betrachtet werden, sondern nur auf einer logischen Ebene. So ist beispielsweise oft nicht von Interesse, ob eine Person einen Meter weiter rechts oder links steht, sondern nur, ob sie sich in einem bestimmten Restaurant oder Geschäft aufhält [NLT07; WJ07]. Derartige logische Positionsdaten lassen sich aus geometrischen Positionsdaten extrahieren [PSR+13; KTE14] oder sind mitunter sogar die einzige freigegebene Information [AAF13].

Da es in der Challenge nicht nur um das Verhalten einzelner Personen ging, sondern kriminelle Aktivitäten zwischen mehreren Mitgliedern einer verdeckten Organisation erkannt werden sollten, muss stets die Möglichkeit gegeben sein, simultane Ereignissequenzen mehrerer unbekannter Personen im Zusammenhang zu betrachten. Unter Berücksichtigung dieser Einschränkung lassen sich aus dem Szenario der Challenge zwei konkrete Anforderungen extrahieren:

- Ereignissequenzen bestimmter Beschaffenheit sollen in der Datensammlung gefunden werden.
- Die Idee, wie eine Ereignissequenz beschaffen sein soll und mit anderen Ereignissequenzen zusammenhängt, soll sich visuell ausdrücken lassen, auch wenn noch kein konkretes Vorkommen einer solchen Gruppe von Ereignissequenzen in der Datenmenge gefunden wurde.

Eine konkrete Anfrage könnte also lauten: „Welche zwei Personen sind von einer Fabrik beziehungsweise einer Bank aus zu einem gemeinsamen Treffpunkt gereist, an dem sie sich vor 15 Uhr getroffen haben?“

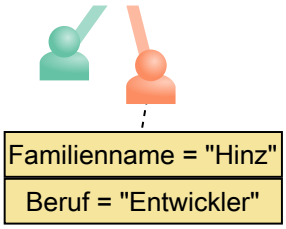
Bisherige Visualisierungen spatio-temporaler Ereignisse wie GeoTime [KW05] legen den Fokus oft auf die geografisch exakte Darstellung des spatialen Aspekts [TSA+12; SLW+14] oder stellen den spatialen Aspekt getrennt vom temporalen Aspekt dar [FMK12; MBB+11]. In beiden Fällen ist es problematisch, einen bestimmten Ort zu repräsentieren, wenn statt dem konkreten Ort nur bestimmte Merkmale des Orts bekannt sind. Lediglich (s|qu)eries stellt Ereignissequenzen grafisch mit Einschränkungen der einzelnen Ereignisse dar [ZDF+15]. Ist eine Visualisierung von Ereignissequenzen mehrerer Personen überhaupt Bestandteil des Konzepts, so beschränkt sich diese meist auf den reinen Vergleich der Ereignissequenzen zwischen Personen. Berührpunkte zwischen den Sequenzen wie den gleichzeitigen Aufenthalt am Treffpunkt werden also nicht explizit gemacht [WPQ+08]. Teilweise werden dann in der Anfrage auch nur Orte repräsentiert, während der temporale Aspekt ausgeklammert wird [CMW87]. Wie bereits in Abschnitt 2.3.5 beschrieben, werden andererseits auch gemeinsame Objekte zwischen mehreren Ereignissen bei bestehenden Konzepten nur indirekt verbunden [JS09]. Eine Ausnahme in dieser Hinsicht stellt GeoTime dar, welches Einschränkungen für Ereignisse visuell an die Ereignissymbole anhängt beziehungsweise durch verbindende Linien zwischen Ereignissen darstellt [KW05].

4.2.2 Visuelle Notation

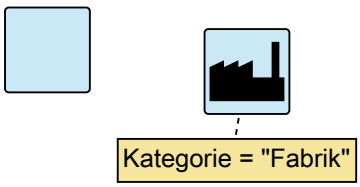
Um die oben genannten Anforderungen zu erfüllen, wurde die visuelle Notation VESPa (Visual Event Sequence Patterns) definiert. Dabei wurde darauf geachtet, die Anzahl der unterschiedlichen visuellen Elemente möglichst überschaubar zu halten. Zudem wurde die Notation dahingehend gestaltet, dass Sequenzen von Ereignissen als visuelle Abfolgen klar erkennbar sind, ähnlich wie es in (s|qu)eries der Fall ist [ZDF+15]. Folgende Notationselemente wurden definiert:

Akteur: In Anlehnung an UML wird ein Platzhalter für eine bestimmte Person in VESPa als Akteur bezeichnet [Obj15, S. 637ff.]. Ein Akteur (Abbildung 4.4a) kann sequenziell mehrere Zustände durchlaufen und dort auch mit anderen Akteuren zusammentreffen. In einem Anfrageergebnis wird jeder Akteur eindeutig auf eine konkrete Person aus der Datensammlung abgebildet.

Zustand: Ein Zustand (Abbildung 4.4b) ist ein Platzhalter für ein Ereignis aus einer Ereignissequenz, ähnlich wie die Ereignisse in anderen Arbeiten [PMR+96; FKS+06; DC96; LAB+14]. Im spatio-temporalen Kontext ist solch ein Ereignis ein Aufenthalt einer oder mehrerer Personen an einem bestimmten Ort zu einer bestimmten Zeit. Entsprechend

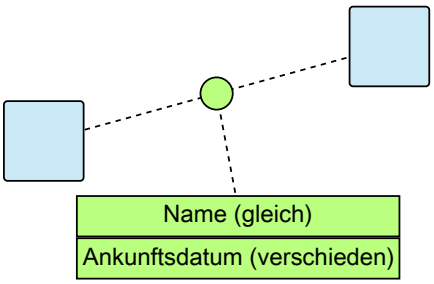


(a) Zwei Akteure, einer davon mit Einschränkungen.



(b) Zwei Zustände, einer davon mit Einschränkungen.

(c) Eine Transition zwischen zwei Zuständen.



(d) Eine Vergleichskante zwischen zwei Zuständen.

Abbildung 4.4: Visuelle Bestandteile von VESPa

wird jeder Zustand in einem Anfrageergebnis auf einen bestimmten Ort während eines bestimmten Zeitabschnitts abgebildet.

Transition: Eine Transition verbindet zwei Zustände (Abbildung 4.4c). Dadurch wird ausgedrückt, dass einer oder mehrere Akteure von einem Zustand zu einem anderen Zustand übergehen. Die einbezogenen Akteure werden als Teil der Anfrage angegeben und, wie in Abbildung 4.5 gezeigt, über die Farbe der Transition visualisiert. Eine beliebige Anzahl von Aufenthalten (normalerweise aber 0) werden durch eine Transition übersprungen.

Vergleichskante: Eine Vergleichskante verbindet Akteure und/oder Zustände, um auszudrücken, dass bestimmte Eigenschaften dieser Akteure oder Zustände miteinander in Verbindung stehen müssen. Welche Eigenschaften dies sind, wird pro Vergleichskante angegeben und dargestellt. Zu dieser Angabe zählt auch jeweils die eigentliche Vergleichsoperation – wie in Abbildung 4.4d gezeigt, können dies beispielsweise Gleichheit oder Ungleichheit sein. Die Vergleichskanten in VESPa entsprechen somit ungefähr Vergleichsknoten [RSB+08] und -kanten [Feg99] aus anderen Objektgraph-basierten Anfragevisualisierungen.

Wie in den Abbildungen 4.4a und 4.4b angedeutet, können Akteure

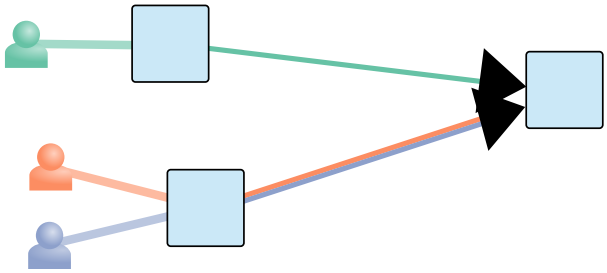


Abbildung 4.5: Transitionen in VESPA geben über ihre Farbe an, welche Akteure an der Transition beteiligt sind. Dabei kann sich eine solche Transition auch auf mehrere Akteure zugleich beziehen, wenn mehrere Akteure ungefähr zur selben Zeit von einem gemeinsamen Zustand zu einem anderen gemeinsamen Zustand wechseln.

und Zustände auch mit expliziten Einschränkungen versehen werden. Zur Verdeutlichung können dabei auch ähnlich wie in OptiqueVQS [SGJ+15] Symbole in den Zuständen angezeigt werden, welche eine oder mehrere der Einschränkungen hervorheben. Da in einer Anfrage mehrere Akteure auftauchen können, werden auch die Einschränkungen für Akteure direkt in der Visualisierung dargestellt, statt neben der Visualisierung als Einstellungsmöglichkeiten in der Benutzeroberfläche aufzutreten [FKS+06]. Auf diese Weise kann die Menge der möglichen Abbildungen für Ergebnisse anhand diverser Eigenschaften der Akteure und Zustände eingeschränkt werden. Die verfügbaren Eigenschaften und Einschränkungen richten sich dabei im konkreten Fall nach der Datensammlung und den darin zur Verfügung stehenden Daten.

Bereits mit dieser minimalen Anzahl an visuellen Elementen wird es möglich, Anfragen wie die oben genannte zu visualisieren. Eine VESPA-Repräsentation der Anfrage „Welche zwei Personen sind von einer Fabrik beziehungsweise einer Bank aus zu einem gemeinsamen Treffpunkt gereist, an dem sie sich vor 15 Uhr getroffen haben?“ ist in Abbildung 4.6 abgebildet.

4.2.3 Abbildungsvorschrift für Ergebnisse

Um die Semantik von VESPA exakt zu definieren, werden im Folgenden die Bedingungen für eine Abbildung eines VESPA-Elements auf ein Element einer Datensammlung erklärt. Die Gesamtmenge der Ergebnisse für eine VESPA-Anfrage ist jedes Ergebnis, dessen Zuordnungen von Datenelementen zu Anfrageelementen diese Bedingungen erfüllen.

Im Folgenden wird ein einfaches Datenmodell für die zu durchsuchende Datensammlung (nicht die Speicherung der Anfrage selber!) angenommen. Es ist in Abbildung 4.7 als Entity-Relationship-Diagramm dargestellt. In ei-

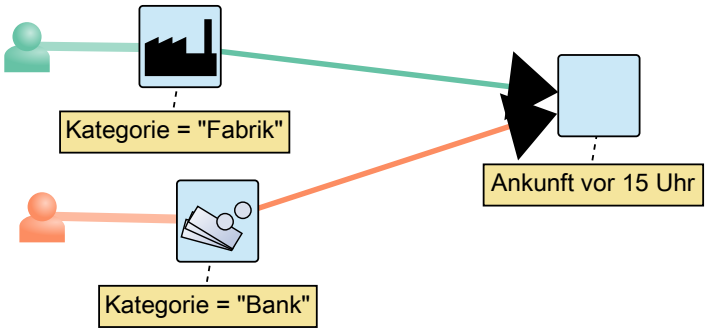


Abbildung 4.6: Die Anfrage „Welche zwei Personen sind von einer Fabrik beziehungsweise einer Bank aus zu einem gemeinsamen Treffpunkt gereist, an dem sie sich vor 15 Uhr getroffen haben?“ mit VESPa ausgedrückt.

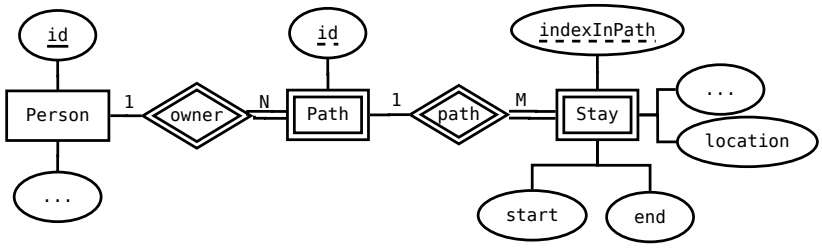


Abbildung 4.7: Für die Auswertung von VESPa-Anfragen wird das hier abgebildete einfache Datenmodell angenommen.

ner Datensammlung stehen folglich zunächst die Mengen Persons_d, Paths_d und Stays_d zur Verfügung[1]. Die gewählte einfache Struktur von Ereignissen ist an verwandte Arbeiten angelehnt [GBG04; MZM11; DLR12]. Datensammlungen, die einen abweichenden Aufbau haben, aber die notwendigen Informationen enthalten, können durch die simple Struktur des hier verwendeten Datenmodells in das erforderliche Schema überführt werden.

Wie im ER-Diagramm erkennbar wird, ist jeder Aufenthalt, *Stay*, einer einzelnen Person zugeordnet. Da wir in VESPa allerdings auch an Treffen zwischen mehreren Personen interessiert sind, ist eine Transformation der Daten notwendig[2]: Bei der Auswertung einer VESPa-Anfrage werden aus einer auf diese Weise strukturierten Datensammlung gemeinsame Aufenthalte errechnet, die mehreren Personen zugeordnet sein können. Aus den ursprünglichen Aufenthalten Stays_d und den gemeinsamen Aufenthal-

[1] Zur Verdeutlichung sind Variablen, welche die Datensammlung oder Teile davon betreffen, mit dem Index $_d$ gekennzeichnet. Entsprechend sind Bestandteile der Anfrage mit dem Index $_q$ markiert.

[2] Ob diese in einem Vorverarbeitungsschritt geschieht oder beim Analysieren der Datenbank „on-the-fly“ berechnet wird, ist ein Implementierungsdetail ohne Einfluss auf das Visualisierungskonzept VESPa.

ten $\text{Stays}_{d,\text{common}}$ ergibt sich die Vereinigungsmenge Stays'_d. Jedes Element aus Stays'_d basiert dann auf einer Menge $\tilde{S}_d \subseteq \text{Stays}_d$ von Aufenthalten aus der unverarbeiteten Datensammlung. Für $\tilde{S}_d$ muss dann gelten, dass sich die Zeitspannen aller enthaltenen Aufenthalte überschneiden und die Orte übereinstimmen, also

$$\begin{aligned}\forall \text{Stay}_{d,\alpha}, \text{Stay}_{d,\beta} \in \tilde{S}_d, \text{Stay}_{d,\alpha} \neq \text{Stay}_{d,\beta} : \\ (\text{Stay}_{d,\alpha}.\text{location} = \text{Stay}_{d,\beta}.\text{location}) \\ \wedge\ (\text{Stay}_{d,\alpha}.\text{start} < \text{Stay}_{d,\beta}.\text{end}) \\ \wedge\ (\text{Stay}_{d,\beta}.\text{start} < \text{Stay}_{d,\alpha}.\text{end})\end{aligned}$$

Jede der einbezogenen Personen muss sich also gleichzeitig mit allen anderen einbezogenen Personen am selben Ort aufgehalten haben. Im Zuge der Erweiterung von Stays_d auf Stays'_d wird die Zuordnung von Aufenthalten zu Pfaden und einem Index in diesen Pfaden aufgehoben. Stattdessen kann für einen Aufenthalt $s_d \in \text{Stays}'_d$ die Menge der anwesenden Personen $s_d.\text{persons} \subseteq \text{Persons}_d$ abgerufen werden. Ebenso kann ermittelt werden, ob für mindestens einen Anwesenden auf einen Aufenthalt $s_{d,1}$ ein Aufenthalt $s_{d,2}$ folgt ($s_{d,1} \xrightarrow{m} s_{d,2}$).

Zu beachten ist hierbei, dass ein konkretes Treffen oder eine Interaktion an einem Ort hierdurch nicht garantiert ist – zwei Personen können zum Beispiel gleichzeitig dasselbe Restaurant besucht haben, ohne sich überhaupt zu sehen. Jedoch liegen in der Realität oft keine Daten darüber vor, ob Personen tatsächlich miteinander interagiert haben. Festzustellen, ob solch eine Interaktion tatsächlich stattgefunden hat, ist Teil der weiteren Datenanalyse, nachdem mittels VESPA die Aufenthaltsorte und Bewegungen von Personen erkannt wurden.

Um die Bedeutung der Anfrageelemente eindeutig zu beschreiben, lässt sich eine VESPA-Anfrage q formal definieren als $q \subseteq \text{States}_q \times \text{Actors}_q$. Dabei wird zusätzlich gespeichert, wie die Zustände States_q miteinander zusammenhängen, und welche Akteure für die dazwischenliegenden Transitionen konfiguriert sind.

Die Zuordnung eines Aufenthalts beziehungsweise einer Person aus der Datensammlung zu einem Zustand beziehungsweise einem Akteur aus der Anfrage lässt sich dann über die Funktion valueOf mit

$$\text{valueOf} : \text{States}_q \cup \text{Actors}_q \rightarrow \text{Stays}'_d \cup \text{Persons}_d$$

ausdrücken.

Für eine erlaubte Zuordnung $\text{valueOf}(x_q) = y_d$ muss zunächst gelten, dass alle Einschränkungen eines Akteurs beziehungsweise eines Zustands x_q von der entsprechenden Person beziehungsweise dem entsprechenden Aufenthalt y_d erfüllt werden. Ferner müssen für jede Vergleichskante zwischen

Akteuren oder Zuständen $x_{d,1}$ und $x_{d,2}$ die Bedingungen, die durch die Vergleichskante spezifiziert sind, von dem Paar $\{\text{valueOf}(x_{d,1}), \text{valueOf}(x_{d,2})\}$ erfüllt werden.

Treffen diese Vorbedingungen zu, so wird im Folgenden davon ausgegangen, dass jedem Akteur eindeutig eine Person zugeordnet ist, dass also

$$\begin{aligned}\forall a_q \in \text{Actors}_q \exists p_d \in \text{Persons}_d : \\ (\text{valueOf}(a_q) = p_d) \\ \wedge (\forall a_{q,0} \in \text{Actors}_q : a_q = a_{q,0} \Leftrightarrow \text{valueOf}(a_q) = \text{valueOf}(a_{q,0}))\end{aligned}$$

erfüllt ist.

In der Datensammlung können dann aufeinanderfolgende Aufenthalte $\dot{S}_d \subseteq \text{Stays}'_d$ gefunden werden, die Paare von Zuständen $\{z_{q,\alpha}, z_{q,\beta}\} \subseteq \text{States}_q$ entsprechen, sodass eine Kette von Aufenthalten gemäß der Einschränkung

$$\begin{aligned}\dot{S}_d &= \{\dot{s}_{d,1}, \ldots, \dot{s}_{d,\dot{n}}\} \\ \text{mit } \forall i \in \{1, \ldots, \dot{n}-1\} : \dot{s}_{d,i} &\xrightarrow{m} \dot{s}_{d,i+1} \\ \text{valueOf}(z_{q,\alpha}) &= \dot{s}_{d,1} \\ \text{valueOf}(z_{q,\beta}) &= \dot{s}_{d,n}\end{aligned}$$

identifiziert wird. Dazu muss eine Transition von $z_{q,\alpha}$ zu $z_{q,\beta}$ existieren und für die Menge $\dot{A}_q \subseteq \text{Actors}_q$, die in diese Transition einbezogen sind, muss

$$\forall \dot{s}_d \in \dot{S}_d : \{p_d | \exists a_q \in \dot{A}_q : \text{valueOf}(a_q) = p_d\} \subseteq \dot{s}_d.\text{persons}$$

gelten. $\dot{n} - 2$ entspricht der Anzahl an Aufenthalten, die im Zuge der Transition übersprungen werden können.

4.2.4 Einschränkungen

Das Ziel beim Entwurf von VESPa bestand darin, eine möglichst einfache visuelle Notation zum Aufspüren von Ereignisabfolgen mit mehreren Akteuren bereitzustellen. Die Ereignisabfolgen sollten sich auf Bewegungen zwischen Orten und Treffen zwischen Akteuren konzentrieren. Um die Einfachkeit der Visualisierung zu gewährleisten, werden alternative Einschränkungen momentan nicht unterstützt. Diese ließen sich möglicherweise über Verzweigungen in den Akteur-bezogenen Pfaden visualisieren. Ebenso erlauben die Vergleichskanten lediglich kommutative Vergleichsoperationen, da keine Reihenfolge der Operanden dargestellt wird.

Ferner werden auch bestimmte temporale Beziehungen nicht unterstützt: Beispielsweise werden mit der aktuellen Definition von VESPa die

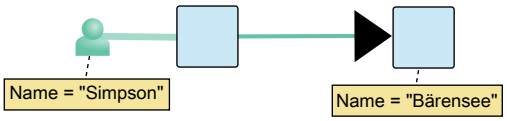


Abbildung 4.8: Eine der einfacheren Verständnisaufgaben aus der VESPA-Benutzerstudie: Der gezeigte Anfragegraph sucht alle Gelegenheiten, bei denen sich eine Person namens „Simpson“ von einem beliebigen Ort aus zum Bärensee begibt.

Aufenthalte aller Personen an einem Ort zusammengefasst, sodass partielle Überschneidungen nicht mehr beachtet werden. Dies lässt sich zwar implementatorisch umgehen, indem auch in der Datensammlung verzweigte Sequenzen vorgesehen werden, jedoch könnte hier der Einsatz bereits definierter Algorithmen hilfreich sein [LAB+14].

4.2.5 Evaluation

Um einen Eindruck von der Verständlichkeit von VESPA zu erhalten, wurde eine kleine qualitative Auswertung mit fünf Teilnehmern durchgeführt. Das Ziel bestand darin, zu ermitteln, ob Benutzer nach einer kurzen Einführung in der Lage sind, Ereignissequenzmuster mit VESPA auszudrücken und in VESPA dargestellte Ereignissequenzen korrekt zu deuten. Dazu wurden mehrere Verständnis- und Kompositionsaufgaben vorbereitet.

4.2.5.1 Verständnisaufgaben

In den sechs Verständnisaufgaben erhielten die Teilnehmer eine gedruckte VESPA-Anfrage. Ihre Aufgabe bestand darin, in natürlicher Sprache möglichst genau zu beschreiben, welche Anfrage mit der Visualisierung ausgedrückt wurde. Zudem wurden sie gebeten, Vermutungen über die Handlungen zu äußern, welche durch die Anfrage gefunden werden, um zu erkennen, ob Beispiele ähnlich den Ereignissequenzen in der realen Welt wiedererkannt werden. Die Komplexität der Aufgaben reichte von sehr einfachen (ähnlich[3] der in Abbildung 4.8 dargestellten) bis hin zu schwierigeren (ähnlich der in Abbildung 4.9 gezeigten) Aufgaben.

[3] Die gezeigten Aufgaben wurden mit den dargestellten Bestandteilen und im Großen und Ganzen demselben Layout in der Studie gezeigt. Während es sich bei den in der Studie verwendeten Grafiken jedoch um rasterbasierte Screenshots handelte, wurden die Grafiken der besseren Druckqualität wegen für dieses Dokument neu als Vektorgrafiken erzeugt.

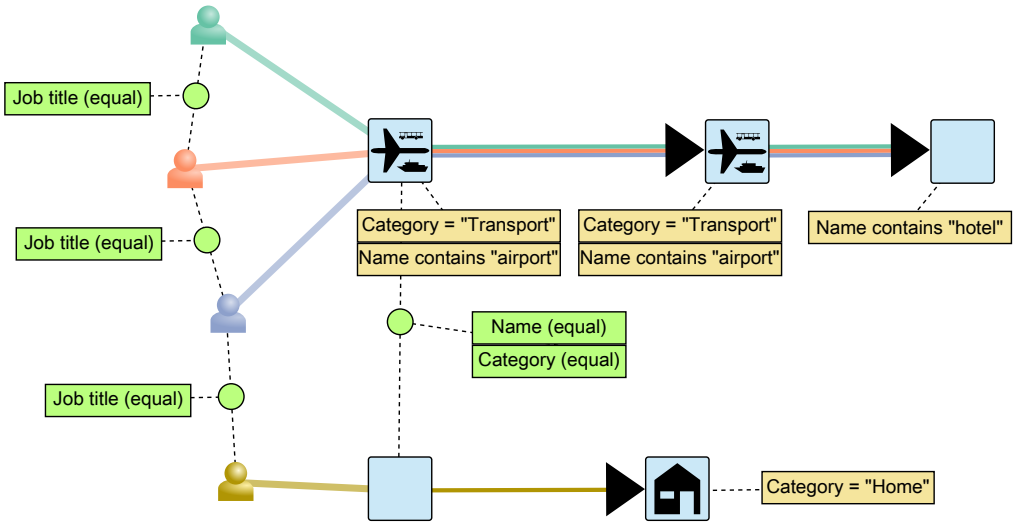


Abbildung 4.9: Eine der komplizierteren Verständnisaufgaben aus der Benutzerstudie: Mit dem gezeigten Anfragegraph kann nach Gelegenheiten gesucht werden, bei denen drei Personen mit demselben Beruf gemeinsam eine Flugreise zu einem anderen Ort unternehmen, von dem aus sie sich zu einem Hotel begeben, während eine vierte Person, die ebenfalls denselben Beruf ausübt, sich zwar am Abflughafen aufhält, die anderen drei Personen dort aber nicht antrifft, und sich wieder nach Hause begibt. Ein möglicher Kontext, in dem dieses Ereignismuster auftreten könnte, wäre eine Dienstreise von vier Kollegen, bei der ein Kollege zu spät zum Flughafen kommt und die Reise somit nicht antreten kann.

4.2.5.2 Kompositionsaufgaben

In den sechs Kompositionsaufgaben erhielten die Teilnehmer jeweils eine gedruckte, natürlichsprachlich ausgedrückte Anfrage. Diese sollte mit einer prototypischen Implementierung von VESPa (siehe Abschnitt 4.2.6) am Computer wiedergegeben werden. Auch hier waren die ersten Anfragen relativ einfach (zum Beispiel „Finden Sie alle Personen, die von einem Restaurant zu einer Fabrik, und von dort weiter zu einem anderen Restaurant fahren.“) und die späteren komplexer (zum Beispiel „Gibt es dreiköpfige Gruppen, deren Mitglieder sich zunächst gemeinsam am selben Ort aufhalten, von denen sich zwei gemeinsam zu einem anderen Ort begeben, während die dritte Person zu einem anderen Ort reist?“). Das Ziel war es jeweils nur, die richtige Anfrage zu erstellen; eine Ausführung der Anfrage war nicht Teil des Konzepts und somit auch nicht in der Evaluation enthalten.

4.2.5.3 Materialien

Alle Materialien der Studie wurden auf Deutsch vorbereitet, was auch die Muttersprache der Teilnehmer war. Eine kurze gedruckte Einführung in die grundlegenden Konzepte von VESPA wurde zusätzlich zu den oben beschriebenen gedruckten Aufgaben vorbereitet. Um die gedankliche Übertragung in die reale Welt einfacher zu gestalten, wurden statt dem in dieser Arbeit verwendeten technischen Vokabular eher alltägliche Begriffe gewählt. So wurde beispielsweise aus einer Vergleichskante (siehe Abschnitt 4.2.2) eine „Vergleichslinie“.

Der implementierte Prototyp wurde auf einem 19”-Monitor angezeigt. Eine explizite Beschreibung der Benutzeroberfläche lag nicht vor, jedoch hatten die Teilnehmer Zeit dazu, sich selbständig mit der Oberfläche vertraut zu machen, bevor mit der Bearbeitung der Aufgaben begonnen wurde. Außerdem durften auch während der Studie Fragen zur Bedienung gestellt werden, da in der Studie nur die visuelle Notation ausgewertet werden sollte, nicht die konkrete Implementierung und ihre Menüs.

4.2.5.4 Teilnehmer

Als Teilnehmer wurden fünf wissenschaftliche Mitarbeiter (vier männlich, eine weiblich) aus dem Gebiet der Visualisierung eingeladen. Jeder Teilnehmer nahm separat an der Evaluation teil. Dabei waren jeweils zwei Beisitzer anwesend, um etwaige Fragen zu beantworten und die Antworten der Teilnehmer zu notieren.

4.2.5.5 Durchführung

Die Teilnehmer bekamen zunächst Zeit, um die gedruckte Einführung in die visuelle Notation zu lesen. Anschließend durften sie die prototypische Implementierung (siehe Abschnitt 4.2.6) explorativ kennenlernen.

Die Teilnehmer erhielten die gedruckten Verständnisfragen (Abschnitt 4.2.5.1) und lösten diese. Dann wurden die Fragebögen mit den Kompositionsfragen (Abschnitt 4.2.5.2) ausgegeben, welche von den Teilnehmern am Computer bearbeitet werden mussten. Die Beisitzer zeichneten die relevanten Aspekte des Lösungswegs auf. Abschließend wurden die Teilnehmer gebeten, ihre Gedanken und Vorschläge zum Konzept mitzuteilen und mögliche Verwendungsszenarien vorzubringen.

4.2.5.6 Ergebnisse

Insgesamt waren die Ergebnisse vielversprechend, da alle Teilnehmer zu jeder Aufgabe eine Lösung finden konnten, die zumindest teilweise korrekt

war. Von den insgesamt 60 Aufgaben wurden 42 richtig beantwortet – 23 Verständnis- und 19 Kompositionsaufgaben – und die übrigen Antworten enthielten zumeist nur einen einzelnen Fehler, welcher sich auf die Interpretation von Zuständen zurückführen ließ. In der jeweils angeschlossenen Frage nach möglichen Hintergrundgeschichten für die Ereignissequenzmuster waren die Teilnehmer fast durchgehend in der Lage, genau das Szenario zu rekonstruieren, das beim Erstellen der Aufgabe zugrunde gelegt worden war.

Die Kommentare zum allgemeinen Konzept fielen überwiegend positiv aus. Mehrere der Teilnehmer gaben an, dass ihnen die Notation nach Hinweisen auf ihre Fehler vollkommen klar sei. Zwei Teilnehmer bezeichneten die Visualisierung als „intuitiv“ und schätzten den Lernaufwand als sehr gering ein. Ein weiterer Teilnehmer kam zu dem Schluss, dass es mit VESPa deutlich einfacher ist, eine Vorstellung von einer Ereignissequenz zu bekommen, als auf Grundlage einer textuellen Beschreibung.

Obwohl darauf hingewiesen worden war, dass im Rahmen der Evaluation Ortsnamen bestimmte Orte eindeutig identifizieren, wünschten sich einige Teilnehmer eine zusätzliche Kante, um zu erzwingen, dass zwei Zustände sich auf denselben Ort beziehen. Ebenso setzten viele Teilnehmer bestimmte Zustände mit bestimmten Orten gleich. Folglich erkannte auch keiner der Teilnehmer, dass in einer der Verständnisaufgaben zwei Zustände denselben Ort zu unterschiedlichen Zeitpunkten repräsentierten. Laut Aussage eines Teilnehmers trugen insbesondere die (ortsbezogenen) Symbole in einigen Zuständen dazu bei, die Zustände als Orte zu interpretieren. Gleichermaßen erzwangen auch einige der Teilnehmer nicht, dass zwei Zustände unterschiedliche Orte ausdrückten, wo dies explizit gefordert war.

Sekundäre Anmerkungen bezogen sich hauptsächlich auf die prototypische Implementierung. Unter anderem wurden dickere Verbindungslinien sowie eine klarere visuelle Trennung zwischen Transitionen und den Verbindungslinien zwischen Akteuren und Zuständen gefordert.

4.2.5.7 Diskussion

Die Hauptschwierigkeit für die Teilnehmer bestand darin, Zustände als einen Ort *in einem bestimmten Zeitraum* zu sehen. Keiner hatte einen konkreten Verbesserungsvorschlag und alle stimmten darin überein, dass die Definition logisch sei, nur nicht intuitiv. Eine mögliche Abhilfe wäre ein expliziter visueller Hinweis auf den temporalen Aspekt, beispielsweise ein kleines Uhr-Symbol.

Zwei der Teilnehmer zeigten auch leichte Verwirrung gegenüber der Anforderung, dass sich zwei Akteure an einem Ort *treffen* müssen. Sie fügten jeweils einen Initialzustand pro Akteur ein, von dem aus eine Transition zum gemeinsamen Zustand führte, statt direkt mit dem gemeinsamen Zu-

stand zu beginnen. Aus einer Diskussion nach Abschluss der Studie ergab sich, dass eine stärkere visuelle Verbindung der Akteure mit den Zuständen selber (statt nur – über ihre Farbe – mit den nachfolgenden Transitionen) zur Wahrnehmung beitragen könnte, dass sich Akteure beim Start in einem gemeinsamen Zustand treffen.

Durch die Antworten eines Teilnehmers zeigte sich auch, wie wichtig eine durchdachte Benennung von Ortskategorien und eine entsprechende Auswahl von Symbolen ist. Der fragliche Teilnehmer interpretierte Zustände, die sich auf Orte der Kategorie „Transport“ bezogen, als Transportvorgang an sich statt als transportbezogene Orte wie Bahnhöfe oder Flughäfen, da das Symbol die Transportvehikel und nicht die entsprechenden Gebäude zeigte.

4.2.6 Implementierung

Eine prototypische Implementierung von VESPa wurde als Bestandteil eines bereits bestehenden, größeren Systems zur visuellen Datenanalyse eingebaut. Der in Abbildung 4.10 gezeigte Prototyp wurde mit Java unter Verwendung der Benutzerschnittstellenbibliothek Swing entwickelt.

Insgesamt bietet die Oberfläche der Anwendung miteinander verknüpfte visuelle Werkzeuge wie eine Karten- und eine Zeitleistenansicht. Die VESPa-Visualisierung ist darin in einem eigenen Bereich untergebracht, der optional eingeblendet werden kann. Die Möglichkeiten zur Benutzerinteraktion sind relativ einfach gehalten, indem VESPa-Elemente über ein Kontextmenü eingefügt und per Drag-und-Drop verschoben werden können. Wird eine Anfrage ausgeführt, so können die Ergebnisse im Zeitleistenbereich angezeigt werden.

Für die Suche nach Resultaten wird die Datensammlung mit Bewegungsdaten zunächst in eine SQLite-Datenbank überführt. Daraufhin wird eine aus der VESPa-Anfrage generierte SQL-Anfrage ausgewertet. Für kleinere Anfragen mit zwei Personen und fünf Knoten pro Person können die Ergebnisse innerhalb weniger Sekunden ermittelt werden. Zur weiteren Beschleunigung erscheint jedoch eine Überarbeitung des Datenbankschemas sinnvoll, durch die auch momentan nicht unterstützte Randfälle (zum Beispiel, dass eine Person eine Gruppe von Personen an einem Ort verlässt und kurz darauf zu diesem Ort zurückkehrt, während die anderen Personen dort bleiben) korrekt ausgewertet werden könnten.

4.2.7 Fazit

VESPa dient zur visuellen Darstellung von Ereignissequenzmustern, die in Datensammlungen zur Suche nach konkreten Ereignissequenzen benutzt

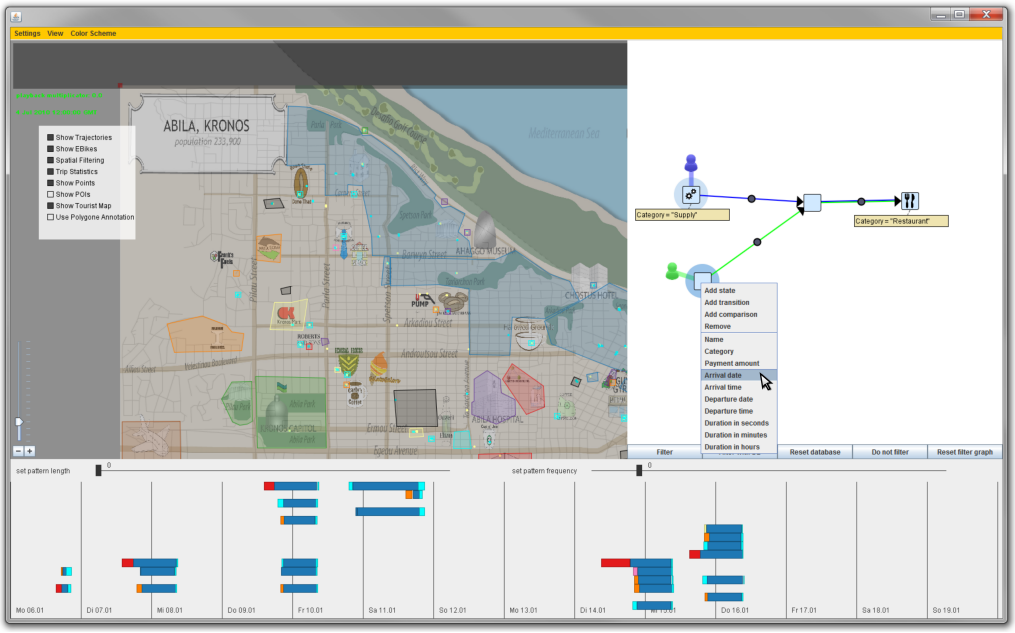


Abbildung 4.10: Eine prototypische interaktive VESPa-Implementierung in einem Visual-Analytics-Werkzeug mit verknüpften Ansichten [KHH+15]. Im rechten Bereich ist eine Anfrage nach Personen zu sehen, die von einem Ort der Kategorie *supply* zu einem Treffpunkt mit einer anderen Person anreisen, mit der sie dann gemeinsam ein Restaurant aufsuchen. Die ermittelten Ergebnisse werden im unteren Bereich des Fensters in Form von Zeitbalken ausgegeben. Die geladene Datensammlung (inklusive der dargestellten Landkarte und ihrer Markierungen) stammt aus der VAST Challenge 2014 [Vis14].

werden können. Durch den Fokus auf mehrere unabhängige Akteure, deren Ereignissequenzen sich überschneiden können, ist es mit VESPa nun möglich, gezielt nach Treffen zwischen Personen oder gemeinsamen Reiseabschnitten zu suchen.

Teil III

Zusammenführung von Anfragekonzepten

5 Kombination der Anfragekonzepte

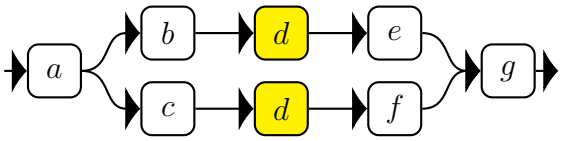
Nachdem in den vorangegangenen Kapiteln eine Reihe neuer Anfragevisualisierungen vorgestellt wurde, soll in diesem Kapitel beleuchtet werden, wie sich diese gemeinsam verwenden lassen. Dazu werden die Techniken zunächst noch einmal systematisch gegenübergestellt (Abschnitt 5.1). Anschließend wird anhand eines kurzen Anwendungsbeispiels erläutert, wie sich mehrere dieser Techniken in verschiedenen Schritten einer Suche einsetzen lassen (Abschnitt 5.2). Letztendlich werden Möglichkeiten vorgestellt, die bestehenden und neuen Techniken direkt visuell miteinander zu kombinieren (Abschnitt 5.3).

5.1 Gegenüberstellung und Vergleich

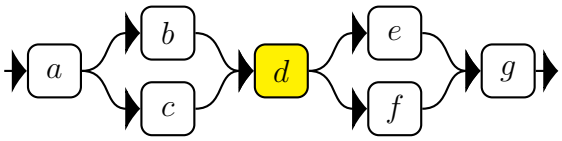
In den vorangehenden Kapiteln wurden insgesamt sechs Ansätze für Anfragevisualisierungen vorgestellt. Das Erweiterte Filter/Flow-Konzept (EFFK) aus Abschnitt 3.1, QueryVOWL aus Abschnitt 3.2 sowie VESPa aus Abschnitt 4.2 sind grundlegend Node-Link-basierte Konzepte, wie in Abschnitt 2.3.1 beschrieben. Magnetic Querying aus Abschnitt 3.3 drückt die Anfrage teilweise durch Positionen von Elementen aus (siehe Abschnitt 2.3.4) und Aspect Grid in Abschnitt 4.1 liegt wie den in Abschnitt 2.3.2 vorgestellten Konzepten eine Tabelle zugrunde, jedoch finden sich in beiden Konzepten auch Node-Link-basierte Bestandteile. Ebenso stellen Filter Dials (siehe Abschnitt 3.4) eine Mischung aus Ansätzen ohne frei kombinierbare Notationselemente und Node-Link-basierten Ansätzen dar.

Das EFFK setzt die Kanten der Node-Link-Darstellung dazu ein, um logische Verknüpfungen auszudrücken. Prinzipiell funktionieren die in Abschnitt 3.4.2 beschriebenen Verbindungen zwischen Filter Dials auf dieselbe Weise. Exakter ausgedrückt wird bei dieser flussbasierten Repräsentation durch die Kanten die schrittweise Auswertung der Datenelemente gemäß der logischen Verknüpfungen zwischen den Filterkriterien repräsentiert. Dies steht im Gegensatz zu booleschen Ausdrucksbäumen, welche die Verschachtelung der Kriterien und logischen Verknüpfungen repräsentieren. Dies ist in Magnetic Querying und auch in einer Variante von Aspect Grid der Fall.

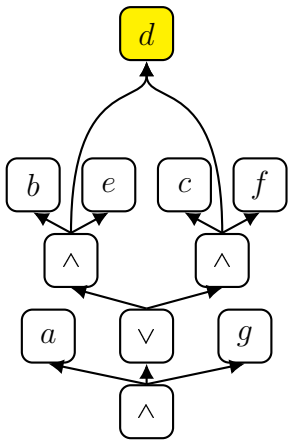
Mit beiden Darstellungsformen logischer Verknüpfungen lassen sich beliebig komplex kombinierte Filterkriterien visualisieren. Die flussbasierte



(a) Der Graph bildet die Anfrage korrekt ab, erfordert aber die duplizierte Darstellung des Kriteriums d in zwei Filterknoten, sofern die Reihenfolge der Kriterien beibehalten werden soll.



(b) Der Graph vermeidet die Duplikation des Filterknotens d; dabei geht allerdings die Information verloren, dass ein gemeinsames Auftreten von b, d und f nicht ausreicht, um den Graphen zu durchlaufen.



(c) Die Darstellung als Ausdrucksbaum ist ohne mehrfache Darstellung des Kriteriums d möglich, wenn Blattknoten wiederverwendet werden.

Abbildung 5.1: Der Ausdruck $a \wedge ((b \wedge d \wedge e) \vee (c \wedge d \wedge f)) \wedge g$, dargestellt als Filter/Flow-Graph und als Ausdrucksbaum.

Darstellung ist für Benutzer direkt visuell nachvollziehbar, ohne dass die Bedeutung der booleschen Verknüpfungsoperatoren wie *und* und *oder* bekannt sein muss [YS93]. Andererseits kann es bei bestimmten Konstellationen von Filterkriterien vorkommen, dass ein Kriterium in einem Filter/Flow-Graphen wie in Abbildung 5.1a mehrfach dargestellt werden muss. Dieser Fall tritt ein, wenn dasselbe Kriterium an unterschiedlichen Stellen im Graphen zum Einsatz kommt und der Graph nicht grundlegend umstrukturiert werden soll. Derselbe Filterknoten kann dann nicht mit den Flüssen an unterschiedlichen Stellen im Filtergraphen verknüpft werden, ohne visuelle Mehrdeutigkeiten wie in Abbildung 5.1b zu erzeugen. In einem Ausdrucksbaum, wie er in Abbildung 5.1c abgebildet ist, ist es jedoch visuell kein Problem, ein Kriterium an mehrere Blätter des Baums anzuschließen. Der Betrachter „verlässt“ einen Knoten in einem Ausdrucksbaum grundsätzlich immer über dieselbe Kante, über die er ihn „erreicht“ hat.

Aus beiden Darstellungsformen logischer Verknüpfungen durch Kanten ergibt sich der Vorteil, dass sich beliebige Kriterien miteinander in Konjunktionen oder Disjunktionen kombinieren lassen. Zusammenhänge aus der gefilterten Graphstruktur werden dabei allerdings nicht ausgedrückt. Dazu wären Querverbindungen wie in Abschnitt 3.1.2.5 beschrieben notwendig.

Die Graphstruktur wird durch Node-Link-basierte Visualisierungen wie QueryVOWL angedeutet. In gewissem Maß ist dies auch durch die in

Abschnitt 3.3.1 beschriebenen Hierarchieknoten von Magnetic Querying der Fall. Aspect Grid visualisiert dahingegen nur noch einen geringen Teil der Graphstruktur. Dazu werden betrachtete Aspekte visuell im Zusammenhang mit den Rezensionen, zu denen sie gehören, dargestellt.

Nach der Klassifikation von Batini et al. enthalten alle vorgestellten Konzepte sowohl Aspekte der Top-Down- als auch der Bottom-Up-Strategie [BCC+91]. In allen sechs Ansätzen werden zunächst einige Bestandteile der Datensammlung ausgewählt, die in der Anfrage verwendet werden sollen (Top-Down-Strategie). Anschließend können aber unabhängig voneinander formulierte Teilanfragen verknüpft werden (Bottom-Up-Strategie). Die Top-Down-Strategie überwiegt allerdings in Aspect Grid, während die Bottom-Up-Strategie wegen der parallelen Auswertung von Subgraphen und der potenziellen späteren Vereinigung der resultierenden Datenmengen insbesondere im EFFK ausgeprägt ist. Die Konzepte fallen somit ebenfalls in die Pattern-Matching-Strategie.

Von den vorgestellten Konzepten bieten prinzipiell zwei die Möglichkeit, die Gesamtmenge der Elemente aus der zu durchsuchenden Datensammlung darzustellen: Magnetic Querying visualisiert alle Datenelemente als magnetische Partikel. Dabei können zur Verbesserung der Skalierungsfähigkeit diejenigen Datenelemente ausgeblendet werden, die von keinem der aktuell vorhandenen Magneten angezogen werden. Aspect Grid zeigt ebenso sämtliche Bewertungen und Produkte auf Grundlage der aggregierten Bewertungsergebnisse an. Im EFFK ist die gesamte initiale Datensammlung nur durch die Dicke etwaiger ungefilterter Flüsse am Anfang des Graphen angedeutet. Dort ließe sie sich auch durch Ergebnisanzeigeknoten im Detail abrufen. Filter Dials stellen jeweils eine aggregierte Information der Ergebnismenge jedes Ergebnisbereichsabschnitts dar, beispielsweise die Anzahl der Datenelemente. Ebenso wird in QueryVOWL zum Teil der Umfang der Ergebnismenge als Zahl und über die Größe von Anfrageelementen repräsentiert.

Aus diesen Beobachtungen lassen sich nur bedingt allgemeine Schlüsse ziehen. Nicht alle grafischen Konzepte eignen sich zur Darstellung der gesamten Datensammlung [Spo93; Cru92]. Dasselbe gilt für tabellenbasierte Ansätze [SCT91]. Aggregierte Aussagen über die Datensammlung, wie die Anzahl der gesamten oder einer Teilmenge der Datenelemente, können andererseits in praktisch jedes Konzept eingefügt werden.

Als generelles Unterscheidungsmerkmal zwischen Anfragevisualisierungen fällt letztendlich auf, dass einige Visualisierungen von der Verfügbarkeit einer Datensammlung abhängen. Nur mit Visualisierungen, die auch ohne Zugriff auf eine Datensammlung einsetzbar sind, lassen sich reine Hypothesen zu Sachverhalten ausdrücken, die sich möglicherweise in keiner Datensammlung wiederfinden [LAB+14]. Diese Fähigkeit fehlt insbesondere Anfragevisualisierungen, bei denen die ursprüngliche Datensammlung

(beziehungsweise ein Ausschnitt daraus) als Eingabe in die Anfrage mit eingeht [CDR+99; HS01; KHS02]. Entsprechend sind derartige Konzepte mitunter gezielt für Datensammlungen vorgesehen, die dem Benutzer schon ungefähr bekannt sind. So besteht in diesen Fällen eine Vorstellung davon, welche Datenelemente überhaupt in der Datensammlung enthalten sind [CDR+99].

5.2 Kombinierte schrittweise Benutzung

Die in Teil II vorgestellten Ansätze können gemeinsam benutzt werden und sich auf diese Weise gegenseitig ergänzen. Für die im vorangegangenen Kapitel erwähnte VAST Challenge 2014 [Vis14] konnten mittels VESPA erfolgreich Ansatzpunkte zu einer Lösung der Aufgabe gefunden werden. Rückblickend betrachtet hätten sich jedoch durch einen kombinierten Einsatz unterschiedlicher Anfragevisualisierungen möglicherweise noch klarere Ergebnisse[1] erzielen lassen. Einige Ideen hierzu werden im Folgenden beschrieben.

In der Challenge wurden Bewegungsdaten zu Fahrzeugen bereitgestellt, die nur zum Teil an Personen gebunden waren. Das heißt, einige der Bewegungsverläufe waren fest mit Fahrzeugen assoziiert. Diese Fahrzeuge könnten aber im Verlauf der Zeit von unterschiedlichen Fahrern bewegt worden sein. Mittels Magnetic Querying wäre es möglich, diverse Anhaltspunkte – Kombinationen aus besuchten Orten oder Uhrzeiten der Benutzung – in mehrere Gruppen zu unterteilen. So könnten unterschiedliche Fahrer identifiziert werden. Ebenso gäbe es die Möglichkeit, Aufenthalte an bestimmten Orten pro Tag als Zeilen in der Visualisierung Aspect Grid vorzusehen. Eine Gewichtung ließe sich dann in Abhängig von der Dauer, Anfangs- oder Endzeit vornehmen. Auf diese Weise könnten beispielsweise Personen abhängig davon geordnet werden, ob sie sich besonders häufig besonders lange an einem Ort aufgehalten haben. Um zu ermitteln, welche Kombinationen aus besuchten Orten überhaupt aufgetreten sind, ließen sich Filter Dials konfigurieren.

Für die Anwendung von VESPA wurden aus der gegebenen Landkarte logische Orte extrahiert. Dazu wurden Informationen über Geschäfte und andere Einrichtungen, die im Rahmen der Challenge zur Verfügung gestellt worden waren, in der Landkarte annotiert. In echten Landkarten lässt sich eine derartige Annotation durch ortsbezogene Daten aus öffentlichen Geschäftsverzeichnissen automatisieren. Zudem lassen sich aus den vorhandenen Daten weitere Beziehungen extrahieren, wie beispielsweise die

[1] Das heißt umfangreichere und exaktere Erkenntnisse, welche einen besseren Eindruck von der zu analysierenden Situation vermittelt hätten.

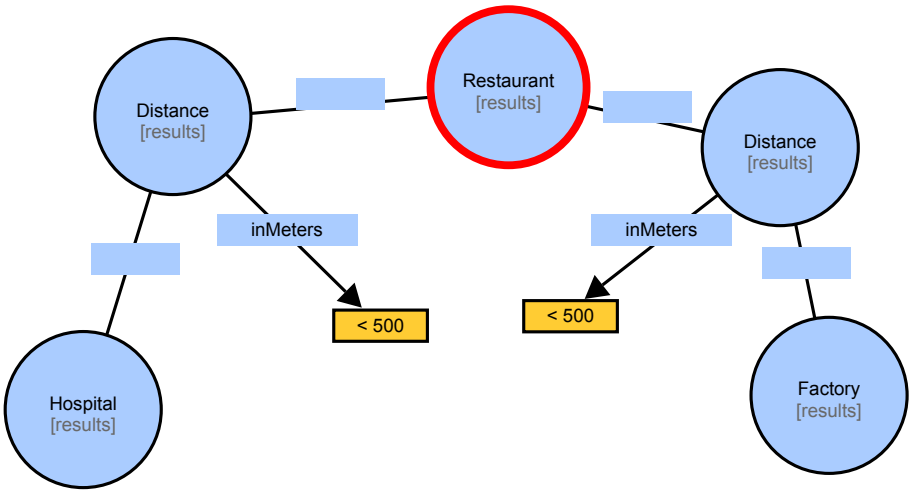


Abbildung 5.2: Eine QueryVOWL-Anfrage für fiktive Daten, wie sie sich aus einer annotierten Landkarte extrahieren ließen: „Finde alle Restaurants, die in einem Umkreis von 500 Metern um ein Krankenhaus und eine Fabrik liegen.“ Die ursprünglichen Kartendaten mit semantischen Annotationen der Ortstypen müssen dazu lediglich in einen RDF-Graphen überführt werden, wobei zu allen annotierten Orten paarweise *Distance*-Individuen generiert werden müssen, die jeweils in einem Attribut *inMeters* die Entfernung zwischen den jeweiligen zwei Orten in Metern angeben.

Entfernung. Diese könnte dann mittels QueryVOWL eingeschränkt werden, um die Auswahl von Orten einzugrenzen. Abbildung 5.2 skizziert eine solche Anfrage.

Letztendlich wäre es auch möglich, dass mehrere alternative auffällige Ereignissequenzen zur Auswahl stehen, an denen jeweils unterschiedliche Personen beteiligt waren. Die entsprechenden Alternativen und die personenbezogenen Einschränkungen, die in jedem Fall zutreffen müssen, wären mit dem EFFK visualisierbar. Ebenso könnten in Verbindung mit Magnetic Querying alle Mitarbeiter der fiktiven Firma aus der Challenge als Partikel visualisiert werden. Eine Gruppierung könnte dann nach durchschnittlichen Arbeitszeiten, häufigen Aufenthaltsorten oder ähnlichen Kriterien erfolgen. Bei ausschließlicher Verwendung von VESPa sind mehrere separate Anfragen und ein manueller Abgleich der Ergebnisse notwendig.

5.3 Verknüpfungspunkte

Das Übertragen von Teilergebnissen aus einer Anfragevisualisierung in eine andere würde einen manuellen Aufwand für Benutzer bedeuten. Aus diesem Grund scheint es angebracht, eine direkte Übertragung von Datenmengen

zwischen den Konzepten zu ermöglichen. Hierzu werden im Folgenden *Verknüpfungspunkte* in den oben beschriebenen Konzepten für Anfragevisualisierungen definiert. Diese können als Schnittstellen zwischen den Konzepten dienen.

Ein Verknüpfungspunkt markiert entweder eine eingehende oder eine abgehende Menge von Datenelementen in einer Anfragevisualisierung. Mehrere Anfragevisualisierungen können somit konzeptuell direkt verbunden werden, indem eine abgehende Datenmenge aus einer Visualisierung direkt als eingehende Datenmenge für eine andere Visualisierung verwendet wird. Diese konzeptuelle Überführung kann auch technisch umgesetzt werden. Dadurch wird die tatsächliche Filterung mit mehreren Visualisierungen ohne manuelles Übertragen von Daten ermöglicht. Die identifizierten Verknüpfungspunkte der beschriebenen neuen Visualisierungen aus Teil II dieser Arbeit werden nachfolgend erläutert. Zudem sind sie in Tabelle 5.1 zusammengefasst.

Tabelle 5.1: Verknüpfungspunkte der vorgestellten Anfragevisualisierungen.

Visualisierung	eingehend	abgehend
EFFK	Rezeptoren	Emitter
QueryVOWL	Klassenknoten, Literalknoten	Klassenknoten, Literalknoten
Magnetic Querying	gesamte Visualisierung, parametrisierbare Magneten	benutzerdefinierte Bereiche in der Visualisierung
Filter Dials	gesamte Visualisierung, Filterschichtabschnitte	Ergebnisbereichsabschnitte
Aspect Grid	gesamte Visualisierung (Zeilen mit Gruppierungsfunktion)	benutzerdefinierte Ausschnitte aus der Ergebnisliste
VESPa	Knoten	Knoten

5.3.1 Filter/Flow

Es kann direkt auf dem Visualisierungskonzept aufgebaut werden, da Rezeptoren bereits innerhalb der Visualisierung als Verknüpfungspunkte für eingehende Datenmengen dienen. Entsprechend können Emitter als Verknüpfungspunkte für abgehende Datenmengen verwendet werden.

5.3.2 QueryVOWL

Da Klassenknoten eigentlich Mengen aus Datenelementen darstellen, sollten diese auch als Verknüpfungspunkte für eingehende Datenmengen fungieren können. Die Klassenknoten lassen sich gleichermaßen als Quellen für abgehende Datenmengen verwenden. Je nach Beschaffenheit der Daten gilt dasselbe für Literalknoten.

5.3.3 Magnetic Querying

Die Datenelemente sind in dieser Visualisierung potenziell über die gesamte Darstellungsfläche verteilt. Visuell existiert somit kein spezifischer Verknüpfungspunkt für die Partikel. Der Verknüpfungspunkt für eingehende Datenmengen, welche als Magnetpartikel verwendet werden, müsste außerhalb der Visualisierung als Datenquelle angegeben werden. Allerdings können Magneten ebenfalls Verknüpfungspunkte für eingehende Daten anbieten, wenn sich die Magneten durch von außen zugeführte Datenmengen parametrisieren lassen.

Der visuelle Filtereffekt in Magnetic Querying ergibt sich vorrangig aus der unterschiedlichen Positionierung der Datenelemente. Gruppen aus Datenelementen werden also normalerweise darauf basierend ausgewählt, ob sie sich nach Anwendung der magnetischen Anziehungskräfte innerhalb eines bestimmten, benutzerdefinierten Bereichs der Visualisierung befinden. Entsprechend wäre solch ein Bereich auch dazu geeignet, als Verknüpfungspunkt zu dienen, der die Menge der eingeschlossenen Datenelemente bereitstellt.

5.3.4 Filter Dials

Ähnlich wie in Magnetic Querying kann einerseits der gesamte Raum um die Filter Dials herum als Quelle der zu filternden Datenmenge und somit als Verknüpfungspunkt für eingehende Daten angesehen werden. Andererseits lässt sich ein parametrisierbarer Filterschichtabschnitt genauso interpretieren. Ein Abschnitt des Ergebnisbereichs kann als Verknüpfungspunkt für abgehende Datenmengen dienen. Prinzipiell entsprechen die Verknüpfungsmöglichkeiten mit Filterschicht- und Ergebnisbereichsabschnitten den bereits innerhalb des Konzepts genutzten Verknüpfungspunkten zwischen Filter Dials, wie in Abschnitt 3.4.2 beschrieben.

5.3.5 Aspect Grid

In Aspect Grid lässt sich ein Verknüpfungspunkt für eingehende Daten wiederum auf die gesamte Visualisierung bezogen als Quelle der einzelnen

Zeilen erkennen. Dazu muss allerdings für die empfangenen Datenelemente eine Gruppierungsfunktion definiert werden, um die Datenelemente Produkten in der Ergebnisliste zuzuordnen. Verknüpfungspunkte für abgehende Datenmengen können Ausschnitte aus der Ergebnisliste sein. Beispielsweise kann die „Menge der ersten zwanzig Ergebniseinträge“ abhängig von der aktuellen Sortierung weiterverarbeitet werden.

5.3.6 VESPa

Jeder der Knotentypen in VESPa kann sowohl als Verknüpfungspunkt für eingehende als auch für abgehende Datenmengen fungieren. Einerseits kann beispielsweise die Auswahl der Personen, die durch einen Akteur repräsentiert werden können, aus einer anderen Filtervisualisierung bezogen werden. Andererseits kann die ermittelte Menge der Personen, welche die Einschränkungen erfüllen, ebenso zur weiteren Filterung weitergeleitet werden. Gleiches gilt für Ereignisse beziehungsweise die Orte, an denen diese stattfinden.

5.3.7 Verknüpfung mit weiteren Konzepten

Abschließend soll nun gezeigt werden, dass das Konzept der Verknüpfungspunkte sich nicht nur auf die hier vorgestellten Anfragevisualisierungen anwenden lässt. Stattdessen können auch bestehende Visualisierungen dabei eingebunden werden.

VQLs, die auf der Grundlage einer Node-Link-Repräsentation von Objektgraphen mit Platzhaltern arbeiten (Abschnitt 2.3.1.1), können grundsätzlich auf dieselbe Weise wie QueryVOWL Verknüpfungspunkte anbieten. Die Platzhalter können durch Verknüpfungspunkte für eingehende Datenmengen eingeschränkt werden. Ebenso können sie die jeweilige Ergebnismenge über Verknüpfungspunkte für abgehende Datenmengen zur weiteren Filterung weiterleiten. Entsprechend können Mengen in Venn-Diagramm-basierten Visualisierungen (Abschnitt 2.3.3) Verknüpfungspunkte für eingehende Datenmengen darstellen und Abschnitte daraus können Datenmengen zurückgeben.

Gleichermaßen gelten die obigen Überlegungen für EFFK-Graphen für alle in Abschnitt 2.3.1.2 beschriebenen Varianten des Filter/Flow-Konzepts. Ohne das EFFK sind nicht immer explizite Rezeptoren und Emitter vorhanden. Dennoch verfügen Filterknoten konzeptuell stets zumindest über einen Ein- und einen Ausgang.

Tabellenbasierte Konzepte und Konzepte ohne feste Notationselemente bieten zum Teil keine visuell abgegrenzten Verknüpfungspunkte für eingehende Datenmengen. Eingehende Datenmengen müssen somit wiederum als Grundlage für die gesamte Visualisierung herangezogen werden. Verknüp-

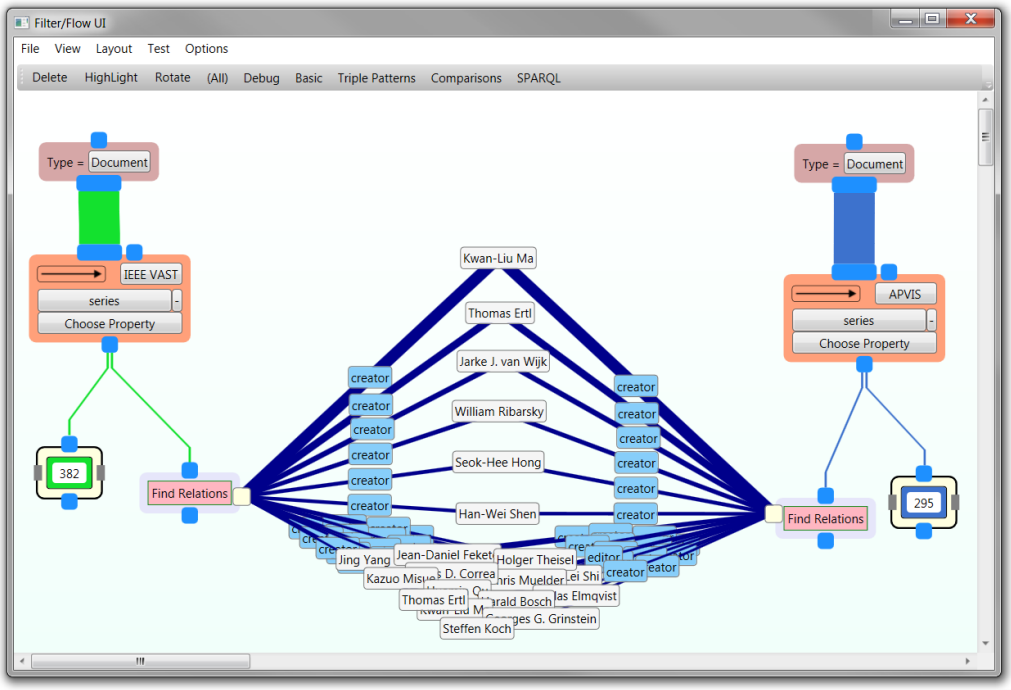


Abbildung 5.3: Die RelFinder-Visualisierung [HHL+09; HLS10], eingebettet in den EFFK-Prototyp: In Faceted DBLP können zwischen allen Veröffentlichungen von der *IEEE VAST* und allen Veröffentlichungen von der *PacificVis* (angezeigt als „APVIS“) einige Verbindungen in Form von Autoren oder Editoren, die auf beiden Konferenzen aktiv waren, gefunden werden. Sechs Autoren mit der größten Anzahl an Publikationen auf beiden Konferenzen sind zur besseren Erkennbarkeit vertikal angeordnet.

fungspunkte für abgehende Datenmengen lassen sich dennoch identifizieren. Beispielweise könnten die Datenmengen aus den einzelnen Tabellenzellen von KMVQL [Huo08] (siehe Seite 29) zur Weiterverarbeitung zurückgegeben werden.

Darüber hinaus können visuelle Bestandteile, die in einer Anfragevisualisierung eigentlich als konkrete Elemente definiert sind, als Verknüpfungspunkte für eingehende Datenmengen dienen. Beispielsweise geht die Visualisierung RelFinder (siehe Seite 21) von mehreren Knoten aus, welche sich jeweils auf genau ein Datenelement beziehen. Daraufhin werden alle bekannten Verbindungen zwischen den betreffenden Datenelementen visualisiert [HHL+09; HLS10]. An Stelle dieser konstanten Knoten lassen sich Platzhalter für Datenmengen setzen – visualisiert werden dann sämtliche Verbindungen zwischen beliebigen Elementen aus den miteinbezogenen Datenmengen. Abbildung 5.3 zeigt eine prototypische Umsetzung dieses Konzepts.

6 Fazit

Durch die stetig zunehmenden Datenmengen erhöht sich die Chance, genau die gesuchten Informationen zu ermitteln. Gleichzeitig wird es jedoch immer schwieriger, diese Informationen gezielt zu finden. Die Organisation der vorhandenen Informationen in Objektgraphen erlaubt eine systematische Filterung nach Attributen von Datenelementen und Zusammenhängen zwischen diesen Datenelementen. Die Erstellung und Bearbeitung der entsprechenden Anfragen mit ihren Filterkriterien ist jedoch für Benutzer mit Problemen verbunden. Eine vielversprechende Möglichkeit, dieser Schwierigkeit zu begegnen, besteht in der visuellen Darstellung der Anfragen.

Zahlreiche Konzepte zu diesem Zweck wurden im Verlauf der vergangenen Jahrzehnte bereits vorgeschlagen. Die bestehenden Konzepte sind teilweise vom Vorhandensein einer Datensammlung abhängig. Zudem eignen sich nicht alle für eine platzsparende Darstellung, welche mit zunehmender Verbreitung mobiler Geräte immer wichtiger wird. Letztendlich haben die unterschiedlichen Anfragevisualisierungskonzepte unterschiedliche Schwerpunkte und Stärken. Es existiert allerdings keine übergreifende Betrachtung, welche die systematische Verbindung mehrerer Visualisierungskonzepte zulässt, um deren jeweilige Stärken gemeinsam zu nutzen.

6.1 Ergebnisse

Nach einer Auseinandersetzung mit bestehenden Arbeiten in Kapitel 2 wurden in Teil II der vorliegenden Arbeit sechs neue beziehungsweise erweiterte Konzepte für die Visualisierung von Suchanfragen vorgestellt. Das Erweiterte Filter/Flow-Modell (EFFK) formalisiert mehrere Verbesserungen des ursprünglichen Filter/Flow-Konzepts, die im Lauf der Zeit in diversen Arbeiten vorgeschlagen wurden, und fügt zudem Erweiterungen hinzu. QueryVOWL erlaubt das Definieren von Graphmustern und lehnt sich an die bereits bestehende Visualisierung VOWL für Ontologien an. Magnetic Querying modifiziert das bestehende Konzept Dust & Magnet für die Unterstüzung von Objektgraphen. Filter Dials lassen die Kombination mehrerer Gruppen von Filterkriterien zugleich zu. Aspect Grid ermöglicht die Priorisierung von Produkten abhängig von einzelnen Aspekten von Produktrezensionen. Letztendlich wurde mit VESPa eine an Objektgraphmuster angelehnte Visualisierung für das Aufspüren bestimmter Ereignissequenzen mehrerer Akteure definiert.

Jede der vorgestellten Visualisierungen erlauben die Darstellung von Suchanfragen auch ohne Vorhandensein einer Datensammlung rein auf Grundlage des Datenschemas. Auf diese Weise lassen sich Hypothesen ausdrücken, die in der Datensammlung nicht zwangsläufig wiederzufinden sind. Ebenso hat dies den Vorteil, dass Anfragen definiert werden können, die in der Zukunft auf unterschiedliche Datensammlungen oder wiederholt auf unterschiedliche zeitabhängige Zustände derselben Datensammlung angewandt werden können. Ist jedoch eine Datensammlung verfügbar, so sind in den vorgestellten Visualisierungen jeweils auch Möglichkeiten vorgesehen, zumindest einen aggregierten Eindruck der Ergebnismenge(n) zu vermitteln, indem beispielsweise deren Größe dargestellt wird.

Für das Filter/Flow-Konzept wurde anschließend angestrebt, eine komplexe Repräsentation von Suchanfragen kompakter zu gestalten. Durch die Erweiterung der Graphstruktur um Rezeptoren und Emitter im EFFK sowie die Zusammenfassung von Subgraphen zu einzelnen Filterknoten kann Platz gespart werden. Wie in einer Benutzerstudie nachgewiesen wurde, bestehen Hinweise darauf, dass die Lesbarkeit durch die kompaktere Darstellung gegenüber der ursprünglichen Variante nicht eingeschränkt wird. Diese Vorgehensweise kann Ansatzpunkte bieten, um Suchanfragen auch auf kleineren Bildschirmen darzustellen, oder zu einem gewissen Grad das Problem der Skalierbarkeit bei Suchanfragen mit zahlreichen separaten Filterkriterien abzuschwächen. Zudem besteht die Möglichkeit, dass sich die Erkenntnisse auch auf andere Anfragevisualisierungen anwenden lassen.

Im Rahmen der Abbildung des EFFK auf SPARQL, aber auch bei der Vorstellung anderer Anfragevisualisierungen wie Magnetic Querying wurden mehrere Ansätze diskutiert, um verschiedene Aspekte von Suchanfragen in den Vordergrund zu rücken. Einerseits scheinen insbesondere flussbasierte Darstellungen für die verständliche Repräsentation von logischen Verknüpfungen geeignet zu sein, während sich Objektbeziehungen über eine Objektgraphstruktur mit Platzhaltern ausdrücken lassen. Durch das Prinzip der Verknüpfungspunkte lassen sich verschiedene Anfragevisualisierungen miteinander kombinieren, um die Stärken der unterschiedlichen Darstellungsformen auszunutzen. Auf diese Weise entsteht die Chance, sowohl logische Verknüpfungen als auch Beziehungen innerhalb von Objektgraphen für Benutzer nachvollziehbar auszudrücken.

6.2 Limitationen und zukünftige Arbeiten

Die in dieser Arbeit vorgestellten Anfragevisualisierungen bieten vielfältige Möglichkeiten zur Formulierung von Suchanfragen. Sie sind dazu jedoch auch auf eine aufgabengerechte Anbindung an Datensammlungen angewiesen, da beispielsweise datentypspezifische Filterknoten für das EFFK zur

Verfügung stehen müssen. Ebenso hängt der erfolgreiche Einsatz komplexer Anfragen auch von der Darstellung der Ergebnisse ab. Um also Filtervisualisierungen wie die vorgestellten nutzbringend einsetzen zu können, muss auch einige Arbeit in andere Komponenten der betreffenden Systeme investiert werden.

Für eine weitere Anpassung an Anwendungsfälle sind zudem zusätzliche Erkenntnisse über den Einsatz von visuellen Suchwerkzeugen notwendig. Da bislang häufig noch mit vergleichsweise rudimentären Mitteln wie Volltextsuche und rein textuellen Tabellen gearbeitet wird, wenn Benutzer mit großen Datenmengen in Berührung kommen, haben selbst die fraglichen Benutzer kaum eine Vorstellung von den Möglichkeiten, die komplexe Anfragen und Visualisierungen davon bieten könnten. Daher ist es zur Zeit auch noch schwierig, überhaupt Schwachstellen für den praktischen Einsatz von Anfragevisualisierungen zu identifizieren, da der reine Gebrauch von Anfragevisualisierungen für Benutzer ohne Visualisierungserfahrung so ungewohnt ist, dass keine zuverlässigen Aussagen über einen routinierten Gebrauch solcher Visualisierungen getroffen werden können. Während für die vorgestellten Techniken also gezeigt werden kann, dass bestimmte komplexe Beziehungen in den Daten, die auch nachvollziehbar für interessant befunden werden können, auffindbar sind, lässt dies dennoch offen, inwieweit die betreffenden Anfragevisualisierungen im täglichen Betrieb noch verbesserungsfähig sind.

Aus der Verwendung von Semantic-Web-Technologien ergeben sich Vorteile wie ein standardisiertes Datenformat, standardisierte Protokolle für die Kommunikation mit Endpunkten für Suchanfragen und die Verknüpfbarkeit von Datensammlungen durch übereinstimmende Bezeichner. Währenddessen treten jedoch auch einige Probleme auf, die außerhalb des Themengebiets der Visualisierung liegen. Eine häufige Schwierigkeit besteht in der niedrigen Verfügbarkeit und langsamen Arbeitsgeschwindigkeit der SPARQL-Endpunkte. Eine Verarbeitungszeit von mehreren Minuten auch für vermeintlich einfache Anfragen ist nicht ungewöhnlich, was eine interaktive Aktualisierung von Visualisierungen unmöglich macht. Ein weiteres Problem ist die Verfügbarkeit entsprechend vernetzter und sauber aufbereiteter Datensammlungen. Zwar ist über das IRI-Konzept des Semantic Web die Möglichkeit gegeben, dass einmal definierte Typen und Eigenschaften in unterschiedlichen Datensammlungen wiederverwendet werden. Unterschiedliche Datensammlungen beziehen sich jedoch nur selten auf dieselben Individuen, und wenn, dann werden auch häufig unterschiedliche IRI-Schemata für die Repräsentation der Individuen eingesetzt, welche erst einzeln zueinander in Bezug gesetzt beziehungsweise transformiert werden müssten. Es bleibt zu hoffen, dass diese Probleme mit der systematischen Übertragung von Datensammlungen in das RDF-Format und der weiteren Entwicklung von SPARQL-Datenbanken mit der Zeit abnehmen.

Anhang

A Bereits veröffentlichte Arbeiten

Während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Visualisierung und Interaktive Systeme habe ich an diversen Publikationen mitgewirkt. Einige bereits veröffentlichte Erkenntnisse aus diesen werden in der vorliegenden Dissertation in einem größeren Kontext zusammengestellt. Im Folgenden soll ein vollständiger Überblick über die vorliegenden Publikationen bereitgestellt werden.

Veröffentlichungen in Konferenzen und Workshops (inklusive Postern und Demos) wurden auf den jeweiligen Veranstaltungen mit einem Vortrag vor einem Publikum vorgestellt. Diejenigen Arbeiten, in denen ich diesen Vortrag nicht selber gehalten habe, sondern durch einen Koautor vertreten wurde, sind mit * markiert.

Artikel in Fachzeitschriften

LNH+16 S. Lohmann, S. Negru, F. Haag und T. Ertl. "Visualizing ontologies with VOWL". In: *Semantic Web Journal* (voraussichtlich 2016). Angenommen am 21.08.2015.

Konferenzbeiträge

Die folgenden Publikationen stellen vollwertige Konferenzbeiträge dar, die einem Peer-Review unterzogen wurden und abgeschlossene Arbeiten behandeln:

HTE11 F. Haag, C. Taras und T. Ertl. "Layout Templates – let users rule user interfaces". In: *Proc. Interfaces and Human Computer Interaction (IHCI '11)*. IADIS, 2011, S. 113–120

NHL13 S. Negru, F. Haag und S. Lohmann. "Towards a unified visual notation for OWL ontologies: insights from a comparative user study". In: *Proc. Semantic Systems (I-SEMANTICS '13)*. ACM, 2013, S. 73–80. DOI: 10.1145/2506182.2506192 *

LNH+14 S. Lohmann, S. Negru, F. Haag und T. Ertl. "VOWL 2: user-oriented visualization of ontologies". In: *Proc. Knowledge Engineering and Knowledge Management (EKAW '14)*. Bd. 8876. LNCS. Springer, 2014, S. 266–281. DOI: 10.1007/978-3-319-13704-9_21 *

HLB+14 F. Haag, S. Lohmann, S. Bold und T. Ertl. "Visual SPARQL querying based on Extended Filter/Flow graphs". In: *Proc. Advanced Visual Interfaces (AVI '14)*. ACM, 2014, S. 305–312. DOI: 10.1145/2598153.2598185

HSE15 F. Haag, T. Schlegel und T. Ertl. "A time-location-based itinerary visualization". In: *Proc. Information Visualization Theory and Applications (IVAPP '15)*. SCITEPRESS, 2015, S. 77–84. DOI: 10.5220/0005199600770084 [1]

HKE16 F. Haag, R. Krüger und T. Ertl. "VESPa: a pattern-based visual query language for event sequences". In: *Proc. Information Visualization Theory and Applications (IVAPP '16)*. Noch nicht erschienen. 2016

Konferenzkurzbeiträge

Die folgenden Publikationen stellen kürzere Arbeiten dar, die einem Peer-Review unterzogen wurden und abgeschlossene, aber weniger umfangreiche Untersuchungen beschreiben:

HLE12 F. Haag, S. Lohmann und T. Ertl. "Simplifying filter/flow graphs by subgraph substitution". In: *Proc. Visual Languages and Human-Centric Computing (VL/HCC '12)*. IEEE, 2012, S. 145–148. DOI: 10.1109/VLHCC.2012.6344501

HLE13b F. Haag, S. Lohmann und T. Ertl. "Evaluating the readability of Extended Filter/Flow graphs". In: *Proc. 2013 Graphics Interface Conference (GI '13)*. CIPS, 2013, S. 33–36

Workshop-Beiträge

Die folgenden Publikationen stellen Beiträge dar, die einem Peer-Review unterzogen wurden und wegen ihres vorläufigen Charakters oder ihres starken Praxisbezugs auf Workshops veröffentlicht wurden:

HRS11 F. Haag, M. Raschke und T. Schlegel. "Ubiquitous alignment". In: *Proc. Workshop on Model-based Interactive Ubiquitous Systems 2011 (MODIQUITOUS '11)*. Bd. 787. CEUR-WS, 2011, S. 33–38

HRE12 F. Haag, M. Raschke und T. Ertl. "Adaptable filter graphs – towards highly-configurable query visualizations". In: *INFORMATIK 2012: Was be-*

[1] Dieser achtseitige Beitrag in einem zweispaltigen Format wurde von der Konferenz selber als *Short Paper* bezeichnet, da *Full Paper* bis zu zwölf Seiten einnehmen konnten. Der Beitrag entspricht aber vom Umfang her den anderen hier aufgeführten Beiträgen voller Länge und wird daher in dieser Liste aufgeführt.

wegt uns in der/die Zukunft? Bd. 208. LNI. Gesellschaft für Informatik e.V. (GI), 2012, S. 1059–1073

HLN+14 F. Haag, S. Lohmann, S. Negru und T. Ertl. "OntoViBe: an ontology visualization benchmark". In: *Proc. Workshop on Visualizations and User Interfaces for Knowledge Engineering and Linked Data Analytics (VOILA '14)*. Bd. 1299. CEUR-WS, 2014, S. 14–27 *

HHJ+14 F. Haag, Q. Han, M. John und T. Ertl. "Aspect Grid: a visualization for iteratively refining aspect-based queries on document collections". In: *INFORMATIK 2014: Big Data – Komplexität meistern*. Bd. 232. LNI. Gesellschaft für Informatik e.V. (GI), 2014, S. 655–660 *

KHH+15 R. Krüger, D. Herr, F. Haag und T. Ertl. "Inspector-Gadget: integrating data preprocessing and orchestration in the visual analysis loop". In: *Proc. EuroVis Workshop on Visual Analytics (EuroVA '15)*. The Eurographics Association, 2015. DOI: 10.2312/eurova.20151096 *

HLS+ F. Haag, S. Lohmann, S. Siek und T. Ertl. "Visual querying of linked data with QueryVOWL". In: *Joint Proceedings of SumPre 2015 and HSWI 2014-15*. Noch nicht erschienen. CEUR-WS

HLN+15 F. Haag, S. Lohmann, S. Negru und T. Ertl. "OntoViBe 2: advancing the ontology visualization benchmark". In: *Proc. Knowledge Engineering and Knowledge Management (EKAW '14) Satellite Events*. Bd. 8982. LNAI. Springer, 2015, S. 83–98. DOI: 10.1007/978-3-319-17966-7_9 *

Poster und Demos

Die folgenden Publikationen stellen Beiträge dar, die als vergleichsweise kurze Beiträge einem Peer-Review unterzogen wurden und auf der jeweiligen Konferenz in Form eines Posters und/oder einer Demo interessierten Konferenzbesuchern vorgestellt wurden:

HLE13a F. Haag, S. Lohmann und T. Ertl. "A Flexible Architecture for Filter/Flow-Based Visual Querying". Graphics Interface (GI '13) Poster Session (unveröffentlicht). 2013

HE14 F. Haag und T. Ertl. "Filter Dials – combine filter criteria, see how much data is available". In: *Proc. Advanced Visual Interfaces (AVI '14)*. ACM, 2014, S. 369–370. DOI: 10.1145/2598153.2600038

HLE14 F. Haag, S. Lohmann und T. Ertl. "SparqlFilterFlow: SPARQL query composition for everyone". In: *The Semantic Web: ESWC 2014 Satellite Events*. Bd. 8798. LNCS. Springer, 2014, S. 362–367. DOI: 10.1007/978-3-319-11955-7_49 [2]*

[2]Beinhaltete eine Demonstration der SparqlFilterFlow-Prototypen.

HLS+15b F. Haag, S. Lohmann, S. Siek und T. Ertl. "QueryVOWL: visual composition of SPARQL queries". In: *The Semantic Web: ESWC 2015 Satellite Events*. Bd. 9341. LNCS. Springer, 2015. DOI: 10.1007/978-3-319-25639-9_12 [3]

Erweiterte eingeladene Beiträge

Die folgenden Publikationen stellen erweiterte Versionen bereits veröffentlichter Publikationen dar. Die ursprünglichen Arbeiten wurden jeweils in Workshops vorgestellt. Es erfolgte eine Einladung zur Veröffentlichung einer erweiterten Fassung. Die erweiterten Versionen wurden keiner erneuten Begutachtung unterzogen und auch nicht in separaten Vorträgen von mir vorgestellt.

HLS+15a F. Haag, S. Lohmann, S. Siek und T. Ertl. "QueryVOWL: a visual query notation for linked data". In: *The Semantic Web: ESWC 2015 Satellite Events*. Bd. 9341. LNCS. Springer, 2015. DOI: 10.1007/978-3-319-25639-9_51

LHN15 S. Lohmann, F. Haag und S. Negru. "Towards a visual notation for OWL: a short summary of VOWL". In: *OWLED '15*. 2015

Buchkapitel

Die folgende Publikation stellt ein Buchkapitel dar, in dem bisherige Erkenntnisse aus dem Themengebiet Multi-Touch zusammengefasst sind:

HBS+13 F. Haag, T. Blascheck, B. Schmitz und M. Raschke. "Berührpunkte mit der Visualisierung". In: *Multi-Touch – Interaktion durch Berührung*. Springer, 2013. Kap. 16, S. 339–367. DOI: 10.1007/978-3-642-36113-5_16

Studentische Arbeiten

Neben den oben aufgelisteten Publikationen, an denen ich direkt mitgewirkt habe, habe ich auch als Betreuer einiger studentischer Arbeiten fungiert. Aus diesen studentischen Arbeiten sind Ideen und Inhalte in diese Dissertation eingeflossen:

AWZ10 O. Alkis, C. Wimmer und F. Zoabi. "IT-Einsatzszenarien zur interaktiven Fahrgastunterstützung". Betreuer: Thomas Schlegel, Michael Raschke, Florian Haag. Fachstudie. Universität Stuttgart, 2010

[3] Beinhaltete eine Demonstration des QueryVOWL-Prototypen.

ABS12 P. Auwärter, P. Berger und O. Sonnauer. “Omnidirektionale Datenvisualisierung – Kollaborative Arbeit auf horizontalen Displays”. Betreuer: Florian Haag. Fachstudie. Universität Stuttgart, 2012

Kni11 M. Knittig. “Entwicklung eines Frontend-Generators für Testanwendungen eines Informationssystems”. Betreuer: Florian Haag. Diplomarbeit. Universität Stuttgart, 2011

Bol13 S. Bold. “Visuelle Filterung von Daten des Semantic Web”. Betreuer: Florian Haag, Steffen Lohmann. Diplomarbeit. Universität Stuttgart, 2013

Fer12 T. Ferber. “Visualisierung von Filter/Flow-Graphen auf mobilen Endgeräten”. Betreuer: Florian Haag. Diplomarbeit. Universität Stuttgart, 2012

Opp14 S. Oppold. “Visuelle Auswahl von Ontologiebestandteilen”. Betreuer: Florian Haag. Bachelorarbeit. Universität Stuttgart, 2014

SSV14 M. Schmidt, S. Simmerling und D. Väth. “In-Situ Visualization of Map Marker Filters on Mobile Devices”. Betreuer: Florian Haag. Projekt-INF. Universität Stuttgart, 2014

Sie14 S. Siek. “VOWL-basierte Visualisierung für SPARQL-Anfragen”. Betreuer: Florian Haag, Steffen Lohmann. Diplomarbeit. Universität Stuttgart, 2014

Raj15 J. Rajamani. “Filter (Dials | ...)” Betreuer: Florian Haag. Masterarbeit. Universität Stuttgart, 2015

Nau15 A. Naumov. “Facettiertes Suchinterface für die explorative Suche in gewichteten Kundenrezensionen”. Betreuer: Florian Haag, Qi Han, Markus John. Bachelorarbeit. Universität Stuttgart, 2015

Vät15 D. Väth. “Query Visualization for Time-Based Graph Data”. Betreuer: Florian Haag. Bachelorarbeit. Universität Stuttgart, 2015

Lin15 V. Link. “Entwicklung eines Benchmark-Generators zum Testen von Ontologievisualisierungen”. Betreuer: Steffen Lohmann, Florian Haag. Bachelorarbeit. Universität Stuttgart, 2015

B Eingesetzte technische Hilfsmittel

Dieser Anhang listet die technischen Werkzeuge auf, die im Rahmen dieses Promotionsvorhabens hauptsächlich zum Einsatz kamen. Diese Auflistung soll einerseits zur Dokumentation dienen, welche zukünftigen Promovierenden unter Umständen bei der Werkzeugwahl behilflich sein kann. Andererseits ist sie als Dank an die Hersteller der genannten Werkzeuge zu verstehen, ohne deren Produkte diese Arbeit nicht möglich gewesen wäre.

Apache Jena zum Ausführen von SPARQL-Anfragen (insbesondere der Triple-Store Fuseki)

Dia für UML- und ER-Diagramme

dotNetRDF für die SPARQL-Anbindung der C#-Prototypen

GhostView zum Betrachten des Dokuments während der Arbeit

Inkscape für die Vektorgrafiken in diesem Dokument

JabRef zur Verwaltung des Literaturverzeichnisses

Microsoft .NET für die Entwicklung und Ausführung der C#-Prototypen

MikTeX für das Setzen dieses Dokuments

Notepad++ für das Verfassen dieses Dokuments

SharpDevelop für die Entwicklung der C#-Prototypen

TortoiseSVN für die Versionsverwaltung der Quelltexte dieses Dokuments

Zudem wurden die folgenden Public-Domain-Grafiken von Wikimedia Commons benutzt:

- `https://commons.wikimedia.org/wiki/File:Civil_Jack_of_the_United_Kingdom.svg` – Landesflagge des Vereinigten Königreichs (verwendet in Abbildung 3.4)
- `https://commons.wikimedia.org/wiki/File:Flag_of_Switzerland.svg` – Landesflagge der Schweiz (verwendet in den Abbildungen 3.5b und 3.6a)

- `https://commons.wikimedia.org/wiki/File:European_Union_map.svg` – Landkarte der Europäischen Union (verwendet in der Abbildung 2.3b)

C Abkürzungen im Literaturverzeichnis

Im Literaturverzeichnis werden an einigen Stellen Abkürzungen für bekannte Organisationen und Ähnliches verwendet. Den meisten Lesern dürften die Abkürzungen vertraut sein und in ihrer abgekürzten Form einen hohen Wiedererkennungswert mit sich bringen. Der Vollständigkeit halber sind die Langformen der Abkürzungen in diesem Anhang dennoch angegeben.

Verlage und Organisationen

ACL: Association for Computational Linguistics

ACS: Australian Computer Society

ACM: Association for Computing Machinery

CEUR-WS: CEUR Workshop Proceedings

CIPS: Canadian Information Processing Society

CSI: Computer Society of India

IEEE: Institute of Electrical and Electronics Engineers

Publikationsorgane

IEEE TVCG: IEEE Transactions on Visualization and Computer Graphics

LNAI: Lecture Notes in Artificial Intelligence (Unterserie von LNCS)

LNCS: Lecture Notes in Computer Science

LNI: Lecture Notes in Informatics

Literatur

[AAF13] N. Andrienko, G. Andrienko und G. Fuchs. “Towards privacy-preserving semantic mobility analysis”. In: *EuroVis Workshop on Visual Analytics (EuroVA ’13)*. Eurographics Association, 2013, S. 19–23.

[ABK+07] S. Auer, C. Bizer, G. Kolibarov, J. Lehmann, R. Cyganiak und Z. Ives. “DBpedia: a nucleus for a web of open data”. In: *International Semantic Web Conference (ISWC ’07): The Semantic Web*. Bd. 4825. LNCS. Springer, 2007, S. 722–735. DOI: 10.1007/978-3-540-76298-0_52.

[ABS12] P. Auwärter, P. Berger und O. Sonnauer. “Omnidirektionale Datenvisualisierung – Kollaborative Arbeit auf horizontalen Displays”. Betreuer: Florian Haag. Fachstudie. Universität Stuttgart, 2012.

[ACK04] N. Athanasis, V. Christophides und D. Kotzinos. “Generating on the fly queries for the semantic web: the ICS-FORTH graphical RQL interface (GRQL)”. English. In: *The Semantic Web – ISWC 2004*. Bd. 3298. LNCS. Springer, 2004, S. 486–501. DOI: 10.1007/978-3-540-30475-3_34.

[ACS90] M. Angelaccio, T. Catarci und G. Santucci. “QBD*: a graphical query language with recursion”. In: *IEEE Transactions on Software Engineering* 16.10 (1990), S. 1150–1163. DOI: 10.1109/32.60295.

[AHK06] J. Abello, F. van Ham und N. Krishnan. “ASK-GraphView: a large scale graph visualization system”. In: *IEEE TVCG* 12.5 (2006), S. 669–676. DOI: 10.1109/TVCG.2006.120.

[Ala03] H. Alani. “TGVizTab: An ontology visualisation extension for Protégé”. Workshop on Visualization Information in Knowledge Engineering (unveröffentlicht). 2003.

[All83] J. F. Allen. “Maintaining knowledge about temporal intervals”. In: *Communications of the ACM* 26.11 (1983), S. 832–843. DOI: 10.1145/182.358434.

[Ama15] Amazon.com, Inc. *Amazon.com*. http://www.amazon.com. 2015.

[AMH10] O. Ambrus, K. Möller und S. Handschuh. “Konduit VQB: a visual query builder for SPARQL on the social semantic desktop”. In: *Proc. Visual Interfaces to the Social and the Semantic Web (VISSW ’10)*. Bd. 565. CEUR-WS. 2010.

[AMT+05] W. Aigner, S. Miksch, B. Thurnher und S. Biffl. “PlanningLines: novel glyphs for representing temporal uncertainties and their evaluation”. In: *Proc. Information Visualisation (IV ’05)*. IEEE, 2005, S. 457–463. DOI: 10.1109/IV.2005.97.

[AP10] G. Allen und J. Parsons. “Is query reuse potentially harmful? anchoring and adjustment in adapting existing database queries”. In: *Information Systems Research* 21.1 (2010), S. 56–77. DOI: 10.1287/isre.1080.0189.

[AS94] C. Ahlberg und B. Shneiderman. “Visual information seeking: tight coupling of dynamic query filters with starfield displays”. In: *Proc. Human Factors in Computing Systems (CHI ’94)*. ACM, 1994, S. 313–317. DOI: 10.1145/191666.191775.

[AWZ10] O. Alkis, C. Wimmer und F. Zoabi. “IT-Einsatzszenarien zur interaktiven Fahrgastunterstützung”. Betreuer: Thomas Schlegel, Michael Raschke, Florian Haag. Fachstudie. Universität Stuttgart, 2010.

[BB10] U. Bojārs und J. Breslin. *SIOC Core Ontology Specification*. http://rdfs.org/sioc/spec/. 2010.

[BBP05] A. Bosca, D. Bonino und P. Pellegrino. “OntoSphere: more than a 3d ontology visualization tool”. In: *Proc. Semantic Web Applications and Perspectives (SWAP ’05)*. Bd. 166. CEUR-WS, 2005.

[BCC+06] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer und D. Sheets. “Tabulator: Exploring and analyzing linked data on the semantic web”. Semantic Web User Interaction Workshop (SWUI ’06) (unveröffentlicht). 2006.

[BCC+91] C. Batini, T. Catarci, M. Costabile und S. Levialdi. “Visual strategies for querying databases”. In: *Proc. Workshop on Visual Languages*. IEEE, 1991, S. 183–189. DOI: 10.1109/WVL.1991.238834.

[BE06] J. Borsje und H. Embregts. *Graphical Query Composition and Natural Language Processing in an RDF Visualization Interface*. Erasmus University Rotterdam. Bachelor thesis. 2006.

[BE10] S. Brody und N. Elhadad. “An unsupervised aspect-sentiment model for online reviews”. In: *Proc. Human Language Technologies: North American Chapter of the Association for Computational Linguistics (NAACL HLT ’10)*. ACL, 2010, S. 804–812.

[BH11] S. Brunk und P. Heim. “Tfacet: hierarchical faceted exploration of semantic data using well-known interaction concepts”. In: *International Workshop on Data-Centric Interactions on the Web (DCI 2011)*. Bd. 817. CEUR-WS. 2011, S. 31–36.

[BH86] D. Bryce und R. Hull. “SNAP: a graphics-based schema manager”. In: *Proc. Data Engineering*. IEEE, 1986, S. 151–164. DOI: 10.1109/icde.1986.7266217.

[BHB09] C. Bizer, T. Heath und T. Berners-Lee. “Linked data – the story so far”. In: *Semantic Services, Interoperability and Web Applications: Emerging Concepts*. IGI Global, 2009, S. 205–227. DOI: 10.4018/jswis.2009081901.

[BIJ01] H. Blau, N. Immerman und D. Jensen. *A Visual Query Language for Relational Knowledge Discovery*. Techn. Ber. University of Massachusetts, 2001.

[BJK+10] T. Boinski, A. Jaworska, R. Kleczkowski und P. Kunowski. “Ontology visualization”. In: *Proc. Information Technology (ICIT ’10)*. IEEE. 2010, S. 17–20.

[BM14] D. Brickley und L. Miller. *FOAF Vocabulary Specification*. http://xmlns.com/foaf/spec/. 2014.

[BNT+08] F. Belleau, M.-A. Nolin, N. Tourigny, P. Rigault und J. Morissette. “Bio2RDF: towards a mashup to build bioinformatics knowledge systems”. In: *Journal of Biomedical Informatics* 41.5 (2008), S. 706–716. DOI: 10.1016/j.jbi.2008.03.004.

[Bol13] S. Bold. “Visuelle Filterung von Daten des Semantic Web”. Betreuer: Florian Haag, Steffen Lohmann. Diplomarbeit. Universität Stuttgart, 2013.

[Bos15] H. Bosch. “Casual Analytics”. Doktorarbeit. Universität Stuttgart, 2015.

[BPL+11] B. Bach, E. Pietriga, I. Liccardi und G. Legostaev. “OntoTrix: a hybrid visualization for populated ontologies”. In: *Proc. Conference Companion on World Wide Web (WWW ’11)*. ACM, 2011, S. 177–180. DOI: 10.1145/1963192.1963283.

[BRZ09] G. Bārzdiņš, S. Rikačovs und M. Zviedris. “Graphical query language as SPARQL frontend”. In: *Proc. Advances in Databases and Information Systems (ABDIS ’09), Workshops and Doctoral Consortium*. Riga Technical University, 2009, S. 93–107.

[BSP+93] E. A. Bier, M. C. Stone, K. Pier, W. Buxton und T. D. DeRose. “Toolglass and Magic Lenses: the see-through interface”. In: *Proc. Computer Graphics and Interactive Techniques (SIGGRAPH ’93)*. ACM, 1993, S. 73–80. DOI: 10.1145/166117.166126.

[BSS+] C. Boscarioli, L. Sánchez García, M. Sfair Sunyé und P. de Bassi. “Towards a Visual Query Tool with Adequate Usability and Communicability to the End User”. http://www.researchgate.net/profile/Laura_Sanchez_Garcia/publication/228768757_Towards_a_Visual_Query_Tool_with_Adequate_Usability_and_Communicability_to_the_End_User/links/09e41513b734497ae8000000.pdf (unveröffentlicht).

[BSW+09] J. Booth, P. Sistla, O. Wolfson und I. F. Cruz. “A data model for trip planning in multimodal transportation systems”. In: *Proc. Extending Database Technology: Advances in Database Technology (EDBT ’09)*. ACM, 2009, S. 994–1005. DOI: 10.1145/1516360.1516474.

[BWW+05] G. Bulter, G. Wang, Y. Wang und L. Zou. “A graph database with visual queries for genomics”. In: *Proc. Asia-Pacific Bioinformatics Conference*. Imperial College Press, 2005, S. 31–40. DOI: 10.1142/9781860947322_0004.

[Car85] F. Carlson. *Picsyms Categorical Dictionary*. Baggeboda Press, 1985.

[CB11] R. Cyganiak und C. Bizer. *Pubby – A Linked Data Frontend for SPARQL Endpoints*. http://wifo5-03.informatik.uni-mannheim.de/pubby/. 2011.

[CC03] L. Chittaro und C. Combi. “Visualizing queries on databases of temporal histories: new metaphors and their evaluation”. In: *Data & Knowledge Engineering* 44.2 (2003), S. 239–264. DOI: 10.1016/S0169-023X(02)00137-4.

[CCD+99] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi und L. Tanca. “XML-GL: a graphical language for querying and restructuring XML documents”. In: *Computer Net-*

works 31.11–16 (1999), S. 1171–1187. DOI: 10.1016/S1389-1286(99)00014-6.

[CCL+97] T. Catarci, M. F. Costabile, S. Levialdi und C. Batini. "Visual query systems for databases: a survey". In: *Journal of Visual Languages and Computing* 8.2 (1997), S. 215–260. DOI: 10.1006/jvlc.1997.0037.

[CCM92] M. P. Consens, I. F. Cruz und A. O. Mendelzon. "Visualizing queries and querying visualizations". In: *SIGMOD Record* 21.1 (1992), S. 39–46. DOI: 10.1145/130868.130874.

[CDD+04] T. Catarci, P. Dongilli, T. Di Mascio, E. Franconi, G. Santucci und S. Tessaris. "An ontology based visual tool for query formulation support". In: *On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops*. Bd. 16. LNCS. Springer, 2004, S. 308–312. DOI: 10.1007/978-3-540-39962-9_15.

[CDN+14] R. Cyganiak, DERI, NUI Galway, D. Wood, 3 Round Stones, M. Lanthaler und Graz University of Technology. *RDF 1.1 Concepts and Abstract Syntax*. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/. 2014.

[CDR+99] M. Czerwinski, S. Dumais, G. Robertson, S. Dziadosz, S. Tiernan und M. van Dantzich. "Visualizing implicit queries for information management and retrieval". In: *Proc. Human Factors in Computing Systems (CHI '99)*. ACM, 1999, S. 560–567. DOI: 10.1145/302979.303158.

[CEC87] D. M. Campbell, D. W. Embley und B. Czejdo. "Graphical query formulation for an entity-relationship model". In: *Data & Knowledge Engineering* 2.2 (1987), S. 89–121. DOI: 10.1016/0169-023X(87)90017-6.

[CFT+08] D. H. Chau, C. Faloutsos, H. Tong, J. Hong, B. Gallagher und T. Eliassi-Rad. "GRAPHITE: a visual query system for large graphs". In: *Proc. Data Mining Workshops (ICDMW '08)*. IEEE, 2008, S. 963–966. DOI: 10.1109/ICDMW.2008.99.

[Cha97] H. C. Chan. "Visual query languages for entity relationship model databases". In: *Journal of Network and Computer Applications* 20.2 (1997), S. 203–221. DOI: 10.1006/jnca.1997.0044.

[CHB09] Y.-X. Chen, M. Hoyer und A. Butz. "CloudMonster: support flexible browsing and searching within music collections". English. In: *Human-Computer Interaction – INTERACT 2009*. Bd. 5726. LNCS. Springer, 2009, S. 428–431. DOI: 10.1007/978-3-642-03655-2_47.

[Che76] P. P.-S. Chen. “The entity-relationship model – toward a unified view of data”. In: *ACM Transactions on Database Systems* 1.1 (1976), S. 9–36. DOI: 10.1145/320434.320440.

[CHM+96] M. J. Carey, L. M. Haas, V. Maganty und J. H. Williams. “PESTO: an integrated query/browser for object databases”. In: *Proc. Very Large Data Bases (VLDB ’96)*. Morgan Kaufmann Publishers Inc., 1996, S. 203–214.

[CL09] P. Castro und A. Lopes. “Magnet Mail: a visualization system for email information retrieval”. English. In: *Proc. Smart Graphics (SG ’09)*. Bd. 5531. LNCS. Springer, 2009, S. 213–222. DOI: 10.1007/978-3-642-02115-2_18.

[CM90] M. P. Consens und A. O. Mendelzon. “GraphLog: a visual formalism for real life recursion”. In: *Proc. Principles of Database Systems (PODS ’90)*. ACM, 1990, S. 404–416. DOI: 10.1145/298514.298591.

[CM93] M. Consens und A. Mendelzon. “Hy+: a hygraph-based query and visualization system”. In: *SIGMOD Records* 22.2 (1993), S. 511–516. DOI: 10.1145/170036.171537.

[CM94] D. Calcinelli und M. Mainguenaud. “Cigales, a visual query language for a geographical information system: the user interface”. In: *Journal of Visual Languages & Computing* 5.2 (1994), S. 113–132. DOI: 10.1006/jvlc.1994.1006.

[CMW87] I. F. Cruz, A. O. Mendelzon und P. T. Wood. “A graphical query language supporting recursion”. In: *SIGMOD Records* 16.3 (1987), S. 323–330. DOI: 10.1145/38714.38749.

[CO12] C. Combi und B. Oliboni. “Visually defining and querying consistent multi-granular clinical temporal abstractions”. In: *Artificial Intelligence in Medicine* 54.2 (2012), S. 75–101. DOI: 10.1016/j.artmed.2011.10.004.

[CRB10] Y.-X. Chen, M. Reiter und A. Butz. “PhotoMagnets: supporting flexible browsing and searching in photo collections”. In: *Proc. Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction (ICMI-MLMI ’10)*. ACM, 2010. DOI: 10.1145/1891903.1891936.

[Cru92] I. F. Cruz. “DOODLE: a visual language for object-oriented databases”. In: *SIGMOD Record* 21.2 (1992), S. 71–80. DOI: 10.1145/141484.130299.

[CS95] T. Catarci und G. Santucci. “Are visual query languages easier to use than traditional ones? an experimental proof”. In: *People and Computers X, Proceedings of HCI ’95*. Cambridge University Press, 1995, S. 323–338.

[CSA93] T. Catarci, G. Santucci und M. Angelaccio. “Fundamental graphical primitives for visual query languages”. In: *Information Systems* 18.2 (1993), S. 75–98. DOI: 10.1016/0306-4379(93)90006-M.

[DB08] J. Diederich und W.-T. Balke. “FacetedDBLP – navigational access for digital libraries”. In: *Bulletin of the IEEE Technical Committee on Digital Libraries* 4.1 (2008).

[DBp15] DBpedia Association. *Data Set 2014 | DBpedia*. 2015. URL: http://wiki.dbpedia.org/data-set-2014.

[DC96] J. D. Dionisio und A. F. Cárdenas. “MQuery: a visual query language for multimedia, timeline and simulation data”. In: *Journal of Visual Languages & Computing* 7.4 (1996), S. 377–401. DOI: 10.1006/jvlc.1996.0020.

[DC99] S. DeRose und J. Clark. *XML Path Language (XPath) Version 1.0*. W3C Recommendation. W3C, 1999. URL: http://www.w3.org/TR/1999/REC-xpath-19991116.

[Dea01] M. Dean. *CIA World Fact Book in DAML*. http://www.daml.org/2001/12/factbook/. 2001.

[Del10] N. Dell. *VIQUEN: A visual query engine for RDF*. Techn. Ber. Computer Science und Engineering, University of Washington, 2010.

[DGJ+95] S. Dar, N. Gehani, H. Jagadish und J. Srinivasan. “Queries in an object-oriented graphical interface”. In: *Journal of Visual Languages & Computing* 6.1 (1995), S. 27–52. DOI: 10.1006/jvlc.1995.1003.

[DKS07] L. Deligiannidis, K. J. Kochut und A. P. Sheth. “RDF data exploration and visualization”. In: *Proc. Workshop on CyberInfrastructure: Information Management in eScience (CIMS ’07)*. ACM, 2007, S. 39–46. DOI: 10.1145/1317353.1317362.

[DLE+11] L. Ding, T. Lebo, J. S. Erickson, D. DiFranzo, G. T. Williams, X. Li, J. Michaelis, A. Graves, J. G. Zheng, Z. Shangguan, J. Flores, D. L. McGuinness und J. A. Hendler. “{twc} logd: a portal for linked open government data ecosystems”. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 9.3 (2011), S. 325–333. DOI: 10.1016/j.websem.2011.06.002.

[DLR12] Q. X. Do, W. Lu und D. Roth. “Joint inference for event timeline construction”. In: *Proc. Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL ’12)*. ACL, 2012, S. 677–687.

[DP05] D. Dotan und R. Y. Pinter. “Hyperflow: an integrated visual query and dataflow language for end-user information analysis”. In: *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*. 2005, S. 27–34. DOI: 10.1109/VLHCC.2005.45.

[DPH12] L. Ding, V. Peristeras und M. Hausenblas. “Linked open government data [guest editors’ introduction]”. In: *IEEE Intelligent Systems* 27.3 (2012), S. 11–15. DOI: 10.1109/MIS.2012.56.

[DS05] M. Duerst und M. Suignard. *RFC 3987: Internationalized Resource Identifiers (IRIs)*. RFC 3987 (Proposed Standard). Internet Engineering Task Force, 2005. URL: http://www.ietf.org/rfc/rfc3987.txt.

[DSB+04] M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider und L. A. Stein. *OWL Web Ontology Language Reference*. http://www.w3.org/TR/owl-ref/. 2004.

[DVF+10] H. F. Deus, D. F. Veiga, P. R. Freire, J. N. Weinstein, G. B. Mills und J. S. Almeida. “Exposing the cancer genome atlas as a SPARQL endpoint”. In: *Journal of Biomedical Informatics* 43.6 (2010), S. 998–1008. DOI: 10.1016/j.jbi.2010.09.004.

[DVZ95] A. Del Bimbo, E. Vicario und D. Zingoni. “Symbolic description and visual querying of image sequences using spatio-temporal logic”. In: *IEEE TKDE* 7.4 (1995), S. 609–622. DOI: 10.1109/69.404033.

[EDG+08] N. Elmqvist, T.-N. Do, H. Goodell, N. Henry und J. Fekete. “ZAME: interactive large-scale graph visualization”. In: *Proc. Pacific Visualization Symposium (PacificVIS ’08)*. IEEE, 2008, S. 215–222. DOI: 10.1109/PACIFICVIS.2008.4475479.

[ENV+07] O. Edsberg, S. J. Nordbø, E. Vinnes und Ø. Nytrø. “Design and evaluation of a temporal, graph-based language for querying collections of patient histories”. In: *Proc. Health (Medical) Informatics (MEDINFO ’07)*. Bd. 129. Studies in Health Technology and Informatics. IOS Press, 2007, S. 402–406.

[EST08] N. Elmqvist, J. Stasko und P. Tsigas. “DataMeadow: a visual canvas for analysis of large-scale multivariate data”. In: *Information Visualization* 7.1 (2008), S. 18–33. DOI: 10.1145/1391107.1391110.

[EW13] S. van den Elzen und J. J. van Wijk. “Small multiples, large singles: a new approach for visual data exploration”. In: *Computer Graphics Forum* 32.3pt2 (2013), S. 191–200. DOI: 10.1111/cgf.12106.

[FC00] S. Fernandes Silva und T. Catarci. “Visualization of linear time-oriented data: a survey”. In: *Proc. Web Information Systems Engineering (WISE ’00)*. IEEE, 2000, S. 310–319. DOI: 10.1109/WISE.2000.882407.

[Feg99] L. Fegeras. “VOODOO: a visual object-oriented database language for ODMG OQL”. In: *W13. The First ECOOP Workshop on Object-Oriented Databases*. 1999.

[Fer12] T. Ferber. “Visualisierung von Filter/Flow-Graphen auf mobilen Endgeräten”. Betreuer: Florian Haag. Diplomarbeit. Universität Stuttgart, 2012.

[FH06] A. Fadhil und V. Haarslev. “GLOO: a graphical query language for OWL ontologies”. In: *Proc. Workshop on OWL: Experiences and Directions (OWLED ’06)*. Bd. 216. CEUR-WS, 2006.

[FH07] A. Fadhil und V. Haarslev. “OntoVQL: a graphical query language for OWL ontologies”. In: *Proc. Workshop on Description Logics (DL’07)*. Bozen-Bolzano University Press, 2007, S. 267–274.

[FKS+06] J. Fails, A. Karlson, L. Shahamat und B. Shneiderman. “A visual interface for multivariate temporal data: finding patterns of events across multiple histories”. In: *Proc. Visual Analytics Science And Technology (VAST ’06)*. IEEE, 2006, S. 167–174. DOI: 10.1109/VAST.2006.261421.

[FMK12] F. Fischer, F. Mansmann und D. A. Keim. “Real-time visual analytics for event data streams”. In: *Proc. Applied Computing (SAC ’12)*. ACM, 2012, S. 801–806. DOI: 10.1145/2245276.2245432.

[Fou95] D. Foulser. “IRIS Explorer: a framework for investigation”. In: *SIGGRAPH Computer Graphics* 29.2 (1995), S. 13–16. DOI: 10.1145/204362.204365.

[FSC97] S. Fernandes, U. Schiel und T. Catarci. “Visual query operators for temporal databases”. In: *Proc. Workshop on Temporal Representation and Reasoning (TIME ’97)*. IEEE, 1997, S. 46–53. DOI: 10.1109/TIME.1997.600781.

[FTH06] F. Frasincar, A. Telea und G.-J. Houben. “Adapting graph visualization techniques for the visualization of rdf data”. English. In: *Visualizing the Semantic Web*. Springer, 2006, S. 154–171. DOI: 10.1007/1-84628-290-X_9.

[GB14] R. Guha und D. Brickley. *RDF Schema 1.1*. http://www.w3.org/TR/2014/REC-rdf-schema-20140225/. W3C Recommendation. 2014.

[GBG04] W. Gaaloul, S. Bhiri und C. Godart. “Discovering workflow transactional behavior from event-based log”. English. In: *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*. Bd. 3290. LNCS. Springer, 2004, S. 3–18. DOI: 10.1007/978-3-540-30468-5_3.

[GDC+90] S. L. Greene, S. J. Devlin, P. E. Cannata und L. M. Gomez. “No IFs, ANDs, or ORs: a study of database querying”. In: *International Journal of Man-Machine Studies* 32.3 (1990), S. 303–326. DOI: 10.1016/S0020-7373(08)80005-3.

[GDH+09] T. Groza, L. Drăgan, S. Handschuh und S. Decker. “Bridging the gap between linked data and the semantic desktop”. English. In: *International Semantic Web Conference (ISWC ’09): The Semantic Web*. Bd. 5823. LNCS. Springer, 2009, S. 827–842. DOI: 10.1007/978-3-642-04930-9_52.

[GDH08] J. Goodwin, C. Dolbear und G. Hart. “Geographical linked data: the administrative geography of great britain on the semantic web”. In: *Transactions in GIS* 12 (2008), S. 19–30. DOI: 10.1111/j.1467-9671.2008.01133.x.

[GFM+11] M. A. Gallego, J. D. Fernández, M. A. Martínez-Prieto und P. de la Fuente. “RDF visualization using a three-dimensional adjacency matrix”. Semantic Search Workshop (SemSearch ’11) (unveröffentlicht). 2011.

[GGS+09] J. Groppe, S. Groppe, A. Schleifer und V. Linnemann. “LuposDate: a semantic web database system”. In: *Proc. Information and Knowledge Management (CIKM ’09)*. ACM, 2009, S. 2083–2084. DOI: 10.1145/1645953.1646313.

[GGS11] J. Groppe, S. Groppe und A. Schleifer. “Visual query system for analyzing social semantic web”. In: *Proc. Companion on World Wide Web (WWW ’11)*. ACM, 2011, S. 217–220. DOI: 10.1145/1963192.1963293.

[Grü99] B. Grünbaum. “The search for symmetric Venn diagrams”. In: *Geombinatorics* 8 (1999), S. 104–109.

[GS02] R. Giugno und D. Shasha. “GraphGrep: a fast and universal method for querying graphs”. In: *Proc. Pattern Recognition (ICPR ’02)*. Bd. 2. IEEE, 2002, S. 112–115. DOI: 10.1109/ICPR.2002.1048250.

[GS14] D. Gotz und H. Stavropoulos. “DecisionFlow: visual analytics for high-dimensional temporal event sequence data”. In: *IEEE TVCG* 20.12 (2014), S. 1783–1792. DOI: 10.1109/TVCG.2014.2346682.

[HB14] A. H. und V. Balasubramanian. “A model independent and user-friendly querying system for indoor spaces”. In: *Proc. Management of Data (COMAD ’14)*. CSI, 2014, S. 17–28.

[HBS+13] F. Haag, T. Blascheck, B. Schmitz und M. Raschke. “Berührpunkte mit der Visualisierung”. In: *Multi-Touch – Interaktion durch Berührung*. Springer, 2013. Kap. 16, S. 339–367. DOI: 10.1007/978-3-642-36113-5_16.

[HC09] O. Hassanzadeh und M. P. Consens. “Linked Movie Data Base”. In: *Proc. Linked Data on the Web (LDOW ’09)*. Bd. 538. CEUR-WS, 2009.

[HE14] F. Haag und T. Ertl. “Filter Dials – combine filter criteria, see how much data is available”. In: *Proc. Advanced Visual Interfaces (AVI ’14)*. ACM, 2014, S. 369–370. DOI: 10.1145/2598153.2600038.

[Hea95] M. A. Hearst. “TileBars: visualization of term distribution information in full text information access”. In: *Proc. Human Factors in Computing Systems (CHI ’95)*. ACM, 1995, S. 59–66. DOI: 10.1145/223904.223912.

[HFM07] N. Henry, J. Fekete und M. McGuffin. “NodeTrix: a hybrid visualization of social networks”. In: *IEEE TVCG* 13.6 (2007), S. 1302–1309. DOI: 10.1109/TVCG.2007.70582.

[HHJ+14] F. Haag, Q. Han, M. John und T. Ertl. “Aspect Grid: a visualization for iteratively refining aspect-based queries on document collections”. In: *INFORMATIK 2014: Big Data – Komplexität meistern*. Bd. 232. LNI. Gesellschaft für Informatik e.V. (GI), 2014, S. 655–660.

[HHL+09] P. Heim, S. Hellmann, J. Lehmann, S. Lohmann und T. Stegemann. "RelFinder: revealing relationships in RDF knowledge bases". In: *Proc. Semantic and Digital Media Technologies (SAMT '09)*. Bd. 5887. LNCS. Springer, 2009, S. 182–187. DOI: 10.1007/978-3-642-10543-2_21.

[HHR+09] M. Hausenblas, W. Halb, Y. Raimond, L. Feigenbaum und D. Ayers. "SCOVO: using statistics on the web of data". English. In: *The Semantic Web: Research and Applications*. Bd. 5554. LNCS. Springer, 2009, S. 708–722. DOI: 10.1007/978-3-642-02121-3_52.

[HKD06] A. Harth, S. R. Kruk und S. Decker. "Graphical representation of RDF queries". In: *Proc. World Wide Web (WWW '06)*. ACM, 2006, S. 859–860. DOI: 10.1145/1135777.1135914.

[HKE16] F. Haag, R. Krüger und T. Ertl. "VESPa: a pattern-based visual query language for event sequences". In: *Proc. Information Visualization Theory and Applications (IVAPP '16)*. Noch nicht erschienen. 2016.

[HLB+14] F. Haag, S. Lohmann, S. Bold und T. Ertl. "Visual SPARQL querying based on Extended Filter/Flow graphs". In: *Proc. Advanced Visual Interfaces (AVI '14)*. ACM, 2014, S. 305–312. DOI: 10.1145/2598153.2598185.

[HLE12] F. Haag, S. Lohmann und T. Ertl. "Simplifying filter/flow graphs by subgraph substitution". In: *Proc. Visual Languages and Human-Centric Computing (VL/HCC '12)*. IEEE, 2012, S. 145–148. DOI: 10.1109/VLHCC.2012.6344501.

[HLE13a] F. Haag, S. Lohmann und T. Ertl. "A Flexible Architecture for Filter/Flow-Based Visual Querying". Graphics Interface (GI '13) Poster Session (unveröffentlicht). 2013.

[HLE13b] F. Haag, S. Lohmann und T. Ertl. "Evaluating the readability of Extended Filter/Flow graphs". In: *Proc. 2013 Graphics Interface Conference (GI '13)*. CIPS, 2013, S. 33–36.

[HLE14] F. Haag, S. Lohmann und T. Ertl. "SparqlFilterFlow: SPARQL query composition for everyone". In: *The Semantic Web: ESWC 2014 Satellite Events*. Bd. 8798. LNCS. Springer, 2014, S. 362–367. DOI: 10.1007/978-3-319-11955-7_49.

[HLN+14] F. Haag, S. Lohmann, S. Negru und T. Ertl. "OntoViBe: an ontology visualization benchmark". In: *Proc. Workshop on Visualizations and User Interfaces for Knowledge Engineering and Linked Data Analytics (VOILA '14)*. Bd. 1299. CEUR-WS, 2014, S. 14–27.

[HLN+15] F. Haag, S. Lohmann, S. Negru und T. Ertl. “OntoViBe 2: advancing the ontology visualization benchmark”. In: *Proc. Knowledge Engineering and Knowledge Management (EKAW ’14) Satellite Events*. Bd. 8982. LNAI. Springer, 2015, S. 83–98. DOI: 10.1007/978-3-319-17966-7_9.

[HLS+] F. Haag, S. Lohmann, S. Siek und T. Ertl. “Visual querying of linked data with QueryVOWL”. In: *Joint Proceedings of SumPre 2015 and HSWI 2014-15*. Noch nicht erschienen. CEUR-WS.

[HLS+15a] F. Haag, S. Lohmann, S. Siek und T. Ertl. “QueryVOWL: a visual query notation for linked data”. In: *The Semantic Web: ESWC 2015 Satellite Events*. Bd. 9341. LNCS. Springer, 2015. DOI: 10.1007/978-3-319-25639-9_51.

[HLS+15b] F. Haag, S. Lohmann, S. Siek und T. Ertl. “QueryVOWL: visual composition of SPARQL queries”. In: *The Semantic Web: ESWC 2015 Satellite Events*. Bd. 9341. LNCS. Springer, 2015. DOI: 10.1007/978-3-319-25639-9_12.

[HLS10] P. Heim, S. Lohmann und T. Stegemann. “Interactive relationship discovery via the semantic web”. In: *Proc. Extended Semantic Web Conference (ESWC ’10)*. Bd. 6088. LNCS. Springer, 2010, S. 303–317. DOI: 10.1007/978-3-642-13486-9_21.

[HMF+10] F. Hogenboom, V. Milea, F. Frasincar und U. Kaymak. “RDF-GL: a SPARQL-based graphical query language for RDF”. English. In: *Emergent Web Intelligence: Advanced Information Retrieval*. Bd. Nicht nummeriert. AIKP. Springer, 2010, S. 87–116. DOI: 10.1007/978-1-84996-074-8_4.

[HR95] S. Hibino und E. Rundensteiner. “A visual query language for identifying temporal trends in video data”. In: *Proc. Workshop on Multi-Media Database Management Systems (IW-MMDBMS ’95)*. IEEE, 1995, S. 74–81. DOI: 10.1109/MMDBMS.1995.520425.

[HRE12] F. Haag, M. Raschke und T. Ertl. “Adaptable filter graphs – towards highly-configurable query visualizations”. In: *INFORMATIK 2012: Was bewegt uns in der/die Zukunft?* Bd. 208. LNI. Gesellschaft für Informatik e.V. (GI), 2012, S. 1059–1073.

[HRS11] F. Haag, M. Raschke und T. Schlegel. “Ubiquitous alignment”. In: *Proc. Workshop on Model-based Interactive Ubiquitous Systems 2011 (MODIQUITOUS ’11)*. Bd. 787. CEUR-WS, 2011, S. 33–38.

[HS01] H. Hochheiser und B. Shneiderman. “Interactive exploration of time series data”. English. In: *Discovery Science*. Bd. 2226. LNCS. Springer, 2001, S. 441–446. DOI: 10.1007/3-540-45650-3_38.

[HS04] H. Hochheiser und B. Shneiderman. “Dynamic query tools for time series data sets: timebox widgets for interactive exploration”. In: *Information Visualization* 3.1 (2004), S. 1–18. DOI: 10.1057/palgrave.ivs.9500061.

[HSE15] F. Haag, T. Schlegel und T. Ertl. “A time-location-based itinerary visualization”. In: *Proc. Information Visualization Theory and Applications (IVAPP ’15)*. SCITEPRESS, 2015, S. 77–84. DOI: 10.5220/0005199600770084.

[HSM+06] T. Hansaki, B. Shizuki, K. Misue und J. Tanaka. “FindFlow: visual interface for information search based on intermediate results”. In: *Proc. Asia-Pacific Symposium on Information Visualisation (APVIS ’06)*. ACS, 2006, S. 147–152.

[HTE11] F. Haag, C. Taras und T. Ertl. “Layout Templates – let users rule user interfaces”. In: *Proc. Interfaces and Human Computer Interaction (IHCI ’11)*. IADIS, 2011, S. 113–120.

[Huo08] J. Huo. “KMVQL: a visual query interface based on Karnaugh map”. In: *Proc. Advanced Visual Interfaces (AVI ’08)*. ACM, 2008, S. 243–250. DOI: 10.1145/1385569.1385609.

[HVA08] T. Hulsen, J. de Vlieg und W. Alkema. “BioVenn – a web application for the comparison and visualization of biological lists using area-proportional Venn diagrams”. In: *BMC Genomics* 9.488 (2008). DOI: 10.1186/1471-2164-9-488.

[HVG14] S. Heggestøyl, G. Vega-Gorgojo und M. Giese. “Visual query formulation for linked open data: the Norwegian entity registry case”. In: *Norsk Informatikkonferanse (NIK ’14)*. 2014.

[HZL08] P. Heim, J. Ziegler und S. Lohmann. “gFacet: a browser for the web of data”. In: *Proc. International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW ’08)*. Bd. 417. CEUR-WS, 2008, S. 49–58.

[Ins02] G. Insana. *MediaGlyphs Project*. http://mediaglyphs.org. 2002.

[ISO99] ISO/IEC JTC 1/SC 32 Data management and interchange. *Information technology – Database languages – SQL*. ISO/IEC, 1999. URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=16663.

[Jac09] K. Jacobson. *MySpO: MySpace Ontology*. http://grasstunes.net/ontology/myspace/myspace.html. 2009.

[JBC+12] C. Jin, S. S. Bhowmick, B. Choi und S. Zhou. "PRAGUE: towards blending practical visual subgraph query formulation and query processing". In: *Proc. Data Engineering (ICDE '12)*. IEEE, 2012, S. 222–233. DOI: 10.1109/ICDE.2012.49.

[JBX+10] C. Jin, S. S. Bhowmick, X. Xiao, J. Cheng und B. Choi. "GBLENDER: towards blending visual query formulation and query processing in graph databases". In: *Proc. Management of Data (SIGMOD '10)*. ACM, 2010, S. 111–122. DOI: 10.1145/1807167.1807182.

[JD08] M. Jarrar und M. D. Dikaiakos. "MashQL: a query-by-diagram topping SPARQL". In: *Proc. Workshop on Ontologies and Information Systems for the Semantic Web (ONISW '08)*. ACM, 2008, S. 89–96. DOI: 10.1145/1458484.1458499.

[JE13] W. Javed und N. Elmqvist. "ExPlates: spatializing interactive analysis to scaffold visual exploration". In: *Computer Graphics Forum* 32.3pt4 (2013), S. 441–450. DOI: 10.1111/cgf.12131.

[JGZ+11] H.-C. Jetter, J. Gerken, M. Zöllner, H. Reiterer und N. Milic-Frayling. "Materializing the query with facet-streams: a hybrid surface for collaborative search on tabletops". In: *Proc. Human Factors in Computing Systems (CHI '11)*. ACM, 2011, S. 3013–3022. DOI: 10.1145/1978942.1979390.

[JHS02] J. A. Jones, M. J. Harrold und J. Stasko. "Visualization of test information to assist fault localization". In: *Proc. Software Engineering (ICSE '02)*. ACM, 2002, S. 467–477. DOI: 10.1145/581339.581397.

[JNS00] K. Järvelin, T. Niemi und A. Salminen. "The visual query language CQL for transitive and relational computation". In: *Data & Knowledge Engineering* 35.1 (2000), S. 39–51. DOI: 10.1016/S0169-023X(00)00021-5.

[Jon98] S. Jones. “Graphical query specification and dynamic result previews for a digital library”. In: *Proc. User Interface Software and Technology (UIST ’98)*. ACM, 1998, S. 143–151. DOI: 10.1145/288392.288595.

[JS09] J. Jin und P. Szekely. “QueryMarvel: a visual query language for temporal patterns using comic strips”. In: *Proc. Visual Languages and Human-Centric Computing (VL/HCC ’09)*. IEEE, 2009, S. 207–214. DOI: 10.1109/VLHCC.2009.5295262.

[JS10] J. Jin und P. Szekely. “Interactive querying of temporal data using a comic strip metaphor”. In: *Proc. Visual Analytics Science and Technology (VAST ’10)*. IEEE, 2010, S. 163–170. DOI: 10.1109/VAST.2010.5652890.

[Kap84] S. J. Kaplan. “Designing a portable natural language database query system”. In: *ACM Transactions on Database Systems* 9.1 (1984), S. 1–19. DOI: 10.1145/348.318584.

[KBG+09] S. Koch, H. Bosch, M. Giereth und T. Ertl. “Iterative integration of visual insights during patent search and analysis”. In: *Proc. Visual Analytics Science and Technology (VAST ’09)*. IEEE, 2009, S. 203–210. DOI: 10.1109/VAST.2009.5333564.

[KBH06] R. Kosara, F. Bendix und H. Hauser. “Parallel Sets: interactive exploration and visual analysis of categorical data”. In: *IEEE TVCG* 12.4 (2006), S. 558–568. DOI: 10.1109/TVCG.2006.76.

[KBS14] C. Keller, S. Brunk und T. Schlegel. “Introducing the public transport domain to the web of data”. English. In: *Proc. Web Information Systems Engineering (WISE ’04)*. Bd. 8787. LNCS. Springer, 2014, S. 521–530. DOI: 10.1007/978-3-319-11746-1_38.

[KCH10] N. W. Kim, S. K. Card und J. Heer. “Tracing genealogical data with TimeNets”. In: *Proc. Advanced Visual Interfaces (AVI ’10)*. ACM, 2010, S. 241–248. DOI: 10.1145/1842993.1843035.

[KCL+15] M. Kolchin, A. Chistyakov, M. Lapaev und R. Khaydarova. “FOODpedia: Russian food products as a linked data dataset”. In: *The Semantic Web: ESWC 2015 Satellite Events*. Bd. 9341. CCNT. Springer, 2015, S. 87–90. DOI: 10.1007/978-3-319-25639-9_17.

[KEC06] R. Keller, C. M. Eckert und P. J. Clarkson. “Matrices or node-link diagrams: which visual representation is better for visualising connectivity models?” In: *Information Visualization* 5.1 (2006), S. 62–76. DOI: 10.1057/palgrave.ivs.9500116.

[KG95] V. Kouramajian und M. Gertz. “A graphical query language for temporal databases”. English. In: *OOER ’95 Object-Oriented and Entity-Relationship Modeling*. Bd. 1021. LNCS. Springer, 1995, S. 388–399. DOI: 10.1007/BFb0020549.

[KHH+15] R. Krüger, D. Herr, F. Haag und T. Ertl. “Inspector-Gadget: integrating data preprocessing and orchestration in the visual analysis loop”. In: *Proc. EuroVis Workshop on Visual Analytics (EuroVA ’15)*. The Eurographics Association, 2015. DOI: 10.2312/eurova.20151096.

[KHS02] E. Keogh, H. Hochheiser und B. Shneiderman. “An augmented visual query mechanism for finding patterns in time series data”. English. In: *Flexible Query Answering Systems*. Bd. 2522. LNCS. Springer, 2002, S. 240–250. DOI: 10.1007/3-540-36109-X_19.

[KK94] D. Keim und H.-P. Kriegel. “VisDB: database exploration using multidimensional visualization”. In: *IEEE Computer Graphics and Applications* 14.5 (1994), S. 40–49. DOI: 10.1109/38.310723.

[KK96] D. Keim und H.-P. Kriegel. “Visualization techniques for mining large databases: a comparison”. In: *IEEE Transactions on Knowledge and Data Engineering* 8.6 (1996), S. 923–938. DOI: 10.1109/69.553159.

[KKS88] H.-J. Kim, H. F. Korth und A. Silberschatz. “PICASSO: a graphical query language”. In: *Software: Practice and Experience* 18.3 (1988), S. 169–203. DOI: 10.1002/spe.4380180302.

[KM89] M. Kuntz und R. Melchert. “Pasta-3’s graphical query language: direct manipulation cooperative queries, full expressive power”. In: *Proc. Very Large Data Bases (VLDB ’89)*. Morgan Kaufmann Publishers Inc., 1989, S. 97–105.

[KM93] J. C. Kwak und S. Moon. “Object query diagram: an extended query graph for object-oriented databases”. In: *Proc. Visual Languages*. IEEE, 1993, S. 44–48. DOI: 10.1109/VL.1993.269577.

[Kni11] M. Knittig. “Entwicklung eines Frontend-Generators für Testanwendungen eines Informationssystems”. Betreuer: Florian Haag. Diplomarbeit. Universität Stuttgart, 2011.

[Koc06] C. Koch. “A Visual Query Language for Complex-Value Databases”. (unveröffentlicht). 2006. URL: http://arxiv.org/abs/cs/0602006.

[KS90] K. Kunkel und T. Strothotte. “Visualization and direct manipulation in user interfaces: are we overdoing it?” English. In: *Visualization in Human-Computer Interaction*. Bd. 439. LNCS. Springer, 1990, S. 183–193. DOI: 10.1007/3-540-52698-6_11.

[KSS04] D. A. Keim, J. Schneidewind und M. Sips. “CircleView: a new approach for visualizing time-related multidimensional data sets”. In: *Proc. Advanced Visual Interfaces (AVI '04)*. ACM, 2004, S. 179–182. DOI: 10.1145/989863.989891.

[KTE14] R. Krüger, D. Thom und T. Ertl. “Visual analysis of movement behavior using web data for context enrichment”. In: *Proc. Pacific Visualization Symposium (PacificVic '14)*. IEEE, 2014, S. 193–200. DOI: 10.1109/pacificvis.2014.57.

[KTW+13] R. Krüger, D. Thom, M. Wörner, H. Bosch und T. Ertl. “TrajectoryLenses – a set-based filtering and exploration technique for long-term trajectory data”. In: *Computer Graphics Forum* 32.3pt4 (2013), S. 451–460. DOI: 10.1111/cgf.12132.

[KW05] T. Kapler und W. Wright. “GeoTime information visualization”. In: *Information Visualization* 4.2 (2005), S. 136–146. DOI: 10.1057/palgrave.ivs.9500097.

[KWV07] S. Krivov, R. Williams und F. Villa. “GrOWL: a tool for visualization and editing of OWL ontologies”. In: *Web Semantics* 5.2 (2007), S. 54–57. DOI: 10.1016/j.websem.2007.03.005.

[LAB+14] T. Lammarsch, W. Aigner, A. Bertone, S. Miksch und A. Rind. “Mind the time: unleashing temporal aspects in pattern discovery”. In: *Computers & Graphics* 38 (2014), S. 38–50. DOI: 10.1016/j.cag.2013.10.007.

[LDA11] S. Lohmann, P. Díaz und I. Aedo. “MUTO: the modular unified tagging ontology”. In: *Proc. Semantic Systems (I-SEMANTICS '11)*. ACM, 2011, S. 95–104. DOI: 10.1145/2063518.2063531.

[LHN15] S. Lohmann, F. Haag und S. Negru. “Towards a visual notation for OWL: a short summary of VOWL”. In: *OWLED '15*. 2015.

[LIJ+15] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer und C. Bizer. "DBpedia – a large-scale, multilingual knowledge base extracted from Wikipedia". In: *Semantic Web Journal* 6.2 (2015), S. 167–195.

[Lin15] V. Link. "Entwicklung eines Benchmark-Generators zum Testen von Ontologievisualisierungen". Betreuer: Steffen Lohmann, Florian Haag. Bachelorarbeit. Universität Stuttgart, 2015.

[Liu12] B. Liu. "Sentiment analysis and opinion mining". In: *Synthesis Lectures on Human Language Technologies* 5.1 (2012), S. 1–167. DOI: 10.1007/978-3-642-19460-3_11.

[LJI11] C.-S. Leung, F. Jiang und P. Irani. "FpMapViz: a space-filling visualization for frequent patterns". In: *Proc. Data Mining Workshops (ICDMW '11)*. 2011, S. 804–811. DOI: 10.1109/ICDMW.2011.86.

[LLM+15] S. Lohmann, V. Link, E. Marbach und S. Negru. "WebVOWL: web-based visualization of ontologies". In: *EKAW 2014 Satellite Events*. Bd. 8982. LNAI. Springer, 2015, S. 154–158. DOI: 10.1007/978-3-319-17966-7_21.

[LNB14] S. Lohmann, S. Negru und D. Bold. "The ProtégéVOWL plugin: ontology visualization for everyone". In: *ESWC 2014 Satellite Events*. Bd. 8798. LNCS. Springer, 2014, S. 395–400. DOI: 10.1007/978-3-319-11955-7_55.

[LNH+14] S. Lohmann, S. Negru, F. Haag und T. Ertl. "VOWL 2: user-oriented visualization of ontologies". In: *Proc. Knowledge Engineering and Knowledge Management (EKAW '14)*. Bd. 8876. LNCS. Springer, 2014, S. 266–281. DOI: 10.1007/978-3-319-13704-9_21.

[LNH+16] S. Lohmann, S. Negru, F. Haag und T. Ertl. "Visualizing ontologies with VOWL". In: *Semantic Web Journal* (voraussichtlich 2016). Angenommen am 21.08.2015.

[LRP95] J. Lamping, R. Rao und P. Pirolli. "A focus+context technique based on hyperbolic geometry for visualizing large hierarchies". In: *Proc. Human Factors in Computing Systems (CHI '95)*. ACM, 1995, S. 401–408. DOI: 10.1145/223904.223956.

[LS93] B. J. Lorch und M. J. Smith. “Pedestrian movement and the downtown enclosed shopping center”. In: *Journal of the American Planning Association* 59.1 (1993), S. 75–86. DOI: 10.1080/01944369308975846.

[LTG+11] M. Lieberman, S. Taheri, H. Guo, F. Mirrashed, I. Yahav, A. Aris und B. Shneiderman. “Visual exploration across biomedical databases”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8.2 (2011), S. 536–550. DOI: 10.1109/TCBB.2010.1.

[LWL+13] C. Langenhan, M. Weber, M. Liwicki, F. Petzold und A. Dengel. “Graph-based retrieval of building information models for supporting the early design stages”. In: *Advanced Engineering Informatics* 27.4 (2013), S. 413–426. DOI: 10.1016/j.aei.2013.04.005.

[LWW+13] S. Liu, Y. Wu, E. Wei, M. Liu und Y. Liu. “StoryFlow: tracking the evolution of stories”. In: *IEEE TVCG* 19.12 (2013), S. 2436–2445. DOI: 10.1109/TVCG.2013.196.

[MAG+04] A. Morris, A. Abdelmoty, B. El-Geresy und C. Jones. “A filter flow visual querying language and interface for spatial databases”. In: *GeoInformatica* 8.2 (2 2004), S. 107–141. DOI: 10.1023/b:gein.0000017744.85002.4c.

[Mar06] G. Marchionini. “Exploratory search: from finding to understanding”. In: *Communications of the ACM* 49.4 (2006), S. 41–46. DOI: 10.1145/1121949.1121979.

[MBB+11] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden und R. C. Miller. “Twitinfo: aggregating and visualizing microblogs for event exploration”. In: *Proc. Human Factors in Computing Systems (CHI ’11)*. ACM, 2011, S. 227–236. DOI: 10.1145/1978942.1978975.

[MC08] S. Mazzocchi und P. Ciccarese. *SIMILE | Welkin*. http://simile.mit.edu/welkin/. 2008.

[MC10] P. Mäder und J. Cleland-Huang. “A visual traceability modeling language”. English. In: *Model Driven Engineering Languages and Systems*. Bd. 6394. LNCS. Springer, 2010, S. 226–240. DOI: 10.1007/978-3-642-16145-2_16.

[MG14] N. Marie und F. Gandon. “Survey of linked data based exploration systems”. In: *Intelligent Exploitation of Semantic Data (IESD ’14)*. Bd. 1279. CEUR-WS, 2014.

[MGA+11] C. Melo, B. Le-Grand, M. Aufaure und A. Bezerianos. “Extracting and visualising tree-like structures from concept lattices”. In: *Proc. Information Visualisation (IV ’11)*. IEEE, 2011, S. 261–266. DOI: 10.1109/IV.2011.46.

[Mic15] Microsoft Corporation. *Microsoft .NET Home*. http://www.microsoft.com/net. 2015.

[Mic82] A. Michard. “Graphical presentation of boolean expressions in a database query language: design notes and an ergonomic evaluation”. In: *Behaviour & Information Technology* 1.3 (1982), S. 279–288. DOI: 10.1080/01449298208914452.

[MK93] L. Mohan und R. Kashyap. “A visual query language for graphical interaction with schema-intensive databases”. In: *IEEE Transactions on Knowledge and Data Engineering* 5.5 (1993), S. 843–858. DOI: 10.1109/69.243513.

[MLL+13] M. Monroe, R. Lan, H. Lee, C. Plaisant und B. Shneiderman. “Temporal event sequence simplification”. In: *IEEE TVCG* 19.12 (2013), S. 2227–2236. DOI: 10.1109/TVCG.2013.200.

[MLM+13] M. Monroe, R. Lan, J. Morales del Olmo, B. Shneiderman, C. Plaisant und J. Millstein. “The challenges of specifying intervals and absences in temporal queries: a graphical language approach”. In: *Proc. Human Factors in Computing Systems (CHI ’13)*. ACM, 2013, S. 2349–2358. DOI: 10.1145/2470654.2481325.

[MPG98] N. Murray, N. Paton und C. Goble. “Kaleidoquery: a visual query language for object databases”. In: *Proc. Advanced Visual Interfaces (AVI ’98)*. ACM, 1998, S. 247–257. DOI: 10.1145/948496.948529.

[MPP+07] C. Morbidoni, A. Polleres, D. L. Phuoc und G. Tummarello. *Semantic Web Pipes*. Techn. Ber. 2007-11-07. Digital Enterpriese Research Institute (DERI), 2007.

[MW95] A. O. Mendelzon und P. T. Wood. “Finding regular simple paths in graph databases”. In: *SIAM Journal on Computing* 24.6 (1995), S. 1235–1258. DOI: 10.1137/S009753979122370X.

[MZM11] A. Makanju, A. N. Zincir-Heywood und E. E. Milios. “Storage and retrieval of system log events using a structured schema based on message type transformation”. In: *Proc. Applied Computing (SAC ’11)*. ACM, 2011, S. 528–533. DOI: 10.1145/1982185.1982298.

[Nau15] A. Naumov. “Facettiertes Suchinterface für die explorative Suche in gewichteten Kundenrezensionen”. Betreuer: Florian Haag, Qi Han, Markus John. Bachelorarbeit. Universität Stuttgart, 2015.

[Neg14] S. Negru. *PersonasOnto*. `http://blankdots.com/open/personasonto.html`. 2014.

[NHL13] S. Negru, F. Haag und S. Lohmann. “Towards a unified visual notation for OWL ontologies: insights from a comparative user study”. In: *Proc. Semantic Systems (I-SEMANTICS ’13)*. ACM, 2013, S. 73–80. DOI: `10.1145/2506182.2506192`.

[NL13a] S. Negru und S. Lohmann. “A visual notation for the integrated representation of OWL ontologies”. In: *Proc. Web Information Systems and Technologies (WEBIST ’13)*. SciTePress, 2013, S. 308–315.

[NL13b] S. Negru und S. Lohmann. *VOWL: Visual Notation for OWL Ontologies*. `http://vowl.visualdataweb.org/v1/`. 2013.

[NLH14] S. Negru, S. Lohmann und F. Haag. *VOWL: Visual Notation for OWL Ontologies*. `http://purl.org/vowl/`. 2014.

[NLT07] T. Nguyen, S. Loke und T. Torabi. “The Community Stack: concept and prototype”. In: *Proc. Advanced Information Networking and Applications Workshops (AINAW ’07)*. Bd. 2. IEEE, 2007, S. 52–58. DOI: `10.1109/ainaw.2007.350`.

[Obj15] Object Management Group. *OMG Unified Modeling Language Version 2.5*. `http://www.omg.org/spec/UML/2.5/`. OMG document number: formal/15-03-01. 2015.

[OBM+13] K. M. de Oliveira, F. Bacha, H. Mnasser und M. Abed. “Transportation ontology definition and application for the content personalization of user interfaces”. In: *Expert Systems with Applications* 40.8 (2013), S. 3145–3159. DOI: `10.1016/j.eswa.2012.12.028`.

[OÖX+99] V. Oria, M. Özsu, B. Xu, L. Cheng und P. Iglinski. “VisualMOQL: the DISIMA visual query language”. In: *Proc. Multimedia Computing and Systems (ICMCS ’99)*. IEEE, 1999, S. 536–542. DOI: `10.1109/MMCS.1999.779258`.

[Ope08] OpenLink Software. *OpenLink iSPARQL*. `http://oat.openlinksw.com/isparql/`. 2008.

[Opp14] S. Oppold. “Visuelle Auswahl von Ontologiebestandteilen”. Betreuer: Florian Haag. Bachelorarbeit. Universität Stuttgart, 2014.

[Ove10] Over the Sun, LLC. *iConji. Connecting the world.* http://www.iconji.com/. 2010.

[PCL14] E. Prud'hommeaux, G. Carothers und Lex Machina, Inc. *RDF 1.1 Turtle*. http://www.w3.org/TR/turtle/. 2014.

[PGP+14] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos und S. Manandhar. "SemEval-2014 task 4: aspect based sentiment analysis". In: *Proc. Workshop on Semantic Evaluation (SemEval '14)*. ACL und Dublin City University, 2014, S. 27–35.

[Pie07] E. Pietriga. *IsaViz: A Visual Authoring Tool for RDF*. http://www.w3.org/2001/11/IsaViz/. 2007.

[PK95] A. Papantonakis und P. King. "Syntax and semantics of Gql, a graphical query language". In: *Journal of Visual Languages & Computing* 6.1 (1995), S. 3–25. DOI: 10.1006/jvlc.1995.1002.

[PMR+96] C. Plaisant, B. Milash, A. Rose, S. Widoff und B. Shneiderman. "LifeLines: visualizing personal histories". In: *Proc. Human Factors in Computing Systems (CHI '96)*. ACM, 1996, S. 221–227. DOI: 10.1145/238386.238493.

[PSR+13] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, Y. Theodoridis und Z. Yan. "Semantic trajectories modeling and analysis". In: *ACM Computing Surveys* 45.4 (2013). DOI: 10.1145/2501654.2501656.

[PW14] A. Perer und F. Wang. "Frequence: interactive mining and visualization of temporal frequent event sequences". In: *Proc. Intelligent User Interfaces (IUI '14)*. ACM, 2014, S. 153–162. DOI: 10.1145/2557500.2557508.

[Raj15] J. Rajamani. "Filter (Dials | ...)" Betreuer: Florian Haag. Masterarbeit. Universität Stuttgart, 2015.

[RC94] R. Rao und S. K. Card. "The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information". In: *Proc. Human Factors in Computing Systems (CHI '94)*. ACM, 1994, S. 318–322. DOI: 10.1145/191666.191776.

[RH04] D. R. Rubenstein und K. A. Hobson. "From birds to butterflies: animal movement patterns and stable isotopes". In: *Trends in Ecology & Evolution* 19.5 (2004), S. 256–263. DOI: 10.1016/j.tree.2004.03.017.

[RKM+94] S. F. Roth, J. Kolojejchick, J. Mattis und J. Goldstein. "Interactive graphic design using automatic presentation knowledge". In: *Proc. Human Factors in Computing Systems (CHI '94)*. ACM, 1994, S. 112–117. DOI: 10.1145/191666.191719.

[RS08] A. Russell und P. Smart. "NITELIGHT: a graphical editor for SPARQL queries". In: *Proc. Poster and Demonstration at the International Semantic Web Conference (ISWC '08)*. Bd. 401. CEUR-WS, 2008.

[RSB+08] A. Russell, P. Smart, D. Braines und N. Shadbolt. "NITELIGHT: a graphical tool for semantic query construction". In: *Proc. Workshop on Semantic Web User Interaction (SWUI '08)*. Bd. 543. CEUR-WS, 2008.

[RWW+11] A. Rind, T. D. Wang, A. Wolfgang, S. Miksch, K. Wongsuphasawat, C. Plaisant und B. Shneiderman. "Interactive information visualization to explore and query electronic health records". In: *Foundations and Trends in Human-Computer Interaction* 5.3 (2011), S. 207–298. DOI: 10.1561/1100000039.

[Say04] C. Sayers. *Node-centric RDF graph visualization*. Techn. Ber. Mobile und Media Systems Laboratory, HP Labs, 2004.

[SBM+93] G. H. Sockut, L. M. Burns, A. Malhotra und K.-Y. Whang. "GRAQULA: a graphical query language for entity-relationship or relational databases". In: *Data & Knowledge Engineering* 11.2 (1993), S. 171–202. DOI: 10.1016/0169-023X(93)90004-9.

[Sch94] A. Schürr. *PROGRES, A Visual Language and Environment for PROgramming with Graph REwriting Systems*. Techn. Ber. AIB 94-11. RWTH Aachen, 1994.

[SCT91] K. Siau, H. Chan und K. Tan. "Visual knowledge query language as a front-end to relational systems". In: *Proc. Computer Software and Applications Conference (COMPSAC '91)*. IEEE, 1991, S. 373–378. DOI: 10.1109/CMPSAC.1991.170205.

[Sei11] I. Seifert. "A pool of queries: interactive multidimensional query visualization for information seeking in digital libraries". In: *Information Visualization* 10.2 (2011), S. 97–106. DOI: 10.1057/ivs.2011.1.

[SF08] A. S. Spritzer und C. M. D. S. Freitas. "A physics-based approach for interactive manipulation of graph visualizations". In: *Proc. Advanced Visual Interfaces (AVI '08)*. ACM, 2008, S. 271–278. DOI: 10.1145/1385569.1385613.

[SF10] P. Sousa und M. J. Fonseca. “Sketch-based retrieval of drawings using spatial proximity”. In: *Journal of Visual Languages & Computing* 21.2 (2010), S. 69–80. DOI: 10.1016/j.jvlc.2009.12.001.

[SGJ+13] A. Soylu, M. Giese, E. Jimenez-Ruiz, E. Kharlamov, D. Zheleznyakov und I. Horrocks. “OptiqueVQS: towards an ontology-based visual query system for big data”. In: *Proc. Management of Emergent Digital EcoSystems (MEDES ’13)*. ACM, 2013, S. 119–126. DOI: 10.1145/2536146.2536149.

[SGJ+15] A. Soylu, M. Giese, E. Jimenez-Ruiz, G. Vega-Gorgojo und I. Horrocks. “Experiencing OptiqueVQS: a multi-paradigm and ontology-based visual query system for end users”. English. In: *Universal Access in the Information Society* (2015), S. 1–24. DOI: 10.1007/s10209-015-0404-5.

[Shn91] B. Shneiderman. “Visual user interfaces for information exploration”. In: *Proc. ASIS Annual Meeting*. Bd. 28. Information Today, 1991, S. 379–384.

[Shn94] B. Shneiderman. “Dynamic queries for visual information seeking”. In: *IEEE Software* 11.6 (1994), S. 70–77. DOI: 10.1109/52.329404.

[SHT+07] P. Sadri, E. Ho, J. Trevor, K. Cheng und D. Raffel. *Yahoo! Pipes*. Seit Ende September 2015 nicht mehr in Betrieb. 2007. URL: http://pipes.yahoo.com/.

[Sie14] S. Siek. “VOWL-basierte Visualisierung für SPARQL-Anfragen”. Betreuer: Florian Haag, Steffen Lohmann. Diplomarbeit. Universität Stuttgart, 2014.

[SLC+08] R. S. Schick, S. R. Loarie, F. Colchero, B. D. Best, A. Boustany, D. A. Conde, P. N. Halpin, L. N. Joppa, C. M. McClellan und J. S. Clark. “Understanding movement data and movement processes: current and emerging directions”. In: *Ecology Letters* 11.12 (2008), S. 1338–1350. DOI: 10.1111/j.1461-0248.2008.01249.x.

[SLW+14] G. Sun, Y. Liu, W. Wu, R. Liang und H. Qu. “Embedding temporal display into maps for occlusion-free visualization of spatio-temporal data”. In: *Proc. Pacific Visualization Symposium (PacificVis ’14)*. IEEE, 2014, S. 185–192. DOI: 10.1109/pacificvis.2014.56.

[SMS+01] M.-A. Storey, M. Musen, J. Silva, C. Best, N. Ernst, R. Fergerson und N. Noy. "Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé". Workshop on Interactive Tools for Knowledge Capture (unveröffentlicht). 2001.

[SMS+05] J. Soo Yi, R. Melton, J. Stasko und J. A. Jacko. "Dust & Magnet: multivariate information visualization using a magnet metaphor". In: *Information Visualization* 4.4 (2005), S. 239–256. DOI: 10.1057/palgrave.ivs.9500099.

[Sno87] R. Snodgrass. "The temporal query language TQuel". In: *ACM Transactions on Database Systems* 12.2 (1987), S. 247–298. DOI: 10.1145/22952.22956.

[SOB+12] N. Shadbolt, K. O'Hara, T. Berners-Lee, N. Gibbins, H. Glaser, W. Hall und m.c. schraefel. "Linked open government data: lessons from data.gov.uk". In: *IEEE Intelligent Systems* 27.3 (2012), S. 16–24. DOI: 10.1109/mis.2012.23.

[Spo93] A. Spoerri. "InfoCrystal: a visual tool for information retrieval & management". In: *Proc. Information and Knowledge Management (CIKM '93)*. ACM, 1993, S. 11–20. DOI: 10.1145/170088.170095.

[SRB+08] P. Smart, A. Russell, D. Braines, Y. Kalfoglou, J. Bao und N. Shadbolt. "A visual approach to semantic query design using a web-based graphical query designer". English. In: *Knowledge Engineering: Practice and Patterns (EKAW '08)*. Bd. 5268. LNCS. Springer, 2008, S. 275–291. DOI: 10.1007/978-3-540-87696-0_25.

[SS93] G. Santucci und P. A. Sottile. "Query by diagram: a visual environment for querying databases". In: *Software: Practice and Experience* 23.3 (1993), S. 317–340. DOI: 10.1002/spe.4380230307.

[SSV14] M. Schmidt, S. Simmerling und D. Väth. "In-Situ Visualization of Map Marker Filters on Mobile Devices". Betreuer: Florian Haag. Projekt-INF. Universität Stuttgart, 2014.

[Sta10] Stack Exchange. *Stack Exchange*. http://stackexchange.com/. 2010.

[STT91] F. Staes, L. Tarantino und A. Tiems. "A graphical query language for object oriented databases". In: *Proc. Workshop on Visual Languages*. IEEE, 1991, S. 205–210. DOI: 10.1109/WVL.1991.238831.

[The13] The W3C SPARQL Working Group. *SPARQL 1.1 Overview*. W3C Recommendation. W3C, 2013. URL: http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/.

[TIC09] M. Tobiasz, P. Isenberg und S. Carpendale. "Lark: coordinating co-located collaboration with information visualization". In: *IEEE TVCG* 15.6 (2009), S. 1065–1072. DOI: 10.1109/tvcg.2009.162.

[TSA+12] C. Tominski, H. Schumann, G. Andrienko und N. Andrienko. "Stacking-based visualization of trajectory attribute data". In: *IEEE TVCG* 18.12 (2012), S. 2565–2574. DOI: 10.1109/tvcg.2012.265.

[TSW+94] L. Tweedie, B. Spence, D. Williams und R. Bhogal. "The Attribute Explorer". In: *Proc. Human Factors in Computing Systems (CHI '94)*. ACM, 1994, S. 435–436. DOI: 10.1145/259963.260433.

[Tun06] D. Tunkelang. "Dynamic category sets: An approach for faceted search". ACM SIGIR '06 Workshop on Faceted Search (unveröffentlicht). 2006.

[UFK+89] C. Upson, J. Faulhaber T.A., D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz und A. van Dam. "The Application Visualization System: a computational environment for scientific visualization". In: *IEEE Computer Graphics and Applications* 9.4 (1989), S. 30–42. DOI: 10.1109/38.31462.

[UZ83] P. Ursprung und C. A. Zehnder. "HIQUEL: an interactive query language to define and use hierarchies". In: *Proc. Entity-Relationship Approach (ER'83)*. North-Holland, 1983, S. 299–314.

[Vät15] D. Väth. "Query Visualization for Time-Based Graph Data". Betreuer: Florian Haag. Bachelorarbeit. Universität Stuttgart, 2015.

[VEC07] K. Vrotsou, K. Ellegård und M. Cooper. "Everyday life discoveries: mining and visualizing activity patterns in social science diary data". In: *Proc. Information Visualization (IV '07)*. IEEE, 2007, S. 130–138. DOI: 10.1109/IV.2007.48.

[Vis14] Visual Analytics Community. *VAST 2014 Challenge – The Kronos Incident*. http://vacommunity.org/VAST+Challenge+2014. 2014.

[VJC09] K. Vrotsou, J. Johansson und M. Cooper. "ActiviTree: interactive visual exploration of sequences in event-based data using graph similarity". In: *IEEE TVCG* 15.6 (2009), S. 945–952. DOI: 10.1109/TVCG.2009.117.

[VKS+12] J. Vermeulen, F. Kawsar, A. Simeone, G. Kortuem, K. Luyten und K. Coninx. "Informing the design of situated glyphs for a care facility". In: *Proc. Visual Languages and Human-Centric Computing (VL/HCC '12)*. IEEE, 2012, S. 89–96. DOI: 10.1109/VLHCC.2012.6344490.

[VZA+15] R. Vesse, R. M. Zettlemoyer, K. Ahmed, G. Moore und T. Pluskiewicz. *dotNetRDF: An Open Source C# .NET Library for RDF*. http://www.dotnetrdf.org/. 2015.

[W3C12] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. http://www.w3.org/TR/owl2-overview/. 2012.

[Wat01] M. Wattenberg. "Sketching a graph to query a time-series database". In: *Extended Abstracts on Human Factors in Computing Systems (CHI EA '01)*. ACM, 2001, S. 381–382. DOI: 10.1145/634067.634292.

[Wat06] M. Wattenberg. "Visual exploration of multivariate graphs". In: *Proc. Human Factors in Computing Systems (CHI '06)*. ACM, 2006, S. 811–819. DOI: 10.1145/1124772.1124891.

[WJ07] U. Westermann und R. Jain. "Toward a common event model for multimedia applications". In: *IEEE MultiMedia* 14.1 (2007), S. 19–29. DOI: 10.1109/mmul.2007.23.

[WMP+05] P. C. Wong, P. Mackey, K. Perrine, J. Eagan, H. Foote und J. Thomas. "Dynamic visualization of graphs with extended labels". In: *Proc. IEEE Information Visualization (INFOVIS '05)*. IEEE, 2005, S. 73–80. DOI: 10.1109/INFVIS.2005.1532131.

[Woo12] P. T. Wood. "Query languages for graph databases". In: *SIGMOD Record* 41.1 (2012), S. 50–60. DOI: 10.1145/2206869.2206879.

[WPQ+08] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, S. Murphy und B. Shneiderman. "Aligning temporal data by sentinel events: discovering patterns in electronic health records". In: *Proc. Human Factors in Computing Systems (CHI '08)*. ACM, 2008, S. 457–466. DOI: 10.1145/1357054.1357129.

[WPT+12] K. Wongsuphasawat, C. Plaisant, M. Taieb-Maimon und B. Shneiderman. "Querying event sequences by exact match or similarity search: design and empirical evaluation". In: *Interacting with Computers* 24.2 (2012), S. 55–68. DOI: 10.1016/j.intcom.2012.01.003.

[XSA08] W. Xu, M. Shehab und G.-J. Ahn. "Visualization based policy analysis: case study in SELinux". In: *Proc. Access Control Models and Technologies (SACMAT '08)*. ACM, 2008, S. 165–174. DOI: 10.1145/1377836.1377863.

[YS93] D. Young und B. Shneiderman. "A graphical filter/flow representation of boolean queries: a prototype implementation and evaluation". In: *Journal of the Association for Information Science and Technology* 44.6 (1993), S. 327–339. DOI: 10.1002/(SICI)1097-4571(199307)44:6<327::AID-ASI3>3.0.CO;2-J.

[YSL+03] K.-P. Yee, K. Swearingen, K. Li und M. Hearst. "Faceted metadata for image search and browsing". In: *Proc. Human Factors in Computing Systems (CHI '03)*. ACM, 2003, S. 401–408. DOI: 10.1145/642611.642681.

[YZL+09] Y. Yue, Y. Zhuang, Q. Li und Q. Mao. "Mining time-dependent attractive areas and movement patterns from taxi trajectory data". In: *Proc. Geoinformatics*. IEEE, 2009, S. 1–6. DOI: 10.1109/GEOINFORMATICS.2009.5293469.

[ZB11] M. Zviedris und G. Bārzdiņš. "ViziQuer: a tool to explore and query SPARQL endpoints". In: *Proc. Extended Semantic Web Conference (ESWC '11)*. Springer, 2011, S. 441–445. DOI: 10.1007/978-3-642-21064-8_31.

[ZDF+15] E. Zgraggen, S. M. Drucker, D. Fisher und R. DeLine. "(s|qu)eries: visual regular expressions for querying and exploring event sequences". In: *Proc. Human Factors in Computing Systems (CHI '15)*. ACM, 2015, S. 2683–2692. DOI: 10.1145/2702123.2702262.

Alle im Literaturverzeichnis sichtbaren URLs, zu denen nicht explizit vermerkt ist, dass sie mittlerweile unerreichbar sind, wurden zuletzt am 26.01.2016 besucht.