Proceedings

23. GI-Workshop Grundlagen von Datenbanken

31.05.2011 - 03.06.2011

Obergurgl, Tirol, Österreich

Wolfgang Gassler, Eva Zangerle, Günther Specht (Hrsg.)

Vorwort

Liebe Teilnehmerinnen und Teilnehmer,

der 23. Workshop Grundlagen von Datenbanken (GvD) 2011 findet in Obergurgl, Tirol, Österreich statt. Dieser viertägige Workshop wird vom GI-Arbeitskreis Grundlagen von Informationssystemen im Fachbereich Datenbanken und Informationssysteme (DBIS) veranstaltet und hat die theoretischen, konzeptionellen und methodischen Grundlagen von Datenbanken und Informationssystemen zum Thema. Organisiert wird der Workshop 2011 von der Forschungsgruppe Datenbanken und Informationssysteme am Institut für Informatik an der Leopold-Franzens-Universität in Innsbruck.

Der Workshop soll die Kommunikation zwischen Wissenschaftlern/-innen im deutschsprachigen Raum fördern, die sich grundlagenorientiert mit Datenbanken und Informationssystemen beschäftigen. Er ist insbesondere als Forum für Nachwuchswissenschaftler/-innen gedacht, die ihre aktuellen Arbeiten in einem größeren Forum vorstellen wollen. Mit der Kulisse der beeindruckenden Tiroler Bergwelt bietet der Workshop auf 2000 Metern Meereshöhe einen idealen Rahmen für die offenen und inspirierenden Diskussionen zu Datenbanken und Informationssystemen. 18 Papiere wurde aus den Einsendungen ausgewählt und werden in Obergurgl vorgestellt. Sie spannen ein weites Feld auf und zeigen, wie aktiv und inspirierend die Datenbankforschung heute sein kann. Gerade das neue Feld der Hauptspeicher- und NoSQL-Datenbanken hat die Auseinandersetzung mit dem, was die eigentlichen Grundlagen der Datenbanken sind, neu beflügelt. Dabei ist der Workshop über die Jahre, inzwischen 23 (!), längst von einem ursprünglich eher theorie- und grundlagenlastigen Workshop auch zu einer Plattform für anwendungsgetriebene und praktische Datenbankforschung geworden und hat sich dabei auch dem damit verbundenen Gebiet der Informationssysteme geöffnet. Wichtig ist die offene Atmosphäre in der man, diesmal zurückgezogen in der Bergwelt, ohne Zeitzwang, intensiv diskutieren kann. Dies bedeutet insbesondere nicht nur die üblichen drei Fragen nach dem Vortrag bevor der nächste kommt, sondern sich die Zeit nehmen, vorgestellte neue Gedanken und Ansätze wirklich gemeinsam zu diskutieren. Davon profitieren alle.

Hinzu kommen drei interessante Keynote-Vorträge von Harald Frick, Thomas Neumann und Wolf-Tilo Balke. Ihnen sei an dieser Stelle für Ihre Bereitschaft und Ihr Kommen gedankt.

Weiters danken wir dem Programmkomitee, das diesmal erstmals auf 18 Mitglieder gewachsen ist und allen Gutachtern für ihre Arbeit. Das Organisations-Komitee und dabei insbesondere Fr. Eva Zangerle und Hr. Wolfgang Gassler haben den Großteil der Arbeit gestemmt. Ohne ihren unermüdlichen Einsatz und ihr Engagement wäre der 23. Workshop nicht zustande gekommen. Herzlichen Dank! Besonderer Dank gilt auch Eike Schallehn und dem GI-Arbeitskreis "Grundlagen von Informationssystemen", der sich mit viel Einsatz um die erfolgreiche jährliche Austragung des Workshops bemüht. Schließlich gilt mein Dank allen denen, die im Hintergrund mitwirkten, dem ganzen DBIS-Team in Innsbruck, dem Haus in Obergurgl und nicht zuletzt allen Autoren und Vortragenden. Sie machen den Workshop erst zu dem, was er ist: Eine inspirierendes und motivierendes Forum für offene Diskussionen über alle neuen Ideen rund um Datenbanken und Informationssysteme. Sicher wird auch die gepflegte Hütten- und Bergatmosphäre und der offene Kamin am Abend das seine dazu beitragen. Ich freue mich darauf.

Mit den besten Grüßen,

Günther Specht

Innsbruck am 26.05.2011

Komitee

Programm-Komitee

- Wolf-Tilo Balke, Universität Braunschweig
- Stefan Brass, Universität Halle
- Erik Buchmann, Universität Karlsruhe
- Stefan Conrad, Universität Düsseldorf
- Johann-Christoph Freytag, Humboldt-Universität Berlin
- Torsten Grust, Universität Tübingen
- Andreas Henrich, Universität Bamberg
- Hagen Höpfner, Universität Weimar
- Harald Kosch, Universität Passau
- Holger Meyer, Universität Rostock
- Klaus Meyer-Wegener, Universität Erlangen
- Bernhard Mitschang, Uni Stuttgart
- Daniela Nicklas, Universität Oldenburg
- Gunter Saake, Universität Magdeburg
- Eike Schallehn, Universität Magdeburg
- Ingo Schmitt, TU Cottbus
- Holger Schwarz, Universität Stuttgart
- Günther Specht, Universität Innsbruck

Organisations-Komitee

- Eva Zangerle, Universität Innsbruck
- Wolfgang Gassler, Universität Innsbruck
- Günther Specht, Universität Innsbruck
- Eike Schallehn, Universität Magdeburg, Speaker of the Working Group Foundations of Information Systems

Weitere Reviewer

- Eva Zangerle, Universität Innsbruck
- Wolfgang Gassler, Universität Innsbruck
- Wolfgang Pausch, Universität Innsbruck
- Sebastian Schönherr, Universität Innsbruck
- Robert Binna, Universität Innsbruck
- Dominic Pacher, Universität Innsbruck

Inhaltsverzeichnis

Massive parallel in-memory database with GPU based query co-processor

Harald Frick
QuiLogic In-Memory DB Technology

ABSTRACT

This talk presents work on transforming SQL-IMDB, a commercial available in-memory database system, into a massive parallel, array structured data processor extending the "classic" query engine architecture with GPU based co-processing facilities. The chosen approach is not just a simple re-implementation of common database functionality like sorting, stream processing and joins on GPUs, instead we take a holistic view and extend the entire query engine to work as a genuine, in-memory, GPU supported database engine. We have partitioned the query engine so that both CPU and GPU are doing what they are best at. The new SQL-IMDBg query execution engine is a "Split-Work" engine which takes care to optimize, schedule and execute the query plan simultaneous and in the most efficient way on two (or more) different memory devices. The principal architecture of the engine, based on simultaneous managing multiple memory devices (local/shared/flash-memory), was a natural fit to include the new GPU/video memory as just another (high speed) memory device. All internal core engine data structures are now based on simple array structures, for maximum parallel access support on multi- and many core hardware. Data tables located on GPU video memory can always queried together with CPU local- and shared-memory tables in "mixed" query statements. Columns on GPU tables are also accessible through GPU based indexes. A special index structure was developed based on sorted containers supporting both CPU and GPU based index lookups. Table data can be manually and automatically split between CPU and GPU and is held in vertically partitioned columns, which ease the stream like processing for basic scan primitives and coalesced memory access mechanism on GPU devices. Based on our experience gained, we see the GPU/video memory as another important high speed memory device for in-memory database systems, but which do not yet fit well into the architecture of current database engines and therefore require a major effort in re-engineering the entire core database architecture.

Efficient Query Processing on Modern Hardware

Thomas Neumann
Lehrstuhl für Informatik III: Datenbanksysteme
Fakultät für Informatik
Technische Universität München

ABSTRACT

Most database systems translate a given query into an expression in a (physical) algebra, and then start evaluating this algebraic expression to produce the query result. The traditional way to execute these algebraic plans is the iterator model: Every physical algebraic operator conceptually produces a tuple stream from its input, and allows for iterating over this tuple stream. This is a very nice and simple interface, and allows for easy combination of arbitrary operators,but it clearly comes from a time when query processing was dominated by I/O and CPU consumption was less important: The iterator interface causes thousands of expensive function calls, degrades the branch prediction of modern CPUs, and ofter results in poor code locality and complex book-keeping.

On modern hardware query processing can be improved considerably by processing tuples in a data centric, and not an operator centric, way. Data is processed such that it can be kept in CPU registers as long as possible. Operator boundaries are blurred to achieve this goal. In combination with an code compilation framework this results in query code that rivals the speed of hand-written code. When using these techniques in the HyPer DBMS, TPC-H Query 1 for example can single-threaded aggregated the scale factor 1GB data set in about 68ms on commodity hardware.

Conceptual Views for Entity-Centric Search

Wolf-Tilo Balke
Databases and Informationsystems
University of Braunschweig

ABSTRACT

The retrieval of entity data has always been a core application of database systems and querying an entity's attributes can be efficiently done using declarative languages like SQL. But today's retrieval tasks more and more focus also on conceptual aspects of entities, which often are not directly expressed by attributes. For instance, users might want to find a 'thrilling' novel, unfortunately there is no 'suspense factor' attribute in today's online book stores. Consequently, entity-centric search suffers from a growing semantic gap between the users' intended queries and the database's schema. In the talk, we will propose the notion of conceptual views, an innovative extension of traditional database views, which aim to uncover those query-relevant concepts that are often only reflected by unstructured data related to some entities. We will also take a look at promising techniques for mining conceptual information and discuss open issues.

Ein Ansatz zu Opinion Mining und Themenverfolgung für eine Medienresonanzanalyse

Thomas Scholz

Heinrich-Heine-Universität Düsseldorf
Institut für Informatik
Universitätsstr. 1
D-40225 Düsseldorf, Deutschland
scholz@cs.uni-duesseldorf.de

pressrelations GmbH
Entwicklung
Klosterstr. 112
D-40211 Düsseldorf, Deutschland
thomas.scholz@pressrelations.de

Zusammenfassung

Heutzutage gibt es eine unüberschaubare Anzahl von Medien mit enorm vielen Artikeln und Beiträgen. Da in ihnen neben vielen anderen potenziell nützlichen Informationen wertvolle Meinungen zu Themen enthalten sind, ist eine automatische Beobachtung dieser Medien sehr interessant, birgt aber zwei große Herausforderungen: eine automatisierte Tonalitätsbestimmung (Opinion Mining) kombiniert mit einer Themenverfolgung. Diese zwei Aufgaben sind Teilgebiete des Text Minings, auch Text Data Mining oder Knowledge Discovery in Texten genannt. Diese Arbeit beschreibt einen Ansatz für Opinion Mining und Themenverfolgung basierend auf einer Information Extraction Architektur. In der Evaluation wird gezeigt, wie dieser Ansatz für Opinion Mining oder eine Themenverfolgung eingesetzt werden kann.

Kategorie

Data Mining and Knowledge Discovery

Schlüsselwörter

Opinion Mining, Topic Tracking, Text Mining

1. EINLEITENDE MOTIVATION

Das Internet, Printmedien, TV, Hörfunk und Soziale Netzwerke sind eine Fundgrube von Meinungen zu bestimm-ten Themen in Form von Artikeln oder Beiträgen. Heutzutage ist es möglich diese in digitaler Form zu erfassen. Im Online Bereich können beispielsweise Crawler eingesetzt werden um die Artikelseiten von Internetnachrichten zu erfassen. Durch Analyse des Seitenquelltextes und den Einsatz von Heuristiken kann der eigentliche Artikeltext gefunden werden. Artikel aus Printmedien können eingescannt und mit optischer Zeichenerkennung (OCR) digitalisiert werden. Artikel und Beiträge in digitaler Textform bieten auch teilweise die TV- und Hörfunksender an. Im Bereich Soziale Netzwerke haben sich Dienstleister darauf spezialisiert, die Diskussionen und Kommentare aus Netzen wie Twitter, Facebook oder bestimmte Foren automatisch zu erfassen und als Daten anzubieten.

Die Auswertung von einer großen Menge dieser Artikel und Beiträge ist hingegen schwierig. Eine manuelle Auswertung ist nur mit erheblichen Aufwand möglich und für eine automatische Auswertung gibt es erste Ansätze, aber längst noch keine allumfassende Lösung. Mit der Aufgabe diese Daten zu erfassen und auszuwerten beschäftigt man sich beim Medienmonitoring bzw. bei der Medienresonanzanalyse. In diesem Bereich arbeiten Ausschnittdienste.

1.1 Medienmonitoring

Die riesigen Ströme aus Artikeln und Beiträgen enthalten viele potenziell wertvolle Informationen zu Themen, Personen, Firmen, Produkten, usw. Besonders die PR- und Marketing-Abteilungen von Unternehmen, Parteien und Verbänden interessieren für diese Daten und deren Auswertung. Dabei interessiert man sich besonders dafür, inwiefern sich das Image des Unternehmens oder das Image von bestimmten Produkten, Marken und Dienstleistungen entwickelt. Aber auch wie bestimmte Personen (Werbeträger, Vorstandsmitglieder, etc.) in diesen Medien wahrgenommen werden. Außerdem von Bedeutung ist die Frage, auf welche Weise bestimmte Themen mit dem Unternehmen verknüpft sind. Beim Medienmoitoring geht es darum Artikel zu erfassen und sammeln, die für PR- und Marketing Abteilungen interessant sind. Dazu werden von ihnen Themen oder Schlagwörter definiert. Dies ist auch interessant für Verbände, Vereine, Parteien, Stiftungen, die teilweise zu klein sind um über eine eigene PR-Abteilung zu verfügen.

1.2 Medienresonanzanalyse

Bei der Medienresonanz geht es darum zu bestimmten Themen das mediale Echo zu analysieren.

Dies kann z. B. auf folgende Art und Weise geschehen: Zunächst werden Themen definiert, die es zu untersuchen gilt. Dies können beispielsweise Marken von Firmen sein oder andere Begriffe wie Produktnamen, Personen oder ähnliches. Bei einem Medienbeobachter und einem Ausschnittdienst würden die Kunden (meist PR-Abteilungen von Firmen oder Organisationen wie Parteien) Themen durch bestimmte Schlagwörter festlegen. Die Schlagwörter werden dann von Crawlern in den Medien gesucht, um die entspre-

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

chenden Artikel zu erfassen.

Dann können diese Artikel bewertet werden, z.B. ob sie wirklich relevant zu diesem Thema sind, wie exklusiv dieser Beitrag ist und am wichtigsten welche Tonalität er besitzt. Die Tonalität beschreibt, ob ein bestimmter Artikel positive oder negative Meinungen zu einem Thema enthält. Es ist auch beides möglich. Oft wird dies mit einem negativen Zahlenwert für eine negative Tonalität und einem positiven Zahlenwert für eine positive Tonalität festgehalten. Diese Bestimmung der Tonalität wird heutzutage bei den Medienbeobachtern noch rein manuell durch Medienanalysten ausgeführt, die die Texte lesen und meinungsbeinhaltende Passagen identifizieren und bewerten.

Allerdings stoßen solche Beobachtungsdienste aufgrund der Menge der Artikel und Beiträge heute an ihre Grenzen, was manuell noch zu bearbeiten ist. Die Nachrichtenanzahl, die diese Dienste bearbeiten, nimmt stetig zu, aufgrund der stärkeren Digitalisierung der Medien wie auch durch Entstehen neuer Medien im Internet wie der Bereich Social Media.

1.3 Opinion Mining und Themenverfolgung als Lösung

Als Lösung für eine automatische Medienresonanzanalyse bieten sich Opinion Mining und Themenverfolgung an, um die doch bisher meist manuell geleistete Arbeit zu unterstützen und perspektivisch zu ersetzen.. Mit Hilfe des Opinion Minings soll es gelingen meinungstragende Passagen innerhalb eines Textes zu finden und dann automatisch mit einem Tonalitätswert zu versehen. Durch die Realisierung einer automatischen Themenverfolgung könnte die manuell vorgenommene Schlagwortverwaltung für die Themenzuordnung abgelöst werden. In der Kombination hätte man dann eine automatische Medienresonanzanalyse.

Der Rest dieser Arbeit kann wie folgt zusammengefasst werden: Zunächst werden die Ansätze aufgezeigt, die schon zum Bereich des Opinion Minings und der Themenverfolgung entwickelt wurden. Dann wird die Architektur entworfen, wodurch eine automatische Medienresonanzanalyse auf Basis von Natural Language Processing und Information Extraction realisierbar wird. Anschließend wird in ersten Versuchen demonstriert, wie die Vorverarbeitung für Opinion Mining und Themenverfolgung eingesetzt werden können. Abschließend werden dann aufgrund dieser Ergebnisse Schlussfolgerungen für das weitere Vorgehen gezogen.

2. VERWANDTE ARBEITEN

2.1 Opinion Mining

Um die Tonalität eines Textes zu bestimmen, benutzen viele Ansätze wie [2, 3, 9, 13] Wörterbücher, in denen Wörter mit einem Tonalitätswert hinterlegt sind. Diese Wörterbücher werden meist so aufgebaut: Man beginnt mit einer kleineren Menge von positiven und negativen Wörtern. Dann wird analysiert, ob neue Wörter oft mit positiven oder negativen Wörtern auftauchen und entsprechend bekommt dann das neue Wort einen positiven oder negativen Tonalitätswert.

Eine typische Menge von Saattonalitätswörtern sieht beispielsweise so aus [16]:

- positiv: {good, nice, excellent, positive, fortunate, correct, superior}
- negativ: {bad, nasty, poor, negative, unfortunate, wrong, inferior}

Diesen Aufbau durch einen Bootstrapping Algorithmus benutzt auch das bekannteste Wörterbuch: SentiWordNET [5]. Als Quelle für neue Wörter benutzt es dazu Glossetexte und Wortbeziehungen in WordNET, dem bekanntesten englischen, digitalen Wörterbuch. Mit diesen Wortbeziehungen wie Synonym, Oberbegriff oder Unterbegriff werden neue Wörter gefunden, die dann die gleiche oder eine ähnliche Tonalität bekommen.

Andere Ansätze benutzen nur WordNET oder ähnliche, allgemeine Wörterbücher [6, 7] oder Textsammlungen [2, 4] oder Suchanfragen [6]. Die Vorgehensweise ähnelt dabei oft der Vergrößerung der Wortmenge durch Bootstrapping.

Viele Ansätze [2, 3, 4, 13] beschränken sich auf das Gebiet der Sentiment Analysis, also der Tonalitätsbestimmung in Kundenrezensionen. Einige Ansätze beschränken sich dabei nur auf Adjektive [3].

Bei Kundenrezensionen ist eine Identifikation von Meinungsblöcken nicht nötig. Eine Rezension besteht nur aus Meinungsblöcken. Um generell Meinungen zu finden, auch wenn dies in einem langen Zeitungsartikel nur ein kleiner Absatz über ein bestimmtes Unternehmen ist, sind zunächst noch anderere Schritte zuvor nötig. Ein satzbasierter Ansatz [7] bestimmt für jeden Satz einen Tonalitätswert basierend auf den in ihm enthalten Wörtern. Dafür werden in zwei Modellen einmal alle Wörter oder nur das stärkste Tonalitätswort herangezogen. Überschreitet der Wert eine gewisse Grenze, dann enthält der Satz eine Meinung. Für einen anderen Ansatz [1] enthält ein Satz eine Meinung, wenn er ein Adjektiv enthält.

Auch kann man davon ausgehen, dass bei Sentiment Analysis Ansätzen man schon mit einem kleineren Wörterbuch mit Tonalitätswerten zurecht kommt, da man es bei den Zielen der Rezension nur mit Produkten und ähnlichem wie Filmen, Hotels usw. zu tun hat. Bei einer Medienresonanzanalyse werden aber gleichzeitig Entitäten wie Personen, Organisationen, Produkte, Events oder Aktionen im Fokus stehen. Somit ändern sich auch die tonalitätsbildenen Wörter, da nicht allein durch eine Beschreibung eines Produktes etc. eine Tonalität ausgedrückt wird.

In einer Rezension will der Autor seine Meinung dem Leser direkt vermitteln. In Zeitungsartikeln beschreibt der Autor nicht nur direkte Meinungen, oft wird eher über Fakten und Handlungen gesprochen, die sich auf bestimmte Personen oder Organisationen beziehen, die dann eine Tonalität entstehen lassen. Darum sollte ein solcher Ansatz auch nicht nur Adjektive, sondern mehr Wortarten miteinbeziehen (z. B. Verben). Die meisten Ansätze [1, 2, 3] sind auf die englische Sprache ausgerichtet. Darüber hinaus gibt es einige Ansätze für die Chinesische Sprache, die allerdings nicht die Güte der Ergebnisse der auf Englisch arbeitenden Ansätze erreichen [4, 9].

2.2 Themenverfolgung

Wissenschaftliche Methoden [10, 14, 17, 18], die eine Themenverfolgung realisieren, stellen ein Thema oft durch Schlagwörter dar. Diese Schlüsselwörter werden dadurch extrahiert, dass die häufigsten Wörter eines Themas genommen werden [14], die TF-IDF Methode zur Gewichtung benutzt wird [10] oder die Wörter ausgewählt werden, die am wahrscheinlichsten in einem Thema vorkommen und am unwahrscheinlichsten in allen anderen Themen [17, 18].

Weiterhin gibt es einen Ansatz [8] einzelne Personen zu verfolgen. In diesem Ansatz geht es dann später um eine Visualisierung der Daten: Wie oft wurde die Person in den beobachteten Medienquellen in einem bestimmten Zeitintervall (beispielsweise an einem Tag) erwähnt.

Eine andere, sehr erfolgreiche Methode ist die Verfolgung von wörtlicher Rede [11] für ein bestimmtes Thema. Die Arbeit beabsichtigt zu erforschen, wie sich Themen zwischen den verschieden Medien (in diesem Fall Onlinenachrichten und Soziale Netzwerke) bewegen. Die Autoren untersuchen nach welcher Zeit Themen in die sozialen Netzwerke gelangen und ob es Themen gibt, die zuerst in den Sozialen Netzen entstehen und dann erst in die herkömmlichen Nachrichten gelangen. Hier werden Zitate aus wörtlicher Rede benutzt, da diese laut den Autoren einfach zu verfolgen sind [11]. Ein Zitat steht dann für ein Thema. Durch einen graphbasierten Ansatz werden Zitate auch wieder erkannt, wenn sie verkürzt oder leicht abgeändert werden.

Selten werden verschiedene Merkmale kombiniert [12], um Themen darzustellen oder zu verfolgen. Allerdings verlangt dies auch größeren Aufwand, da man zunächst mittels Information Extraktion Methoden viele Informationen im Vorlauf erfassen muss, damit man daraus entsprechende Merkmale generieren kann. Hier sind auch begrenzte Rechnerkapazitäten ein nicht zu vernachlässigender Aspekt.

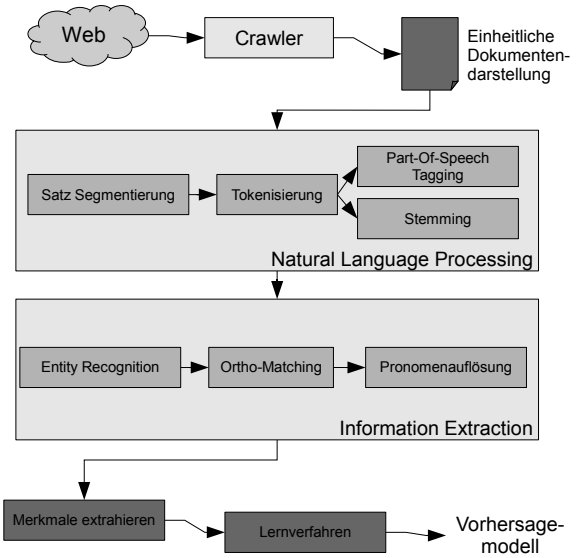


Abbildung 1: Ablauf der Verarbeitungsschritte

3. ANFORDERUNGEN FÜR DEN ANSATZ

Die verschiedenen Arbeiten [1, 2, 3, 7, 12] zu diesen Bereichen zeigen oft, dass es sinnvoll ist, die Texte einer Vorverarbeitung zu unterziehen (s. Abbildung 1). Sehr vorteilhaft erscheint der Einsatz von Natural Language Processing und Information Extraction. Wenn ein Ansatz diese Vorgaben für eine Vorverarbeitung erfüllt, dann entstehen neue Möglichkeiten, die später aufgeführt werden.

3.1 Natural Language Processing

Gerade Natural Language Processing (NLP) wird in vielen anderen Ansätzen benutzt, um unter anderem Adjektive zu identifizieren [1, 2, 3]. Beim Natural Language Processing ist nach dem simplen Aufteilen des Textes in Sätze und Wörter das sogenannte Part-Of-Speech Tagging der wichtigste Analyseschritt. Dabei werden die Wörter aufgrund von Wahrscheinlichkeitsmodellen wie Hidden Markov Modellen grammatikalischen Wortarten wie Nomen, Verben, Adjektiven usw. zugeordnet. Außerdem sollte ein Stemming durchgeführt werden, damit alle Wörter auch in Ihrer Grundform verfügbar sind. Mit dieser Zurückführung werden viele Methoden vereinfacht, die auf der Identifikation von bestimmten Wörtern beruhen oder einen Text als Wortlisten mit Häufigkeiten darstellen.

3.2 Information Extraction

Durch Information Extraction (IE) [15] ist es darüber hinaus möglich, Entitäten im Text wie Personen, Organisationen und Orte zu erkennen. Diese Named Entity Recognition (NER) ist ein gutes Beispiel, wie IE auf NLP aufbaut: Zuerst werden Nomen identifiziert und z.B. durch Listen genauer bestimmt, ob es eine Person ist und eventuell zusätzlich, ob es nur ein Vorname, Vor- und Nachname ist usw. Diese Entitäten werden dann über den Text verfolgbar, wenn weitere Techniken wie Ortho-Matching und eine Pronomenauflösung durchgeführt wird. Ortho-Matching beschreibt das Erkennen der Entität im selben Text an mehreren Stellen, wenn auch nicht exakt die selbe Zeichenkette verwendet wird. Im Falle einer Person könnte z.B. erst der komplette Name und später im Text nur noch der Nachname benutzt werden. Kommt eine Pronomenauflösung hinzu, dann wird eine Entität auch dann weiterverfolgt, wenn im Text für die Entität nur noch Pronomen stehen wie "sie" oder "ihn", die sich aber auf die entsprechende Person beziehen.

Darüber hinaus kann es noch sehr nützlich sein, weitere Informationen wie den grammatikalischen Fall oder das Geschlecht von Wörtern zu bestimmen. Auch Textpassagen mit wörtlicher Rede oder die Extraktion von Nomenphrasen sind weitere nützliche Informationsbausteine, die in einer weiteren Verarbeitung aufgegriffen werden können.

3.3 Neue Möglichkeiten

Diese Vorverarbeitung lässt sich für das Opinion Mining und die Themenverfolgung folgendermaßen einsetzen:

Durch die Bestimmung der verschiedenen Wortarten lassen sich für das Opinion Mining Wörterbücher aus Adjektiven, Verben und Adverben extrahieren. Hier stellt sich noch die Frage, wie man diesen Tonalitätswert bestimmt. Viele Ansätze bilden dazu Maße, die auf das Zusammenauftreten mit positiven bzw. negativen Tonalitätswörtern beruhen. Wenn man viele annotierte Meinungsblöcke besitzt, kann man auch Standardansätze aus dem Information Retrieval wie TF-IDF darauf anwenden. Ebenso kann man Bootstrapping einsetzen.

Auch Nominalphrasen können in dem Wörterbuch aufgenommen werden. Diese können auch für die Darstellung eines Themas von großem Nutzen sein.

Durch die Erkennung von Personen, Organisationen usw. kann man bei der Tonalitätsbestimmung unterscheiden, ob nun eine Person oder ein Produkt besprochen wird. Dadurch kann man die Bewertung des Vokabulars darauf anpassen.

Bei der Themenverfolgung kann man Themen durch die Anwesenheit von Entitäten beschreiben. Das Vorhandensein bzw. die Abwesenheit einer Entität kann ebenso wie Schlagwörter dazu benutzt werden ein Thema zu beschreiben und

damit auch zu verfolgen. Auch könnte man genauso wie bei Schlagwörtern das Auftreten mit TF-IDF gewichten, was nun konkret bedeuten würde: Die Häufigkeit einer Entität multipliziert mit der inversen Dokumentenfrequenz in der sie vorkommt. Bei dieser Gewichtung ist noch zu klären, was den gesamten Dokumentenkorpus darstellt. Dies könnte ein zeitlicher Ausschnitt sein (z. B. ein Monat).

Zusätzlich könnte diese Gewichtung interessante Informationen über die Entwicklung eines Themas liefern, weil es anzeigt welche Entitäten in welchen Themen eine starke Rolle spielen.

Dies kann man zusätzlich mit bisherigen Ansätzen für Themendarstellung durch Schlagwörter sowie die Verfolgung von wörtlicher Rede kombinieren.

Diese Vorverarbeitungsschritte benötigen natürlich auch Rechenzeit. Da allerdings dazu auf einem riesigen Datenfeld (den Texten) nur gelesen werden muss, ist eine Parallelisierung der Vorverarbeitung durchführbar.

4. EVALUATION

Um zu überprüfen, wie dieser Ansatz mit NLP und IE für Opinion Mining und eine Themenverfolgung eingesetzt werden kann, wird evaluiert, wie mit bestimmten Wortarten automatisch eine Tonalität bestimmt werden kann und ob mit Entitäten eine thematische Zuordnung möglich ist. Dafür werden zuvor klassifizierte Daten benutzt.

Diese Evaluation soll erste Hinweise geben, ob es grundsätzlich mit diesen Merkmalen möglich ist, die fundamentalen Bausteine einer Medienresonanzanalyse maschinell durchzuführen: Tonalitätsbestimmung und Themenzuordnung.

Dabei geht es auch weniger um die Bestimmmung des optimalen Lernverfahrens. Der Einfachheit halber wurden dazu im ersten Schritt drei typische Klassifikationsverfahren verwendet. Diese bieten sich an, weil die Daten schon vor der Evaluation mit entsprechenden Klassen versehen sind.

4.1 Tonalitätsbestimmung

Für diesen Test wurden 1600 Nachrichtenmeldungen mit 800 positiven und 800 negativen Meldungen analysiert. Um eine Tonalität zu erhalten, wurden mittels NLP Adjektive, Adverbien und Verben aus dem Text extrahiert und mittels Stemming auf ihre Stammform zurückgeführt. Danach wurden invertierte Listen von diesen Dokumenten erzeugt und die einzelnen Terme mittels TF-IDF gewichtet.

Nach der Erzeugung dieser Attribute wurden die Daten in einer 10-fach-über-Kreuz-Validierung durch drei Klassifikationsverfahren getestet: Support Vector Machine (SVM), Naive-Bayes und k-Nearest-Neigbours mit k=7.

Bei den Resultaten zeigte sich, das diese doch recht naive Methode (es wird beispielsweise nicht betrachtet, ob irgendwelche Negationen anwesend sind) tatsächlich erste brauchbare Hinweise geben kann.

Es zeigte sich, dass die Vermutung, nur Adjektive allein würden die Tonalität bestimmen, nicht zutrifft. Die Gruppe der Verben schneidet schon besser ab. Hier scheint der Unterschied zwischen Sentiment Analysis bezogen auf Kundenrezensionen und Opinion Mining bezogen auf Nachrichten deutlich zu werden. Kundenrezensionen beziehen ihre Tonalität wohl eher durch Adjektive ("die Bildqualität ist super" oder "der Autofokus ist zuverlässig")[1], während in Nachrichten dies nicht unbedingt der Fall ist ("Verbraucherschützer warnen vor der 'Umfrucht'-Falle" oder "Rekordjahr: Audi belohnt Mitarbeiter mit Millionen")[2].

Wortart	Klassifikationsverfahren	Genauigkeit
Adjektive		
	Support Vector Machine	80,81 %
	Naive-Bayes	68,34 %
	k-Nearst-Neighbour	53,60 %
Verben		
	Support Vector Machine	82,07 %
	Naive-Bayes	72,29 %
	k-Nearst-Neighbour	56,05 %
Adverbien		
	Support Vector Machine	75,61 %
	Naive-Bayes	66,08 %
	k-Nearst-Neighbour	53,79 %

Abbildung 2: Tonalitätsbestimmung

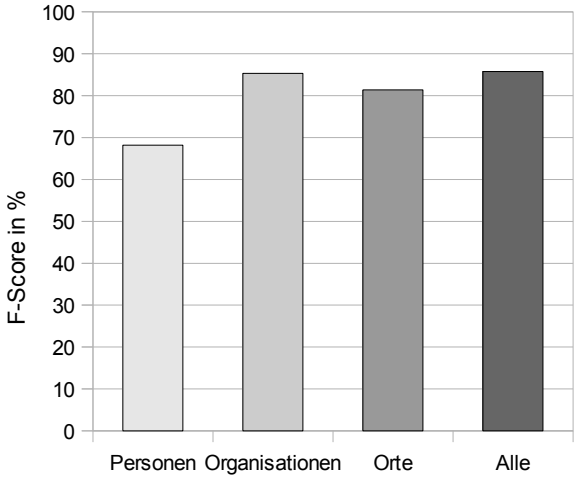


Abbildung 3: Themenzuordnung

Das beste Klassifikationsverfahren der drei hier getesteten Verfahren ist eindeutig die Support Vector Machine. Darum wird sie nun auch im zweiten Teil der Evaluation angewendet.

4.2 Themenzuordnung

Im zweiten Test wurden 204 Texte zu zwei sehr allgemeinen Themen gesammelt. Die Themen waren "Finanzmarkt" und "Parteien".

Dann wurden aus diesen Texten die Entitäten Personen, Organisationen und Orte extrahiert. Diese Entitäten wurden dann wiederum als Terme genommen, die mittels TF-IDF gewichtet wurden. Abschließend wurde eine SVM benutzt um die Daten in einer 10-fach-über-Kreuz-Validierung zu evaluieren.

Die Ergebnisse veranschaulicht Abbildung 3. Interessant ist hier, dass bei den einzelnen Merkmalen die Organisationen das beste Resultat liefern (ca. 85,29 %). Dies kann aber mit der Themenauswahl ("Finanzmarkt" und "Parteien") zu tun haben, in der wahrscheinlich generell mehr Organisatio-

[1]Beispiele aus einer Amazon.de Kundenrezension

[2]Beispiele von Spiegel.de am 4.3.2011

nen eine Rolle spielen bzw. die Organisationen das trennende Kriterium sind.

Auch Orte scheinen charakteristisch für Themen zu sein (ca. 81,37 %). Kaum überraschend spielt der Ort "Frankfurt" im ersten Thema eine wichtigere Rolle als im zweiten Thema und für "Berlin" ist es umgekehrt.

Das Merkmal Person ist nicht so erfolgreich (nur ca. 68,14 %). Bei den hier vorliegenden Themen gab es hinsichtlich der Personen auch durchaus Überschneidungen, weil mehrere Personen in beiden Themen auftauchten. Eine interessante Frage stellt sich nun dahingehend, ob dies bei kleineren Themen vielleicht seltener der Fall ist.

Insgesamt zeigt sich das wünschenswerte Resultat: Mit allen Entitäten gemeinsam wird das beste Ergebnis erzielt (ca. 85,78 %).

5. SCHLUSSFOLGERUNG UND WEITERFÜHRENDE FRAGESTELLUNGEN

Die Ergebnisse der Evaluation lassen darauf schließen, dass sich aufbauend auf dem beschriebenen Anforderungsprofil eine automatische Tonalitätsbestimmung und Themenverfolgung realisieren lässt.

Zu dem Aspekt des Opinion Minings fehlen noch viele Bestandteile, die in einem Text die Tonalität verändern können. Es hat sich gezeigt, dass die Worte allein schon im Ansatz funktionieren, aber noch großes Verbesserungspotenzial vorhanden ist.

Dazu ist zu erarbeiten, ob es noch bessere Methoden der Gewichtung gibt als der Standardansatz über TF-IDF. Außerdem muss überlegt werden, wie man die Tonalitätswörter beispielsweise in einem Wörterbuch verwalten kann. Als nächste Fragestellung schließt sich dann an, wie man mit semantischen Merkmalen wie Negation oder dem Bezug zu Entitäten umgeht.

Darüber hinaus ist ein weiteres spannendes Problem die Identifizierung der Meinungsblöcke, also der Textpassagen, die eine Meinung beinhalten. Ein Tonalitätsgrenzwert für Abschnitte und Sätze ist denkbar, aber auch die Lokalisierung durch die Entitäten im Text, für die man sich erstens verstärkt interessiert und die sich zweitens mit ausreichend vielen tonalitätsbildenen Wörtern umgeben.

Bei der Themenverfolgung haben die Experimente zunächst nur den Wert von Entitätenerkennung in einem einfachen Beispiel gezeigt. Hier müsste die Kombination mit klassischen Schlagwortansätzen und neueren Ansätzen, wie die Einbeziehung von wörtlicher Rede, genutzt werden, um eine bessere Themendarstellung zu erhalten und zusätzlich interessante Fakten über ein Thema zu sammeln. Diese Fakten können Folgendes beinhalten: Wie stark sind welche Personen mit welchen Themen verbunden? Oder gibt es zentrale Zitate/Aussagen, die immer wieder aufgegriffen werden.

Allerdings muss zunächst die Frage beantwortet werden, wie man die Entitäten sinnvoll mit Ansätzen wie Schlagwörtern und die Verfolgung von Zitaten verbinden kann. Dies wird Gegenstand der zukünftigen Arbeit sein, wobei auch zu klären ist, wie man diese Kombination für die Verwaltung einer Themenverfolgung sinnvoll einsetzen kann.

Weiterhin ist dabei die Größe eines Themas zu beachten (für die Defintion der Größe eines Themas gibt es viele Möglichkeiten, die Anzahl der Artikel zu einem Thema ist eine nahe liegende Lösung). Wie wirkt sich die Größe der Themen auf die Verwaltung aus? Und wie verhält sich die Themendarstellung mit Merkmalen dadurch? In der Evaluation kam schon die Frage auf, ob Personen bei kleineren Themen nicht eine wichtigere Rolle zur Themenbeschreibung spielen.

Insgesamt zeigt sich aber, dass die Vorverarbeitung durch Natural Language Processing und Information Extraction von großem Vorteil ist, da sie für beide Aufgabenstellungen, Opinion Mining und Themenverfolgung, viele neue Möglichkeiten eröffnet und diese im Ansatz für eine Medienresonanzanalyse funktionieren.

6. LITERATUR

[1] L. Dey and S. K. M. Haque. Opinion mining from noisy text data. In *Proc. of the 2nd workshop on Analytics for noisy unstructured text data*, AND '08, pages 83–90, 2008.

[2] X. Ding, B. Liu, and P. S. Yu. A holistic lexicon-based approach to opinion mining. In *Proc. of the international conference on Web search and web data mining*, WSDM '08, pages 231–240, 2008.

[3] X. Ding, B. Liu, and L. Zhang. Entity discovery and assignment for opinion mining applications. In *Proc. of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 1125–1134, 2009.

[4] W. Du and S. Tan. An iterative reinforcement approach for fine-grained opinion mining. In *Proc. of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 486–493, 2009.

[5] A. Esuli and F. Sebastiani. Determining the semantic orientation of terms through gloss classification. In *Proc. of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 617–624, 2005.

[6] X. Huang and W. B. Croft. A unified relevance model for opinion retrieval. In *Proc. of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 947–956, 2009.

[7] S.-M. Kim and E. Hovy. Automatic detection of opinion bearing words and sentences. In *Companion Volume to the Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, 2005.

[8] M. Krstajic, F. Mansmann, A. Stoffel, M. Atkinson, and D. A. Keim. Processing online news streams for large-scale semantic analysis. In *ICDE Workshops*, pages 215–220, 2010.

[9] L.-W. Ku, Y.-T. Liang, and H.-H. Chen. Opinion extraction, summarization and tracking in news and blog corpora. In *AAAI Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW)*, pages 100–107, 2006.

[10] S. Lee and H.-J. Kim. News keyword extraction for topic tracking. In *Proc. of the 4th International Conference on Networked Computing and Advanced Information Management - Volume 02*, pages 554–559, 2008.

[11] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proc. of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*,

KDD '09, pages 497–506, 2009.

[12] B. Li, W. Li, Q. Lu, and M. Wu. Profile-based event tracking. In *Proc. of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 631–632, 2005.

[13] M. M. S. Missen, M. Boughanem, and G. Cabanac. Comparing semantic associations in sentences and paragraphs for opinion detection in blogs. In *Proc. of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '09, pages 80:483–80:488, 2009.

[14] X. Tang, C. Yang, and J. Zhou. Stock price forecasting by combining news mining and time series analysis. In *Proc. of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '09, pages 279–282, 2009.

[15] J. Turmo, A. Ageno, and N. Català. Adaptive information extraction. *ACM Comput. Surv.*, 38, July 2006.

[16] P. D. Turney and M. L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21:315–346, October 2003.

[17] X. Wang, C. Zhai, X. Hu, and R. Sproat. Mining correlated bursty topic patterns from coordinated text streams. In *Proc. of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 784–793, 2007.

[18] J. Zeng, C. Wu, and W. Wang. Multi-grain hierarchical topic extraction algorithm for text mining. *Expert Syst. Appl.*, 37:3202–3208, April 2010.

Representing Perceptual Product Features in Databases

Joachim Selke
Institut für Informationssysteme
Technische Universität Braunschweig
Braunschweig, Germany

ABSTRACT

Many modern goods have both factual and perceptual features. While factual features such as technical specifications can easily be handled by existing database technology, perceptual features such as design or usage experience are very hard to deal with. However, with the huge success and growing market share of online shopping, retailers face the need to provided detailed and structured information about perceptual product features to their customers. In this paper, we analyze why dealing with perceptual product features in databases is difficult and summarize our current efforts on tackling this problem.

1. INTRODUCTION

Marketing theory distinguishes between two types of product features: Factual and perceptual ones [4, 8]. Factual features are those that can easily be named and specified. Typical factual features are technical specifications (e.g., length, height, and weight) and traditional publication metadata (e.g., authors, number of pages and year of publication). Perceptual features are those that usually are hard to describe and tend to involve an emotional reaction or physical contact to the respective product. Typical perceptual features are artistic or stylistic properties such as the mood of songs, the sophistication of novels, and the character depth in movies.

While factual product features can easily be represented and managed by existing database technology (e.g., by introducing a database attribute per feature), working with perceptual product features is much more complicated. This is mainly because perceptual features tend to be vague and defy precise definitions (e.g. the borders of literary genres). However, paradoxically, there are established ways to express perceptual features using natural language (e.g., *sporty* car or *clunky* cell phone), which surprisingly mostly are not a matter of taste but are based on general agreement. Therefore, we strongly believe that established database technology is indeed able and suited to store, process, analyze, and answer queries based on perceptual product features. We just have to find out how this can be done in practice.

In this paper, we survey existing approaches to handling perceptual product features in databases, point out their limitations, and summarize our own recent work towards solving this problem. In particular, we present a series of use cases illustrating the benefits of our approach.

In this following, we use movies as a running example. Movies are particularly suited for this task as they appeal to a wide range of people and provide a large variety of both factual and perceptual features. In addition, movies perfectly illustrate the problem of lacking support for perceptual features in databases: It has been shown that, when selecting movies, consumers rely far more on perceptual movie features (funny, romantic, scary, ...) than factual ones (actors, directors, release year, ...) [3]. However, the ideas and results presented in this paper can easily be transferred to other types of products.

2. EXISTING APPROACHES

In this section, we take a close look at existing approaches to handling perceptual product features in information systems. We identified three different groups of approaches: those based on explicit data provided by experts, those based on textual data provided by users, and special-purpose approaches that implicitly deal with perceptual product features. In addition, in domains where products can be represented in digital form (e.g., music or movies), (low-level) features can be extracted automatically.

2.1 Explicit Modeling by Experts

Besides the traditional classification of movies into a small number of major genres [2, 9], many movie databases recently adopted more refined classification schemes. While some just introduced a larger number of possible genres (e.g., the rental service Netflix[1] expanded its simple genre list into a taxonomy covering 485 genres), others decided to describe movies using generally applicable description attributes. Popular examples are the metadata provider AllMovie[2], which classifies its 440,000 movies with respect to more than 5,000 different moods, themes, tones, and types (e.g., Ensemble Film, Haunted By the Past, and Intimate), and the recommendation service Clerkdogs[3], which rates each movie with respect to 37 different attributes (e.g., Character Depth, Geek Factor, and Violence) on a 12-point scale. Essentially, all these approaches try to capture a movie's perceptual features by means of a set of predefined databases attributes, which can either contain binary values (as in AllMovie) or numbers (as in Clerkdogs).

Although this approach looks rather straightforward and seems to be easy to implement in practice, it comes with many problems. First of all, clearly identifying and narrowing down the most relevant

[1] `http://www.netflix.com`
[2] `http://www.allmovie.com`
[3] `http://www.clerkdogs.com`

23rd GI Workshop on Foundations of Databases
(Grundlagen von Datenbanken),
31.5.2011–3.6.2011, Obergurgl, Austria.

individual perceptual features tends to be difficult. However, even if a comprehensive and generally understandable classification system has been developed and experts have been trained how to use it correctly and consistently, manually classifying all movies is a huge amount of work.

An even worse problem is the actual consistency of these movie classifications. We recently compared the genre judgments made by three major movie databases and found that the agreement among them is moderate at best, being just slightly less directed towards to completely random genre assignments than to perfect agreement [11, 12]. As we restricted our analysis only to the most popular movie genres, even worse results can be expected for less established and/or more complex classification schemes.

2.2 Textual Descriptions by Users

An alternative approach to making perceptual movie features available to movie databases has been adopted by movie portals such as the Internet Movie Database[4] (IMDb) or Rotten Tomatoes[5] (RT). Instead of trying to represent movies in a structured fashion by means of explicit database attributes, they focus on textual descriptions, usually in the form of reviews provided by arbitrary users (IMDb) or (semi-)professional critics (RT).

Although textual descriptions give users a comprehensive and helpful characterization of each individual movie, it is difficult to search for movies or provide targeted movie recommendations given only textual data. One the the rare services offering movie search based on movie reviews is Nanocrowd[6], which applies information retrieval methods to extract so-called nanogenres from textual data. Each movie is characterized by a set of nanogenres, where each nanogenre is represented by a three-word group (e.g., sports/ballpark/loves or chemistry/adorable/formulaic). However, these nanogenres tend to be much less informative and understandable than explicit database attributes that have been manually created by experts.

Another drawback of text-based movie descriptions is the lack of data. While blockbusters are commented by a large number people, less popular movies often receive just a very small number of reviews, which tend to provide only a partial movie description and are too short to effectively apply methods of text analysis.

2.3 Implicit Modeling for Special Purposes

The third major approach is collaborative filtering as used in the area of recommender systems [1]. Here, the only data available about movies are numerical ratings provided by users (e.g., on a scale ranging from one to five stars), where each user assigns just a single number to each movie he rated. As rating movies is an almost effortless task, usually there is a large number of ratings from many different users available. For example, in IMDb, there are about a hundred times more ratings than reviews, while even relatively unknown movies still receive a substantial number of ratings.

So far, this kind of data has only been used for special problems such as similarity search (finding those movies that are most similar to a given one) or recommendations (providing a list of movies that are likely to appeal to a given user). Here, the basic idea is to analyze the ratings for systematic patterns indicating similar taste across a group of users or similar properties in a group of movies. For example, to provide recommendations to some user u, one might first look for other users who rated most of the movies rated by u in a similar way, and then recommend those movies to u that have been liked by most of these other users. In a way, movie features and user tastes are modeled implicitly when using collaborative filtering.

Recently, a series of recommendation algorithms has been developed that try to decompose the rating matrix (movies are rows, users are columns, and ratings are entries) into the product of two smaller matrices [6]. These so-called factor models have an important by-product, which usually is neglected by recommendation algorithms: the representation of each movie as points in some abstract coordinate space. Here, movies with similar coordinates tend to be rated similarly by different users, whereas users with very different coordinates tend to be perceived very differently. From this perspective, one can think of these coordinates as a embedding of movies into some abstract *semantic space*.

Our own analysis of the semantic spaces produced by recent recommender algorithms showed that these spaces indeed capture major perceptual features of movies [10, 11, 12]. However, the main problem of semantic spaces hindering their use for general purpose database applications is the total lack of intuitive understandability. To illustrate this problem, Table 1 show the first three dimensions of a 100-dimensional semantic space extracted from the Netflix Prize ratings data set[7] (about 20k movies, 500k users, and 100M ratings). For each dimension, we listed the those popular movies that received the five highest and five lowest scores with respect to this dimension. Clearly, these axes do not offer any intuitive interpretation. However, the relative positions in semantic spaces are indeed meaningful. To give an example, Table 2 shows the five nearest neighbors of three popular movies.

2.4 Content-Based Feature Extraction

In some domains, one can provide a (near-)complete description of each product in digital form. Prime examples are images, music, and movies. In these cases, it is possible to automatically derive so-called low-level features from the products itself, thus avoiding any dependence on external product descriptions. For example, common low-level features of images are color histograms, symmetry properties, and measures for contrast. Low-level features are contrasted by high-level features (concepts), which describe those aspects of content objects a user is interested in.[8] For example, high-level features of images are the types of objects (sun, beach, mother, child, ...), events (playing, talking, ...), or abstract concepts (family, fun, ...) associated with a photo. The multimedia content description standard MPEG-7 defines a large number of low-level features and also provides a language to annotate multimedia content with custom-defined high-level features.

In state-of-the-art content-based multimedia retrieval systems, low-level features are usually extracted automatically from the available content, whereas the use of high-level features tends to require a significant amount of human interaction. Although there are initial approaches to automatically derive selected high-level features from low-level features, there is still a large discrepancy between the limited information that one can extract from the available multimedia data and the interpretation that the same data has for users [7]. This problem is usually referred to as *semantic gap*.

When comparing content-based feature extraction to the three approaches discussed previously, we see that low-level features loosely correspond to semantic spaces and high-level features to explicitly modeled attributes. However, there are important differences:

- As low-level features must be extracted by means of spe-

[4] `http://www.imdb.com`
[5] `http://www.rottentomatoes.com`
[6] `http://www.nanocrowd.com`
[7] `http://www.netflixprize.com`
[8] Sometimes, the distinction into low-level and high-level features is refined to a 10-layer pyramid structure for classifying different feature types of multimedia content [5].

Axis	Popular high-scoring movies	Popular low-scoring movies
1	Indiana Jones and the Temple of Doom (1984), The Godfather (1972), American Pie (1999), Top Gun (1986), The Silence of the Lambs (1991)	Eternal Sunshine of the Spotless Mind (2004), Garden State (2004), Two Weeks Notice (2002), Bend It Like Beckham (2002), Miss Congeniality (2000)
2	Twister (1996), Titanic (1997), Lost in Translation (2003), Napoleon Dynamite (2004), Ghost (1990)	Ocean's Twelve (2004), Mission: Impossible (1996), Paycheck (2003), Anger Management (2003), Ocean's Eleven (2001)
3	The League of Extraordinary Gentlemen (2003), Chicago (2002), Van Helsing (2004), Steel Magnolias (1989), Ocean's Twelve (2004)	American Pie (1999), Big Daddy (1999), Mr. Deeds (2002), The General's Daughter (1999), Lethal Weapon 4 (1998)

Table 1: Popular movies receiving high and low scores on the first three coordinate axes.

Rocky (1976)	Dirty Dancing (1987)	The Birds (1963)
Rocky II (1979)	Pretty Woman (1990)	Psycho (1960)
Rocky III (1982)	Footloose (1984)	Vertigo (1958)
Hoosiers (1986)	Grease (1978)	Rear Window (1954)
The Natural (1984)	Ghost (1990)	North By Northwest (1959)
The Karate Kid (1984)	Flashdance (1983)	Dial M for Murder (1954)

Table 2: Three popular movies and their respective five nearest neighbors in semantic space.

cialized extraction algorithms, they are tied to a particular representation of the original content. Consequently, low-level features extracted from images cannot be compared to low-level features extracted from songs. In contrast, semantic spaces are derived from user feedback which can be provided for any product type in the same way, thus enabling the direct comparison of images and music. In addition, the design of effective low-level extraction algorithms is a complex task, which must be hand-crafted for each product domain under consideration.

- Semantic spaces are derived directly from human feedback (e.g., star ratings), which is turn is based on the most relevant perceptual product properties. Low-level features only capture statistical properties of the data representation such as color histograms. Therefore, the semantic gap between semantic spaces and user perception can be expected to be lower than the semantic gap present in current content-based multimedia retrieval systems.

For these reasons, we decided to put aside content-based feature extraction for the moment and focus on the three remaining approaches discussed above. However, in future work we plan to compare the ideas presented in this paper to existing methods from content-based multimedia retrieval where this is possible.

2.5 Conclusion

We can draw the following conclusions from the findings presented in this section:

- Modeling perceptual movie features by explicit attributes requires a huge amount of manual work but still leads to data of questionable quality. However, users can easily understand the meaning of these attributes.
- Capturing perceptual movie features by means of textual descriptions is helpful for users when looking for information about each individual movie. However, this kind of data is difficult to process automatically, cannot be understood as easily as explicit attributes, and the amount of available data is scarce for less popular movies.
- Semantic spaces created from a large number of user-provided ratings capture major perceptual features of movies. However, semantic spaces as such do not offer any intuitive interpretation and thus cannot be used to communicate with users.

3. PROPOSED SOLUTION

At first view, the result of our above analysis is rather disillusioning. Intuitively understandable models of perceptual movie properties are expensive to create and lack data quality, semantically meaningful models cannot be understood, and the third option seems to combine both disadvantages.

However, there is still hope. In [11] we introduced a data model that tries to combine the strengths of the approaches mentioned above. To be more precise, we propose to represent each movie by three different types of database attributes:

A. attributes describing factual movie properties,

B. attributes making a selected number of perceptual properties explicit (manual classification), and

C. attributes containing the movie's coordinates in some semantic space.

This approach brings several advantages. Probably most important is that the three different types of attributes can work together to reduce the weaknesses of each of them. In the following, we show a series of examples illustrating this idea (for technical details, please see [11]).

Enhancing the data quality in type-B attributes.

By aligning the manual classification of movies as expressed in type-B attributes to the semantic space, we are able to detect are large number of possibly misclassified movies. The basic idea is that movies that are classified into the same category should also be located close together in the semantic space. If we find a movie *m* that has the same value with respect to some type-B attribute but is very different from other movies having this value with respect to the semantic space, then *m* is likely to be misclassified by the experts. By identifying such movies and giving human experts

Figure 1: Genre clouds for Rocky (1976) and Star Trek (1979).

a chance carefully re-check problematic movies, the data quality can be increased. In our experiments on genre classifications [11], we have been able to detect possibly misclassified movies with a mean precision of about 55% and a mean recall of about 25%, which is significantly better than drawing random samples (the only alternative approach available). In summary, with the help of type-C attributes we are able to reduce a significant weakness of type-B attributes (data quality).

Saving manual work in creating type-B attributes.

To significantly reduce the amount of work required to manually classify all movies with respect to the type-B attributes, automatic classification can be applied. Here, given a binary type-B attribute (e.g., the genre *Action*), a human experts provides a small number (e.g., 10) of clearly positive examples (i.e., typical Action movies) and the same number of clearly negative examples (i.e., obvious non-Action movies). Using a support vector machine classifier that categorizes all remaining movies based on the training data and the type-C semantic space representation of movies, we have been able to produce results being only of slightly lower quality than those created by human experts [11]. By means of the method described previously, the data quality can easily be increased incrementally. In summary, with the help of type-C attributes we are able to reduce another significant weakness of type-B attributes (amount of work).

Enriching type-B attributes.

Again, by comparing type-B attributes to the semantic space represented by type-C attributes we are able to determine to what degree a type-B attribute value applies to each movie. For example, IMDb only assigns binary genre judgments to its movies, which leads to the classification Drama/Romance/Sport for the movie Rocky (1976) and Action/Adventure/Mystery/Sci-Fi for the movie Star Trek (1976). Although this classification is justified, there are several problems: Rocky contains romantic elements but it is a highly untypical Romance movie. It is most well-known for being a typical sports movie with dramatic activities. Similarly, Sci-Fi is widely recognized as Star Trek's most prominent genre, while it is a rather untypical Mystery movie. By analyzing the semantic space for where typical movies of genre *X* are located, we are able to judge how typical an assigned genre for each movie really is. To illustrate this, Figure 1 depicts a "genre clouds" for the above two movies. We automatically generated it from IMDb's binary genre assignments (type B) in combination with a semantic space extracted from ratings (type C) [11].

Enabling conceptual queries.

When describing their movie preferences, users often refer to factual movie properties as means attributes that approximately characterize an intuitive concepts that they are unable to express otherwise. For example, movies in the style typically associated with the director Quentin Tarantino could be called *Tarantino-ish* movies. In fact, Google counts 4530 Web pages mentioning this term. We refer to database queries in this style as conceptual queries. We are able to answer such queries by first finding out where movies directed by Quentin Tarantino are typically located in the semantic space (by identifying a small continuous region in space), and then looking for other movies that are located close to the center to this region. By applying a simple weighting scheme, we are able to produce a, say, top-10 list of the most Tarantino-ish movies. To give an example, Table 3 shows our results for Tarantino and two popular actors. Here, we used a support vector machine to learn where movies directed by Tarantino tend to be located in semantic space and used this information to find very similar movies that have not been directed by Tarantino [11]. Apart from minor exceptions (in particular, The Professional and Dragon: The Bruce Lee Story), these results look very promising. In summary, we have been able to understand users' implicit concepts by of mapping type-A attributes to the semantic space.

4. CONCLUSION AND OUTLOOK

In this paper, we have discussed the problem of representing perceptual product features in databases. We concluded that each existing approach alone does not provide an acceptable solution to this problem as it comes with severe disadvantages. However, by combining several methods into a joint data model, we have been able to reduce the weaknesses of each individual approach and boost its strengths. Our examples show promising results, which we are going to analyze in detail in future work. In addition, as already indicated in Section 2.4, we plan to compare our work to approaches from content-based multimedia retrieval. For example, for genre classification tasks, it would be interesting to compare semantic spaces derived from ratings to low-level features extracted from the actual movies.

5. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] D. Chandler. An introduction to genre theory, 1997. Available from `http://www.aber.ac.uk/media/Documents/intgenre`.

[3] E. Cooper-Martin. Consumers and movies: Some findings on experiential products. In *Advances in Consumer Research*, volume 18, pages 372–378. 1991.

[4] E. C. Hirschman and M. B. Holbrook. Hedonic consumption: Emerging concepts, methods and propositions. *Journal of Marketing*, 46(3):92–101, 1982.

[5] C. Jörgensen, A. Jaimes, A. B. Benitez, and S.-F. Chang. A conceptual framework and empirical research for classifying visual descriptors. *Journal of the American Society for Information Science and Technology*, 52(11):938–947, 2001.

[6] Y. Koren and R. Bell. Advances in collaborative filtering. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

[7] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, 2007.

[8] P. Nelson. Information and consumer behavior. *Journal of Political Economy*, 78(2):311–329, 1970.

[9] C. Preston. Film genres. In W. Donsbach, editor, *The International Encyclopedia of Communication*. Blackwell, 2008.

[10] J. Selke and W.-T. Balke. Extracting features from ratings: The role of factor models. In *Proceedings of M-PREF 2010*, pages 61–66, 2010.

Stallone-ish	Tarantino-ish	Jim Carrey-ish
Universal Soldier (1992)	True Romance (1993)	EDtv (1999)
Commando (1985)	GoodFellas (1990)	Innerspace (1987)
Missing in Action (1984)	The Usual Suspects (1995)	Bedazzled (2000)
Red Heat (1988)	Casino (1995)	Cadillac Man (1989))
Raw Deal (1986)	Desperado (1995)	Pleasantville (1998)
Bloodsport (1988)	The Professional (1994)	Dragon: The Bruce Lee Story (1993)
The Last Boy Scout (1991)	Killing Zoe (1994)	Honey, I Shrunk the Kids (1989)
The Running Man (1987)	Full Metal Jacket (1987)	Alive (1993)
Kickboxer (1989)	2 Days in the Valley (1996)	Shallow Hal (2001)
The Delta Force (1986)	Go (1999)	Punchline (1988)

Table 3: Top 10 results for three different conceptual queries.

[11] J. Selke and W.-T. Balke. TEAMWORK: A data model for experience products. ifis technical report, Institut für Informationssysteme at Technische Universität Braunschweig, 2011.

[12] J. Selke, S. Homoceanu, and W.-T. Balke. Conceptual views for entity-centric search: Turning data into meaningful concepts. In *Proceedings of BTW 2011*, pages 327–346, 2011.

Acknowledgments

I am very grateful to Prof. Dr. Wolf-Tilo Balke for providing valuable guidance and supervising my doctoral thesis, which will be based partly on the work presented in this paper.

Echtzeitüberwachung und Langzeitanalyse mittels eingebetteter Systeme

Tino Noack
TU Cottbus
Institut für Informatik, Informations- und Medientechnik
Lehrstuhl Datenbank- und Informationssysteme
Tino.Noack@tu-cottbus.de

Kurzfassung

Der vorliegende Beitrag skizziert ein interdisziplinäres Forschungsvorhaben im Rahmen einer Doktorarbeit. Einer der Forschungsbeiträge ist die Kombination von Echtzeitüberwachung und Langzeitanalyse. Diese Kombination basiert auf existierenden Ansätzen und umfasst Event-Condition-Action-Regeln (ECA-Regeln), Data-Mining-Technologien sowie Complex Event Processing (CEP). Im vorliegenden Beitrag werden zunächst drei grundlegende Annahmen und fünf Überwachungsanforderungen erarbeitet. Darauf aufbauend wird die Forschungsfrage detailliert betrachtet. Die Grundlage für die vorgestellte Idee bildet ein mathematisches Modell (der Zustandsraum), welches das Wissen über das zu überwachende System repräsentiert. Mit Hilfe dieses Zustandsraums werden durch die Anwendung von Data-Mining-Technologien ECA-Regeln erzeugt und an eine CEP-Anwendung, die sich auf einem eingebetteten System befindet, übertragen. Dieser Teilschritt bezieht sich auf die Langzeitanalyse. Die CEP-Anwendung wertet anschließend die übertragenen ECA-Regeln auf einem kontinuierlichen Strom von Sensordaten aus und erzeugt Aktionen. Dieser Teilschritt bezieht sich auf die Echtzeitüberwachung. Weiterhin wird eine Prozesskette vorgestellt, die zyklisch durchlaufen wird und zur Kombination von Echtzeitüberwachung und Langzeitanalyse dient. Hier wird ein dynamischer und flexibler Überwachungsansatz vorgestellt.

Schlüsselwörter

Überwachung, Echtzeit, Langzeit, Eingebettete Systeme, Datenströme, Data Mining, Complex Event Processing

1. EINLEITUNG

Viele Produkte, in denen sich eingebettete Systeme verbergen, sind sicherheitskritisch und unterliegen Echtzeitanforderungen wie z.B. Kraft-, Schienen-, Luft- oder Raumfahrzeuge. Eingebettete Systeme werden oft für Regelungs-, Kontroll- und Überwachungsfunktionalitäten eingesetzt. Die Überwachung technischer Systeme ist ein sehr weit verbreitetes Forschungsfeld und bezieht sich auf viele heterogene Anwendungsdomänen. Häufig werden Überwachungssysteme für spezielle Anwendungen entworfen, entwickelt und implementiert. Dies führt zu erhöhten Entwicklungskosten und gleichzeitig zur Abnahme der Flexibilität bzw. der Wiederverwendbarkeit. Bedeutende Anwendungen sind z.B. die Überwachung von Raumfahrzeugen [21], [22] oder die Überwachung von Schienenfahrzeugen [16]. Die Überwachung von Raumfahrzeugen ist besonders herausfordernd, da komplette Systemtests in der vorgesehenen Systemumwelt (dem Weltraum) und kontinuierliche Wartung unpraktisch bzw. unmöglich sind.

Aufgrund der steigenden Komplexität heutiger Produkte werden verbesserte Überwachungsansätze benötigt, die heutige und zukünftige Anforderungen berücksichtigen. Im vorliegenden Beitrag steht die Überwachung des zu überwachenden Systems, welches im Weiteren als Produkt bezeichnet wird, im Vordergrund. Das Produkt besteht aus einer Menge von Systemkomponenten. Nur aufgrund des Zusammenspiels der einzelnen Systemkomponenten untereinander genügt das Produkt einer vorher definierten Funktion bzw. Aufgabe. Zusätzlich wirken externe Einflüsse aus der umgebenden Produktumwelt auf das Produkt (vgl. [15], [17]). Somit bezieht sich die Überwachung des Produkts je nach Anwendungsdomäne und je nach Überwachungsziel zusätzlich auf externe Einflüsse und auf die korrekte Arbeitsweise der beteiligten Systemkomponenten. Eine strikte Trennung der Überwachung externer Einflüsse, des Produkts selbst und der einzelnen Systemkomponenten, aus denen das Produkt besteht, kann nicht immer vollzogen werden.

Der vorliegende Beitrag skizziert ein interdisziplinäres Forschungsvorhaben im Rahmen einer Doktorarbeit. Einer der Forschungsbeiträge ist die Kombination von existierenden, gut bekannten und bereits praktisch angewendeten Ansätzen, die für die Kombination von Echtzeitüberwachung und Langzeitanalyse eingesetzt werden können. Anhand des Einsatzes von existierenden Ansätzen sind Einsparungen im Bereich der Entwicklungskosten möglich. Das Forschungsvorhaben umfasst die Erstellung von Event-Condition-Action-Regeln (ECA-Regeln) [10], Data-Mining-Technologien [27] sowie Complex Event Processing (CEP) [12]. Hier wird ein dynamischer und flexibler Überwachungsansatz vorgestellt, der auf den drei folgenden Annahmen basiert:

1. Anwendungsübergreifend werden ähnliche Methodiken und Algorithmen für die Überwachung technischer Systeme eingesetzt.

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

2. Das Auftreten von Fehlern im laufenden Betrieb lässt sich nicht ausschließen. Daher muss durch die Änderung des Systemverhaltens so schnell wie notwendig eine angemessene Aktion ausgelöst werden.

3. Teile des gesamten Überwachungsprozesses sind semimanuell. Informationssysteme werden nur zur Unterstützung des Überwachungsprozesses angewendet.

Der Rest des vorliegenden Beitrags ist wie folgt organisiert. Kapitel 2 beschreibt ein Anwendungsbeispiel. In Kapitel 3 wird der Begriff eines eingebetteten Systems definiert, so wie es für die Forschungsarbeit verwendet wird. Kapitel 4 fasst Anforderungen an die Überwachung zusammen und aufbauend darauf wird in Kapitel 5 die Forschungsfrage detailliert betrachtet. Kapitel 6 beschreibt das Systemmodell, welches dem vorgeschlagenen Überwachungsansatz zu Grunde liegt. In Kapitel 7 wird der vorgeschlagene Überwachungsansatz detailliert beschrieben. Kapitel 8 fasst existierende Lösungen zusammen und schließlich wird in Kapitel 9 eine Zusammenfassung gegeben.

2. ANWENDUNGSBEISPIEL: ZUGUNGLÜCK VOM ICE 884 IN ESCHEDE

Das Zugunglück vom ICE 884 in Eschede ist ein sehr praxisnahes Anwendungsbeispiel. Die Hauptursache des katastrophalen Zugunglücks war der Bruch eines gummigefederten Radreifens. Dieser Bruch war die Folge von langfristigen Verschleißerscheinungen (z.B. Verringerung der Radreifendicke und Korrosion). Bereits einige Monate vor dem Unglück wurden während der Wartung anomale Messwerte an dem besagten Radreifen festgestellt. Die detaillierte Bruchflächenanalyse stellte heraus, dass die langfristigen Verschleißerscheinungen zu einem Riss in dem Radreifen, lange vor dem Unglück, führten. Der Bruch des Radreifens führte zur Entgleisung des Zuges ([24], [13]). Der beschriebene Anwendungsfall deutet auf langfristige und auf kurzfristige Einflussfaktoren hin. Verschleißerscheinungen sind langfristige Einflussfaktoren und der Bruch des Radreifes bzw. die Zugentgleisung sind kurzfristige Einflussfaktoren.

Durch die Langzeitanalyse können langfristige Verschleißerscheinungen erkannt, analysiert und bewertet werden. Der Bruch des Radreifens und die nachfolgende Entgleisung des Drehgestells haben zu einer plötzlichen und signifikanten Veränderung des Systemverhaltens geführt (bspw. Schlingerbewegung des entgleisten Drehgestells). Anhand der Anwendung der Echtzeitüberwachung mittels eines eingebetteten Systems kann diese plötzliche Veränderung des Systemverhaltens erkannt und in einem angemessenen Zeitraum eine Aktion (z.B. Notbremsung) durchgeführt werden.

3. EINGEBETTETES SYSTEM

Abbildung 1 skizziert die abstrakte Architektur eines eingebetteten Systems, wie sie hier Einsatz findet. Eingebettete Systeme sind in ein umgebendes Produkt eingebettet. Das Produkt ist in eine Produktumgebung eingebettet. Eingebettete Systeme enthalten elektronische Baugruppen (Hardware), die die Systemkomponenten repräsentieren. Zusätzlich sind diese elektronischen Baugruppen mit Software ausgestattet. Eingebettete Systeme unterliegen eingeschränkten Systemressourcen wie z.B. Prozessorleistung, Strom- und Speicherverbrauch. Das eingebettete System steht mittels

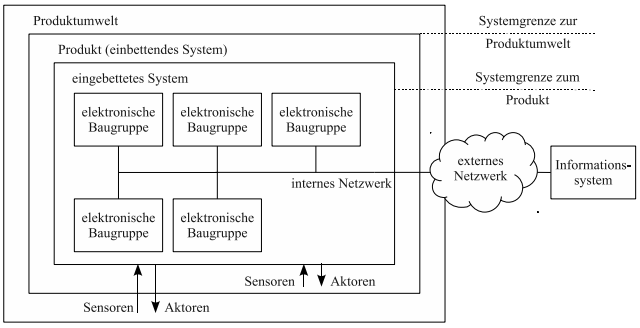


Abb. 1: Eingebettetes System

Sensoren und Aktoren mit dem Produkt und der Produktumgebung in Interaktion. Die elektronischen Baugruppen können mittels eines internen Netzwerkes miteinander verbunden sein. Zusätzlich kann eine temporäre Verbindung zu einem externen Informationssystem vorhanden sein. Weitere Informationen zu eingebetteten Systemen finden sich u.a. in [28], [20] und [23].

4. ÜBERWACHUNGSANFORDERUNGEN

Entsprechend des vorgestellten Anwendungsbeispiels und in Anbetracht der abstrakten Architektur eines eingebetteten Systems werden hier folgende fünf Überwachungsanforderungen erarbeitet: Zeit, Lokalität, Wissen, Systemressourcen und Schärfe. Abbildung 2 fasst die genannten Anforderungen zusammen.

Zeit: Diese Anforderung bezieht sich auf die zeitliche und kontinuierliche Veränderung der Bauteile.

- *kurzfristig*: Es können plötzliche Änderungen der Bauteile (z.B. Bruch des Radreifens) auftreten. Es ist notwendig, diese in Echtzeit zu erkennen.
- *langfristig*: Zur Erkennung langfristiger Einflussfaktoren und Veränderungen (z.B. Verschleiß und Alterung) sind Langzeitanalysen notwendig.

Lokalität: Diese Anforderung bezieht sich auf Wechselwirkungen der Einflussfaktoren und die räumliche Lokalität der Überwachung.

- *lokal*: Fehler, die sich z.B. auf wenige Bauteile beziehen, müssen durch eine lokale Überwachung erkannt werden.
- *global*: Aufgrund der steigenden Komplexität von Produkten und eingebetteter Systeme korrelieren die Einflussfaktoren zunehmend. Somit entstehen komplexe Zusammenhänge zwischen den Bauteilen, die durch eine globale Analyse erfasst und erkannt werden müssen.

Wissen: Diese Anforderung bezieht sich auf das vorhandene Wissen über das eingebettete System, das Produkt und die Produktumwelt.

- *bekannt*: Es ist notwendig das bekannte Wissen über das eingebettete System, das Produkt und die Produktumwelt möglichst umfassend und zielorientiert für die Überwachung einzusetzen.
- *unbekannt*: Aufgrund unbekannter bzw. unvorhersehbarer Umstände ist ein dynamischer und flexibler Überwachungsprozess notwendig.

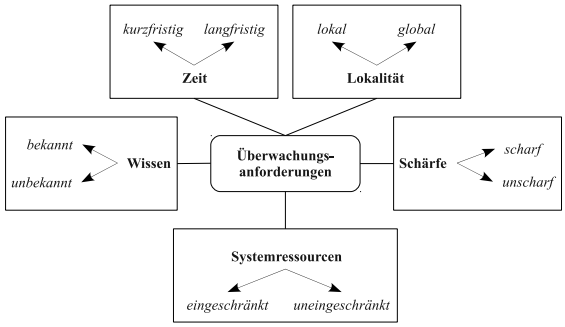


Abb. 2: Überwachungsanforderungen

Systemressourcen: Diese Anforderung bezieht sich auf die vorhandenen Ressourcen, die für die Überwachung zur Verfügung stehen.

- *uneingeschränkt*: Die Überwachung von Systemen benötigt äußerst viele Systemressourcen. Somit ist eine Kombination von interner und externer Überwachung (hybrides Überwachungssystem [26]) notwendig, um ausreichend Ressourcen für die Überwachung zur Verfügung zu stellen.
- *eingeschränkt*: Aufgrund der eingeschränkten Systemressourcen eingebetteter Systeme ist es notwendig, diese angemessen und zielführend für die Überwachung einzusetzen.

Schärfe: Diese Anforderung bezieht sich auf die Auswertung von Bedingungen (vgl. [25], [4]).

- *scharf*: Systemzustände müssen exakt und zuverlässig durch eine exakte binäre Auswertung von Bedingungen (Boolesches Modell) erkannt werden.
- *unscharf*: Diese scharfe Grenze zwischen Systemzuständen ist nicht immer gegeben. Um dies zu berücksichtigen, wird die exakte binäre Auswertung mittels Zugehörigkeitsgrade zwischen 0 und 1 verallgemeinert. Der Wert 1 wird als volle Zugehörigkeit und der Wert 0 als nicht zugehörig interpretiert.

5. FORSCHUNGSFRAGE

Es gibt eine Lücke zwischen Echtzeitüberwachung und Langzeitanalyse von Ereignissen, die die Zuverlässigkeit von Produkten beeinträchtigen. Dies ist die Motivation für unsere Forschung an der Kombination von Echtzeitüberwachung und Langzeitanalyse von Ereignissen. In dem ersten Schritt werden hier alle Überwachungsanforderungen außer der Schärfe betrachtet. Abbildung 3 fasst die Forschungsfrage grafisch zusammen.

Langzeitanalyse benötigt meist sehr viele Systemressourcen. Zusätzlich sind Data-Mining-Technologien semi-manuell und müssen durch Fachpersonal betreut und gepflegt werden. Aus diesem Grund ist eine Offlineverarbeitung auf einem externen Informationssystem mit nahezu uneingeschränkten Systemressourcen notwendig. Data-Mining-Technologien werden hier für das Erlernen von Klassifikatoren eingesetzt, die anschließend durch ECA-Regeln repräsentiert werden. Die persistent gespeicherten Daten umfassen alle gesammelten Attribute und geben eine globale Sicht über das gesamte zu überwachende Produkt. Dabei ist die Anzahl der Attribute je nach Anwendungsdomäne und Überwachungsziel unterschiedlich. Diese Daten können zur Identifikation von relevanten Wechselwirkungen Verwendung finden. Data-Mining-Technologien werden eingesetzt, um das Wissen über das Produkt mit der Zeit zu erhöhen.

In Bezug zum genannten Anwendungsbeispiel werden die Daten mittels eines externen Informationssystems gesammelt. Diese persistent gespeicherten Daten werden eingesetzt, um einen Klassifikator zu erlernen, der zwischen bekanntem und unbekanntem Verhalten des Zuges unterscheiden kann. Dies wird in [8] als Anomalieerkennung bezeichnet. Weiterhin können diese gespeicherten Daten zur Erkennung gradueller Änderungen der Systemkomponenten in Bezug zur Zeit genutzt werden. Somit können langfristige Einflussfaktoren wie z.B. Verschleiß erkannt werden.

Echtzeitüberwachung wird auf dem eingebetteten System durchgeführt. Dieses unterliegt eingeschränkten Systemressourcen. Die Echtzeitüberwachung wird automatisch, online und ohne Benutzerinteraktion durchgeführt. Plötzliche Änderungen des Systemverhaltens müssen so schnell wie notwendig erkannt werden. Anschließend ist eine angemessene Aktion notwendig. Die gelernten Klassifikatoren bzw. ECA-Regeln werden hier zum eingebetteten System übertragen und anschließend zur Erkennung von Änderungen des Systemverhaltens bzw. zur Anomalieerkennung eingesetzt. CEP ist hier ein ausgewähltes Werkzeug, um die ECA-Regeln auf den kontinuierlichen Datenströmen anzuwenden. ECA-Regeln repräsentieren das Wissen über das Produkt. Verhalten, welches nicht zu diesen Regeln passt, kann als unbekannt bzw. anomal gekennzeichnet werden. Dies ist ein lokaler Aspekt, da nur eine Teilmenge der vorhandenen Attribute für die Definition eines speziellen Verhaltens mittels ECA-Regeln Verwendung findet. Wie auch bei der Langzeitanalyse ist die Anzahl der Attribute je nach Anwendungsdomäne und Überwachungsziel unterschiedlich, aber geringer als für die Anwendung der Langzeitanalyse.

In Bezug zum genannten Anwendungsbeispiel stellen das Brechen des Radreifens und die anschließende Entgleisung signifikante und plötzliche Änderungen der Fahreigenschaften des Zuges dar. Nachfolgend wird in Bezug zum Anwendungsbeispiel der ECA-Ansatz kurz erläutert. Ein Ereignis (Event) ist hier das Verhalten des Zuges zu einer bestimmten Zeit. Die Bedingung (Condition) bezieht sich auf die gelernten Klassifikatoren bzw. die Regeln, die ermittelt wurden, um das Verhalten des Zuges zu einem bestimmten Zeitpunkt zu klassifizieren. Eine Aktion (Action) kann bspw. die Verringerung der Geschwindigkeit des Zuges oder das Auslösen der Notbremse sein, um materielle Schäden und menschliche Opfer zu vermeiden.

6. SYSTEMMODELL

Ein wesentlicher Punkt ist das Verständnis der Eingangsdaten. Sensoren erzeugen kontinuierliche Daten. Diese kontinuierlichen Sensordaten werden hier als Datenströme interpretiert. Ein Datenstrom besteht aus einer Sequenz von Datenelementen. Häufig ist diese Sequenz sehr lang. Ein System, welches Datenströme verarbeitet, hat keine A-Priori-Kontrolle über die Reihenfolge der eintreffenden Datenelemente. Die erneute Übertragung von verlorenen Datenelementen ist nicht möglich. Weitere Informationen über Datenströme finden sich u.a. in [1], [7], [3] und [14].

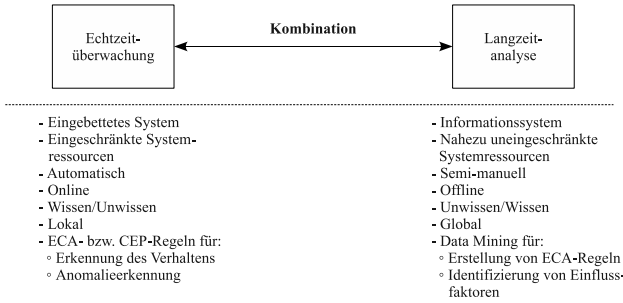


Abb. 3: Kombination von Echtzeitüberwachung und Langzeitanalyse

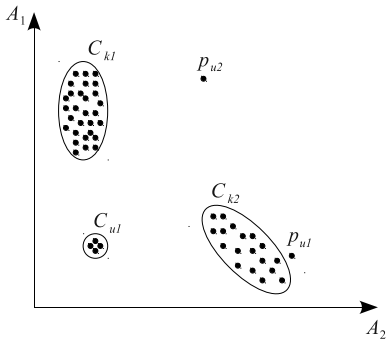


Abb. 4: Zustandsraum [8]

Die Menge von Eigenschaften, die das Zielsystem beschreiben, wird hier als eine Menge von Attributen $A_1, ..., A_n$ interpretiert. Diese Attribute können u.a. nominal, ordinal oder metrisch sein. Attributwerte sind Funktionen der Zeit, so dass Werte von A_i einer Funktion $a_i : T \rightarrow \mathbb{R}$ entsprechen. Dabei ist T die Zeit und $\mathbb{R}$ die Menge der reellen Zahlen. Somit ist ein Zustand in Bezug zur Zeit ein Zustandsvektor

$$\vec{a}(t) = \begin{pmatrix} a_1(t) \\ a_2(t) \\ \vdots \\ a_n(t) \end{pmatrix}.$$

Der Raum, der durch die Attribute aufgespannt wird, heißt Zustandsraum. Die Anzahl der Attribute definiert die Anzahl der Dimensionen des Zustandsraums. Eine Menge von Zustandsvektoren im Zustandsraum, die ähnliche Arten von Zuständen repräsentieren, können geometrisch interpretiert werden. Diese geometrische Interpretation wird im Rahmen von Data-Mining-Technologien als Cluster bezeichnet ([27], [8], [5], [2]).

Abbildung 4 veranschaulicht den Zustandsraum in einem Zeitfenster unter Berücksichtigung der zwei Attribute A_1 und A_2. Zur besseren Übersichtlichkeit sind die Zustandsvektoren als Punkte dargestellt. Sei S die Menge aller möglichen Systemzustände bzw. der gesamte Zustandsraum, C_k die Menge der bekannten Cluster und C_u die Menge aller unbekannten Cluster, so dass $C_k \bigcup C_u = S$ und $C_k \bigcap C_u = \varnothing$. Somit sind bekannte Cluster komplementär zu unbekannten Clustern. In Abbildung 4 repräsentieren die Cluster C_{k1} und C_{k2} Mengen von bekannten Systemzuständen. Der Cluster C_{u1} sowie die Punkte p_{u1} und p_{u2} stehen exemplarisch für unbekannte Systemzustände. In [8] werden diese unbekannten Systemzustände als Anomalien bezeichnet. Das Ziel des gelernten Systemmodells ist die Klassifizierung eines Zustandsvektors zu einem Zeitpunkt t zu einem bekannten Cluster. Kann dieser Zustandsvektor keinem bekannten Cluster zugeordnet werden, so ist dieser Zustandsvektor unbekannt und wird als eine Anomalie gekennzeichnet. Somit repräsentieren die ECA-Regeln den Klassifikator, der mittels der Data-Mining-Technologien erlernt wurde.

7. KOMBINATION VON ECHTZEITÜBERWACHUNG UND LANGZEITANALYSE

Wie bereits beschrieben, liegt die Kombination von Echtzeitüberwachung und Langzeitanalyse im Fokus des Interesses. Ziel ist es, ein Modell bzw. einen Zustandsraum zu erlernen, der das Wissen über das Produkt repräsentiert. Zunächst wird in diesem Kapitel eine Prozesskette für die Überwachung beschrieben. Anschließend wird diese Prozesskette in eine abstrakte Überwachungsarchitektur überführt.

Die Prozesskette ist in Abbildung 5 grafisch verdeutlicht. Sie ist in zwei Teile gegliedert. Der obere Teil repräsentiert die Echtzeitüberwachung auf dem eingebetteten System. Der untere Teil repräsentiert die Langzeitanalyse auf einem externen Informationssystem. Zur besseren Übersichtlichkeit ist die untere Teilkette in umgekehrter Reihenfolge dargestellt.

Im ersten Schritt startet die Prozesskette mit Ereignissen bzw. Zustandsvektoren. Die Vorverarbeitung ist der zweite Schritt. Dieser kann u.a. zur Filterung, zur Selektion oder für Fensterfunktionen, zur Verringerung des Verarbeitungsaufwands, Verwendung finden. Die Ausführung der Regeln ist der dritte Schritt. In diesem dritten Schritt werden die im Voraus definierten Regeln auf dem Datenstrom angewendet. Der vierte Schritt umfasst das Senden von Nachrichten zu den Aktoren. Der fünfte Schritt wird für die temporäre Speicherung verwendet. Das schließt Datenaggregation zur Minimierung des Speicherbedarfs sowie angemessene Speicherstrategien wie z.B. Ringpuffer oder eingebettete Datenbanken mit ein. Der letzte Schritt der obersten Teilkette bezieht sich auf das Senden der Daten vom eingebetteten System zum stationären System. Aufgrund der temporären Verbindung mittels des externen Netzwerkes können die Daten nur von Zeit zu Zeit an das externe Informationssystem gesendet werden. Die genannten Schritte sind automatisch. Es ist notwendig, dass jeder Teilschritt austauschbar und konfigurierbar (z.B. Plug-in-System) ist, um einen dynamischen und flexiblen Überwachungsansatz bereitzustellen. Somit ist es möglich, das CEP-System auf die vorhandene Hardware und den beabsichtigten Überwachungszweck zuzuschneiden.

Der erste Schritt der Langzeitanalyse betrifft das Laden der empfangenen Daten vom eingebetteten System in ein persistentes Datenverzeichnis wie z.B. ein Data Warehouse (DWH). Der zweite Schritt umfasst die Erstellung der Regeln mittels Data-Mining-Technologien. Dazu gehört die Integration der empfangenen Daten in den Zustandsraum. Dabei steigt das Wissen über das Produkt durch die Integration von neuen und noch unbekannten Zustandsvektoren in den Zustandsraum. Aktuell werden hier folgende Algorithmen zur Klassifikation bzw. überwachtes Lernen eingeschlossen: Regelinduktion, Support Vector Machine und k-nächste Nachbarn. In vielen Fällen müssen die genannten Algorithmen ebenfalls kombiniert werden, um einen ange-

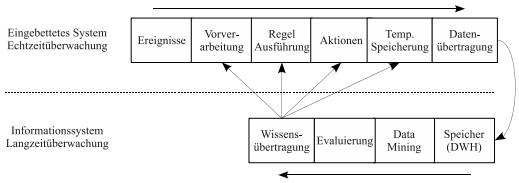


Abb. 5: Prozesskette

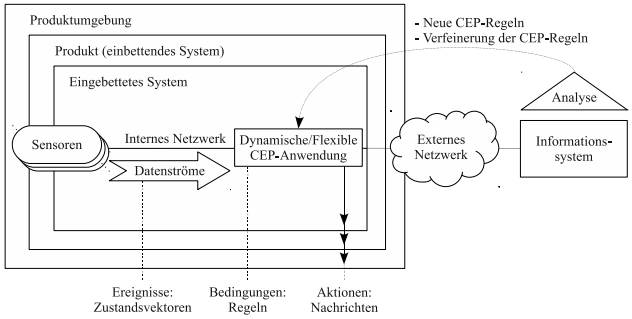


Abb. 6: Überwachungsarchitektur

messenen Klassifikator bereitzustellen ([8], [27]). Der dritte Schritt dient zur Evaluierung der neu ermittelten Regeln und zum Testen mit vorhandenen Regeln, um evtl. Seiteneffekte auszuschließen. Der letzte Schritt der unteren Teilkette betrifft die Übertragung des so ermittelten Wissens zum eingebetteten System. Dies schließt die Anpassung und die Rekonfiguration des bestehenden Überwachungssystems auf Basis des neuen Wissens mit ein. Die genannten Schritte sind semi-manuell und werden durch Fachpersonal betreut.

Die komplette Prozesskette wird zyklisch durchlaufen. So kann mit der Zeit das Wissen über das zu überwachende Produkt gesteigert werden.

Die vorgeschlagene Überwachungsarchitektur ist in Abbildung 6 grafisch verdeutlicht. Sie basiert auf der Prozesskette, die bereits beschrieben wurde. Sensoren erzeugen kontinuierlich Datenströme, die über das interne Netzwerk übertragen werden. Es ist notwendig, diese Ereignisse bzw. Zustandsvektoren kontinuierlich unter Berücksichtigung von Echtzeitbedingungen zu verarbeiten. Das CEP-System muss entsprechend der festgelegten Regeln Aktionen auslösen. Weiterhin wird der Datenstrom aggregiert und temporär gespeichert, bevor er zum externen Informationssystem übermittelt wird. Das externe Informationssystem wird für die Langzeitanalyse und zur Ermittlung neuer bzw. zur Verfeinerung bestehender Regeln eingesetzt. Anschließend ist die Evaluierung der Regeln und die Übertragung zum eingebetteten System notwendig.

Für das beschriebene Forschungsvorhaben können folgende zwei Herausforderungen identifiziert werden.

1. Übersetzung der erlernten Klassifikatoren in verfügbare Anfragesprachen bzw. Funktionen.

2. Erstellung einer dynamischen und flexiblen CEP-Anwendung, die stetig an neue Anforderungen anpassbar ist. Weiterhin muss unter Berücksichtigung der eingeschränkten Systemressourcen und Echtzeitanforderungen ein kontinuierlicher Strom von Zustandsvektoren zuverlässig klassifiziert werden können.

8. EXISTIERENDE LÖSUNGEN

Zur Analyse von Datenströmen werden Datenstrom-Management-Systeme (DSMS), z.B. STREAM [1] oder Aurora [6], eingesetzt. Aurora enthält ein Pfeil-Box-Architektur-Modell, welches einem Plug-in-System ähnlich ist. Ein Überblick über DSMS wird u.a. in [14] gegeben. CEP-Systeme wie CAYUGA [9] oder ESPER [11] werden für das Anwenden von Regeln auf Datenströme mittels Anfragesprachen verwendet. Ein Überblick über CEP-Systeme wird in [12] gegeben. Die genannten DSMS und CEP-Systeme sind nicht für Überwachung mittels Data-Mining-Technologien konzipiert.

NanoMon [29] ist eine sehr spezielle Überwachungssoftware für Sensornetzwerke. MobiMine [19] ist ein mobiles Data-Mining-System für den Aktienhandel. Beide Überwachssysteme unterstützen die genannten Überwachungsanforderungen nicht. Weiterhin enthalten NanoMon und MobiMine keine Anfragesprache.

VEDAS [18] ist ein Datenstrom-Mining-System, welches einigen der hier erarbeiteten Überwachungsforderungen entspricht. Die Erkennung von ungewöhnlichem Fahrerverhalten ist eines der Hauptaugenmerke von VEDAS. Wie auch hier kommen bei VEDAS Data-Mining-Technologien zum Einsatz. Der Unterschied liegt in der Verwendung von unüberwachtem Lernen für Datenstrom-Mining. Weiterhin gibt es keine strikte Trennung zwischen Echtzeitüberwachung und Langzeitanalyse sowie zwischen automatischen und semiautomatischen Funktionen. Dieses Argument kann durch die interaktive Verbindung vom externen Informationssystem zum eingebetteten System untermauert werden. Weiterhin wird bei VEDAS die Evaluierung vernachlässigt. Zusätzlich wird die Überwachungsanforderung Lokalität nicht berücksichtigt. In VEDAS ist das eingebettete System so konfiguriert, dass alle Attribute für die Überwachung Verwendung finden. Dies kann unter Umständen zu sehr hohem Rechenaufwand führen.

9. ZUSAMMENFASSUNG

Es besteht ein Bedarf an neuen Lösungen für die Überwachung von Systemen, die heutige und zukünftige Anforderungen in Betracht ziehen. Der vorliegende Beitrag skizziert ein interdisziplinäres Forschungsvorhaben im Rahmen einer Doktorarbeit. Einer der Forschungsbeiträge ist die Kombination von Echtzeitüberwachung und Langzeitanalyse mittels eingebetteter Systeme, ECA-Regeln, Data-Mining-Technologien und CEP. Drei Annahmen bilden die Basis für den beschriebenen Überwachungsansatz. Weiterhin wurden hier fünf Überwachungsanforderungen erarbeitet. Die Analyse bestehender Lösungen zeigt, dass die dargestellten Überwachungsanforderungen nur unzureichend in Betracht gezogen werden. Aufbauend darauf wurde hier ein dynamischer und flexibler Überwachungsansatz vorgestellt. Der hier vorgestellte Überwachungsansatz basiert auf einem mathematischen Modell, welches als Zustandsraum bezeichnet wird. Dieser Zustandsraum repräsentiert das Wissen über das Produkt, welches im laufenden Betrieb überwacht wird. Weiterhin wurde eine Prozesskette erläutert. Diese Prozesskette wird zyklisch durchlaufen und somit das Wissen über das Produkt mit der Zeit gesteigert. Der Zustandsraum wird mit der Hilfe von Data-Mining-Technologien in ECA-Regeln übersetzt und an eine CEP-Anwendung, die sich auf einem eingebetteten System befindet, übertragen. Durch die CEP-

Anwendung werden die ECA-Regeln verwendet, um die kontinuierlich eintreffenden Zustandsvektoren als bekannt oder unbekannt zu klassifizieren.

10. LITERATUR

[1] Babcock, B. ; Babu, S. ; Datar, M. ; Motwani, R. ; Widom, J. : Models and Issues in Data Stream Systems. In: *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ACM, 2002, S. 1–16

[2] Bellmann, R. : *Adaptive Control Processes*. Princeton University Press, 1961

[3] Bifet, A. ; Kirkby, R. : Data Stream Mining - A Practical Approach / Centre for Open Software Innovation (COSI) - Waikato University. Version: 2009. `http://moa.cs.waikato.ac.nz/wp-content/uploads/2010/05/StreamMining.pdf`. – Forschungsbericht

[4] Borgelt, C. ; Klawonn, F. ; Kruse, R. ; Nauck, D. : *Neuro-Fuzzy-Systeme: Von den Grundlagen künstlicher Neuronaler Netze zur Kopplung mit Fuzzy-Systemen*. Vieweg, 2003

[5] Boslaugh, S. ; Watters, P. A.: *Statistics in a Nutshell*. O'Reilly, 2008

[6] Carney, D. ; Çetintemel, U. ; Cherniack, M. ; Convey, C. ; Lee, S. ; Seidman, G. ; Stonebraker, M. ; Tatbul, N. ; Zdonik, S. : Monitoring Streams: A New Class of Data Management Applications. In: *VLDB '02: Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB Endowment, 2002, S. 215–226

[7] Chakravarthy, S. ; Jiang, Q. : *Stream Data Processing: A Quality of Service Perspective*. Springer, 2009

[8] Chandola, V. ; Banerjee, A. ; Kumar, V. : Anomaly detection: A survey. In: *ACM Comput. Surv.* 41 (2009), S. 15:1–15:58

[9] Demers, A. J. ; Gehrke, J. ; Panda, B. ; Riedewald, M. ; Sharma, V. ; White, W. M.: Cayuga: A General Purpose Event Monitoring System. In: *CIDR*, 2007, S. 412–422

[10] Dittrich, K. R. ; Gatziu, S. ; Geppert, A. : The Active Database Management System Manifesto: A Rulebase of ADBMS Features. In: *SIGMOD Rec.* 25 (1996), Nr. 3, S. 40–49

[11] EsperTech: *Esper*. `http://www.espertech.com/products/esper.php`. Version: 2011. – Online: 30.03.2011

[12] Etzion, O. ; Niblett, P. : *Event Processing in Action*. Manning Publications Co., 2010

[13] Fischer, G. ; Grubisic, V. : Praxisrelevante Bewertung des Radbruchs vom ICE 884 in Eschede. In: *Materialwissenschaft und Werkstofftechnik* 38 (2007), Nr. 10, S. 789–801

[14] Golab, L. ; Özsu, M. T.: *Data Stream Management*. Morgan & Claypool Publishers, 2010

[15] Gordon, G. : *Systemsimulation*. Oldenbourg, 1972

[16] Guo, Y. : *Algorithmen zur On-Board-Diagnose von Fahrwerksschäden an Schienenfahrzeugen*, TU Berlin, Diss., 2005. `http://opus.kobv.de/tuberlin/volltexte/2005/1120/`

[17] Imboden, D. M. ; Koch, S. : *Systemanalyse*. Springer, 2003

[18] Kargupta, H. ; Bhargava, R. ; Liu, K. ; Powers, M. ; Blair, P. ; Bushra, S. ; Dull, J. ; Sarkar, K. ; Klein, M. ; Vasa, M. ; Handy, D. : VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. In: *Proceedings of the Fourth SIAM International Conference on Data Mining*, 2004

[19] Kargupta, H. ; Park, B.-H. ; Pittie, S. ; Liu, L. ; Kushraj, D. ; Sarkar, K. : MobiMine: Monitoring the Stock Market from a PDA. In: *SIGKDD Explor. Newsl.* 3 (2002), S. 37–46

[20] Marwedel, P. : *Eingebettete Systeme*. Springer-Verlag, 2007

[21] Noack, E. ; Belau, W. ; Wohlgemuth, R. ; Müller, R. ; Palumberi, S. ; Parodi, P. ; Burzagli, F. : Efficiency of the Columbus Failure Management System. In: *AIAA 40th International Conference on Environmental Systems*, 2010

[22] Noack, E. ; Noack, T. ; Patel, V. ; Schmitt, I. ; Richters, M. ; Stamminger, J. ; Sievi, S. : Failure Management for Cost-Effective and Efficient Spacecraft Operation. In: *Proceedings of the 2011 NASA/ESA Conference on Adaptive Hardware and Systems*, IEEE Computer Society, 2011 (AHS '11). – To appear

[23] Peckol, J. K.: *Embedded Systems: A Contemporary Design Tool*. John Wiley & Sons, 2007

[24] Richard, H. ; Fulland, M. ; Sander, M. ; Kullmer, G. : Fracture in a rubber-sprung railway wheel. In: *Engineering Failure Analysis* 12 (2005), Nr. 6, S. 986 – 999

[25] Schmitt, I. : QQL: A DB&IR Query Language. In: *The VLDB Journal* 17 (2008), S. 39–56

[26] Tsai, J. J. P. ; Yang, S. J. H.: *Monitoring and Debugging of Distributed Real-Time Systems*. IEEE Computer Society Press, 1995

[27] Witten, I. H. ; Frank, E. ; Hall, M. A.: *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier, 2011

[28] Wolf, F. : *Behavioral Intervals in Embedded Software: Timing and Power Analysis of Embedded Real-Time Software Processes*. Kluwer Academic Publishers, 2002

[29] Yu, M. ; Kim, H. ; Mah, P. : NanoMon: An Adaptable Sensor Network Monitoring Software. In: *IEEE International Symposium on Consumer Electronics (ISCE)*, 2007

Analyse und Vergleich von Zugriffstechniken für funktionale Aspekte in RDBMS

Matthias Liebisch
Friedrich-Schiller-Universität Jena
Lehrstuhl für Datenbanken und Informationssysteme
Ernst-Abbe-Platz 2
07743 Jena
m.liebisch@uni-jena.de

KURZFASSUNG

Neben klassischen fachlichen Anforderungen existieren in Anwendungssystemen oft auch querschnittliche Belange, deren Funktionalität sich nicht einfach kapseln bzw. modularisieren lässt. Vertreter dieser sogenannten *funktionalen Aspekte* sind beispielsweise die mehrsprachige oder versionierte Darstellung und Verwaltung von Anwendungsdaten. Nachdem sich in der Software-Entwicklung seit einigen Jahren die Aspektorientierte Programmierung als Lösung etabliert hat, bietet das neuartige Paradigma der Aspektorientierten Datenhaltung ein entsprechendes Konzept zur Abbildung querschnittlicher Belange in einem relationalen Datenmodell. Dabei stehen vor allem die Unabhängigkeit vom Prozess der fachlichen Modellierung und ein hoher Wiederverwendungsgrad im Vordergrund. Basierend auf dem zu diesem Zweck entwickelten Referenzmodell untersucht der vorliegende Beitrag unterschiedliche Techniken für den Zugriff auf jene funktionalen Aspekte. Diese werden anschließend anhand wesentlicher Bewertungskriterien einer Evaluation unterzogen und miteinander verglichen.

Kategorien und Themenbeschreibung

H.4 [**Information Systems Applications**]: Miscellaneous; H.2.3 [**Database Management**]: Languages—*Query languages*

Allgemeine Bestimmungen

Design, Languages, Performance

1. EINLEITUNG

Seit der Beschreibung des relationalen Modells[4] Anfang der 1970er Jahre ist die Bedeutung auf diesem Modell basierender Datenbankmanagementsysteme (RDBMS) als Persistierungsebene stetig gewachsen. Heutzutage bilden relationale Datenbanksysteme die Grundlage für die vielfältigsten Anwendungssysteme und sind damit aus den meisten alltäglichen Prozessen nicht mehr wegzudenken. Die Entwicklung derartiger Applikationen ist deswegen auch immer mit dem Entwurf eines für den Einsatzzweck geeigneten Datenmodells verbunden. Dieses sollte unter Beachtung verschiedener Kriterien, wie beispielsweise Benutzbarkeit und Wiederverwendbarkeit, die modularisierte Speicherung der fachlichen Datenobjekte in relationalen Strukturen optimal unterstützen. Neben dieser Abbildung existieren jedoch häufig zusätzlich anwendungsweite Anforderungen wie beispielsweise die Unterstützung von Mehrsprachigkeit oder Versionierung, welche als sogenannte **funktionale Aspekte**[11] Einfluss auf das gesamte Datenmodell haben.

Dieses Problem der *cross-cutting concerns* ist bereits aus dem Umfeld der Objektorientierten Programmierung seit einigen Jahren bekannt und hat zur Entwicklung der Aspektorientierten Programmierung[3] geführt. Im übertragenen Sinne stellt die Aspektorientierte Datenhaltung[11] ein Modellierungsparadigma dar, um funktionale Aspekte in einem Datenmodell gekapselt und unabhängig von den fachlichen Datenobjekten zu integrieren. Triviale Ansätze, wie beispielsweise die Erweiterung relevanter Tabellen um eine zusätzliche Spalte zur Festlegung der Locale im Fall mehrsprachiger Datenhaltung, sind meist nur auf konkrete Anwendungsfälle zugeschnitten und versagen zudem bei der Unterstützung beliebig vieler funktionaler Aspekte unter den Anforderungen des Paradigmas der Aspektorientierten Datenhaltung [19]. Ein generischer Ansatz zur Lösung der angedeuteten Herausforderungen ist das in [12] beschriebene Referenzmodell. Darauf basierend zeigt der vorliegende Beitrag verschiedene Alternativen für den Zugriff und die Nutzung funktionaler Aspekte aus Sicht der Anwendung auf.

Nachfolgend werden in Abschnitt 2 das erwähnte Referenzmodell sowie ein kleines Anwendungsbeispiel kurz vorgestellt. Die darauf aufbauenden Zugriffstechniken stehen im Fokus von Abschnitt 3, bevor sie in Abschnitt 4 einer Bewertung unterzogen werden. Schließlich fasst Abschnitt 5 die Ergebnisse der Arbeit nochmal zusammen.

2. REFERENZMODELL

Für die vom fachlichen Datenmodell unabhängige und gekapselte Persistierung aspektspezifischer Daten wurde in [12] ein Referenzmodell vorgestellt und beschrieben, welches mit geringfügigen Anpassungen bezüglich der Fremdschlüsseldefinitionen in den Tabellen zur Aspektverknüpfung auch in diesem Beitrag zum Einsatz kommt.

23^{rd} *GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken)* 31.5.-03.06.2011, Obergurgl, Austria.

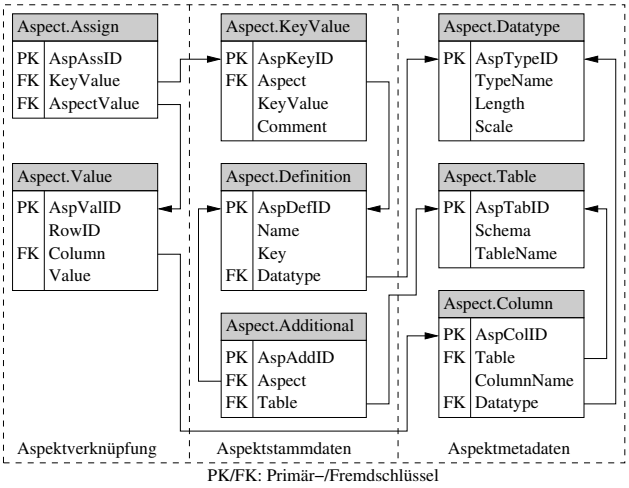
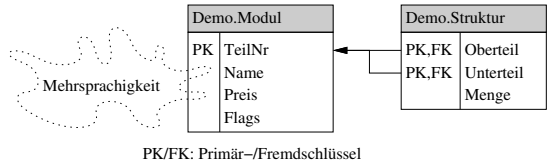


Abbildung 1: Referenzmodell

Zentraler Bestandteil dieses in Abbildung 1 skizzierten Referenzmodells sind die beiden Tabellen ASPECT.VALUE und ASPECT.ASSIGN, welche die Speicherung aspektspezifischer Attributwerte sowie deren Zuordnung zu einer konkreten Aspektausprägung (z.B. Locale 'en' im Aspekt Mehrsprachigkeit) für eine Fachtabellenzelle realisieren. Daneben existieren weitere Tabellen zur Verwaltung von Metadaten, wie beispielsweise ASPECT.KEYVALUE für die Spezifikation von Ausprägungswerten zu allen im System definierten Aspekten oder ASPECT.COLUMN zur Hinterlegung aspektrelevanter Attribute der Fachtabellen.

Die Anforderungen aus dem Paradigma der Aspektorientierten Datenhaltung werden insbesondere durch das Entity-Attribute-Value-Konzept[15] (EAV) gewährleistet, welches für ASPECT.VALUE und ASPECT.KEYVALUE zur Anwendung kommt. Die damit verbundenen Konsequenzen[7] bezüglich der Komplexität von direkten SQL-Anfragen im Referenzmodell erfordern die Analyse alternativer Zugriffsarten.



►Anfrage: *Ermittle für das Modul mit TeilNr=4711 alle mehrsprachigen Daten (*NAME *und* PREIS*) sowie den jeweiligen Aspektschlüsselwert als Locale.*

Abbildung 2: Beispiel für Datenmodell mit Anfrage

Zur Veranschaulichung der in Abschnitt 3 folgenden Techniken soll das in Abbildung 2 dargestellte Beispiel eines vereinfachten Datenmodells zur Verwaltung von Stücklisten dienen, in dem der Aspekt „Mehrsprachigkeit" für die Attribute DEMO.MODUL.NAME sowie DEMO.MODUL.PREIS aktiviert wurde. Darauf aufbauend soll jeweils die Beantwortung der zugehörigen Beispiel-Anfrage erläutert werden, welche das Prinzip für den Zugriff auf aspektspezifische Attributwerte in einem konkreten fachlichen Anwendungskontext verdeutlichen soll.

3. ZUGRIFFSTECHNIKEN

Dieser Abschnitt beschreibt verschiedene Möglichkeiten für den Zugriff auf funktionale Aspekte, welche mit Hilfe des in Abbildung 1 präsentierten Referenzmodells in ein fachliches Datenmodell integriert werden. Aufgrund der Tatsache, dass das relationale Modell als Grundlage dient, ist der direkte Zugriff mittels SQL auf die entsprechenden Strukturen die naheliegendste Möglichkeit. Allerdings stellt die zentrale Tabelle ASPECT.VALUE eine neue Herausforderung für die Anfragegenerierung dar, weil darin enthaltene Daten durch das verwendete EAV-Prinzip[15] einer sogenannten unpivotisierten („gekippten") Speicherungsform unterliegen. Dies hat zur Konsequenz, dass zu jedem Attribut (COLUMN) eines traditionellen Tupels (identifizierbar über ROWID) der jeweilige Wert (VALUE) in einer eigenen Zeile gespeichert wird. Da jedoch die klassische relationale Verarbeitung von Datensätzen mit zugehörigen Attributen als Tabellenspalten ausgeht, ist eine Pivotisierung („rows to columns") notwendig, sobald in der Anfrage die Tabelle ASPECT.VALUE involviert wird. Die anschließenden Abschnitte beschreiben drei Konstrukte in SQL sowie einen applikativen Ansatz, um die genannte Transformation zu unterstützen.

3.1 SQL mit JOIN

Bei einer Beschränkung auf normierte Sprachmittel ist die erforderliche Transformation nur mittels JOIN-Operatoren realisierbar, da weder im SQL:92-Standard[6] als Grundlage für das Paradigma der Aspektorientierten Datenhaltung[11] noch in der aktuellen SQL:2008-Norm[10] dedizierte Operatoren zur Pivotisierung einer Tabelle existieren. Das prinzipielle Vorgehen ist exemplarisch für die in Abbildung 2 formulierte Anfrage in Abbildung 3 dargestellt. Dabei wurde auf die Formatierung der Ergebnisattribute entsprechend den zugeordneten Datentypen in Tabelle ASPECT.DATATYPE verzichtet und zwecks Übersichtlichkeit die Kenntnis gewisser Metadaten wie Idents von Aspekten und Tabellenspalten als bekannt vorausgesetzt.

```
SELECT T1.Value AS Name, T2.Value AS Preis,
       T5.KeyValue AS Locale
  FROM Aspect.Value T1
       INNER JOIN Aspect.Value T2
         ON T1.RowID = T2.RowID
       INNER JOIN Aspect.Assign T3
         ON T1.AspValID = T3.AspectValue
       INNER JOIN Aspect.Assign T4
         ON T2.AspValID = T4.AspectValue
       INNER JOIN Aspect.KeyValue T5
         ON T3.KeyValue = T5.AspKeyID
 WHERE T1.Column = 1 -- /* 'Name' */
   AND T2.Column = 2 -- /* 'Preis' */
   AND T3.KeyValue = T4.KeyValue
   AND T5.Aspect = 1 -- /* 'Mehrsprachigkeit' */
   AND T1.RowID = 4711
```

Abbildung 3: Anfrage mit JOIN

Bereits für das simple in Abbildung 3 präsentierte Beispiel ist die Komplexität der Anfragegenerierung inklusive Pivotisierung der Tabelle ASPECT.VALUE erkennbar. Insbesondere skalieren die notwendigen JOIN-Operatoren linear mit den Attributen im Ergebnisschema. Dabei werden jeweils

die Tabellen ASPECT.VALUE und ASPECT.ASSIGN miteinander verbunden, welche für die Speicherung aller aspektspezifischen Ausprägungen von Attributwerten in Fachtabellen zuständig sind und dadurch mit Abstand die umfangreichsten Mengengerüste aufweisen. Erwartungsgemäß haben derartige Anweisungen eine oft inakzeptable Verarbeitungszeit wie entsprechende Analysen gezeigt haben[13].

3.2 SQL mit PIVOT

Verlässt man die SQL-Norm auf der Suche nach adäquater Unterstützung für die Pivotisierung von EAV-Tabellen, dann zeigt sich, dass DBMS-Hersteller bereits produktspezifische SQL-Erweiterungen mit den Operatoren PIVOT und UNPIVOT[20] anbieten. Unter anderem finden sich derartige Implementierungen in Datenbanksystemen wie Microsoft SQL Server 2005[18] oder Oracle 11g[14]. Ein typisches Anwendungsgebiet für diese Transformationen sind OLAP-Anfragen im Bereich Data Warehouse[17], deren Blickwinkel geändert werden soll (beispielsweise die Gruppierung nach Regionen statt Produkten in einer Umsatzübersicht). Durch Nutzung von PIVOT und UNPIVOT kann eine gezielte Optimierung auf Basis der klassischen Operatoren wie Verbund oder Projektion erfolgen[5].

```
SELECT PivotedData.[1] AS Name,
       PivotedData.[2] AS Preis,
       PivotedData.KeyValue AS Locale
  FROM
  (
    SELECT T1.RowID, T1.Column, T1.Value,
           T3.KeyValue
      FROM Aspect.Value T1
           INNER JOIN Aspect.Assign T2
             ON T1.AspValID = T2.AspectValue
           INNER JOIN Aspect.KeyValue T3
             ON T2.KeyValue = T3.AspKeyID
     WHERE T3.Aspect = 1 -- /*'Mehrsprachigkeit'*/
       AND T1.RowID = 4711
  ) AS JoinData
  PIVOT
  (
    MAX(JoinData.Value)
    FOR JoinData.Column IN ([1], [2])
    -- /* 1 und 2 = relevante Column-IDs */
  ) AS PivotedData
```

Abbildung 4: Anfrage mit PIVOT

Aber auch die Verarbeitung von EAV-Tabellen wie hier im Kontext der Aspektorientierten Datenhaltung ist mit Hilfe des PIVOT-Operators möglich. Abbildung 4 demonstriert dessen Verwendung unter Microsoft SQL Server 2005 für die Beispielanfrage in Abbildung 2. Auf den ersten Blick verwirrt dabei der Ausdruck `MAX(JoinData.Value)`, welcher die notwendige Aggregatbildung bei der Pivotisierung übernimmt. Diese Funktion lässt das Ergebnis jedoch inhaltlich korrekt, solange jede Gruppe bezüglich der gruppierten Attribute (`T1.RowID, T3.KeyValue`) und den zu Spalten umgewandelten Werten aus `T1.Column` nur einen Datensatz enthält. Um dies unter Beachtung der getrennten Speicherung der aspektspezifischen Werte (ASPECT.VALUE) und den tatsächlichen Fachtabellen-Zuordnungen (ASPECT.ASSIGN) sicherzustellen, müssen die beiden genannten Tabellen zusammen mit der Tabelle ASPECT.KEYVALUE verbunden werden. Anschließend kann über die projezierte Attributmenge sinnvoll pivotisiert werden.

Zusätzlich ist bei der Nutzung des PIVOT-Operators zu beachten, dass für die `IN`-Klausel nur eine fest definierte Spaltenmenge angegeben werden kann. Ein Ausdruck der Form `SELECT Column FROM Aspect.Value` ist beispielsweise nicht zulässig. Dies gilt sowohl für Microsoft SQL Server 2005[1] als auch für Oracle 11g[2]. Wird eine derartige Dynamik dennoch benötigt, lässt sich diese nur über eine Stored Procedure, vorgeschalteten Anwendungscode oder im Fall von Oracle unter Verwendung von XML realisieren.

3.3 SQL mit Spracherweiterung

Aufgrund der fehlenden Normierung des PIVOT-Operators einerseits und dessen Nutzungs-Einschränkungen im Kontext der Aspektorientierten Datenhaltung andererseits, verfolgt dieser Abschnitt die Idee einer SQL-Erweiterung für einen adäquaten Zugriff auf funktionale Aspekte im Referenzmodell. Die neuen Sprachelemente beeinflussen sowohl den DML-Teil als auch den DDL-Bereich, um beispielsweise für eine Tabellenspalte relevante Aspekte definieren zu können. Hier soll jedoch aus Platzgründen nur das `SELECT`-Statement im Fokus stehen.

```
<table reference> ::=
   <table name> [ <correlation specification> ]
 | <derived table> <correlation specification>
 | <joined table>
 | <aspect view>

<aspect view> ::=
  ASPECTVIEW <identifier> BASED ON <table name>
  GROUP BY <aspect key> [{, <aspect key>} ... ]

<aspect key> ::=
  ASPECTKEY(<character string literal>)
  AS <identifier>
```

Abbildung 5: SQL-Erweiterung ASPECTVIEW

Aufbauend auf dem SQL:92-Standard[6] wird das Nichtterminalsymbol `<table reference>`[3] um eine weitere Alternative `<aspect view>` ergänzt, deren Definition in Abbildung 5 dargestellt ist. Das neue Schlüsselwort `ASPECTVIEW` erzeugt dabei eine Sicht auf alle aspektspezifischen Daten der über `BASED ON` angegebenen (fachlichen) Basistabelle. Da es möglich ist, einer Tabellenspalte verschiedene Aspekte zuzuordnen, erfolgt die Aufbereitung der Daten in gruppierter Form bezüglich des einattributigen Primärschlüssels[11] und der über `<aspect key>` spezifizierten Aspekte. Diese stehen anschließend im Rahmen der Anfrageformulierung als reguläre Attribute der Sicht zur Verfügung, für eine Tabelle mit n Attributen und insgesamt k zugeordneten Aspekte ergibt sich also das in Abbildung 6 dargestellte Relationsschema. Voraussetzung hierfür ist eine im Referenzmodell ver-

[1] `http://msdn.microsoft.com/en-us/library/ms177634.aspx`

[2] `http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/statements_10002.htm#CHDFAFIE`

[3] `http://savage.net.au/SQL/sql-92.bnf`

ankerte UNIQUE-Bedingung auf ASPECT.DEFINITION.KEY, dessen Werte über den Ausdruck `ASPECTKEY(<character string literal>)` referenziert werden.

$$(\overbrace{Att_1, \ldots, Att_n}^{\text{Basistabelle}}, \overbrace{Asp_1, \ldots, Asp_k}^{\text{Aspekte}})$$

Abbildung 6: Relationsschema von ASPECTVIEW

Angewendet auf das in Abbildung 2 skizzierte Beispiel ergibt sich eine gegenüber den bisherigen Zugriffstechniken sehr kompakte Anfrage, wie Abbildung 7 zeigt. Neben dem Aufwand zur Pivotisierung für die Aspektsicht `T1` ist unabhängig von der Anzahl der Attribute oder Aspekte nur noch eine JOIN-Operation erforderlich, um für die Idents in Asp_i die eigentlichen Aspektschlüsselwerte anzeigen zu können.

```
SELECT T1.Name, T1.Preis, T2.KeyValue AS Locale
  FROM Aspect.KeyValue T2 INNER JOIN
       (
         ASPECTVIEW ModulAspects
         BASED ON Demo.Modul
         GROUP BY ASPECTKEY('Locale') AS AspLoc
       ) T1
       ON T2.AspKeyID = T1.AspLoc
 WHERE T1.TeilNr = 4711
   AND T2.Aspect = 1 -- /*'Mehrsprachigkeit'*/
```

Abbildung 7: Anfrage mit ASPECTVIEW

Sollte sich eine derartige Spracherweiterung wie angedeutet für alle Bereiche (DDL und DML) als geeignetes Ausdrucksmittel im Umgang mit funktionalen Aspekten beweisen, bleibt die praktische Nutzung stark eingeschränkt, solange eine Umsetzung in SQL-Norm oder DBMS-Produkten fehlt. In solch einem Szenario kann der Einsatz sogenannter Proxys[1] oder Query Transformation Layer[2] zwischen Anwendung und Datenbank eine Lösung darstellen. Im vorliegenden Fall müssten alle neu definierten SQL-Ausdrücke, beispielsweise `ASPECTVIEW`, durch einen Parser auf effiziente Art und Weise in genormte oder produktspezifische SQL-Anweisungen transformiert werden. Damit wäre zumindest eine benutzerfreundliche Schnittstelle im Umgang mit funktionalen Aspekten unter Nutzung der trivialen Umformung (Anfrage mit JOIN, siehe Abschnitt 3.1) geschaffen. Die nachfolgend beschriebene Zugriffstechnik ist ebenfalls ein Vertreter dieser Kategorie, allerdings erfolgt die Anfrageformulierung nicht auf Basis von SQL.

3.4 Funktionsbaustein mit API

Gegenüber den bisher vorgestellten Möglichkeiten, den Zugriff auf funktionale Aspekte allein mit SQL-Mitteln auf der Persistierungsebene zu realisieren, wird mit dem vierten Ansatz eine applikationsseitige Verarbeitung verfolgt. Analog zum Konzept mit JDBC[9] eine einheitliche Schnittstelle für relationale Datenbanken zu etablieren, ermöglicht der hier vorgestellte Funktionsbaustein den Zugriff auf funktionale Aspekte, welche im Referenzmodell persistiert sind. Die zugehörige API[16] umfasst Methoden sowohl zur Verwaltung der Aspekte inklusive ihrer Metadaten als auch zur Abfrage und Änderung aspektspezifischer Attributwerte unter Beachtung sogenannter Aspektfilter und Aspektkontexte.

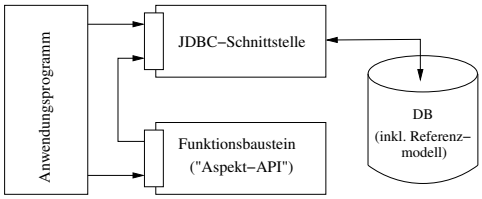


Abbildung 8: Anwendungs-Integration der API

Um einen möglichst plattformunabhängigen und universellen Funktionsbaustein bereitstellen zu können, ist dieser inklusive seiner API in Java spezifiziert. Da in den meisten Fällen bei Verwendung einer Java-Bibliothek das umgebende Anwendungsprogramm ebenfalls auf Java basiert und für Datenbankzugriffe die JDBC-Schnittstelle genutzt wird, zeigt Abbildung 8 ein typisches Integrations-Szenario. Dabei können in einer ersten Ausbaustufe über die API tatsächlich nur aspektspezifische Daten abgefragt und geändert werden, die Verarbeitung der fachlichen Daten erfolgt unverändert über die bereits existierende Datenbank-Schnittstelle. Aufgabe der Anwendung ist damit also die Zusammenführung der Ergebnisse beider Datenquellen, abhängig natürlich von den inhaltlichen Anforderungen.

```
1  // Deklarationen
   AspectManager am = /* Verweis auf Instanz holen */
3  AspectCatalogManager acm =
     am.getAspectCatalogManager();
5  int aspLang = acm.lookupAspectID("Language");
   int tID = acm.lookupTableID("Demo","Modul");
7  int cNameID = acm.lookupColumnID(tID, "Name");
   int cPriceID = acm.lookupColumnID(tID, "Preis");
9
   // Anfragespezifikation
11 QueryStatement st = am.createQueryStatement(tID);
   st.setRowIDs(new long[] {4711});
13 st.setColumnSet(am.createColumnSet(
     new int[] {cNameID, cPriceID}));
15
   // Ergebnisverarbeitung
17 ResultRows result = st.execute();
   AspectSet aSet = st.getAspectSet();
19 ColumnSet cSet = st.getColumnSet();
   int aspLangIdx = aSet.getIndex(aspLang);
21 int cNameIdx = cSet.getIndex(cNameID);
   int cPriceIdx = cSet.getIndex(cPriceID);
23 while (result.next())
   {
25   long rowID = result.getRowID();
     AspectContextElement e =
27     result.getContext().getElement();
     String name = result.getValueString(cNameIdx);
29   String price = result.getValueString(cPriceIdx);
     int localeID = e.getIndexKeyValue(aspLangIdx);
31 }
```

Abbildung 9: Anfrage mit Funktionsbaustein

Eine Abfrage aspektspezifischer Daten wie für das Referenz-Beispiel wird prinzipiell durch das in Abbildung 9 dargestellte Code-Fragment realisiert. Der Einstieg erfolgt mit einer implementierungsspezifischen Instanz der zentralen Klasse `AspectManager` (Zeile 2), welche im nächsten Schritt zur Erzeugung einer `AspectCatalogManager`-Instanz benötigt wird und zudem auch für die Verwaltung einer Datenbank-Session zuständig ist. Schließlich werden zum Aspekt Mehrsprachigkeit sowie für die Attribute NAME und PREIS der Tabelle MODUL die Idents ermittelt (Zeilen 5-8). Hierfür können entsprechende Methoden an der Aspektkatalogmanager-Klasse genutzt werden, damit sind alle für die Anfrage notwendigen Metadaten bekannt.

Für die Anfragespezifikation wird über den `AspectManager` eine Instanz der Klasse `QueryStatement` angelegt (Zeile 11). Da in der relevanten Fachtabelle DEMO.MODUL das Attribut TEILNR bereits die Anforderung eines einattributigen Primärschlüssels erfüllt, entspricht die Filterung mit der Methode `setRowIDs` in Zeile 12 genau der Einschränkung auf das Modul mit *TeilNr=4711* in einer WHERE-Klausel. Die zu projezierenden Spalten werden analog in Zeile 13 festgelegt. Weitere zusätzliche aspektspezifische Beschränkungen beispielsweise über die `AspectFilter`-Klasse entfallen, da alle mehrsprachigen Daten für die Attribute NAME und PREIS bereitzustellen sind.

Die Ausführung der Anfrage erzeugt ein `ResultRows`-Objekt (Zeile 17), welches analog zum entsprechenden Konstrukt in JDBC zeilenweise über den `next`-Iterator verarbeitet werden kann (Zeile 23). Zuvor sind für die korrekte Auswertung der Ergebnisstruktur die darin enthaltenen Indexe des Mehrsprachigkeits-Aspekts sowie der beiden angefragten Attribute zu ermitteln (Zeilen 18-22). Mit Hilfe dieser Indexe kann nun auf die entsprechenden Informationen zugegriffen werden (Zeilen 25-29). Weiterführende Details zu Datenstrukturen, Klassen und Methoden sowie zu deren Verwendung finden sich bei PIETSCH[16].

4. VERGLEICH

Nachdem der vorangegangene Abschnitt 3 die unterschiedlichen Zugriffstechniken präsentiert hat, dient dieser Abschnitt der Bewertung und dem Vergleich jener Techniken. Zuerst werden die dafür notwendigen Kriterien im Folgenden beschrieben und anschließend für jeden Ansatz evaluiert.

4.1 Kriterien

Ausgehend vom Paradigma der Aspektorientierten Datenhaltung spiegelt sich die Forderung nach Universalität[11] bezüglich SQL:92 im Kriterium der **Standardisierung** wider, welches hier jedoch auch bezüglich der Existenz einer ISO-Norm erweitert werden soll. In direktem Zusammenhang dazu stellt sich die Frage der **Praxisrelevanz** eines Ansatzes, d.h. ob mit diesem der Zugriff auf funktionale Aspekte unter den aktuellen Gegebenheiten überhaupt praktisch realisierbar ist. Eine große Bedeutung spielen natürlich auch Ausagen zur **Performance**, allerdings liegen nur für den ersten Ansatz (SQL mit JOIN) tatsächlich konkrete Messwerte[13] vor, sodass für alle anderen Varianten nur eine qualitative Abschätzung möglich ist.

Inwieweit die Strukturen des Referenzmodells zur Persistierung funktionaler Aspekte dem Nutzer bzw. der Anwendung im Rahmen der Verarbeitung bekannt sein müssen oder verborgen bleiben können, zeigt sich in der **Transparenz** einer Technik. Weiterhin wird geprüft, ob im jeweiligen Ansatz auf die bereits vorhandene Mächtigkeit eines zu Grunde gelegten RDBMS in angemessener Weise zurückgegriffen wird (**Funktionsadäquatheit**) oder ob Funktionalität, wie beispielsweise das Parsen von Sprachausdrücken und das Caching von Daten, reimplementiert werden muss. Hierbei spielen auch prinzipielle Möglichkeiten zur **Erweiterbarkeit** des Funktionsumfangs eine Rolle, erwartungsgemäß werden diese durch Standardisierung beschränkt. Schließlich soll noch mit der **Benutzerfreundlichkeit** beurteilt werden, wie sich letztlich die gesamte Auswertung funktionaler Aspekte für den Anwendungsentwickler darstellt.

4.2 Bewertung

Eine detaillierte Bewertung jeder einzelnen Zugriffstechnik bezüglich der zuvor aufgestellten Kriterien kann hier aus Platzgründen nicht beschrieben werden. Stattdessen enthält Tabelle 1 einen Überblick mit den Ergebnissen.

Zugriffstechnik / **Kriterium**	Standardisierung	Praxisrelevanz	Performance	Transparenz	Funktionsadäquatheit	Erweiterbarkeit	Benutzerfreundlichkeit
JOIN	+	+	−	−	+	−	−
PIVOT	o[1]	o[1]	o	o	+	−	o
ASPECTVIEW	−	o[3]	o	+	o[3]	+	+
API	o[2]	+	+[4]	+	−	+	o

+ : ja/hoch o : neutral/mittel − : nein/niedrig

[1] unterstützt durch Oracle 11g und MS SQL Server 2005
[2] de-facto Standard[8] aufgrund enormer Verbreitung
[3] nur bei Nutzung von Zwischenschichten[1, 2]
[4] begründet durch Ergebnisse in [7]

Tabelle 1: Bewertung der Zugriffstechniken

Dabei fällt auf, dass zwar der **JOIN-Ansatz** (Abschnitt 3.1) als einziger Vertreter auf standardisierte Sprachkonstrukte zurückgreift und damit eine große Praxisrelevanz bzw. Produktunterstützung besitzt, allerdings weder performant noch benutzerspezifisch erweiterbar ist. Zudem erzwingt die Nutzung des Verbund-Operators genaue Kenntnisse über die Aspekt-Tabellen, wodurch die Transparenz und Benutzerfreundlichkeit verloren geht.

Dagegen verspricht die **PIVOT-Methode** (Abschnitt 3.2) sowohl den transparenteren Zugang als auch eine performantere Verarbeitung der aspektspezifischen Daten in den EAV-Tabellen[5]. Dennoch fehlt hier ebenfalls die Möglichkeit zur Erweiterbarkeit, zudem ist die Anwendbarkeit aufgrund der (noch) fehlenden Normierung an Hersteller wie Oracle oder Microsoft und deren DBMS-Produkte gekoppelt.

Ähnliche Charakteristik besitzt auch der **ASPECTVIEW-Vorschlag** (Abschnitt 3.3), er soll jedoch als Erweiterung

von SQL die Konsequenzen der Aspektorientierten Datenhaltung berücksichtigen. Daher ist der praktische Einsatz momentan nur über die genannten Zwischenschichten realisierbar. Andererseits wird dem Anwendungsentwickler ein intuitives und adäquates Konstrukt zur Verfügung gestellt, um transparent auf funktionale Aspekte von Fachtabellen zugreifen zu können.

Schließlich ergibt sich für den **API-Ansatz** (Abschnitt 3.4) eine mindestens ebenso gute oder bessere Bewertung gegenüber den anderen Zugriffstechniken in vielen Kriterien, insbesondere ist ein performanter Aspekt-Zugriff möglich. Der fehlenden Standardisierung lässt sich z.B. durch Portierung auf die gängigsten Programmiersprachen begegnen. Großer Aufwand ist zudem notwendig, um der Mächtigkeit von SQL auf Seiten der API gerecht zu werden.

5. ZUSAMMENFASSUNG

Der vorliegende Beitrag hat vier verschiedene Techniken für den Zugriff auf funktionale Aspekte aufgezeigt, welche im Kontext der Aspektorientierten Datenhaltung über ein ebenfalls kurz beschriebenes Referenzmodell auf relationalen Datenbanken abgebildet werden können. Durch die Fokussierung auf RDBMS und deren hohen Verbreitungsgrad ist die Grundlage für den praktischen Einsatz gewährleistet. Die anschließende Bewertung anhand zuvor aufgestellter Kriterien sollte weitere Klarheit über die Potentiale der einzelnen Ansätze liefern. Dabei hat sich herausgestellt, dass vor allem für eine performante und transparente Auswertung funktionaler Aspekte die standardisierten Mittel von SQL nicht ausreichen.

Unter den möglichen Alternativen erscheinen der Vorschlag zur Erweiterung von SQL mit ASPECTVIEW und die applikative Verarbeitung in einem Funktionsbaustein vielversprechend. Dabei ist der Aufwand zur Erweiterung der SQL-Norm ungleich höher und nicht automatisch von Erfolg gekrönt bzw. wie bereits vorgeschlagen nur durch Implementierung einer transformierenden Zwischenschicht praktikabel. Aus diesem Grund werden sich nachfolgende Arbeiten vor allem der prototypischen Realisierung, quantitativen Performance-Vergleichen sowie funktionellen Weiterentwicklungen bezüglich aktueller Einschränkungen der vorgestellten API für funktionale Aspekte widmen.

6. LITERATUR

[1] A. Adam, S. Leuoth, and W. Benn. Nutzung von Proxys zur Ergänzung von Datenbankfunktionen. In *Proceedings of the 22nd GI Workshop Grundlagen von Datenbanken (GvDB 2010)*, pages 31–35, Bad Helmstedt, Germany, May 2010.

[2] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger. Multi-tenant databases for software as a service: schema-mapping techniques. In *SIGMOD Conference*, pages 1195–1206, 2008.

[3] R. Chitchyan, I. Sommerville, and A. Rashid. An Analysis of Design Approaches for Crosscutting Concerns. In *Workshop on Aspect-Oriented Design (held in conjunction with the 1st Aspect Oriented Software Development Conference (AOSD 2002)*, 2002.

[4] E. F. Codd. A relational model of data for large shared data banks. *CACM*, 13(6):377–387, 1970.

[5] C. Cunningham, G. Graefe, and C. A. Galindo-Legaria. PIVOT and UNPIVOT: Optimization and Execution Strategies in an RDBMS. In *(e)Proceedings of the 30th International Conference on Very Large Data Bases (VLDB 2004)*, pages 998–1009, 2004.

[6] C. Date and H. Darwen. *SQL - Der Standard. SQL/92 mit den Erweiterungen CLI und PSM*. Addison-Wesley, 1998.

[7] V. Dinu, P. Nadkarni, and C. Brandt. Pivoting approaches for bulk extraction of Entity-Attribute-Value data. *Computer Methods And Programs in Biomedicine*, 82(1):38–43, 2006.

[8] T. Egyedi. Why Java Was Not Standardized Twice. *Hawaii International Conference on System Sciences*, 5(1):5015–5025, 2001.

[9] M. Fisher, J. Ellis, and J. Bruce. *JDBC API Tutorial and Reference (Java Series)*. Addison-Wesley, 2003.

[10] *ISO/IEC 9075-2:2008. Information technology – Database languages – SQL – Part 2: Foundation (SQL/Foundation)*. ISO, Geneva, Switzerland, 2008.

[11] M. Liebisch. Aspektorientierte Datenhaltung - ein Modellierungsparadigma. In *Proceedings of the 22nd GI Workshop Grundlagen von Datenbanken (GvDB 2010)*, pages 13–17, Bad Helmstedt, Germany, May 2010.

[12] M. Liebisch. Supporting functional aspects in relational databases. In *Proceedings of the 2nd International Conference on Software Technology and Engineering (ICSTE 2010)*, pages 227–231, San Juan, Puerto Rico, USA, Oct. 2010.

[13] M. Liebisch and M. Plietz. Performance-Analysen für Realisierungsansätze im Kontext der Aspektorientierten Datenhaltung. Institut für Informatik, Friedrich-Schiller-Universität Jena, Nov. 2010.

[14] D. Lorentz. *Oracle Database SQL Language Reference 11g Release 1*. Oracle, Aug. 2010.

[15] P. Nadkarni, L. Marenco, R. Chen, E. Skoufos, G. Shepherd, and P. Miller. Organization of Heterogeneous Scientific Data Using the EAV/CR Representation. In *JAMIA*, 6, pages 478–493, 1999.

[16] B. Pietsch. Entwurf einer Zugriffsschicht für funktionale Aspekte in DBMS. Studienarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, Mar. 2011.

[17] A. B. Rashid and M. Islam. Role of Materialized View Maintenance with PIVOT and UNPIVOT Operators. In *Proceedings of the 1st IEEE International Advance Computing Conference (IACC 2009)*, pages 915–955, Mar. 2009.

[18] T. Rizzo, A. Machanic, and J. Skinner. *Pro SQL Server 2005*. Apress, 2005.

[19] T. Schilling. Realisierungskonzepte für die Aspektorientierte Datenhaltung. Studienarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, Apr. 2011.

[20] C. M. Wyss and E. L. Robertson. A formal characterization of PIVOT/UNPIVOT. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 602–608, 2005.

Verbindung relationaler Datenbanksysteme und NoSQL-Produkte

Ein Überblick

Andreas Göbel
Friedrich-Schiller Universität Jena
Lehrstuhl für Datenbanken und Informationssysteme
Ernst-Abbe-Platz 2
07743 Jena, Germany
andreas.goebel@uni-jena.de

KURZFASSUNG

In den letzten Jahren entstanden verschiedene Open-Source-Systeme, die mit fundamentalen Konzepten und Regeln relationaler Datenbanksysteme brachen, um die Verwaltung von Daten in speziellen Einsatzbereichen zu optimieren. Die wesentlichen Gründe für die Entwicklung dieser so genannten NoSQL-Systeme sind jedoch nicht SQL oder das relationale Datenbankmodell, sondern sie ist auf die Implementierung relationaler Datenbanksysteme zurückzuführen. Der Beitrag verdeutlicht durch eine Gegenüberstellung von Oracle Real Application Cluster, IBM DB2 PureScale und MySQL Cluster die gegensätzlichen Implementierungen relationaler Clusterlösungen. An die Motivation der NoSQL-Produkte sowie einen Überblick ihrer Zielstellung, Vor- und Nachteile schließt sich das Aufzeigen von Möglichkeiten an, um Konzepte und Implementierungen beider Welten miteinander zu verbinden und so die Vorzüge zu vereinen.

Kategorien und Themenbeschreibung

H.2.4 [**Database Management**]: Systems—*Parallel databases* ; H.3.5 [**Database Management**]: Systems and Software—*Distributed systems*

Allgemeine Bestimmungen

Theory, Design, Reliability

Stichworte

Parallel Databases, NoSQL, Postrelational, Hybrid

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

1. EINLEITUNG

Die zunehmende Verbreitung von Unternehmensnetzwerken, globalen Netzwerken wie dem Internet und mobilen Endgeräten gepaart mit dem Wunsch vieler Unternehmen nach Globalisierung führt vermehrt zur Nutzung zentraler (Datenbank-)Services für eine Vielzahl von Nutzern. Die unter dem Begriff Web 2.0 zusammengefassten Entwicklungen ermöglichen zunehmend Interaktion und Verknüpfungen in Netzwerken, was sowohl die Gestalt als auch die Menge der Daten auffallend beeinträchtigt. So werden Inhaber erfolgreicher Web-Anwendungen mit beachtlichen Datenmengen konfrontiert, die das Datenaufkommen in klassischen Anwendungen um ein Vielfaches übersteigen können.

Relationale Datenbanksysteme sind zentraler Bestandteil des Software-Stacks vieler Unternehmen und Behörden. Mittels der Verbindung eines mathematischen Fundaments, der Gewährleistung der ACID-Eigenschaften und der standardisierten deskriptiven Abfragesprache SQL stellen sie die Verfügbarkeit, Korrektheit und Auswertbarkeit der Unternehmensdaten sicher. Der vorliegende Beitrag motiviert, warum Betreiber vieler Web-Anwendungen trotz der auf der Hand liegenden Vorteile bewährter relationaler Produkte Eigenentwicklungen propietärer Spezialsysteme zur Datenverwaltung vorantreiben, die bewusst auf wesentliche Merkmale relationaler Systeme verzichten.

Nach einer Gegenüberstellung relevanter Implementierung relationaler Clusterdatenbanken werden die Herausforderungen und Einschränkungen der zu dem Schlagwort NoSQL zusammengefassten Systeme herausgearbeitet, einige aktuelle Entwicklungen zur Verbindungen von NoSQL und RDBMS zusammengefasst und die Notwendigkeit flexiblerer Implementierungen relationaler Datenbanksysteme aufgezeigt.

2. HERAUSFORDERUNGEN

Die Charakteristika zu verarbeitender Daten bei Web-Anwendungen führen zu folgenden Kern-Herausforderungen an zu verwendende Datenbanksysteme bzw. Datenspeicher.

Performance und Skalierbarkeit kennzeichnen die bedeutendsten Herausforderungen. Die damit verbundene Verringerung der Latenzzeit in Web-Anwendungen steht häufig in direktem Zusammenhang mit der Nutzerzufriedenheit und ist insbesondere in Bereichen wie Suchmaschinen oder dem E-Commerce-Sektor von essentieller Bedeutung. Die Performance resultiert aus der Grundperformance für Anfra-

gen und der Verarbeitungsgeschwindigkeit für steigende Datenvolumina, welche allgemeinhin als Skalierbarkeit bezeichnet wird. Eine zunehmende Datenmenge kann hierbei die Dauer von Aufgaben, die Anzahl der Aufgaben oder beides erhöhen. Die Skalierbarkeit eines Rechensystems kann durch den Einsatz leistungsfähigerer Hardware (vertikale Skalierbarkeit) oder durch das Verteilen der Aufgaben auf weitere Rechenressourcen (horizontale Skalierbarkeit) erzielt werden. Die Vorgänge müssen jeweils transparent zur Anwendung geschehen.

Ausfallsicherheit ist für jedes zentrale (Datenverarbeitungs-)System eine wesentliche Herausforderung, um Nutzern dauerhafte Verfügbarkeit zu bieten. Neben ungeplanten Ausfällen eines Systems in Folge von Hardwaredefekten oder Systemfehlern müssen auch geplante Ausfälle – beispielsweise zur Aktualisierung des Systems – vermieden werden. Beide Ausfallarten können erheblichen wirtschaftlichem Schaden durch Kundenverlust oder Pönalen bei Verstoß gegen Service Level Agreements nach sich ziehen. Um Hochverfügbarkeit zu erreichen, sollten Single Points of Failure (SPoF) in einem System vermieden sowie binnen kurzer Zeit und automatisiert auf jegliche Art von Fehlern reagiert werden.

Schemaflexibilität bezeichnet den Verzicht auf ein vordefiniertes und stets omnipräsentes Datenbankschema, um den Umgang mit Datenbanken und -speichern flexibler zu gestalten. Dies ermöglicht die adäquate Verwaltung semistrukturierter und dokumenten-orientierter Daten, die nicht zuletzt aufgrund von Web-Standards und Auszeichnungssprachen wie XML oder RDF in Web-Anwendungen weit verbreitet sind. Schemaflexibilität spielt des Weiteren eine wichtige Rolle bei der Konsolidierung von heterogenen Nutzerdaten innerhalb eines Systems.

Kosten: Für viele Betreiber von Web-Anwendungen ist der Einsatz kostengünstiger Hard- und Software eine Grundvoraussetzung. Lizenz-, Support- und Administrationskosten für Datenbanksysteme sowie die Anschaffungs-, Administrations- und Betriebskosten von Datenbankservern machen meist einen nicht unerheblichen Teil der IT-Gesamtaufwendungen aus. Aus diesem Grund wird für Unternehmen die Nutzung von kostengünstigen Cloud-Services oder -Storages stets lukrativer. Entsprechend sollte ein geeignetes Lizenzierungskonzept angeboten und der Einsatz auf Commodity-Servern unterstützt werden.

Für viele Provider ist die Antwortzeit der Web-Anwendung derart wichtig, dass sie Einschränkungen der Datenkonsistenz in Kauf nehmen oder gar auf die Realisierbarkeit des Definierens von Konsistenzsicherungen verzichten, wenn diese einen Performance-Overhead mit sich bringen. Dies ist bemerkenswert, denn es kennzeichnet einen wahrnehmbaren Wandel der Anforderungen an Datenbanksysteme. In klassischen Unternehmensanwendungen stellt die Forderung nach Datenkonsistenz die oberste Prämisse dar und ist unentbehrlich. Die Herausforderung besteht hierbei im Wesentlichen in der Optimierung der Performance. Dementgegen verdeutlichen die obigen Herausforderungen, dass die Hauptaufgaben vermehrt in der Optimierung der Antwortzeit oder nach [3] gar in der Minimierung der (Hardware-)Kosten und Erhöhung des Konsistenzniveaus bei gegebenen Performance-Vorgaben zu sehen ist.

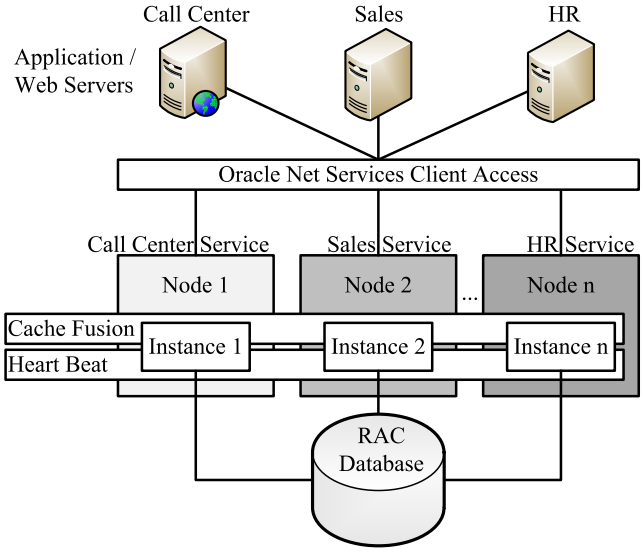


Abbildung 1: Architektur von Oracle RAC (nach [12])

3. RELATIONALE DATENBANKCLUSTER

Horizontale Skalierbarkeit und Hochverfügbarkeit unter Einsatz kostengünstiger Hardware bilden nach Abschnitt 2 die wesentlichen Herausforderungen an die Speicherung und Verarbeitung der Daten von Web-Anwendungen. Beinahe alle relationalen Datenbanksysteme bieten Mittel, um ihre Systeme vor Ausfällen und Datenverlust in diversen Fehlerszenarien zu schützen und eine hohe Verfügbarkeit zu erzielen. Sie bieten für dieses Ziel neben Sicherungs- und Wiederherstellungsmöglichkeiten von Datenbanken verschiedene Techniken zur Replikation. Zumeist erkennen sie ein Problem jedoch erst beim erfolglosen Datenzugriff statt unmittelbar nach dem Auftreten und erfordern im Fehlerfall einen manuellen Eingriff zum Umleiten auf das Replikat. Zudem sind die Replikate bei einigen Systemen ausschließlich im Fehlerfall einsetzbar und dienen im Normalbetrieb nicht der Lastbalancierung. Im Folgenden werden die Eckpunkte vorherrschender Hochverfügbarkeitslösungen gegenübergestellt, die diese Mängel nicht aufweisen und zudem horizontale Skalierbarkeit in Mehrrechnersystemen ermöglichen.

3.1 Oracle Real Application Cluster

Oracle Real Application Cluster (RAC) ermöglicht bis zu 100 Datenbankinstanzen einen parallelen Zugriff auf denselben Datenbestand und realisiert somit eine Shared-Disk-Architektur. Wie Abbildung 1 verdeutlicht, greifen die Application Server und Web Server über eine gemeinsame Service-Schnittstelle auf das System zu, die u.a. der Lastbalancierung dient. Sämtliche Dateien für Daten, Verwaltung und Konfigurationsparameter werden auf einem clusterfähigen Storage-System gespeichert und sind von allen Servern les- und schreibbar. Lediglich die Undo- und Redo-Logs bilden eine Ausnahme: Sie werden stets von der Besitzerinstanz geschrieben und können nur von deren Nachbarinstanzen gelesen werden, um die Besitzerinstanz bei einem Ausfall automatisch wiederherstellen zu können. Der Ausfall eines Knotens wird durch eine Heartbeat-Netzwerkverbindung in kürzester Zeit erkannt. Extended Distance Cluster bietet durch

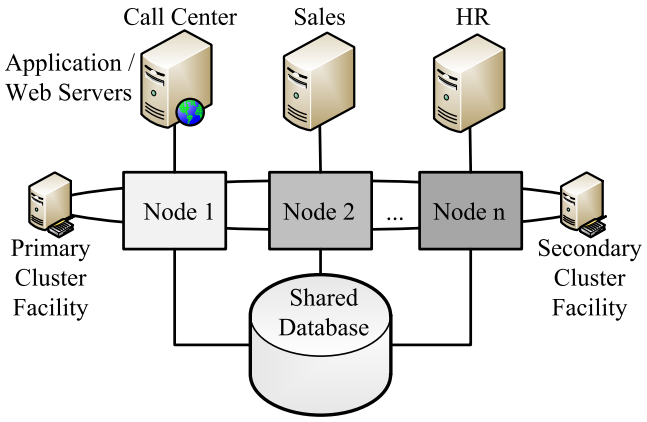


Abbildung 2: Architektur von IBM DB2 PureScale (nach [9])

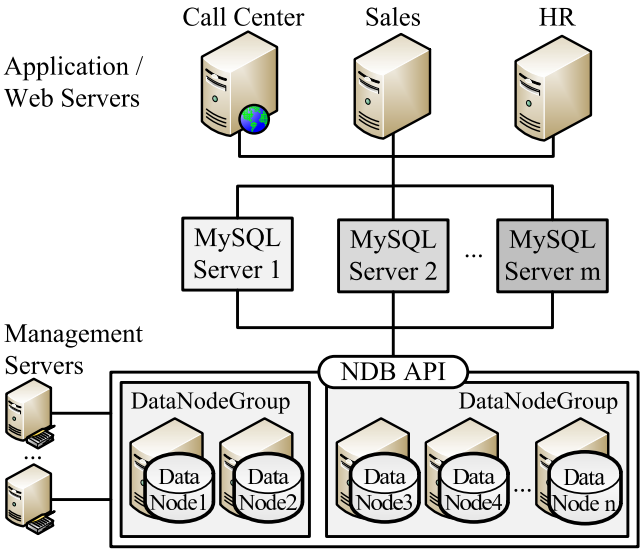


Abbildung 3: Architektur von MySQL Cluster (nach [10])

eine Systemspiegelung auf ein aktives System innerhalb weniger Kilometer das Reagieren auf Fehlerszenarien, die zum Ausfall des kompletten Clusters führen. Oracle Data Guard ermöglicht darüber hinaus das Spiegeln auf ein weiter entferntes Standby-System zur Realisierung einer Disaster Recovery.[12, 5]

Oracle RAC bietet Skalierbarkeit durch das Hinzufügen neuer Nodes, einen automatischen Lastausgleich und die parallelisierte Ausführung von Operationen auf mehreren Servern. Für den parallelen Zugriff mehrerer Instanzen auf denselben Datenbestand nutzt Oracle im Falle einer Datenmodifikation den Global Cache Service (GCS), um zu bestimmen, in welchen lokalen Knotencaches die betroffenen Blöcke liegen bzw. ob sie sich gegebenenfalls bereits auf dem Storage-System befinden. Nachdem die Position bekannt ist, werden die Blöcke durch ein In-Memory-Blockinventar (Global Resource Directory, GRD) und Global Enqueue Service auf aktive Schreibsperren und weitere wartende Instanzen geprüft, um anschließend eigene Schreibsperren zu setzen, die wiederum im GRD vermerkt und anderen Nodes bekannt gemacht werden. Die verwendeten Komponenten werden unter dem Begriff Cache Fusion zusammengefasst und ermöglichen zudem beim Datenzugriff das direkte Versenden von Daten zwischen Buffer-Caches verschiedener Nodes. Oracle RAC vermeidet somit Cache-Kohärenz und ein SPoF durch einen globalen Cache, jedoch auf Kosten von sehr viel Kommunikation.[12, 5]

3.2 IBM DB2 PureScale

Das Design der IBM-Clusterlösung DB2 pureScale basiert auf der Architektur des bewährten Parallel Sysplex für System z[1]. Sie ermöglicht durch eine Shared-Disk-Architektur den gemeinsamen Zugriff von bis zu 128 Datenbankservern auf einen gemeinsamen Datenbestand, der durch IBMs General Parallel File System zur Verfügung gestellt wird. Die Abbildung 2 zeigt, dass der Cluster neben den Datenbankservern aus Cluster Facilities (CF) besteht. Um einen SPoF zu verhindern, ist diese Komponente meist doppelt ausgelegt. Sie kann ein eigenständiges System sein oder auf einem Clusterknoten betrieben werden. Die CF ermöglicht die zentrale Verwaltung der Sperren (Global Lock Table, GLT) und des Caches (Group Buffer Pool, GBP), wodurch analog zum GRD und GCS des Oracle RAC die Informationen der Datenblöcke verwaltet und allen Servern zur Verfügung gestellt werden. Die Server sind untereinander sowie mit der CF mittels eines Hochleistungsnetzwerks verbunden, welches einen direkten Fernzugriff auf den Arbeitsspeicher (RDMA) in wenigen Mikrosekunden ermöglicht. Bei einem Schreibvorgang ermöglicht diese schnelle Verbindung das synchrone Aktualisieren der zentralen Sperrtabelle in Form von Zeilen- und Seitensperren und des zentralen wie auch anderer relevanter Caches. Beim Lesevorgang eines Nodes wird nach erfolgloser Suche im lokalen Cache im GBP nach den Blöcken gesucht. Werden die Daten vom Festspeicher in den lokalen Cache geladen, wird dies ebenfalls dem GBP bekannt gemacht. [9, 6]

Ein integrierter Watchdog-Prozess überwacht permanent die Verfügbarkeit sämtlicher Knoten. Wird der Ausfall eines Knotens bemerkt, stehen bis zum Instanzneustart lediglich die momentan von diesem Knoten aktualisierten Tupel nicht zur Verfügung. Logs werden im Gegensatz zu Oracle RAC auf den gemeinsamen Festspeicher geschrieben und sind für die Recovery von anderen Knoten lesbar.

3.3 MySQL Cluster

MySQL Cluster basiert im Gegensatz zu den Lösungen von IBM und Oracle auf einer Shared-Nothing-Architektur, weshalb die bis zu 255 Datenknoten nicht parallel auf einen gemeinsamen Datenbestand zugreifen, sondern jeder Datenknoten einen Teil des Gesamtdatenbestands verwaltet. Die Tabellen werden bei diesem Ansatz horizontal partitioniert. MySQL Cluster stellt keine spezifischen Voraussetzungen an zu verwendende Netzwerke oder Server und unterstützt In-Memory- als auch Festpeicher-Datenspeicherung. Auf das System wird über vollwertige MySQL-Server zugegriffen. Sie sind mit einer Schnittstelle zur NDB-Engine versehen und werden zudem für verschiedene Funktionen wie Views, Trigger oder Volltext-Indizes verwendet, die von der NDB-Engine nicht unterstützt werden. Die Management-Server sind für die Konfiguration des Clusters zuständig, während die

[1] Auf eine gesonderte Beschreibung des Parallel Sysplex wird aufgrund analoger Konzepte verzichtet.

Datenknoten zur Speicherung der Daten und der Verwaltung von Transaktionen dienen. Knoten, die deckungsgleiche Inhalte verwalten, werden zu einer Datenknotengruppe zusammengefasst, in der die synchrone Replikation der Knoten dazu führt, dass der Ausfall von Knoten keine aufwändige Instanz -Wiederherstellung nach sich zieht. Somit müssen Undo- und Redo-Dateien anderen Knoten nicht sichtbar gemacht werden und das System ist verfügbar, solange ein Datenknoten je Gruppe erreichbar ist. Da beim Ausfall eines Knotens die Aktualisierung einer zentralen Sperr- und Pufferverwaltung nicht nötig ist, können sehr geringe Failover-Zeiten erzielt werden. Zudem werden asynchron Checkpoints auf einen Festspeicher geschrieben, um auf den Ausfall kompletter Gruppen reagieren zu können bzw. einen System-Reboot zu ermöglichen. Durch den Einsatz der MySQL Cluster Carrier Grade Edition kann Hochverfügbarkeit durch die Realisierung geografischer Replikation erzielt werden.[10, 11, 5]

3.4 Bewertung

Die vorgestellten Datenbankcluster ermöglichen Skalierbarkeit sowohl durch Einsatz leistungsstärkerer Server als auch durch das Hinzufügen weiterer Server. Trotz verschiedener Realisierungen verfügen sie über effiziente Strategien für die wesentlichen Herausforderungen im Kontext der Skalierbarkeit: Logging, Locking und die Verwaltung von Zwischenspeichern[13]. Im Gegensatz zum Shared-Nothing-Ansatz von MySQL Cluster basieren Oracle RAC und DB2 pureScale auf einer Shared-Disk-Architektur und benötigen wegen ihrer nahen Knotenkopplung schnelle Kommunikation mittels Hochleistungsnetzwerken. Diese ist für die aufwändige Kommunikation der Sperr- und Cachingverwaltung bei gegebener Performance notwendig.

Alle Systeme bieten hohe Ausfallsicherheit bis hin zu Unterstützung einer Disaster-Recovery über die Anbindung entfernter Standby-Systeme. Serverausfälle werden beinahe unmittelbar erkannt, die Wiederherstellung ist in kürzester Zeit möglich und führt kaum zu wenig Einschränkungen. Während bei Oracle RAC im Fehlerfall bis zum Neuaufbau des CGS für einen Augenblick keine Datenmodifikation durchgeführt werden können, stehen bei DB2 PureScale die vom ausgefallenen Knoten aktuell veränderten Daten bis zur Instanz-Wiederherstellung nicht zur Verfügung. MySQL Cluster besitzt durch den Shared-Nothing-Ansatz in Verbindung mit synchroner Replikation der In-Memory-Daten im Fehlerfall kaum Einschränkungen.

Die wesentlichen Nachteile von Oracle RAC und DB2 pureScale bestehen im Kontext der Anforderungen in Abschnitt 2 vor allem in den enormen Kosten für Lizenzen, spezielle Hardware und Wartung im Vergleich zu MySQL Cluster. Insbesondere sind hier der vor Ausfällen zu schützende Shared Storage, das Cluster-Dateisystem sowie leistungsstarke Netzwerke für die Clusterkommunikation und Cache Fusion bzw. die Cluster Acceleration Facilities zu nennen. Oracle RAC wurde zudem in den vergangenen Jahren um diverse Features ergänzt, die zu einer System-Komplexität führten, die eine intensive Einarbeitungszeit unabdingbar macht.

Ein wesentlicher Vorteile von Oracle RAC und DB2 pureScale ist hingegen die einfache Migration von Anwendungen auf die Clustersysteme, da keine Änderung des Anwendungscodes notwendig ist. Da die NDB-Engine von MySQL Cluster nur einen Teil der Funktionen von InnoDB und MyISAM unterstützt, müssen fehlende Funktionalitäten auf die MySQL-Server ausgelagert werden, um deren Performance und Verfügbarkeit manuell gesorgt werden muss. Daher bietet sich der MySQL Cluster vor allem in Szenarien mit einer Vielzahl simpler Anfragen und hohen Latenz- und Verfügbarkeitsanforderungen an, während die Einsatzmöglichkeiten von Oracle RAC und DB2 pureScale kaum begrenzt sind.

4. NOSQL-BEWEGUNG

In den letzten Jahren gewinnen so genannte NoSQL-Systeme zur Verwaltung von Daten zunehmend an Bedeutung. Einige Kritikpunkte bei der Verwendung relationaler (Cluster-)Systeme in der Welt der Services regten Unternehmen zur Eigenentwicklung von Systemen zur Datenspeicherung und -verarbeitung an, die bewusst auf Merkmale relationaler DBMS verzichten, um sich auf einen Anwendungsfall zu spezialisieren. Ausgehend von den technischen Beschreibungen von Systemen bekannter Internetgrößen entstanden im Laufe der letzten Jahre eine Vielzahl von Open-Source-Systemen. Diese kopierten, kombinierten und erweiterten die Konzepte der Ausgangssysteme mit dem Ziel, den Anforderungen der Unternehmen gerecht zu werden. Der Begriff „NoSQL“ umfasst all jene Systeme und wird inzwischen üblicherweise als „Not only SQL“ ausgelegt. Das Ziel dieser Systeme besteht im Aufzeigen von Alternativen zu relationalen Datenbanksystemen und nicht in deren Ablösung.

4.1 Zielstellungen

Mangels einer anerkannten Definition des Begriffs „NoSQL“ werden im Folgenden entsprechend der in Abschnitt 2 beschriebenen Herausforderungen die wesentlichen Zielstellungen der NoSQL-Systeme zusammengefasst, wobei diese als Obermenge der Ziele jedes einzelnen NoSQL-Systems zu sehen sind.

Performance, Skalierbarkeit: Untersuchungen wie [14] zeigen, dass die Performance moderner RDBMS in verschiedenen Bereichen um ein Vielfaches übertroffen werden kann. Als Grund wird vor allem die nach wie vor auf System R basierende und stets erweiterte Architektur gesehen, welche in der Client-Server-Welt hervorragende Dienste leistet, für die Welt der Services und die verschiedenden Leistungs- und Kapazitätsverhältnisse von Prozessoren, Fest- und Arbeitsspeicher jedoch neuer Architekturansätze bedarf [16]. Das Hauptziel der meisten NoSQL-Datenspeicher ist das Erreichen linearer horizontaler Skalierbarkeit zur Verarbeitung riesiger Datenmengen. Sie nutzen hierfür überwiegend Shared-Nothing-Architekturen in Verbindung mit horizontaler Partitionierung der Daten. Das im Jahre 2002 bewiesene Eric Brewers CAP-Theorem besagt, dass nur zwei der drei folgenden Eigenschaften eines verteilten Systems erfüllt sein können [4].

- Consistency: Zu jedem Zeitpunkt sehen alle Knoten denselben Datenbestand.
- Availability: Knoten können Datenbestände jederzeit schreiben und lesen.
- Partition tolerance: Das System arbeitet trotz einer Zerteilung in Teilsysteme weiter.

Während relationale Datenbanksysteme stets auf die Wahrung der Konsistenz bestehen und dies zur Beeinträchtigung der Performance und Skalierbarkeit nach sich zieht, verfolgen viele NoSQL-Systeme den im Abschnitt 2 aufgefassten

Ansatz, strenge Konsistenzforderungen zugunsten der Performance aufzugeben.

Ausfallsicherheit: Ein Großteil der NoSQL-Systeme bietet hervorragende Replikations- und Failovertechniken, um Ausfälle von Knoten innerhalb einer Shared-Nothing-Architektur zu kompensieren, indem das System vor Datenverlust geschützt und der laufende Betrieb minimal beeinflusst wird.

Schemaflexibilität: NoSQL-Systeme verdeutlichen, dass neben dem relationalen Datenbankmodell andere Datenmodelle existieren, die Daten gemäß ihrer Eigenschaften adäquat speichern, ohne sie in ein fixes Datenbankschema zu fügen. Für einfache, schemafreie Daten bieten Key-Value-Stores die Möglichkeit, mehrattribute Objekte anhand eines eindeutigen Schlüssels zu speichern und abzufragen. Dokumenten-basierte Systeme erlauben zudem das Speichern komplexerer Inhalte wie verschachtelte Daten und bieten durch leistungsfähigere Abfragesprache beispielsweise das Suchen auf beliebigen Attributen. Wide-Column-Stores vereinen hingegen Vorzüge des Relationenmodells mit Funktionalitäten wie flexiblen Schemata und Versionierung. Diese Datenmodelle werden beispielweise durch die Graphen-DBS ergänzt. Die Komplexität des Datenmodells spiegelt sich meist in der zur Verfügung gestellten Programmierschnittstelle bzw. Abfragesprache wieder, es existieren für die Datenmodelle kaum standardisierte Notationen und standardisierte, deskriptive Sprachen. Entsprechend ihrer Zielstellung bieten sie häufig auf REST basierende Schnittstellen.[15]

Kosten: Das Gros der Systeme wird als Open Source und mit wenigen Nutzungseinschränkungen zur Verfügung gestellt. Die Installation und Verwendung der Systeme ist meist unkompliziert. Zudem ist häufig ein Betrieb auf günstigen Commodity Servern möglich, da Einschränkungen bezüglich der zu verwendenen Hardware kaum vorhanden sind und geläufige Betriebssysteme unterstützt werden.

4.2 Bewertung

NoSQL-Datenspeicher sind hervorragend geeignet, um kostengünstig skalierbare und hochverfügbare Datenspeicherung und -verarbeitung in einem begrenzten Anwendungsfall bereitzustellen. Aus Sicht dieser Systeme sind die größten Hürden beim Einsatz relationaler Systeme für Web-Applikationen nicht das relationale Datenbankmodell, ACID oder gar SQL. So zeigen aktuelle Entwicklungen im Bereich relationaler Datenbanksysteme wie VoltDB[2] oder HyPer[3], dass diese Merkmale eine lineare Skalierbarkeit nicht zwingendermaßen ausschließen. Im Zentrum der Beanstandungen stehen hingegen die Tatsachen, dass keine bewährten parallelen und hochverfügbaren Open Source RDBMS existieren und die *Implementierung* bewährter relationaler DBMS häufig keine hinreichende Skalierbarkeit zulässt.

Die Spezialisierung der NoSQL-Systeme auf eine wenige Anwendungsgebiet verwehrt in vielen Fällen den Einsatz bei sich ändernden Anforderungen, wie beispielsweise dem Wunsch komplexer Abfragen auf Daten bei simplen Datenmodellen. Bei der Nutzung eines relationalen DBMS wären hierbei kaum Änderungen vonnöten, während ein NoSQL-System angepasst oder gar ausgetauscht werden muss. Ein Austausch gestaltet sich zudem schwierig, da es den Systemen an standardisierten Notationen und Schnittstellen mangelt. Zudem werden statt deskriptiven Sprachen je nach Datenmodell meist Low-Level-Abfragesprachen verschiedenen Komplexitäts- und Mächtigkeitsgrades genutzt, was aus Sicht des Programmierers ein Fortschritt, aus Sicht eines Datenbänklers aber durchaus als Rückschritt gesehen werden kann [2]. Insbesondere der Verzicht einiger NoSQL-Systeme auf die Gewährleistung der ACID-Eigenschaften führt dazu, dass ein Großteil von Unternehmen den Einsatz dieser Systeme ausschließen wird.

[2]http://voltdb.com/

[3]http://www3.in.tum.de/research/projects/HyPer/

5. VERBINDUNG BEIDER WELTEN

Relationale Datenbanksysteme bieten aufgrund jahrelanger Forschung und Entwicklung u.a. eine enorme Verbreitung und Bekanntheit, ein ausgereiftes mathematisches Fundament, die Datenbanksprache SQL und nicht zuletzt zugesicherte Transaktionseigenschaften durch ACID. Auf der anderen Seite existieren NoSQL-Systeme, deren Verbreitung sich in der Regel auf wenige Web-Anwendungen beschränkt. Charakterisiert durch die in Abschnitt 4 zusammengefasste Eigenschaften sowie die verwendeten Konzepte, weisen sie zum Teil Zielstellungen auf, die sich deutlich von der Zielstellung klassischer relationaler Datenbanksysteme unterscheidet.

Eine Verbindung von Konzepten und Implementierungen relationaler Datenbanksysteme und NoSQL Data Stores kann dazu genutzt werden, die Vorteile beider Welten zu vereinen. Im Folgenden werden mögliche Ansätze zur Vereinigung anhand stellvertrender Beispiele vorgestellt.

5.1 Erweiterungen von NoSQL-Produkten

NoSQL-Systeme wurden in der Regel für ein spezielles Anwendungsgebiet entwickelt. Durch eine Erweiterung der Systeme kann ihr Einsatzbereich vergrößert werden, wodurch sie die Aufmerksamkeit von mehr Unternehmen auf sich ziehen können. Somit wird neben der Erweiterung der Funktionalität auch die Bekanntheit des Produkts gesteigert. Ein Beispiel für diesen Ansatz ist das Produkt Hive[17], welches das NoSQL-System Hadoop um die deskriptive, SQL-ähnliche Sprache HiveQL erweitert und Schnittstellen in Form eines CLIs, einer Web-Gui und JDBC/ODBC bietet. Zudem schafft es durch komplexe Analysen und das Absetzen von Ad-hoc-Abfragen die Voraussetzung, Hadoop für Data Warehousing zu nutzen.

5.2 Hybridsystem

Hybride Systeme wie HadoopDB[1] führen zu einem Kompromiss zwischen zwei unterschiedlichen Produktwelten und erschaffen dabei Produkte mit neuen Funktionalitäten. Das Open-Source-Produkt HadoopDB kombiniert MapReduce in Form der Implementierung Hadoops sowie Hive und PostgreSQL, wobei das System bereits mit anderen Datenbanksystemen getestet wurde. HadoopDB kann sowohl SQL-Anfragen als auch MapReduce-Jobs entgegennehmen und bietet den Zugriff auf Hadoops verteiltes Dateisystem HDFS oder alternativ auf ein Datenbanksystem wie PostgreSQL an. In der Folge sind Nutzer durch die Verwendung von HadoopDB in der Lage, mittels SQL auf ein Shared-Nothing-DBMS zuzugreifen.

5.3 Anpassung von RDBMS

Der Abschnitt 4 verdeutlicht, dass die bewährten fundamentalen Konzepte hinter dem relationalen Datenbankmodell mit Anforderungen wie enormer Skalierbarkeit vereinbar sind, es hierzu jedoch einer Anpassung der von System R abstammenden Architektur bedarf. Die Implementierungen

von DBMS müssen sich durch geeignete Konfigurationsmöglichkeiten weit mehr als bisher an verschiedene Einsatzzwecke anpassen lassen. Realisiert werden kann dies beispielsweise durch die Ausnutzung der Austauschbarkeit von Komponenten in modularen DBMS-Architekturen wie [7] oder die Implementierung von adaptierbaren DBMS-Komponenten.

Ein möglicher Ansatzpunkt dieses Konzepts könnte das Anbieten einer wahlweisen Speicherung auf langsamen, persistenten Festspeichern oder im schnellen, flüchtigen Arbeitsspeicher oder einer kombinierten Lösung sein, was bereits in einigen Systemen wie dem in Abschnitt 3.3 beschriebenen MySQL Cluster möglich ist. Hierdurch bieten sich entsprechend der Charakteristika und des Umfang der zu speichernden Daten sowie den Zugriffseigenschaften verschiedene Einsatzmöglichkeiten. Orthogonal kann wahlweise eine spalten- oder zeilenbasierte Speicherung angeboten werden, um sowohl im OLTP- als auch im OLAP-Bereich überzeugende Leistungskennzahlen zu erzielen. Für die Implementierung bieten einige Systeme bereits verschiedene Storage-Engines innerhalb eines Systems.

Auch die Transaktionsverwaltung relationaler Datenbanksysteme bietet sich bezüglich einer Erweiterung an, indem neben den harten Anforderungen von ACID und den heute wählbaren Isolationsszenarien weitere Transaktionskonzepte mit schwächeren Anforderungen integriert werden und Administratoren die Wahl des Transaktionskonzepts überlassen wird. Aus Sicht des in Abschnitt 4 angesprochenen CAP-Theorems könnten je nach Konfiguration des Systems verschiedene CAP-Eigenschaften erfüllt werden und somit das Datenbanksystem an verschiedene Einsatzzwecke angepasst werden. Die Realisierung kann beispielsweise über ein autonomes Modul zur Transaktionsverwaltung in einer modularen DBMS-Architektur gemäß [8] erfolgen.

6. ZUSAMMENFASSUNG

In diesem Beitrag wurde die Notwendigkeit adaptierbarer flexibler RDBMS-Implementierungen aufgezeigt. Als Grundlage diente der Vergleich von Oracle RAC, IBM DB2 PureScale und MySQL Cluster. Er verdeutlichte, dass die Hersteller zum Erreichen des Ziels eines horizontal skalierbaren und hochverfügbaren Clusterdatenbanksystems gemäß verschiedener Implementierungsansätze verfahren. Während Oracle RAC und IBM DB2 PureScale sich durch gute Lastbalancierung, effizientes Logging, Locking und Caching sowie einfache Migration von Anwendungen auf die Clustersysteme hervorheben, ist MySQL Cluster vor allem durch geringe Ansprüche bezüglich der verwendeten Hardware und unkomplizierte Fehlerbehandlung aufgrund der Shared-Nothing-Architektur gekennzeichnet.

NoSQL Data Stores stellen vermehrt eine Alternative zu RDBMS dar, die Systeme sind jedoch meist auf den Einsatz in wenigen Anwendungsgebieten limitiert. Zudem mangelt es ihnen an Standardisierung und vor allem die Low-Level-Abfragesprachen sind aus Sicht der Datenbankforschung als Rückschritt zu werten.

Durch die Verknüpfung bewährter Konzepte und Implementierungen der RDBMS mit Ansätzen der NoSQL-Bewegung können Vorteile beider Welten vereint werden. Die Erweiterung eines NoSQL-Systems führt nicht nur zu zusätzlichen Funktionalitäten, sondern steigert zudem die Bekanntheit und eröffnet neue Einsatzbereiche. Eine weitere Möglichkeit stellt eine Kombination von RDBMS- und NoSQL-Implementierungen in Form eines hybriden Systems dar, wovon bereits wenige Beispiele existieren. Als Mittel der Wahl zur Vereinigung von Konzepten relationaler Datenbanksysteme und NoSQL-Systeme zeichnen sich jedoch aus Sicht des Autors flexible RDBMS-Implementierungen ab, die sich gezielter als in aktuellen Systemen an verschiedene Einsatzzwecke anpassen lassen. Als mögliche Ansatzpunkte wurden die Implementierung verschiedener Storage-Engines und weiterer Transaktionskonzepte vorgeschlagen.

7. LITERATUR

[1] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin. HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. In *VLDB '09*, pages 922–933. VLDB Endowment, 2009.

[2] D. J. DeWitt and M. Stonebraker. MapReduce: A major step backwards, 2008.

[3] D. Florescu and D. Kossmann. Rethinking cost and performance of database systems. *SIGMOD Rec.*, 38:43–48, June 2009.

[4] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33:51–59, June 2002.

[5] T. Grebe. Gruppendynamik – Oracle Real Application Cluster vs. MySQL Cluster. *databasepro*, (6):46–63, 2010.

[6] IBM. Transparent Application Scaling with IBM DB2 pureScale. Technical report, IBM, 2009.

[7] F. Irmert, M. Daum, and K. Meyer-Wegener. A new approach to modular database systems. In *EDBT Workshop der SETMDM '08*, pages 40–44, New York, NY, USA, 2008. ACM.

[8] F. Irmert, C. P. Neumann, M. Daum, N. Pollner, and K. Meyer-Wegener. Technische Grundlagen für eine laufzeitadaptierbare Transaktionsverwaltung. In *BTW '09*, pages 227–236, Münster, Germany, 2009.

[9] A. Maslo. Unendliche Weiten – IBM DB2 pureScale für Power Systems. *databasepro*, (1):82–86, 2010.

[10] MySQL. Hochverfügbarkeitslösungen von MySQL – Ein Überblick über die Hochverfügbarkeitslösungen von MySQL. Technical report, MySQL AB, 2007.

[11] Oracle. MySQL Cluster 7.0 & 7.1: Architektur und neue Funktionen. Technical report, Oracle, Inc., 2010.

[12] Oracle. Oracle Real Application Clusters Administration and Deployment Guide, 11g Release 2. Technical report, Oracle Corporation, 2010.

[13] M. Stonebraker. The NoSQL Discussion has nothing to do with SQL. Blog-Eintrag, 2010.

[14] M. Stonebraker, C. Bear, U. Çetintemel, M. Cherniack, T. Ge, N. Hachem, S. Harizopoulos, J. Lifter, J. Rogers, and S. Zdonik. One size fits all - Part 2: benchmarking results. In *In CIDR*, 2007.

[15] M. Stonebraker and R. Cattell. Ten Rules for Scalable Performance in „Simple Operation" Datastores. *Communications of the ACM*, 2010.

[16] M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland. The end of an architectural era (it's time for a complete rewrite). In *VLDB '07*, pages 1150–1160. VLDB Endowment, 2007.

[17] A. Thusoo. Hive - A Petabyte Scale Data Warehouse using Hadoop. Technical report, Facebook Inc., 2009.

Ad-hoc Datentransformationen für Analytische Informationssysteme

Christian Lüpkes
OFFIS - Institut für Informatik
Escherweg 2
26121 Oldenburg, Deutschland
christian.luepkes@offis.de

ABSTRACT

Beim Betrieb von Data Warehouse Systemen kann es zu einem Semantic Shift kommen. Dieser bezeichnet eine Veränderung der Bedeutung von Dimensionselementen und kann bei Nichtbeachtung zu Informationsverlust und fachlich inkorrekten Analyseergebnissen führen. In dieser Arbeit wird ein graph-basierter Ansatz vorgeschlagen, welcher die Änderungen zwischen Dimensionen als Überleitungen verwalten und für Analysen zur Verfügung stellen kann. Dadurch wird es möglich, Anfragen in Analytischen Informationssystemen unter Berücksichtigung eventueller Semantic Shifts zu beantworten. Dieser Ansatz verzichtet dabei auf eine kennzahlbasierte Approximation und nutzt die Überleitungen klassischer Adaptionsverfahren. Der eingeführte Ansatz wird kritisch hinsichtlich bestehender Ansätze diskutiert und exemplarisch in verschiedenen Domänen durchgeführt.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Administration — *Data warehouse and repository*; H.2.8 [**Database Management**]: Applications; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Design, Data Analysis

Keywords

Data warehouse, Schema Versioning, OLAP, Temporal Data Warehouse

1. EINLEITUNG

Die am meisten verwendete Architektur für Analytische Informationssysteme ist die des Data Warehouses mit Metadaten, welche die gespeicherten Daten beschreiben und einer auf diesen Metadaten aufbauender Auswertungssoftware. Die Metadaten werden dabei in streng hierarchisch organisierten Taxonomien, sogenannten Dimensionen, gespeichert. Dimensionen beschreiben, wie die Daten analysiert werden können.

Das Data Warehouse ist dabei nach der Definition von Inmon eine themenorientierte, integrierte, stabile Sammlung zeitbezogener Daten, welche als Datenbasis zur Analyse dient [9]. Data Warehouses haben also immer einen Zeitbezug, bieten aber keine hochentwickelten Konzepte, um mit Änderungen in den Metadaten über die Zeit umzugehen. Klassisch wird davon ausgegangen, dass die Metadaten über die Zeit weitgehend stabil sind [7] [11]. Falls die Metadaten in Einzelfällen doch angepasst werden müssen, werden die gespeicherten Daten einfach den neuen Metadaten entsprechend umcodiert, die sogenannte *Instanzadaption* [2] [11]. Der Nachteil dieses Ansatzes ist, dass beim Umcodieren üblicherweise ein Informationsverlust entsteht. Zudem wird durch die Änderung der Metadaten und die Instanzadaption eine Wiederholung früherer Anfragen unmöglich. Außerdem besteht bei klassischen Systemen keine Möglichkeit die spezifischen Informationen der Metadatenänderungen zu speichern, da die Metadaten selbst nicht zeitbezogen gespeichert werden [12].

2. PROBLEMBESCHREIBUNG

Um das identifizierte Problem des *Semantic Shift* bei Datenanalysen zu verdeutlichen, soll an dieser Stelle zunächst ein Beispiel aus der Arbeit des Autors im deutschen Gesundheitswesen gegeben werden. Dort werden alle Diagnosen nach der *ICD*-Klassifikation, der ***I**nternational Statistical **C**lassification of **D**iseases and Related Health Problems*, codiert. Die Klassifikation selbst beinhaltet sowohl beschreibende als auch ordnende Metadaten und wird als Dimension zur Datenanalyse verwendet. Die deutsche Modifikation der *WHO*-ICD, ICD-GM (***G**erman **M**odifikation*), wird dabei jedes Jahr durch eine Expertengruppe des DIMDI, ***D**eutsches **I**nstitut für **M**edizinische **D**okumentation und **I**nformation* aktualisiert [3] [4] [5].

Die Aktualisierungen bestehen darin, dass neu identifizierte Erkrankungen einen Code zugewiesen bekommen, Erkrankungen zusammengefasst werden oder einzelne Krankheitsbereiche neu unterteilt werden. So wurde zum Beispiel im Jahr 2006 der Code *J09* für die neu identifizierte Vogelgrippe eingeführt.

Um die Daten zwischen den Jahren transformieren zu können, stellt das DIMDI zusätzlich sogenannte Überleitungen in einem Datenbankformat zur Verfügung. In den Abbildungen 1 und 2 sind diese exemplarisch in Ausschnitten für die Jahre 2005 bis 2007 gezeigt. Der Buchstabe A zeigt dabei an, ob eine Überführung automatisch in der durch die Spal-

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

icd_code2005	icd_code2006	auto2005_2006	auto2006_2005
J10.0	J10.0	A	A
J10.0	J09		A
J10.1	J10.1	A	A
J10.8	J10.8	A	A

Abbildung 1: Ausschnitt der offiziellen ICD Überleitungen zwischen den Jahren 2005 und 2006

tenüberschrift festgelegten Richtung möglich ist.

icd_code2006	icd_code2007	auto2006_2007	auto2007_2006
J09	J09	A	A
J10.0	J10.0	A	A
J10.1	J10.1	A	A
J10.8	J10.8	A	A

Abbildung 2: Ausschnitt der offiziellen ICD Überleitungen zwischen den Jahren 2006 und 2007

Im Jahr 2005 auf 2006 ist es zum Beispiel möglich, den Code *J10.0* zwischen den Jahren umzucodieren. Allerdings gibt es noch einen zweiten Eintrag von *J10.0* auf *J09* bei dem nur eine Umcodierung von 2006 auf 2005 zugelassen ist.

In Analytischen Informationssystemen werden die Daten meist nach der aktuellsten ICD-Definition gespeichert. Daten aus dem Jahr 2005 würden also im Jahr 2006 und 2007 umcodiert. Entsprechend den Überleitungsregeln aus den Abbildungen 1 und 2 würde der Wert *J10.0* syntaktisch gleich bleiben und nicht umcodiert werden.

Eine typische Anfrage wäre nun der Art *„Zeig mir die jährliche Summe aller behandelten J10.0 Patienten der Jahre 2005, 2006 und 2007“* welche das in Abbildung 3 gezeigte Ergebnis liefert. Im Jahr 2006 gab es also im Vergleich zum Vorjahr eine Abnahme von *J10.0* Patienten die sich im Jahr 2007 noch deutlicher fortsetzt.

Summe der behandelten J10.0 Patienten nach Jahr

2005	2006	2007
18347	17913	17548

Abbildung 3: Summe aller behandelten J10.0 Patienten der Jahre 2005, 2006 und 2007

2.1 Darstellung als Graph

In Abbildung 4 ist exemplarisch ein Ausschnitt der ICD-Klassifikation für nachgewiesene sonstige Influenzaviren der Jahre 2005 bis 2007 abgebildet. Der Graph repräsentiert dabei die offizielle Taxonomie der ICD Codes und die gerichteten Kanten repräsentieren die offiziell als gültig definierten Transformationsregeln für Umcodierugnen der drei abgebildeten Jahre 2005, 2006 und 2007, wie sie in den Tabellen der Abbildungen 1 und 2 definiert sind.

In einem Data Warehouse werden die zu analysierenden Daten in der Regel auf der feinsten verfügbaren Klassifikationsstufe vorgehalten. Veranschaulicht handelt es sich also um die Ausprägungen der Blätter. Falls eine Analyse der Erkrankungen *J10.1* oder *J10.8* durchgeführt werden soll,

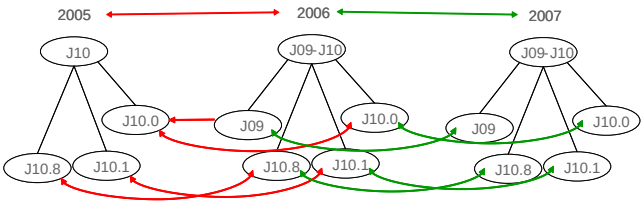


Abbildung 4: Darstellung dreier Teilgraphen der ICD-GM Metadaten für Influenzaviren und deren Überleitung über die Jahre 2005 bis 2007

ist dies unproblematisch, da es keinerlei Änderungen in der Datenbeschreibung gab; zu erkennen an der Existenz der bijektiven Kanten zu den gleichen Knoten jedes Jahres. Das Problem des *Semantic Shift* tritt auf, wenn *J10.0* für die Jahre 2005 bis 2007 untersucht werden soll. Es ist nicht eindeutig, welche Bedeutung von *J10.0* verwendet werden soll. Falls die Bedeutung des Codes *J10.0* vom Jahr 2005 beabsichtigt ist, sollte für die Folgejahre auch der Code *J09* berücksichtigt werden. Falls die Semantik von 2006 oder 2007 gemeint ist, muss dem menschlichen Analysten bewusst sein, dass es eine inhaltliche Änderung vom Jahr 2005 zu 2006 für *J10.0* gab auch wenn die Daten syntaktisch identisch sind und Transitionen in beide Richtungen existieren.

Das analysespezifische Hintergrundwissen des Fachexperten ist, dass *J10.0* ein Sammelknoten für nicht genauer bestimmte Influenzaviren ist. Wie bereits oben erwähnt, wurde in 2006 die Vogelgrippe identifiziert und als neuer Code *J09* eingefügt. Dadurch wurde die Bedeutung von *J10.0* als alle unbestimmten Influenzaviren zwar nicht verändert, aber verglichen mit 2005 *fehlen* nun die Vogelgrippefälle. Für statistische Analysen auf solch einer feingranularen Ebene würden daher fehlerhafte Ergebnisse geliefert. Bei einer Analyse auf den Elternknoten *J10* des Jahres 2005 oder *J09-J10* der Jahre 2006 und 2007 wären die Resultate korrekt, da alle Transformationskanten auf Kindknoten verweisen.

Für die Ergebnisse in Abbildung 3 bedeutet dies, dass die Abnahme der *J10.0* Erkrankungen auch darin begründet liegt, dass Krankheitsfälle in *J09* codiert wurden, die vorher in *J10.0* enthalten waren.

2.2 Weitere Domänen

Der *Semantic Shift* kann nicht nur in der medizinischen Dokumentation beobachtet werden, sondern auch in anderen Bereichen. So kann man z.B. für die Entwicklung der Länder Europas von 1988 bis 2000 bedingt durch den Zusammenbruch des Warschauer Pakts ähnliches feststellen. Allerdings muss dort beachtet werden, dass es sich bei den abgeleiteten und angepassten Metadaten nicht um gesetzlich vorgegebene Dimensionsstrukturen handelt, sondern um von Fachexperten erstellte Dimensionen. Dies ist der Normalfall bei Data Warehouses. Die Dimension soll alle Länder im Herzen Europas widerspiegeln. Bis zum Jahr 1991 gab es die beiden eigenständigen deutschen Staaten *BRD* und *DDR*. In der Dimension wären diese dann als Blätter verfügbar. Mit der Wiedervereinigung wird das Blatt *DDR* gelöscht und die dazugehörigen Daten der *BRD* zugeordnet. Der Begriff *BRD* ist also syntaktisch gleich geblieben, beschreibt nun aber einen deutlich größeren Bereich.

Würde man die Daten der BRD betrachten, so könnte man z.B. in 1991 eine deutliche Steigerung der Einwohner-

zahl feststellen. Dies wäre aber nicht durch hohe Geburtsraten begründet, sondern durch die größere betrachtete Fläche infolge der Wiedervereinigung mit der DDR.

Die umgekehrte Richtung kann man bei der Tschechoslowakei beobachten. Bis 1990 war es die *ČSSR*, dann wurde das gleiche Land umbenannt in *ČSFR* und im Jahr 1992 aufgeteilt in die zwei Staaten Slowakei *SR* und Tschechien *ČR*. Für den letzt genannten Fall würde in der Dimension ein Blatt gelöscht und dafür zwei neue Blätter eingefügt. Die dazugehörige Transformationsregel wäre, dass es keine Möglichkeit gibt, *ČSFR* auf *SR* und *ČR* abzubilden, wohl aber in der Gegenrichtung.

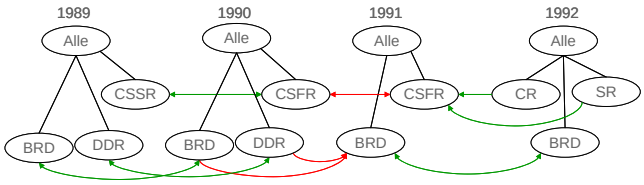


Abbildung 5: Darstellung von vier Ausschnitten einer Länderdimension für die Jahre 1989 bis 1992 und deren Überleitungen

Die graph-basierte Visualisierung der beschriebenen Dimensionsentwicklung ist in Abbildung 5 zusehen. Dazu muss gesagt werden, dass der Aufbau und die Entwicklung der Dimension von Fachexperten für Analysezwecke durchgeführt wurde. Die Dimensionen und Transformationen hätten auch auf andere Arten modelliert werden können.

3. EXISTIERENDE LÖSUNGANSÄTZE

Der erste Lösungsansatz für das Problem der sich ändernden Dimensionen wurde 1993 von Kimball postuliert [11]. Die Lösung besteht in der Umcodierung der Daten nach der jeweilig neuesten Dimensionsbeschreibung. Dies kann dabei in drei verschiedenen Arten geschehen. Der *Type 1* Ansatz überschreibt die alten Werte mit den neuen, umcodierten Werten. Die *Type 2* und *Type 3* Ansatz behält die alten Werte zusätzlich bei. Auf diese Weise können alte Werte in die neue Dimension transformiert, bzw. eingebunden werden. Der Nachteil aller dieser Ansätze ist aber, dass sie nicht in der Lage sind mit dem *Semantic Shift* syntaktisch gleicher Ausprägungen umzugehen. Es gibt also keine Unterstützung für Datenanalysen, die über verschiedene Versionen der Dimensionen hinausgehen, wenn sich die Bedeutung der Daten geändert hat.

Das Problem der Anfragen über mehrere Dimensionsversionen wurde 2006 in [8] als graphentheoretisches Problem diskutiert. Dabei wurden die Metadaten als sogenannte *Schemagraphen* repräsentiert. Für die Graphen wurden erlaubte Modifikationen definiert, welche die potentiellen Änderungen der Dimensionen wiedergeben. Wird eine Dimension durch eine Modifikation geändert, wird dies als neue Version in einem Graphen gespeichert. Basierend auf einer Graphenalgebra ist es dadurch möglich, Anfragen über verschiedene Dimensionsversionen hinweg zu stellen. Diesem Ansatz fehlt zum einen der Umgang mit dem *Semantic Shift* der Daten. Zum anderen erscheint er nicht praxisgerecht, da für historische Daten neu hinzugekommene Angaben nachträglich eingepflegt werden müssen, um Vergleiche über verschiedene Versionen zu ermöglichen.

Die am weitest gehende Lösung für das präsentierte Problem wurde 2002 in [12] veröffentlicht. Ein formales Temporal-Modell für die Beschreibung von Änderungen in den Dimensionen wurde dazu eingeführt [7]. Es wurden entsprechende Transformationsfunktionen definiert, welche die erlaubten Datenänderungen beschreiben. Der Ansatz ermöglicht dabei Anfragen über verschiedene Versionen der Dimensionen hinweg, indem die Daten zur Anfragezeit adaptiert werden. Der Nachteil des Ansatzes liegt in der Realisierung der Instanzadaption durch die Verwendung von Matrizenmultiplikation. Jeder Wert einer Dimensionsversion muss von Fachexperten mit einem Koeffizienten versehen werden, der aussagt wie ähnlich der Wert dem Nachfolger in der verbundenen Dimensionsversion ist. Dies erlaubt eine Abschätzung, um den *Semantic Shift* zu lösen. Jedoch hat dies zwei Nachteile. Zum einen muss der Koeffizient für jede Verwendung der Dimension in einer Kennzahl individuell angegeben werden, da sich die Koeffizienten für z.B. Erkrankungs- und Sterberisiko unterschiedlich verhalten und deshalb die Koeffizienten nicht für alle Analysen gleich sind. Zum anderen wird das in den Transformationsdaten inhärente Wissen nicht dazu genutzt, genaue anstatt approximierten Ergebnissen zu liefern.

4. DER GRAPH-BASIERTE ANSATZ

Wie in der Problembeschreibung ausgeführt und in den Abbildungen 4 und 5 veranschaulicht, lassen sich Dimensionen als streng hierarchische Bäume mit einem Wurzelknoten darstellen. Die Blätter repräsentieren dabei in der Regel die im Data Warehouse speicherbaren Werte. Falls Analysen auf den Elternknoten durchgeführt werden sollen, werden diese standardmäßig aus den Kindelementen berechnet [2].

Bei den Dimensionen handelt es sich um von Fachexperten modellierte Metadaten, die nur zu bestimmten Zeitpunkten geändert werden. Deshalb ist es möglich, die Änderungen einer Dimension zusammen mit einer Versionsnummer zu speichern. Dieser Ansatz zur Beschreibung der zeitlichen Entwicklung wurde auch von [12] und [8] verfolgt. Anders aber als bei [11] soll keine Instanzadaption mit Informationsverlust vorgenommen werden, sondern die Transformationsregeln als gerichtete Kanten zwischen den Blättern zweier Dimensionsversionen gespeichert werden. Es wird verlangt, dass jede neue Version einer Dimension Transformationsregeln zu mindestens einem Vorgänger definiert. Dies ist keine Einschränkung, da es beim Fehlen von Transformationsregeln nicht um einen Nachfolger der Dimension sondern um eine vollständig neue, andere Dimension handelt.

Bei einer Anfrage an das Analytische Informationssystem soll ein Interpreter zwischen den Anwender und das Auswertungssystem geschaltet werden. Dieser Interpreter wertet die Transformationsregeln aus und stellt fest, ob in dem angefragten Zeitraum für die auszuwertenden Daten eine Änderung stattgefunden hat. Wenn dies nicht der Fall ist, wird die Anfrage ohne Nutzerinteraktion und ohne Änderungen durchgeführt. Falls jedoch zwei oder mehrere Dimensionsversionen von der Anfrage betroffen sind, wird der Interpreter mittels der ein- und ausgehenden Kanten der Knoten prüfen, ob auf zusätzliche Knoten über die Kanten zugegriffen werden kann. Wenn die Knoten für den gewünschten Zeitraum stabil sind, wird dem Nutzer die Veränderung der Dimension für seinen angefragten Ausschnitt als sogenannter Evolutionspfad angezeigt.

Da es nicht beabsichtigt ist, die Definition der Transfor-

mationsregeln auf genau einen Vorgänger und Nachfolger zu beschränken, kann es durchaus mehrere unterschiedliche Evolutionspfade geben, die zu unterschiedlichen Mengen von Knoten führen. Deswegen sollen die gefunden Evolutionspfade dem anfragenden Nutzer angezeigt werden, der dann den für seine Anfrage geeignetsten auswählen kann. Dabei ist festzustellen, dass die Bedeutung der Evolutionspfade immer der modellierten Realwelt einer Dimensionsversion entspricht. Dies führt dazu, dass die Daten dann ad-hoc zum Anfragezeitpunkt unter die ausgewählte Dimensionsversion transformiert werden. Die Datentransformation ist allerdings keine Instanzadaption, sondern eine Transformation eines Wertes auf eine Menge von Werten.

4.1 Beispiel der Lösungsidee

Falls der Nutzer eine Anfrage der Art *„Gib mir die Summe aller behandelten J10.0 Patienten der Jahre 2005 bis 2007“* stellt, wird der Interpreter die Werte *J10.0* und die ICD-GM Dimensionsversionen 2005, 2006 und 2007 identifizieren. Den Transformationsregeln in Abbildung 4 folgend, wird der Interpreter zwei verschiedene Arten von J10.0 feststellen: Die Version 2005 hat zwei eingehende Kanten aus der Version 2006, einmal vom *J10.0* als auch vom *J09* Knoten. Der Interpreter kann also feststellen, dass der Knoten *J10.0* Version 2005 geteilt wurde. Nun prüft der Interpreter die identifizierten Knoten des Jahres 2006 und findet zusätzlich nur bidirektionale Kanten zu den Knoten des Jahre 2007, was bedeutet, dass keine Änderung stattgefunden hat.

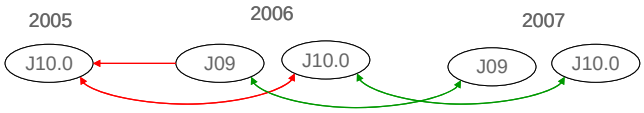


Abbildung 6: Lösungsvorschlag mit Erweiterung der Anfragemenge, Konzept *J10.0* Version 2005

Abbildung 7: Lösungsvorschlag mit Beschränkung des Anfragebereichs, Konzept *J10.0* Version 2006 und 2007

Dem Nutzer wird als Zwischenergebnis seiner Anfrage mitgeteilt, dass zwei verschiedene Interpretationen von *J10.0* für den Zeitraum 2005 bis 2007 identifiziert wurden. Es werden dann diese zwei verschiedenen Evolutionspfade zur Auswahl angeboten: In Abbildung 6 wird die Erweiterung der Anfrage für die Jahre 2006 und 2007 um den Wert *J09* vorgeschlagen, damit die Anfrage der Bedeutung von *J10.0* im Jahr 2005 entspricht. Die zweite angebotene Lösung ist in Abbildung 7 zu sehen. Hier wird die Bedeutung von *J10.0* der Jahre 2006 und 2007 vorgeschlagen.

Werden Anfragen auf höheren Ebenen der Dimension, wie z.B. die *Gib mir die Summer aller behandelten Fälle von „Grippen durch nachgewiesene Influenzaviren“* welche dem ICD Gruppencode *J10* des Jahres 2005 bzw. *J09-J10* der Jahre 2006 und 2007 entspricht, ist dies auch ohne weiteres möglich. Für alle Kindelemente von *J10* werden die Evolutionspfade ausgewertet. Da alle Kindelemente in den Dimensionsversionen unter einem Elternelement sind, wird davon ausgegangen, dass die Elternelemente gleich sind. Die Anfrage wird dann direkt ausgeführt. Sollte es in irgendeinem gültigen Evolutionspfad eines beliebigen Kindes mehrere Elternknoten geben, würden dem Anwender wieder die verschiedenen Optionen wie bei den Einzelelementen angezeigt. Der Anwender wird also immer über Bedeutungsänderungen und Systembrüche automatisch graphisch informiert und kann die für seine Zwecke geeignete Anfrage auswählen.

4.2 Vorteile des Ansatzes

Es wird erwartet und angestrebt, dass der vorgestellte Ansatz die folgenden Vorteile bietet:

- Durch den graphenbasierten Ansatz, der auf eine kennzahlabhängige Approximation verzichtet, ist es möglich die Überführungsregeln für alle Analyseanfragen zu verwenden, welche die Dimension beinhaltet. Dies ist eine deutliche Erweiterung gegenüber [12].
- Da die Überfühungsregeln auch in klassischen Adaptionsverfahren wie [11] benötigt werden, ist kein zusätzlicher Arbeitsaufwand der Fachexperten notwendig, um die Kanten bereit zustellen.
- Durch den graphenbasierten Ansatz ist es bei mehreren Überleitungsregeln pro Dimensionsversion möglich, gröbere Versionen einer Dimension zu überspringen. Gröber meint dabei, dass Fehlen einzelner Knoten, die in späteren Versionen wieder eingefügt wurden. Bei einer Umcodierung des Datenbestandes wäre dies ein irreversiebler Informationsverlust.
- Der Import und die Haltung der Daten wird vereinfacht, da die Daten in ihrer originären Version gespeichert werden können. Die Daten müssen nicht in eine einzige Version umcodiert werden.
- Da der Nutzer zwischen verschiedenen inhaltlichen Interpretationen eines Wertes wählen kann, ist das Anfragesystem mächtiger als klassische Systeme. Zudem erlaubt dies die Wiederholung historischer Analysen, da die Datenbasis nicht umcodiert und die Dimensionsdaten genauso erhalten bleiben.

4.3 Zu untersuchende Fragestellungen

Um sicherzustellen, dass ein Data Warehouse zusammen mit einer OLAP Analyse Anwendung die vorgestellten Funktionen und insbesondere Vorteile erfüllen kann, muss untersucht werden, welche Konsistenzbedingungen die Überleitungsregeln als auch die Metadaten einhalten müssen. Zudem sind die Anforderungen an die Datenrepräsentation und Speicherung der Transformationsregeln und zusätzlichen Versionsinformationen in den Metadaten als auch der Datenhaltung zu untersuchen. Ein weiterer Bereich ist, wie sich die Methoden auf verschieden Datenarten (Integer, Boolean, Nominal) als auch verschiedene Analyse Operationen (Sum, Max, Min, Average) anwenden lassen. Da in Analysen auch oft mehrere verschiedene Dimensionen genutzt werden, muss als letzter wichtiger Punkt noch die Anwendbarkeit auf mehrere Dimensionen durchdacht werden.

4.4 Evaluation

Um den Ansatz mit seinen Konzepten und festgelegten Anforderungen zu evaluieren, wird ein Prototyp auf Basis

von *MUSTANG - Multidimensional Statistical Data Analysis Engine* [1] [13] umgesetzt werden. Dies ist ein kommerzielles Daten Analyse Tool, welches insbesondere für Analysen im Gesundheitswesen, z.B. Krebsregistern, eingesetzt wird.

Da das vorgestellte Thema durch zwei Projekte mit Klinikdaten motiviert wurde, bei denen sich der *Semantic Shift* als problematisch erwiesen hatte, soll das Konzept in diesen evaluiert werden. Dabei handelt es sich zum einen um Daten deutscher Kliniken der Jahre 2006 bis 2010. Hier sollen Fragen der Versorgungsforschung auf einer feingranularen Ebene ausgewertet werden, was bisher nicht möglich war. Zum anderen geht es in einem Forschungsprojekt der EU darum, für spezielle Herzschrittmacherpatienten statistisch valide Muster zu identifizieren, die in historischen Patientendaten früherer Fälle enthalten sind. Die Patientendaten stammen dabei aus den Jahren 2006 bis 2011 eines österreichischen Universitätsklinikums, in dem später die Anwendung erfolgt. Hier liegt der Fokus darauf, alte Codierungen akkurat unter die aktuellste Version zu subsumieren, damit die Muster auf aktuelle Fälle angewendet werden können.

5. ZUSAMMENFASSUNG

Dieses Paper stellt einen Ansatz vor, der akkurate Datenanalysen in einem Analytischen Informationssystem über sich ändernde Datengrundlagen ermöglicht. Die Datenänderungen können dabei sowohl syntaktischer als auch semantischer Natur sein. Änderungen der Daten werden dabei als verbindende Kanten zwischen verschiedenen Versionen einer Dimension modelliert und diese Dimensionen dabei als Graphenstruktur aufgefasst. Durch die Interpretation der Verbindungen zum Zeitpunkt einer Analyseanfrage, werden die möglichen Evolutionspfade identifiziert. Die Evolutionspfade repräsentieren dabei domänspezifisches Hintergrundwissen, wie z.B. die Bedeutungsänderung von Werten, den *Semantic Shift*. Der Nutzer kann dieses Hintergrundwissen visuell erfassen und sich für einen geeigneten Evolutionspfad entscheiden. Die Analyseanfrage wird dann zur Anfragezeit so umgewandelt, dass die Daten ad-hoc unter die gewählte Bedeutung des Evolutionspfads transformiert werden. Da die Evolutionspfade so berechnet werden, dass Sie inhaltlich identische und vergleichbare Mengen repräsentieren, sind die Anfrageergebnisse akkurat. Dies wird dadurch ermöglicht, dass die Daten in ihrem Originalformat gespeichert und die Transformationsregeln nur gespeichert aber nicht direkt auf die Daten angewendet werden. Mit dem vorgestellten Modell und den dazugehörenden Methoden sind keine verlustbehafteten Datentransformationen oder Abschätzungen notwendig.

6. REFERENCES

[1] Appelrath, H.-J., Rohde, M. , Thoben, W. , OFFIS e.V., MUSTANG - Multidimensional Statistical Data Analysis Engine: `http://www.offis.de/en/offis_in_portrait/structure/projects/detail/status/mustang.html`, (2011)

[2] Bauer, A., Günzel, H.: Data Warehouse Systeme. dpunkt.verlag, 3. überarbeitete und aktualisierte Auflage, (2009)

[3] DIMDI - Deutsches Institut für Medizinische Dokumentation und Information: ICD-10-GM 2005 Systematisches Verzeichnis. Systematisches Verzeichnis zur Internationalen statistischen Klassifikation der Krankheiten und verwandter Gesundheitsprobleme - German Modification. Deutsche Krankenhaus Verlags-Gesellschaft, (2004)

[4] DIMDI - Deutsches Institut für Medizinische Dokumentation und Information: ICD-10-GM Version 2006. Systematisches Verzeichnis. Deutsche Krankenhaus Verlags-Gesellschaft, (2005).

[5] DIMDI - Deutsches Institut für Medizinische Dokumentation und Information: ICD-10-GM Version 2007. Band I: Systematisches Verzeichnis. Deutsche Krankenhaus Verlags-Gesellschaft, (2006)

[6] DIMDI - Deutsches Institut für Medizinische Dokumentation und Information: ICD-10-GM Version 2011: Band I: Systematisches Verzeichnis. Deutsche Krankenhaus Verlags-Gesellschaft, (2010)

[7] Eder, J, Koncilia, C., Morzy, T.,: The ℂOMET Metamodel for Temporal Data Warehouses. In Proc. of the 14th Int. Conference on Advanced Information Systems Engineering (CAISE02), pp. 83–99.Springer Verlag (LNCS) (2002)

[8] Golfarelli, M., Lechtenbörger, J., Rizzi, S., Vossen, G.: Schema versioning in data warehouses: enabling cross-version querying via schema augmentation. In DataKnowl. Eng., 59(2):435–459, 2006. Elsevier Science Publishers B. V., Amsterdam, (2006)

[9] Inmon, W. H.: Building the data warehouse (2nd ed.). John Wiley & Sons, Inc.,New York, NY, USA, (1996)

[10] Inmon, W. H., Strauss, D., Neushloss, G.: DW 2.0: The Architecture for the Next Generation of Data Warehousing. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, (2008)

[11] Kimball, R.: Slowly Changing Dimensions. In DBMS online, `http://www.dbmsmag.com/9604d05.html` (1996)

[12] Koncilia; C. A.: The ℂOMET Temporal Data Warehouse. PhD thesis, Universität Klagenfurt (2002)

[13] Teiken, Y., Rohde, M., Mertens, M.: Mustang - Realisierung eines analytischen informationssystems im kontext der gesundheitsberichtserstattung. In K.-P. Fähnrich and B. Franczyk, editors, GI Jahrestagung (1), volume 175 of LNI, pages 253–258. GI, (2010)

APPENDIX

A. ACKNOWLEDGMENTS

The research leading to these results has received in part funding from the European Community's Seventh Framework Programme (FP7/ 2007-2013) under grant agreement no. ICT-248240, iCARDEA project.

Wissensbasiertes Business Intelligence für die Informations-Selbstversorgung von Entscheidungsträgern

Matthias Mertens
OFFIS - Institut für Informatik
Escherweg 2
26121 Oldenburg, Germany
mertens@offis.de

ABSTRACT

Im Bereich der Business Intelligence haben sich Analytische Informationssysteme (AIS) mit dem Ziel entwickelt, verschiedene Datenquellen integriert analysieren zu können und Informationen zu gewinnen die Business User, in ihrem Entscheidungsfindungsprozess unterstützen. Sowohl die hohe Komplexität, die sich aus der Flexibilität und Mächtigkeit solcher Systeme ergibt, als auch die hohe notwendige Interaktion mit dem Nutzer zur Durchführung adäquater Analysen, bedingen entsprechendes Analyse- und Domänenwissen sowie ein tiefergehendes konzeptionelles und technisches Verständnis. Dieses ist meistens bei Business Usern ohne entsprechende Schulung nicht gegeben, wodurch eine eigenständige Informationsversorgung mittels AIS behindert wird. Erschwerend kommt hinzu, dass AIS in der Regel keine zusätzlichen Metadaten zu Business Regeln, Strategien oder Hintergrundinformationen erfassen, verwalten und für die Analyseunterstützung bereitstellen können.
Idealerweise sollten Business User auf Basis einer Analyseunterstützung des AIS dazu befähigt werden, adäquate Analysen durchzuführen, ohne zwingend über Analyse- und Domänenwissen verfügen zu müssen. Diese analyseunterstützenden Funktionalitäten können von weiterführenden Informationen, über eine Navigationsunterstützung für Analysepfade bis hin zu einer Vorschlagsgenerierung von Analyseschritten reichen.
In diesem Paper werden Konzepte eines Analyseprozesses und darauf aufbauend analyseunterstützende Funktionen vorgestellt, die eine Informations-Selbstversorgung des Business Users erlauben. Der Fokus wird hierbei auf die Erweiterung eines AIS um semantische Metadaten gelegt, um eine Erfassung, Verwaltung und Nutzung von Analyse- und Domänenwissen zu ermöglichen.

Categories and Subject Descriptors

J.1 [**Administrative data processing**]: Business; H.4.2 [**Information Systems Applications**]: Types of Systems—*Decision support*

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

Keywords

Semantic Metadata, Data Warehouse, Analytical Information System, Decision support, Business Intelligence

1. EINLEITUNG

Die Unterstützung des Managements und die Verbesserung des Entscheidungsfindungsprozesses werden als Schlüsseleigenschaften der Business Intelligence (BI) gesehen [6]. Entscheidungsträger, sogenannte Business User, sollen befähigt werden, alle benötigten Informationen zur richtigen Zeit zu erhalten. In der BI wurden analytische Informationssysteme (AIS) entwickelt, die es Business Usern erlauben große Datenmengen zu visualisieren, zu handhaben und zu analysieren. AIS bestehen aus einem Data Warehouse (DWH) und darauf aufbauenden Analysekomponenten. Während das DWH es ermöglicht verschiedene Datenquellen qualitätsgesichert zu integrieren und multidimensional aufzubereiten, erlauben die Analysekomponenten Online Analytical Processing (OLAP) Operatoren, komplexe statistische Verfahren sowie geografische Operatoren in verschiedenen Visualisierungen auf den integrierten Daten durchzuführen.

AIS haben jedoch auch verschiedene Mängel, die im nächsten Abschnitt 2 näher betrachtet werden. Aus diesen leitet sich die Forschungsfrage sowie die zugehörigen Anforderungen an einen Ansatz ab, welcher in Abschnitt 3 diskutiert wird. Im Anschluss wird der eigene Ansatz mit zugehörigen Konzepten und Funktionalitäten zur Analyseunterstützung in Abschnitt 4 mit der Anwendungsdomäne „Krankenhausmarktanalyse“ (KMA) präsentiert, bevor im Abschnitt 5 verwandte Arbeiten in diesem Forschungsumfeld aufgezeigt werden. Abschließend erfolgt in Abschnitt 6 eine Zusammenfassung sowie ein Ausblick des Forschungsvorhabens.

2. MÄNGEL ANALYTISCHER INFORMATIONSSYSTEME

In dem am OFFIS entwickelten AIS - Multidimensional Statistical Data Analysis Engine (MUSTANG) [9] - konnte beobachtet werden, dass sich aus der hohen Flexibilität und Mächtigkeit von AIS eine Komplexität ergibt, welche zu einer signifikanten Herausforderung für Business User werden kann, wenn diese eigenständig adäquate explorative Analysen durchführen möchten. Im Gegensatz zu Analysten verfügen Business User in der Regel über ein geringeres konzeptionelles Verständnis des multidimensionalen Datenmodells (MDM) sowie geringeres notwendiges Analyse- und

Domänenwissen. Hierbei enthält das Domänenwissen Informationen darüber, welche Fragestellungen einer zu untersuchenden Analyse zu Grunde liegen, mit welchen Kennzahlen Analysen zu spezifischen Fragestellungen möglich sind, welche explizite Semantik diese Kennzahlen haben und in welchen Beziehungen diese zueinander stehen. In Abgrenzung dazu umfasst Analysewissen Informationen zu den Analyseinstrumenten, d.h den Analyseoperationen, -verfahren und den möglichen Visualisierungen. Schließlich wird in den AIS für die Durchführung komplexer Analysen eine hohe Interaktion mit den Business Usern benötigt, wodurch diese ohne entsprechende Schulung schnell überfordert sind.

Ein weiterer Mangel von AIS wird in der geringen Berücksichtigung von Metadaten gesehen [2], die weiterführende Informationen über die quantitativen DWH Daten sowie die DWH Struktur (Kennzahlen und Dimensionselemente) bereitstellen. Zu diesen zählen Annahmen, Definitionen, Business Regeln, Terminologien und Hintergrundinformationen [10]. Daher müssen Business User sich die Semantiken von Daten und Strukturen durch Zuhilfenahme externer Quellen selbst erschließen [2].

Des weiteren sind AIS derzeit nicht in der Lage, Analyse- und Domänenwissen zu importieren, zu verwalten und für weiterführende Analyseunterstützung der Business User zu nutzen. Insbesondere ist die Expertise aus dem Bereich der Analysestrategien und den „Best Practices“ für Analysen in spezifischen Domänen von Interesse. Wissen, das durch Analysten in AIS eingebracht wird, geht in der Regel verloren [1].

Idealerweise sollten Business User befähigt werden, multidimensionale quantitative Daten zu analysieren, ohne zwingend über Analyse- und Domänenwissen sowie ein tiefgehendes konzeptionelles Verständnis verfügen zu müssen. Das AIS sollte den Business User in seinen explorativen ad-hoc Analysen unterstützen, indem es modelliertes semantisches, maschinenlesbares und -verständliches Wissen ausnutzt. Analyseunterstützende Funktionalitäten für eine Business User Self-Information Service [2, 12], könnten von weiterführenden Informationen zu DWH Entitäten, über eine Navigationsunterstützung für Analysepfade bis hin zu einer Vorschlagsgenerierung von weiteren sinnvollen Analyseschritten reichen.

In diesem Paper werden nun im Folgenden Konzepte für eine semantische Metadatenbasis sowie darauf aufbauende analyseunterstützende Funktionalitäten für ein AIS vorgestellt. Ziel ist es, dass Business User sich selbst mit Informationen auf eine effiziente und intuitive Art und Weise versorgen können. Exemplarisch werden Beispiele aus der Domäne Krankenhausmarktanalyse (KMA) gebracht, da hier Business User, z.B. Krankenhauscontroller durch die Veränderungen im deutschen Gesundheitswesen gezwungen werden, zur Sicherung der Wettbewerbsfähigkeit die Potentiale ihres Krankenhauses zielgerichtet zu erschließen und Leistungsangebote konkurrenzfähig auszurichten.

3. FORSCHUNGSFRAGE UND ANFORDERUNGEN

Motiviert aus den in Abschnitt 2 genannten Mängeln wird die folgende Fragestellung abgeleitet:

> Wie kann eine Komplexitätsreduktion von AIS mit dem Ziel erfolgen, ungelernte Business User zu befähigen, selbst adäquate explorative Analysen für eine intuitive und effiziente Informations-Selbstversorgung durchzuführen.

Um die Forschungsfrage zu beantworten, sieht der verfolgte Lösungsansatz die Verwendung von verschiedenen Analyseunterstützungsfunktionalitäten vor, welche explizites modelliertes semantisches Domänen- und Analysewissen ausnutzen. Dieses wird in ein semantisches Datenmodell des AIS importiert, dort verwaltet und durch verschiedene Services genutzt werden.

Folgenden Anforderungen werden an einen Ansatz gestellt:

- Der Ansatz soll es einem domänenunabhängigen AIS ermöglichen Business User im Kontext domänenspezifischer Analyseaufgaben zu unterstützen. Dafür wird ein Konzept bestehend aus einem generischen Datenmodell in Verbindung mit einem domänenunabhängigen AIS benötigt. Das Konzept wird für eine spezifische Domäne instanziiert.

- Der Ansatz soll den Lernprozess von Analyse- und Domänenwissen unterstützen und den initialen Einarbeitungs- und Trainingsaufwand ins AIS reduzieren. Business User sollen von der inhärenten Expertise des AIS lernen.

- Der Ansatz soll eine Komplexitätsreduktion eines AIS verfolgen, ohne jedoch die Flexibilität und die Mächtigkeit des AIS einzuschränken. Das AIS soll für Business User mit geringer Expertise intuitiv benutzbar sein, so dass diese adäquate explorative Analysen auf dem DWH durchführen können. Des Weiteren soll der Ansatz die Anzahl der Nutzer-Interaktionen für das Erreichen gleicher Analyseergebnisse reduzieren, wodurch die Effektivität des AIS gesteigert werden soll.

- Das Konzept soll es ermöglichen, mit verschiedenen Metadatenarten umzugehen und diese in einer intelligenten Art und Weise für eine Analyseunterstützung zu verknüpfen. Oftmals sind Ansätze in der Literatur zu finden [2, 7, 8, 11], die Hintergrundinformationen sowie Regeln über die DWH Struktur bereithalten. Ferner sollen aber Metadaten zum Analyseprozess berücksichtigt werden, die den Analyseprozess von einer Fragestellung über verschiedene Analyseschritte hin zu einem Analyseergebnis beschreiben und wesentliche Informationen für eine Analyseunterstützung bereithalten. Diese Metadaten werden auch Strategien oder „Best Practices“ genannt und beschreiben die Expertise bzw. das Analyse- und Domänenwissen eines Analysten. Schließlich soll das AIS ebenfalls Metadaten zu quantitativen Daten des DWH, sprich zu Analyseergebnissen wie z.B. Trends, spezifische Zusammenhänge, etc. verarbeiten.

Im nächsten Abschnitt 4 werden verschiedene Konzepte eines Datenmodells sowie die darauf anwendbaren Funktionalitäten zur Analyseunterstützung und zur Erfüllung der Anforderungen diskutiert.

4. WISSENSBASIERTE FUNKTIONEN UND KONZEPTE FÜR EINE ANALYSEUNTERSTÜTZUNG

Ziel des hier diskutierten Ansatzes ist es, Business User im Kontext domänenspezifischer Analyseaufgaben zu unterstützen. Hierzu sollen verschiedene unterstützende Funktionalitäten auf Basis eines modellierten semantischen Metadatenmodells bereitgestellt werden (s. Abschnitt 4.1). Als Wissensrepräsentationssprache zur Modellierung des Metadatenmodells, in Form von mehreren miteinander verknüpften Ontologien, kommt OWL-DL zum Einsatz. Eine DWH-Ontologie bildet das MDM des jeweils zu Grunde liegenden DWHs als Instanzen ab und ermöglicht es, in den darauf aufbauenden Ontologien das MDM zu referenzieren. Die Analyse-Ontologie modelliert abstrakt Entitäten und deren Beziehungen, die die Analyseprozesse in einem MDM beschreiben. Diese decken den gesamten Analyseprozess von Fragestellungen, über Analyseketten hin zu Analyseergebnissen ab, beschreiben aber auch die enthaltenden Operatoren, Visualisierungen, Verfahren und Business Rules. Die wichtigsten Konzepte werden in Abschnitt 4.2 erläutert. Die auf der Analyse-Ontologie aufbauende Domänen-Ontologie beinhaltet die konkreten Instanzen der Analyse-Ontologie für eine Domäne. Sie bildet also das für Analyseunterstützung genutzte Analyse- und Domänenwissen ab. Für eine detaillierte Beschreibung der Ontologien und deren Zusammenhänge sei auf [9] verwiesen.

Im Metadatenmodell zu speicherndes Wissen soll Aussagen zu konkreten Fragestellungen, über durchzuführende Analyseschritte bis hin zu erkennbaren Analyseergebnissen enthalten. Soll z.B. die Fragestellung „Welchen Marktanteil hat mein Krankenhaus (KH) im Einzugsgebiet?“ untersucht werden, so ist als Wissen modelliert, dass das Einzugsgebiet, bestehend aus Kernmarkt, erweiterter Kernmarkt und Peripheriemarkt, für das KH über dessen Fallzahlen ermittelt werden muss und dass der Marktanteil eine berechnete Kennzahl im DWH ist. Diese berechnet sich aus dem Verhältnis der erwarteten Fallzahl und den behandelten Fällen des KH. Des Weiteren ist z.B. als Wissen modelliert, dass sich aus der ersten Fragestellung weitere relevante Fragestellungen ableiten können, z.B. „Wie verhält sich die Fragestellung für Konkurrenten oder aber für spezifische Fachabteilungen (FA) des eigenen KH?“. Für letzteres müssen insbesondere die behandelten Diagnosen (ICD-Codes) und die vorhandene Ausstattung / Verfahren (OPS-Codes) sowie der Versorgungsschwerpunkt der FA im MDM berücksichtigt werden. Besonders relevant ist das Wissen zu weiterem Analysevorgehen im Kontext von Analyseergebnissen, wie z.B. erkannten Auffälligkeiten: Änderungen im Patientenspektrum / Einweiserverhalten; oder Erkennen von Versorgungslücken oder Regionen mit „stillen Reserven“.

Das Konzept sieht einen initialen Aufbau der Wissensbasis, mit Hilfe einer Expertengruppe vor. Diese können im KMA Umfeld z.B. einer Krankenhauskette oder einem Beratungsunternehmen angehören und ihr Analyse- und Domänenwissen in den Ontologien persistieren. Da das modellierte Wissen eine Allgemeingültigkeit in der modellierten Domäne haben soll, ist eine spätere personifizierte Anpassung bzw. die Erfassung zusätzlicher personenbezogener Metadaten derzeit nicht vorgesehen.

4.1 Analyseunterstützende Funktionalitäten

Um den genannten Mängeln aus Abschnitt 2 unter Berücksichtigung der in Abschnitt 3 erläuterten Anforderungen zu begegnen, werden die Metadaten in Form einer semantischen Suche, Navigation und eines Recommendings genutzt. Die Metadaten werden an den Business User über diese Funktionalitäten kommuniziert.

Semantische Suche: Das übergeordnete Ziel der semantischen Suche ist, eine Suchfunktion auf definierten Metadaten anzubieten. Instanzen des Metadatenmodells sollen anhand ihrer Semantik gesucht, gefunden und anschließend visualisiert werden.
Ein spezifischeres Ziel ist unter anderem die Suche nach bestehenden Fragestellungen und damit verbundenen Analyseketten, die sich auf die jeweilige Analysesituation adaptieren lassen. Auch semantisch verknüpfte Fragestellungen, die in einer Vorgänger-, Nachfolgerbeziehung stehen, können gefunden werden. Daneben spielt auch das Finden von Analyseergebnissen und damit verbundenen quantitativen Daten des DWH eine Rolle. Annotationen auf den quantitativen Daten können mit ihrer Semantik gefunden und in der zugehörigen Analysevisualisierung wieder dargestellt werden. Ebenfalls lassen sich die durchlaufende Analysekette samt Fragestellung und analysierendem Business User ermitteln und als Ausgangsbasis für anknüpfende Analysen verwenden.

Semantische Navigation: Werden die Klassen und Relationen des Metadatenmodells für eine konkrete Domäne instanziiert, so kann eine Navigation von einer Klasseninstanz zur Nächsten, entlang der dazwischen definierten semantischen Relation, erfolgen. Im Kontext eines AIS kann diese semantische Navigation zum einen als eine reine Navigation innerhalb der Metadaten erfolgen und zum anderen können zusätzliche assoziierte Operationen des AIS ausgeführt werden.
Für Ersteres sei auf Abb. 1 verwiesen. Wird die Fragestellung zur Kennzahl B mit Hilfe der semantischen Suche gefunden, so kann der Business User sowohl zu den notwendigen Vorgänger-Fragestellungen als auch den Nachfolger-Fragestellungen oder zu weiteren verbundenen Instanzen navigieren.

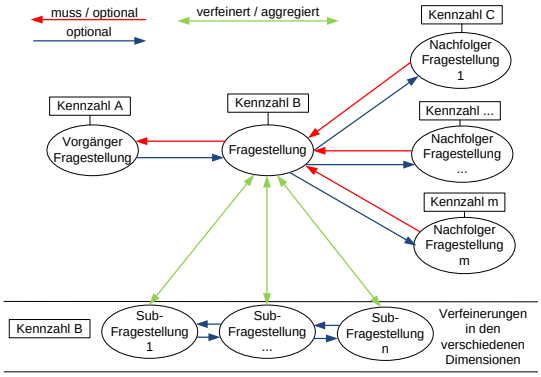


Abbildung 1: Beziehungen von Fragestellungen

Möchte der Business User basierend auf einer Fragestellung eine konkrete Analyse durchführen, so kann er von der

Fragestellung zu der verbundenen Start-Analysevisualisierung wechseln und dann entlang eines spezifischen Analysepfades zu einer Ende-Analysevisualisierung navigieren (s. Abb. 2). Diese Art der semantischen Navigation ist immer mit realen Aktionen und Daten des AIS verbunden, da Analysevisualisierungen quantitative Daten des DWH anzeigen und beim Analysevisualisierungswechsel durch die Domänenoperationen ein oder mehrere OLAP Operationen im AIS angewendet werden.

Vorschlagsgenerierung: Eine wichtige Funktion ist das Geben von Hinweisen und Vorschlägen durch das AIS im Kontext einer Fragestellung und einer Analysevisualisierung. Diese werden aus dem im Metadatenmodell hinterlegten Analyse- und Domänenwissen abgeleitet. Vor allem weiterführende Hintergrundinformationen, Business Rules und Analysestrategien sowie weitere Fragestellungen können wertvolle Informationen im Analyseprozess sein.
Hinweise zu weiterführenden sinnvollen Analysevisualisierungen können durch das AIS gegeben werden, indem mögliche Domänenoperationen und die enthaltenen Business Rules auf ihre Anwendbarkeit in einer Analysevisualisierung überprüft werden. Ziel der Vorschlagsgenerierung soll es sein, dass relevante Analysevisualisierungen erreicht werden, die eine Interpretation der quantitativen Daten hinsichtlich der Fragestellung zulassen. Wichtig ist, dass beim Geben der Hinweise und Vorschläge sowie bei der Anwendung von Domänenoperationen die Hintergründe kurz erläutert werden, damit diese für den Business User nachvollziehbar bleiben und er sich zusätzlich Analyse- und Domänenwissen aneignen kann.

4.2 Wissensbasierte Konzepte

Im Folgenden werden die für eine Analyseunterstützung notwendigen Entitäten des Analyseprozesses in einem AIS näher betrachtet.
Fragestellung: Eine Fragestellung gibt einer Hypothese Ausdruck, die es zu klären gilt und auf deren Ergebnis eine Entscheidung basieren kann. Als Beispiel sei "Was sind die Marktanteile in meinem Einzugsgebiet?" genannt. Hierbei ist eine einzelne Fragestellung in der Regel nicht losgelöst von Anderen zu betrachten (s. Abb. 1). Vielmehr kann eine Fragestellung über Vorgänger verfügen, wie z.B. "Was ist mein Einzugsgebiet?", auf welche zuvor eine Antwort gefunden werden muss. Sie kann über weiterführende Nachfolger-Fragestellungen verfügen, die sich aus dem Analyseergebnis ergeben und optional weiter analysiert werden können. Beispiele sind: "Was sind die Marktpotenziale in meinem Einzugsgebiet?" oder "Wie wird sich mein Marktanteil entwickeln?". Neben Vorgängern und Nachfolgern kann es auch Subfragestellungen geben, welche die gleiche Kennzahl(en) wie die Originalfrage behandeln, jedoch diese hinsichtlich ein oder mehrerer Dimensionen weiter verfeinern.

Analysekette: Eine Analysekette wird für die Analyse von Fragestellungen verwendet. Diese bildet den logischen Rahmen der Analyse und enthält weitere Konzepte wie Analysepfade, Analysevisualisierungen, Domänenoperationen und Business Rules. Die Zusammenhänge sind in der Abb. 2 dargestellt. Ziel der Analysekette ist das Finden ein oder mehrerer Analysevisualisierungen, die eine Interpretation der quantitativen Daten des DWH erlauben.

Analysevisualisierung: Eine Analysevisualisierung ist die grafische Repräsentation der quantitativen Daten eines Analyseschrittes z.B. in Form einer Pivottabelle, eines Diagramms oder einer Karte. Sie besteht aus 1 bis n Kennzahlen, die wiederum in 1 bis m Dimensionen aufgespannt sind. Von diesen Dimensionen sind jeweils 1 bis z Dimensionselemente gewählt. Im Kontext der Analysekette sind sogenannte Start- und Ende-Analysevisualisierungen definiert. Erstere dienen als Einstiegspunkte für die Analysen von Fragestellungen. Von ihnen aus können die Analysepfade mit ihren verschiedenen Analysevisualisierungen durchlaufen werden. Letztere ermöglichen das Interpretieren der quantitativen Daten hinsichtlich der Fragestellung und das Ableiten von Analyseergebnissen.

Domänenoperation: Eine Domänenoperation ermöglicht das Navigieren zwischen zwei Analysevisualisierungen und kommt zum Einsatz, wenn spezifische Ausprägungen des MDM in einer Analysevisualisierungen eintreten. Domänenoperationen können aus einer Menge auszuführender OLAP-Operatoren, Business Rules und Visualisierungswechseln bestehen. Ihr Zweck ist es, die notwendigen Schritte der Kennzahlen-, Dimensionen-, Dimensionselemente- und Visualisierungsauswahl bzw. des -wechsels für den Endanwender durchzuführen und somit die Komplexität des MDM und der Analysedurchführung zu verbergen. So kann der Business User direkt von einer Analysevisualisierung zu einer nächsten sinnvollen Analysevisualisierung gelangen. Ob eine Analysevisualisierung als sinnvoller weiterer Analyseschritt gesehen werden kann, ist über die Business Rules definiert.

Business Rules: Business Rules repräsentieren konkretes Analyse- und Domänenwissen und werden als semantische Metadaten zu den domänenspezifischen Inhalten eines AIS modelliert. Sie beschreiben weiterführende Informationen und Regeln und kommen im Analyseprozess in den Domänenoperationen zur Anwendung. Business Rules lassen sich aus den „Best Practices" eines Analysten ableiten und beziehen sich in der Regel auf eine Fragestellung, eine Analysevisualisierung oder ein Analyseergebnis, oder aber auf eine Kombination aus diesen. Die Business Rule ist anwendbar, wenn das MDM einen spezifischen definierten Zustand erreicht. Mehrere Business Rules können in unterschiedlichen Domänenoperationen anwendbar sein, woraus sich mehrere Möglichkeiten für den Business User ergeben können, wie er seine Analyse fortsetzen möchte. Durch die Gewichtung der Business Rules kann hierbei ein Ranking entstehen.

Analysepfad: Unter einem Analysepfad wird eine Menge von Analysevisualisierungen verstanden, die durch Domänenoperationen zu einem Pfad in einer definierten Reihenfolge verbunden werden. Typischerweise wird ein Analysepfad in einer konkreten Analyse von einer Start-Analysevisualisierung zu einer Ende-Analysevisualisierung durchlaufen. Hierbei ist zu beachten, dass Analysepfade nicht zwingend zuvor definiert sein müssen, sondern sich aus der Anwendbarkeit von Domänenoperationen auf Analysevisualisierungen im Kontext einer spezifischen Fragestellung ergeben können. Die Menge aller Analysepfade zu einer Fragestellung bildet die Analysekette.

Analyseergebnis: Analyseergebnisse beschreiben eine Interpretation von quantitativen Daten im MDM des DWH.

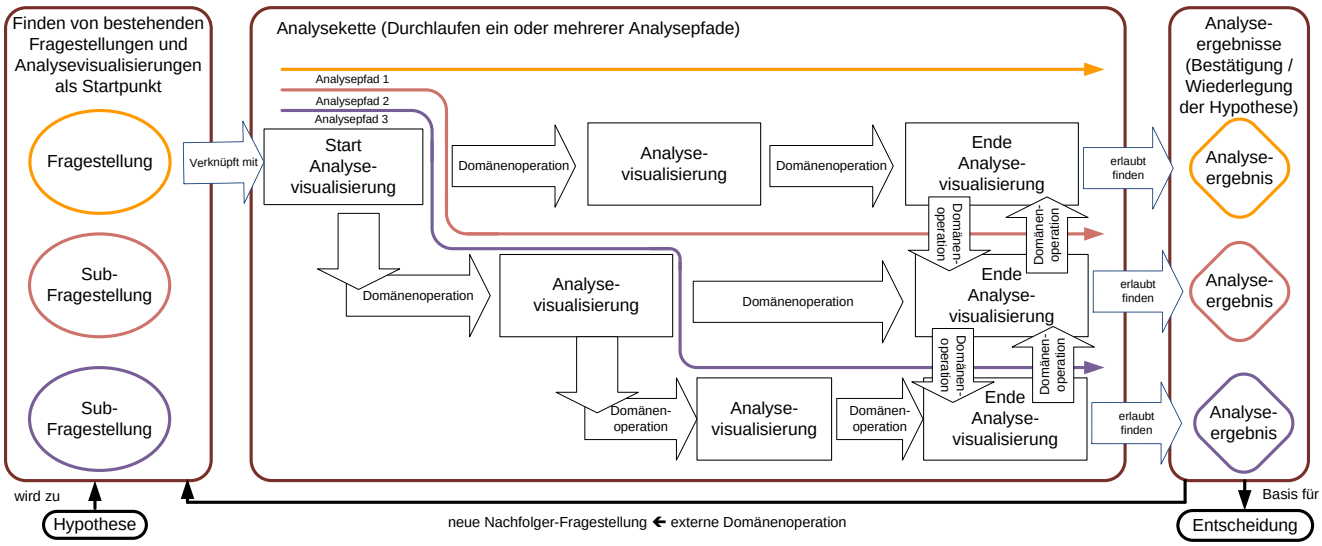


Abbildung 2: Zusammenhang von Konzepten im Analyseprozess

Dies können Korrelationen und Zusammenhänge zwischen den Daten, aber auch Trends, Einbrüche, Zunahmen, Auffälligkeiten, geografische Ballungen, etc. sein.

5. VERWANDTE ARBEITEN

Im Kontext der Business Intelligence gibt es vielfältige Forschungsfragen, die von der Erschließung des Analysewissens [1], über die Anpassung der BI-Tools an Business Anforderungen [11, 2] und die integrierte Anfrage von BI-Tools [12, 4] bis hin zur Annotation von Datenschemata mit weiterführenden Informationen zur Analyseunterstützung reichen [2, 7, 8, 11]. In vielen Fällen soll der Analyst und teilweise auch der Business User [2, 12] stärker in den Fokus rücken, indem die BI-Software besser an individuelle Informationsbedürfnisse angepasst werden kann oder aber eine Analyseunterstützung zur individuellen Befriedigung des Informationsbedarfs angeboten wird. Viele der verwandten Arbeiten versuchen die jeweiligen adressierten Probleme durch die Nutzung von semantischen Metadaten und damit verbundenen Semantic Web Technologien zu lösen.

Im Bereich der Dokumentation von Analyseprozessen und -ergebnissen ist die Arbeit [1] zur Distribution von Business-Intelligence-Wissen zu nennen. Diese zielt auf die kontrollierte organisationsweite Verbreitung und Weiterverwendung von Berichten und Analyseansätzen durch das Einstellen der BI-Inhalte in Wissensmanagementsysteme ab, jedoch ohne auf eine konkrete technische Umsetzung einzugehen.

Im Kontext von Analyseprozessen ist auch die Arbeit von [3] relevant, da diese den Begriff der Analysekette einführt. Fokus der Arbeit ist die Mensch-Maschine Interaktion, wobei das Finden, Aufbereiten und Darstellen von Daten als technisch / operative Tätigkeit gesehen wird. Zu den kognitiven Tätigkeiten zählen die Bildung eines anwenderorientierten Analysemodells sowie dessen Prüfung und Verfeinerung.

Während der Durchführung von Analysen werden eine Reihe von OLAP Anfragen gesendet, um durch den multidimensionalen Datenraum zu navigieren und um die benötigten Informationen zu erfragen. Hierbei ist laut [5] das Erstellen entsprechender Anfragen eine schwierige Aufgabe. Daher wird in dieser Arbeit ein Framework vorgestellt, dass den Nutzer in der Analyse unterstützt, indem es passende OLAP Operatoren vorschlägt. Diese werden durch die Auswertung des OLAP Server Query Logs abgeleitet.

Eine andere Art der Analyseunterstützung für eine höhere Nutzerfreundlichkeit wird im Kontext der BI in der Trennung von Business und IT Belangen gesehen. Anders als in den zuvor genannten Arbeiten spielen hier semantische Metadaten eine entscheidende Rolle. Über die Metadatenmodelle können die Entitäten der zugrunde liegenden BI-Systeme mit ihren zugehörigen Relationen modelliert werden. Ebenfalls erlauben diese flexiblen, erweiterbaren Metadatenstrukturen weiterführende Informationen zu den Entitäten, wie beispielsweise Business Rules oder Expertenwissen festzuhalten. In [7] und [8] werden diese Möglichkeiten weiter vertieft und als Anwendung z.B. die Nutzung der Metadaten im Extraktions-, Transformations- und Ladeprozess (ETL) für ein DWH angeführt.

In der Arbeit von [11] wird auf dieser Semantic Web Basis eine Architektur für analytische Tools vorgeschlagen, um Effizienzsteigerungen im Entscheidungsfindungsprozess zu erzielen. Durch die Nutzung einer Domänen-Ontologie, welche die Entitäten der zu untersuchenden Domäne, deren Relationen sowie weitere Informationen bereitstellt und durch die Nutzung einer Business-Intelligence-Ontologie, welche Informationen zu den Datenstrukturen des multidimensionalen Modells vorhält, soll eine Unterstützung im Analyseprozess erfolgen. Zum einen soll die Auswahl von Dimensionselementen durch proaktive Vorschläge erleichtert werden und zum anderen sollen durch ein semiautomatisches Umschreiben der Anfragen Analyseergebnisse aufgewertet werden. Die Entitäten werden in den jeweiligen Domänen-Ontologien oftmals mit ihrem natürlichsprachlichen Namen versehen, da diese eher die Business Semantik ausdrücken als die korrespondierenden technischen Bezeichner des MDM. Nutzer

können so Anfragen in einer für sie vertrauten Terminologie an das System stellen. Dieser Ansatz wird in [4] und [12] verwendet, um auf einer abstrakten Ebene unternehmensweite Informationen integriert aus verschiedenen BI-Systemen wie DWH, ERP, CRM, etc. anzufragen. In diesen Ansätzen werden semantische Metadaten auch für die Integrationsaufgaben und das Umschreiben von Anfragen verwendet.

Die Unterstützung eines technisch unversierten Business Users wird insbesondere in [2] und [12] fokussiert. In [2] liegt der Schwerpunkt auf einer kollaborativen ad-hoc Entscheidungsunterstützung, in der Daten integriert aus den verschiedenen BI-Tools dargestellt und über semantische Metadaten mit weiterführenden Informationen versehen werden. Insbesondere sollen über Web 2.0 Technologien Information Mash-Ups gebildet, aber auch eine Kollabration zwischen verschieden Business Usern erzielt werden.

Auch wenn, wie in vielen anderen Arbeiten, die semantischen Metadaten eine entscheidende Rolle spielen, grenzt der eigene Ansatz sich von diesen durch die Modellierung von Konzepten und deren Instanziierung in Form von Analyse- und Domänenwissen ab. Im Fokus steht dabei das Wissen zu Analyseprozessen, das von verschiedenen Komponenten eines AIS zur Unterstützung des Business Users genutzt werden kann. Eine technologisch ähnliche Umsetzung mit Hilfe von verschiedenen Ontologien wie sie in [11] und [12] genannt werden, wird angestrebt.

6. FAZIT UND AUSBLICK

Analytische Informationssysteme haben sich im Kontext der Business Intelligence als Systeme zur Informationsgewinnung für Business User im Entscheidungsfindungsprozess etabliert. Allerdings setzen AIS aufgrund ihrer hohen Interaktionsmöglichkeiten und Komplexität entsprechendes Analyse- und Domänenwissen sowie ein tiefergehendes technisches Verständnis voraus, um adäquate Analysen durchführen zu können (s. Abschnitt 1). Dieses liegt jedoch ohne entsprechende Schulungen bei Business Usern nicht vor. Auch werden in der Regel keine Metadaten zu Business Regeln, Strategien oder Hintergrundinformationen durch AIS bereitgestellt, die den Business User unterstützen können. Da jedoch Business User befähigt werden sollen, sich selbst mit Informationen zu versorgen, wurde in dieser Arbeit als Forschungsfrage untersucht, wie die Komplexität von AIS reduziert werden kann, damit Business User eine adäquate explorative ad-hoc Analyse durchführen können. Hierzu wurden zunächst in Abschnitt 3 Anforderungen an einen entsprechenden Ansatz definiert. Darauf aufbauend wurden in Abschnitt 4 verschiedene Konzepte aus dem Bereich der Datenanalyse definiert und erläutert, die in einem semantischen Metadatenmodell für ein AIS modelliert und instanziiert werden können und somit als Basis für weiterführende Analyseunterstüzungsfunktionalitäten dienen. Von zentraler Bedeutung waren die Konzepte der Fragestellung, Analysekette, Analysepfad und Analysevisualisierung sowie Domänenoperationen, Business Rules und Analyseergebnisse. Als Funktionen wurden die Semantische Suche, die Semantische Navigation und die Vorschlagsgenerierung präsentiert, bevor in Abschnitt 5 verwandte Arbeiten vorgestellt und gegen den eigenen Ansatz abgegrenzt wurden. Als weiterer Schritt im Forschungsvorhaben werden die vorgestellten Konzepte in einem Metadatenmodell mittels Wissensrepräsentationssprachen modelliert und für die Domäne der Krankenhausmarktanalyse instanziiert. Dieses Metadatenmodell wird in eine zu implementierende semantische Metadatenebene des am OFFIS entwickelten AIS - Multidimensional Statistical Data Analysis Engine (MUSTANG) - eingebettet, um Analyse- und Domänenwissen zu erfassen, zu verwalten und für die genannten Analyseunterstüzungsfunktionalitäten zu verwenden. Eine Umsetzung und Evaluierung wird im Rahmen einer laufenden Dissertation und einer studentischen Projektgruppe erfolgen.

7. REFERENCES

[1] H. Baars. Distribution von Business-Intelligence-Wissen. *Analytische Informationssysteme*, pages 409–424, 2005.

[2] H. Berthold, P. Rösch, S. Zöller, F. Wortmann, A. Carenini, S. Campbell, P. Bisson, und F. Strohmaier. An Architecture for ad-hoc and collaborative Business Intelligence. In *Proceedings of the 2010 EDBT/ICDT Workshops*, EDBT '10, pages 13:1–13:6, New York, NY, USA, 2010. ACM.

[3] N. Bissantz. Deltaminer. *Wirtschaftsinformatik*, 43(1):77–80, 2001.

[4] L. Cao, C. Zhang, und J. Liu. Ontology-based Integration of Business Intelligence. *Web Intelligence and Agent Systems*, Volume 4, 2006.

[5] A. Giacometti, P. Marcel, und E. Negre. A Framework for Recommending Olap Queries. In *Proceeding of the ACM 11th international workshop on Data warehousing and OLAP*, DOLAP '08, pages 73–80, New York, NY, USA, 2008. ACM.

[6] P. Gluchowski und H.-G. Kemper. Quo Vadis Business Intelligence? Aktuelle Konzepte und Entwicklunstrends. *Business Intelligence Spektrum*, 1. Jg., Heft 1, Mai 2006:12 – 19, 2006.

[7] N. Inference. Ontology and Data Warehousing. Technology white paper, NETWORK INFERENCE, INC., 2004.

[8] L. Ludwig. Business Intelligence und das Semantic web: Ein Traumpaar. 2005.

[9] M. Mertens, Y. Teiken, und H.-J. Appelrath. Semantische Anreicherung von strukturierten Daten und Prozessen in Analytischen Informationssystemen am Beispiel von Mustang. In *Forschungskolloquium der GI Fachgruppe 5.8 - Management Support Systems, Dortmund, Deutschland*. Universität Dortmund, 2009.

[10] B. ONeil. Semantics and business. *The Data Administration*, 2007.

[11] D. Sell, L. Cabral, E. Motta, J. Domingue, F. Hakimpour, und R. Pacheco. A semantic web based Architecture for Analytical Tools. In *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*, pages 347–354, Washington, DC, USA, 2005. IEEE Computer Society.

[12] M. Spahn, J. Kleb, S. Grimm, und S. Scheidl. Supporting Business Intelligence by providing ontology-based End-User Information Self-Service. In *OBI '08: Proceedings of the first international workshop on Ontology-supported business intelligence*, pages 1–12, New York, NY, USA, 2008. ACM.

Towards Efficiently Running Workflow Variants by Automated Extraction of Business Rule Conditions

Markus Döhring
SAP Research Darmstadt
Bleichstraße 8
64283 Darmstadt, Germany
markus.doehring@sap.com

Christo Klopper
SAP Deutschland
Hasso-Plattner-Ring 7
69190 Walldorf, Germany
christo.klopper@sap.com

Birgit Zimmermann
SAP Research Darmstadt
Bleichstraße 8
64283 Darmstadt, Germany
birgit.zimmermann@sap.com

ABSTRACT

Efficient workflow variant management is becoming crucial especially for enterprises with a large process landscape. Our research fosters the combination of business rules for adapting reference workflows at runtime and tailoring them to many different situations. A main goal is to optimize the performance of workflow instances w.r.t. different aspects, e.g., branching decisions, throughput time or compliance.

Having a data mining procedure at hand which can automatically extract potentially useful conditions from execution logs to create new variants is therefore a very significant benefit. The extracted conditions could be conveniently reused within the business rules of our framework, which can handle the deviations at runtime for those special situations. However, most existing data-mining techniques do not describe a continuous mining pipeline how to get from workflow logs to problematic context conditions for new variant creation or are difficult for business people to interpret.

Therefore we present an integrated rule mining methodology, starting with the semi-automatic discovery of "hot spots" within workflow instance logs. Then, data variables of instances related to these hot-spots are translated into a data mining classification problem. Other than related approaches, we employ a fuzzy rule learning algorithm, yielding easily interpretable and reusable conditions for variants. We also provide first insights from a case study at a consulting company and corresponding open research challenges.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; H.4.1 [**Information Systems Applications**]: Office Automation—*Workflow Management*; D.2.2 [**Software Engineering**]: Design Tools and Techniques

Keywords

workflow, business rules, process mining, process performance, rule learning

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

1. INTRODUCTION

Workflow management systems (WfMS) are becoming an essential part of most industrial IT system landscapes [19]. For some domains, traditional WfMS have already been determined as unsuitable to cover prevalent requirements w.r.t. the flexibility of workflows [7]. In order to address the challenge of managing workflow variants (i.e. workflows with slight deviations from a "reference workflow") at design-time as well as their dynamic adaptation at runtime due to changing data contexts, we have proposed the integration of business rules containing adaptation operations on adaptive segments in reference workflows [10].

In many practical scenarios, it is unrealistic that process analysts are able to define all variants and exceptions in a workflow. Especially when a WfMS is introduced in a company, but also if workflow models are already mature, environmental changes may lead to shifts in the impact factors on process performance. A potential relief for making such blind spots in workflow execution visible is the application of process mining techniques. The goal is to find data-dependencies for weak spots in the workflows and making them available as conditions for additional business rules leading to new workflow variants. Existing work has partly addressed these issues each with a relatively isolated view on e.g. bottleneck detection or dependency mining. Results w.r.t. to an integrated "mining pipeline" for a business user are however still quite unsatisfying. For example, prevalent approaches leave the user with a mined decision tree which, as we will show, might be hard to read for real-world workflow logs. Instead, we aim at a pipeline from a workflow definition in an understandable notation over automated mining application to interpretable business (variant) rules.

Our approach is based on the general idea of rule-based workflow adaptation as described in Section 2. As a solution to the above challenges, in Section 3 we present a mining methodology which we consider promising as a suitable mining pipeline for a business user. For each of the methodology's three generic steps, concrete technologies and their wiring are explicated, especially the employment of a fuzzy mining approach for ruleset extraction. We then present first learnings from a case study on real-world workflow execution data building upon our methodology in Section 4 and summarize challenges which have to be solved to fully implement our methodology in Section 5. In Section 6 we discuss related research, before we conclude in Section 7 and state remaining issues for future work.

2. FLEXIBILIZATION OF WORKFLOWS BY ADAPTATION RULES

Our methodology for condition extraction is motivated by a general approach for workflow adaptation [10, 9]. It is considered essential to establish a basic understanding of the nature of business(variant) rules as targeted for being automatically mined. Our framework as well as the examples in this paper rely on BPMN2 [1], because its notation is a de-facto industry standard which was designed to be understandable for business users. Basically, the framework consists of three conceptual building blocks for workflow variant management and flexible workflow adaptation:

1. Adaptive Segments in BPMN2 Reference Workflows: An adaptive segment demarcates a region of a workflow which may be subject to adaptations at runtime when entering the segment. It corresponds to a block-structured part of the workflow, i.e. a subgraph which has only one incoming and one outgoing connection. In special cases, adaptive segments can also be "empty". What matters is that they correspond to valid BPMN2 workflow definitions and not to a kind of white box which is left empty for later filling. We have extended the BPMN2 metamodel to capture the special semantics of adaptive segments [9].

2. Workflow Adaptations Defined in BPMN2: The actual definition of potential adaptations which can take place at runtime have been proposed as a pattern catalogue [10] which also relies on BPMN2 notation, with the benefit that adaptation patterns are comprehensible and extensible. The catalogue contains basic adaptations like SKIP or INSERT, but also more sophisticated event- and time- related patterns, like "event-based cancel and repeat" or "validity period". Every adaptation pattern has the block-structured adaptation segment as an obligatory input parameter. As such, patterns can be conveniently nested and combined.

3. Linking Adaptations to Data Contexts by Business Rules: Business (variant) rules are used to apply suitable adaptations for different situations expressed by data context conditions. The data context can be globally valid (like a date) or workflow instance specific (like an order value). A pseudo-syntax for variant rules, where $*$ stands for 0-n repetitions, can be defined as: **ON** *entry-event* **IF** <*data-context*> **THEN APPLY** [<*pattern(segment, (parameter, value)**>]$*$ Once the general relations of adaptive segments and potential adaptations have been established by a process analyst, the conditions could be maintained by a business user e.g., via a domain-specific language. For automatic rule extraction, in this work we therefore especially focus on the IF-part of potentially newly discovered variant rules and aim at revealing data dependencies for variants which are not a-priori known, but have significant implicit impact on the overall business performance of workflow execution.

Figure 2 exemplifies the above concepts based on a ship engine maintenance workflow fragment. The actual conduction of engine tests for a ship may depend on the harbor in which it currently resides. Due to environmental restrictions, many different harbors impose specific time constraints on ships conducting engine tests. In Hamburg for example, ships may only have 12h time, after which devices need to be reset and the tests need to be restarted. For adapting the workflow correspondingly, a generic parameterizable template is used and weaved with the segment at runtime.

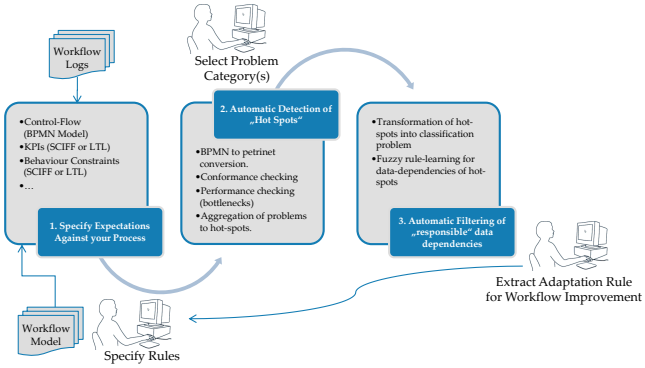


Figure 1: Outline of the Rule Extraction Procedure

3. METHODOLOGY FOR VARIANT RULE CONDITION EXTRACTION

As already stated, we are interested in automatically extracting condition constraints (the "IF-part") for potentially useful workflow adaptation rules within our framework. Useful in this respect means, that the condition constraints should describe eventually problematic situations in workflow instances by means of their data context values, such that a timely adaptation of a workflow instance can eventually prevent such a situation. Our proposed methodology is illustrated in Figure 1 in a circular manner. The methodology is divided into three main phases explained in detail in the following subsections. For each phase, concrete concepts and technologies for implementing the methodology are discussed and open challenges are outlined where existing.

3.1 Formulation of Log Expectations

The first phase of our methodology consists in the definition of expectations towards a set of workflow instance logs. Correspondingly, there are two obligatory input components for the extraction pipeline: a workflow model and a sufficiently large set of workflow instance logs belonging to the model. The instance logs must contain workflow-relevant events like at least the start or finishing timestamp of particular task types and must also carry a number of data context variables[1]. Since we want to target business users with our rule extraction approach, we consider BPMN as an appropriate input format for the expected control-flow logic restricing the expected order of task executions and event occurrences in the input logs.

As an optional input, additional constraints w.r.t. workflow execution can be provided in some form of logic. These constraints may concern time-related interdependencies of events within a workflow instance log, whereas typical key performance indicators (KPIs) like throughput times can be understood as a subset of such time constraints. But also other more sophisticated circumstances which are hard to model in BPMN2 graph structures can be provided as logical constraints, as for instance that a task A should be executed N times after the occurrence of task B. Suitable logics to formulate such process-related constraints can for example be based on the SCIFF framework [5] or linear temporal logic (LTL) [17]. Since a regular business user may not

[1]It is hard to give generally valid recommendations on data size characteristics, but from experience reasonable mining can start from 1000 instances with about 5 context variables.

be familiar or feel comfortable with such logics, it is recommended to provide constraint templates, i.e. small chunks of logic mapped to easily parameterizable pieces of restricted natural language for constraint maintenance.

3.2 Automatic Discovery of "Hot Spots"

For the ability to apply established mining and analysis techniques on the instance logs in combination with the workflow model, it is useful to first transform the BPMN workflow definition into a pure formal representation, e.g. in terms of petri net graphs which are backed by a long trail of research and corresponding toolsets. Transformation mechanisms which are able to map a large part of BPMN constructs to petri net constructs exist [8] and can be employed within our methodology. The next phase of our methodology then consists in the automatic discovery of problematic spots in the instance logs, relating to different issues:

1. **Non-conformance to defined workflow model**: Using log-replay approaches on the petri net model as presented in [16], it can be determined whether instances behave exactly according to the underlying model or whether there are deviations. Provided the petri net has been suitably constructed, such deviations can be structurally spotted as petri net places where tokens are left over after an instance has been finished or where tokens often are missing when a transition should be fired. In most of the latter cases, a distinct transition (=BPMN task) can be "blamed" for causing the non-conformance. Places and transitions with a relatively high error-rate are kept for further analysis within our methodology.

2. **Disproportionate delays (bottlenecks)**: Similar to the above petri net log-replay techniques, the sojourn times of tokens in places and the times it takes to execute transitions can be stored [12]. Based on this computed data, it can be determined where instances on average get stuck for a disproportionate amount of time related to the average overall throughput time. The corresponding threshold values can be computed automatically if they are not explicitly formulated as KPI constraints, which is discussed below. Again, concerned places and transitions are kept for analysis.

3. **Non-conformance to execution constraints**: SCIFF or LTL constraints can be checked on the instances logs using approaches from [5] resp. [17] with respect to their violation. The employment of constraint checking allows for a very broad range of non-conformance types being checked. Three of the most important ones are:

   - The violation of KPIs by the use of time-related constraints (for example, task B has to be executed 1h after task A latest).
   - The deviation from expected routing decisions (for example if orderValue>10.000 in a sales order, always choose the "priority shipment" branch after an exclusive gateway).
   - Data- or organizational incompliance like the violation of the "four-eyes principle" for some tasks.

   In contrast to the checking mechanisms for issues (1.) and (2.), a challenge consists in the spotting of the actual source for a constraint violation. For our KPI example (B 1h after A), if B is not executed at all, it has to be decided whether *A* or *not B* or *both* are to be considered as the actual error source and kept for further analysis. Potentials lie in the partly automated mapping of constraint predicates to places or transitions in the underlying model and the consideration of "what happened first". Research is still ongoing here.

As a final step of this phase, the user is confronted with issues which have a particular degree of "severity" (e.g. exceed a predefined fraction of instances which are non-conformant) and gets the corresponding "hot-spots" based on average instance execution marked in the BPMN process model. The proper automatic accumulation and back-projection of issues to the BPMN workflow model remains an open issue. The user may then select one or several hot spots and one or several problem types for these hot-spots for further analysis by mining data dependencies as business rule conditions as described in the next subsection.

3.3 Automatic Extraction of Rules for "Hot Spot Occurrences"

For the selected hot-spots and problem types, the instance data from the workflow logs is transformed into a classification problem for machine learning algorithms. A classification problem consists of a number of cases (=workflow instances), each made up of a number of numeric or nominal data variable values (=workflow instance or task context, e.g. order value, customer priority or shipment partner) and a single class in terms of a category for a learning instance. The class can be determined in a binary manner as *problematic* or *non-problematic* from the problem types connected to the hot spots, but also the distinction of finer-granular problem classes can be considered. The variable values for a learning instance can be constructed by looking at their occurrence when an instance has reached a hotspot in the petri net. Special challenges in this conversion step concern the treatment of some control-flow constructs, as for example a loop which may cause multiple visits of a hotspot in a workflow, whereas the context variables may have changed meanwhile. Such problems and solution approaches, for instance creating a separate training instance for each loop execution, are discussed, e.g., in [15].

Having the training set for a machine learning classifier at hand, established algorithms like C4.5 decision tree [14] or rule learners [6, 11] can be applied. In fact recent research mostly favors decision trees for presenting mining results to the business user [18]. However, we have tested the C4.5 decision tree learner on a real-world dataset (see Section 4) and found its results not interpretable for the business user to draw any reasonable conclusions from it mainly due to the size and complexity of the overall decision tree. Despite ex-post global optimization heuristics in C4.5, local feature selection often leads to redundant splits in the initial decision trees. As rules can only be extracted one-by-one along paths in the decision tree [11], they are of rather less use for directly extracting conditions for use in adaptation rules that might eventually tackle the problematic situation at workflow runtime. The problem with established rule learners like RIPPER [6] in turn is that they generate *ordered* rulelists, which means each rule in the list covers only those

learning instances which are not covered by the previous rule. This characteristic makes the corresponding output rules also hard to read and interpret for an end user. Potential relief consists in the employment of a fuzzy learning approach which generates globally valid rules that have a probabilistic certainty factor to hold on the dataset or not. We are currently evaluating a novel algorithm [13] w.r.t. the suitability for being employed within our methodology, which is subject to discussion in the following section.

4. CASE-STUDY

The first feasibility study for our methodology was conducted at a large globally operating IT consulting company. In the following, we report on the input dataset, the realization of our methodology in the ProM[2] framework, and our preliminary results and findings.

4.1 Description of the Dataset

The focus of the case study is on a staffing workflow for serving customer and company-internal human resource requests for different type of IT projects. A simplified corresponding model in BPMN notation is shown in Figure 3. The first three sequential steps are creating and submitting the request and then having it validated by an authorized person. Resources can be found by three different strategies: by company-internal broadcasts, by external broadcasts to partner consulting companies or by directly contacting a potentially suitable resource. After at least one such search procedure has been triggered, different reactions can occur, namely the acceptance, rejection, withdrawal or feedback of non-availability for a particular resource. At anytime during these search procedures, an initial proposition of currently gathered resources can be made to the customer. After the request is closed, it is marked as either successfully or not staffed. The input dataset consisted of 13225 workflow instance logs each with up to 50 data context variable values attached. In this case, context variables concern for example the country a request is sent from, the concerned industry profile or the overall duration of the project.

4.2 Realizing the Methodology based on ProM

For some basic analysis techniques, we rely on functionality provided by ProM. The translation of the BPMN model into a petri net was done manually, as automated mapping approaches still generated too complex results which could make first mining and analysis efforts more difficult. The resulting petri net is shown in the upper middle of Figure 4. Black boxes indicate "silent" transitions which do not correspond to any task in the BPMN model. On the left upper side, one of the additional constraints provided by the consulting company for its staffing workflows is shown, i.e. that before or at least in parallel to an external broadcast, there should also be an internal broadcast trying to gather the required resources. The lower left window shows the evaluation results of these rules. In the right window, the petri net-based bottleneck analysis indicates an overproportional waiting time between request submission and request validation (concrete values in the figure have been changed for anonymization purposes). In the lower middle window, we see an instance marked with a conformance issue, namely that the request validation sometimes has been left out or was conducted only after another task already was executed. Combining these information types, we would identify the validation task as a "hot spot" in the process.

For our first analysis purpose however, we have concentrated on the decision whether a request has been staffed or not. Following [15], we turn the decision into a binary classification problem using a manually selected subset of context variables that have occurred while instance execution. The results are presented in the following.

4.3 Preliminary Results

Running a C4.5 decision tree (J48 implementation) learner with standard parameters yields a decision tree of size 757 with 644 leaves. It is quite obvious that this output type would need a considerable time to be interpreted for a business user. Leaving aside the rule learning algorithms for ordered rule lists, we instead applied the fuzzy rule induction algorithm presented in [13]. Results were very promising, for example generating the following output (some context values changed for anonymization):

```
(Remote = Y) and (ReqingSRegion = DUCKBURG) and (ReqType = Project)
    => class=Branch 4.1 { ROLE_Closed (Not Staffed)/complete } (CF = 0.61)
(ReqingSRegion = NA) and (StartDateFlexible = Yes) and
  (ReqingLOB = FS__Consulting) and (CustIndustry = )
    => class=Branch 4.1 { ROLE_Closed (Not Staffed)/complete } (CF = 0.71)
(Remote = N) and (ContractType = ) and (CustIndustry = UTILITIES) and
  (JobText = B) and (Requestor = ABC) and (StartDateFlexible = No)
    => class=Branch 4.1 { ROLE_Closed (Not Staffed)/complete } (CF = 0.53)
(Remote = ) => class=Branch 4.2 { ROLE_Staffed/complete } (CF = 0.73)
(Remote = Y) => class=Branch 4.2 { ROLE_Staffed/complete } (CF = 0.7)
(ReqingSRegion = GOTHAM_CITY) => class=Branch 4.2 { ROLE_Staffed/complete } (CF = 0.72)
(StartDateFlexible = No) => class=Branch 4.2 { ROLE_Staffed/complete } (CF = 0.72)
```

Manual inspection of the instances characterized e.g. by the first two rules immediately showed that they in fact constitute problematic situations in the staffing workflows. In a flexible WfMS according to Section 2, these conditions could now be reused as a condition for a variant rule with the click of a button, for example inserting addtional activities in the workflow to handele the problematic situation or not even trying specific activities because of potential waste of time.

5. OPEN CHALLENGES

For a better overview and to motivate future work in this area, the main challenges we experienced while setting up the mining pipeline are briefly recapitulated:

- A petri net conversion most useful for mining purposes has to be determined, as straight-forward mappings have problems with more advanced BPMN constructs or generate valid but overcomplex petri nets.
- The accumulation and aggregation of hot spots from the petri net-based and especially the constraint-based checking methods has to be defined in more detail. This challenge is connected to linking back hot spots to the BPMN model for further investigation.
- The conversion of hot spots to a classification problem has to be advanced w.r.t. problematic control-flow structures as for example loop or special joins.
- For the classification problem, the selection of context variables and algorithm parameters has to be made accessible for a business user. Experiments also showed that the rule output may vary significantly w.r.t. the predicates used in the rules. We have to find a way for stabilizing the rule output, e.g. by modifying the learning algorithm w.r.t. this goal and not only taking prediction accuracy into account.

[2] http://www.promtools.org/prom5/

6. RELATED WORK

Due to space restrictions, we do not cover the broad range of general process mining approaches in this section, but rather elaborate on selected approaches which tackle the issue of dependency- or constraint-extraction in workflow logs:

The authors of [15] present the idea of decision point mining in workflows by translating a routing decision into a classification problem for machine learning. In this work, we generalize this idea also for problem domains in workflow execution like bottlenecks or general rule compliance. In [18], a pipeline for analyzing influential factors of business process performance is presented. Some of the steps resemble that of our approach, however e.g. decision trees are used for dependency analysis. The approach is evaluated on a simulated dataset. As we have motivated, decision trees are rather unsuited for direct extraction of globally valid "hot-spot" conditions for a business user on real-world data. An approach for learning constraints for a declarative workflow model is presented in [4], however focusing on control-flow constraints and neglecting data-dependencies. In [3], related to HP's solution for business operation management, an overview on the suitability of different mining techniques for specific analysis types are discussed. Rule extraction is mentioned, but only as rules derived from decision trees which as discussed may get too complex for our purposes. The approach in [2] focuses on dependencies of service-level agreements for service compositions and analyzes reasons for SLA violations. In contrast to our approach, where dependencies are extracted from historic data, the dependencies in [2] are identified at design time for later comparison with monitoring results at runtime.

7. CONCLUSION

We motivated the need for automated extraction of condition constraints for problematic "hot spots" in workflows by the initial uncertainty of a modeler when introducing a flexible WfMS and by rapidly changing impact factors on workflow execution performance. Existing approaches for data dependency extraction have turned out not to deliver conveniently interpretable results on real-world datasets and were considered generally hard to employ for business users.

Therefore in this work we have proposed a methodology which starts from a BPMN workflow definition with a set of additional template-based constraints and transforms the workflow into a petri net for automatic hot-spot discovery according to rule-conformance, control-flow-conformance and bottleneck detection. The hot-spots in turn are transformed into a classification problem for further mining algorithms which should explain the data-dependencies characterizing the problem. One key differentiator to other approaches is the use of a fuzzy rule induction approach, which delivers globally valid and interpretable rules. Our approach especially aims at providing the corresponding conditions for reuse in adaptation rules which improve the overall workflow performance by circumventing critical situations.

However, some integration steps between the phases of our methodology, like a BPMN to petri net translation suitable for mining purposes, the aggregation of problem situations to hot-spots or the guided parameter selection for the rule mining algorithm remain subject to future work.

8. REFERENCES

[1] Business Process Model and Notation (BPMN) - Version 2.0 11-01-03, 2011.

[2] L. Bodenstaff, A. Wombacher, M. Reichert, and M. C. Jaeger. Monitoring Dependencies for SLAs: The MoDe4SLA Approach. *SCC'08*, pages 21–29, 2008.

[3] M. Castellanos, F. Casati, U. Dayal, and M.-C. Shan. A Comprehensive and Automated Approach to Intelligent Business Processes Execution Analysis. *DAPD*, 16(3):239–273, Nov. 2004.

[4] F. Chesani, E. Lamma, P. Mello, M. Montali, F. Riguzzi, and S. Storari. *Exploiting Inductive Logic Programming Techniques for Declarative Process Mining*, pages 278–295. Springer, 2009.

[5] F. Chesani, P. Mello, M. Montali, F. Riguzzi, and S. Storari. Compliance Checking of Execution Traces to Business Rules. In *BPM'08 Workshops*, pages 129—-140, Milan, 2008. Springer.

[6] W. W. Cohen. Fast Effective Rule Induction. In *ML'95*, pages 115—-123, 1995.

[7] P. Dadam and M. Reichert. The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support. *CSRD*, 23(2):81–97, 2009.

[8] R. M. Dijkman, M. Dumas, and C. Ouyang. Semantics and Analysis of Business Process Models in BPMN. *IST*, 50(12):1281—-1294, 2008.

[9] M. Döhring and B. Zimmermann. vBPMN: Event-Aware Workflow Variants by Weaving BPMN2 and Business Rules. In *EMMSAD'11*, London, 2011. Springer.

[10] M. Döhring, B. Zimmermann, and L. Karg. Flexible Workflows at Design- and Runtime using BPMN2 Adaptation Patterns. In *BIS'11*, Poznan, 2011. Springer.

[11] E. Frank and I. H. Witten. Generating Accurate Rule Sets Without Global Optimization. In *ICML'98*, Madison, 1998.

[12] P. Hornix. Performance Analysis of Business Processes through Process Mining. (January), 2007.

[13] J. Hühn and E. Hüllermeier. FURIA: an algorithm for unordered fuzzy rule induction. *DMKD*, 19(3):293–319, Apr. 2009.

[14] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc, 1993.

[15] A. Rozinat and W. van Der Aalst. Decision mining in business processes, 2006.

[16] A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *IS*, 33(1):64–95, 2008.

[17] W. M. P. van der Aalst, H. T. de Beer, and B. F. van Dongen. Process Mining and Verification of Properties. In *OTM Conferences (1)*, pages 130—-147, Agia Napa, 2005. Springer.

[18] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, and F. Leymann. Monitoring and Analyzing Influential Factors of Business Process Performance. *EDOC'09*, pages 141–150, 2009.

[19] P. Wolf, C., Harmon. The State of Business Process Management 2010, 2010.

IF dockyardStation==*Hamburg* **THEN APPLY** NonFailableTimedHandler(measurements, time=*12h*, handlerTask=*ResetDevices*)

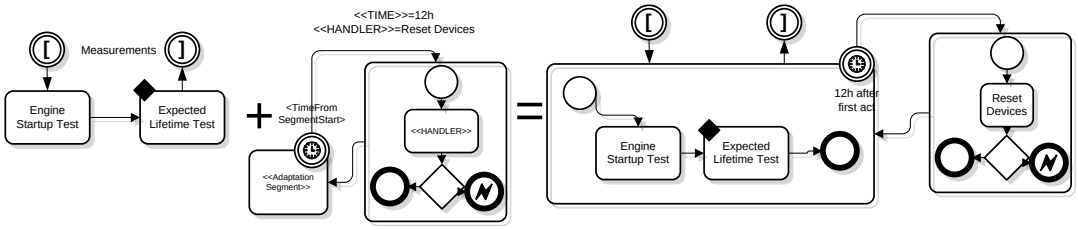


Figure 2: Example of a Rule-Based Workflow Adaptation

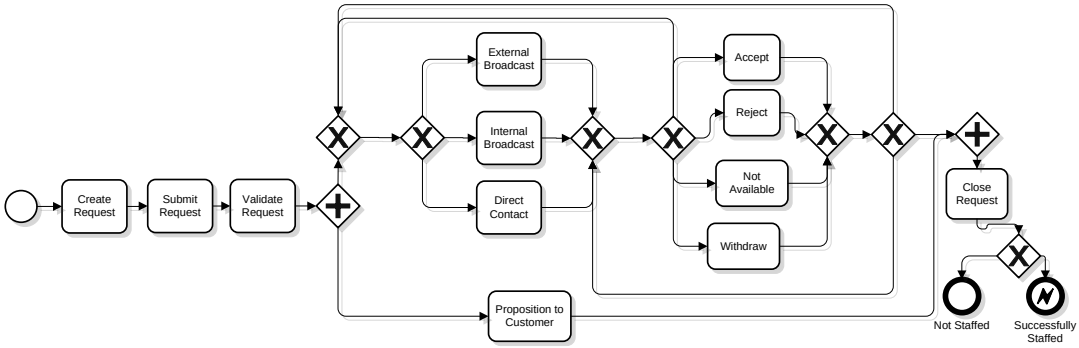


Figure 3: Staffing Workflow of a Large Globally Operating IT Consulting Company

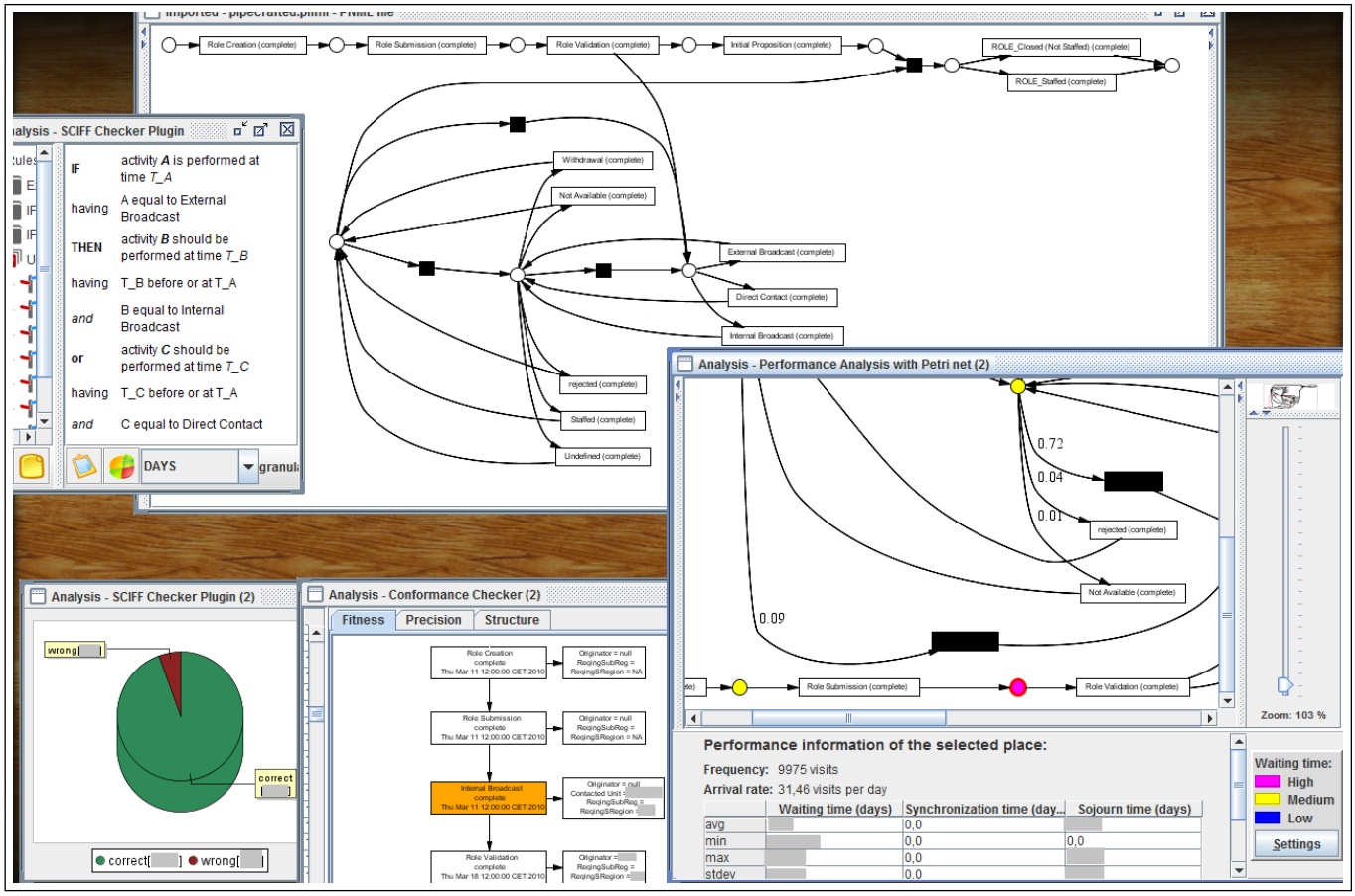


Figure 4: Screenshot of ProM with Most Relevant Workflow Analyses within our Methodology

Vorschlag Hypermodelling: Data Warehousing für Quelltext

Tim Frey
OvG University Magdeburg, Germany
tim.frey@tim-frey.com

ABSTRACT
This paper explains the idea to load source code into a Data Warehouse. First, separation of concerns is explained. Following, the motivation to load source code in a Data Warehouse is briefly presented. Afterwards, the multi-dimensionality of software is discussed. Also, a first model for software in a Data Warehouse is shown. Nearby, the challenge that multiple cubes will be needed in order to load software in a Data Warehouse is elucidated. Thereafter, related work is shown and its relation to the revealed idea is explained. Finally, conclusions are done and future work paths are described.

Categories and Subject Descriptors
D.2.3 [Software Engineering]: Coding Tools and Techniques; D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement

General Terms
Human Factors

Keywords
Separation of Concerns, OLAP

1. EINLEITUNG
Bei der Softwareentwicklung ist das Prinzip der Separation of Concerns (deutsch: Trennung der Belange, kurz SOC) etabliert. Dieses Prinzip entspricht der Handlungsweise, ein Softwaresystem aus verschiedenen Blickwinkeln zu betrachten und die Codierung für jeden Blickwinkel, in einzelnen Modulen zu erstellen [1]. Diese Blickwinkel werden oftmals auch Belang oder Concern genannt. Verschiedene Programmiersprachen stellen dabei verschieden mächtige Mechanismen zur Verfügung, um Module für die einzelnen Blickwinkel zu erstellen. Durch die Modularisierung wird eine erhöhte Wiederverwendbarkeit und Verständlichkeit erreicht. Trotz der Anwendung von SOC ist die Erstellung und die Untersuchung von Software eine große Herausforderung. Ferner ist es nicht einfach möglich, einzelne Concerns in Module zu kapseln, weil eine "absolute" Modularität mit gängigen eingesetzten Programmierparadigmen nicht möglich ist [2,4]. So ist es zum Beispiel bei der Kodierung von Problemstellungen durch das objektorientierte Paradigma nicht möglich, alle Concerns in einzelne Module zu verpacken. Ein Modul ist oftmals für verschiedene Zuständigkeiten zugleich programmiert [2, 3, 4]. Folglich sind verschiedene Concerns in Modulen vermischt und werden nicht in einzelnen Modulen, wie es eigentlich die Idee von SOC ist, kodiert. Um die Analyse von Software im Zusammenhang mit Concerns zu ermöglichen, muss folglich ein Analyseverfahren den Umstand der Vermischung von Concerns in Modulen beachten. Der Beitrag dieses Papiers ist daher die Idee, mehrdimensionale Analyseverfahren, die solche Umstände beachten können, aus dem betriebswirtschaftlichen Kontext für Quelltext einzusetzen. Es werden dabei erste Überlegungen zu deren Einsatz präsentiert. Das Fernziel des Einsatzes dieser Analyseverfahren ist es, Data Warehouse Technologie als Werkzeug zur Quelltextanalyse nutzen zu können.[1]
Im Folgenden wird zuerst die Motivation und Vision, Quelltext in ein Data Warehouse zu laden, beschrieben. Danach wird die Mehrdimensionalität von Software erläutert, um den Bezug zu mehrdimensionalen Daten im Data Warehouse zu geben. Im Anschluss werden erste relationale Schemata präsentiert und diskutiert, die Quelltext in einem Data Warehouse abbilden können. Durch deren Darstellungen werden weitere Anforderungen für zukünftige Schemata aufgedeckt. Nachfolgend werden verwandte Arbeiten, im Vergleich zu der in diesem Papier vorgeschlagenen Idee, erläutert. Am Ende werden Rückschlüsse und weitere Arbeitspfade beschrieben.

GvD Workshop'11, 31.05.-03.06.2011, Obergurgl,Österreich.

2. MOTIVATION UND VISION
Im betriebswirtschaftlichen Bereich existiert eine Vielzahl von Systemen, deren Daten in einem Data Warehouse zusammengefasst und homogenisiert werden. Dies ermöglicht es, diese zu aggregieren, Mining zu betreiben und strategische Entscheidungen zu treffen [5]. Des Weiteren werden Data Warehouses zur integrierten Unternehmensplanung verwendet. Dabei werden Planziele in einem Data Warehouse hinterlegt [6].[2]

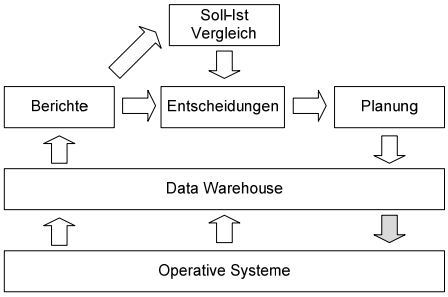


Abbildung 1: Prozesse mit einem Data Warehouse

Dargestellt ist dies in Abbildung 1. Daten werden aus operativen Systemen in ein Data Warehouse geladen. Diese Daten

[1] Aufgrund des Bezuges des Papieres auf SOC und Data Warehouse Technologien wird im weiteren Verlauf von einer Vertrautheit mit SOC [1] und den hierbei auftretenden Problemen [2,4], Frameworks [8,9], aspekt-orientierter Programmierung [12], domänen-spezifischen Sprachen [11], Annotationen [10] und Grundkenntnissen im Data Warehouse Bereich [5,7] ausgegangen.

[2] Eine integrierte Unternehmensplanung ist die Hinterlegung von Planzielen in einem Data Warehouse. In diesem werden hierbei Merkmalkombinationen in Verbindung mit Kennzahlen gespeichert, wobei zumindest ein Merkmal ein zukünftiges Zeitdatum darstellt. Der Zusammenhang zwischen strategischen Entscheidungen und Planung ist, dass strategische Entscheidungen zu Planzielen führen können.

werden dann genutzt, um Berichte zu erzeugen. Aufgrund dieser Berichte werden Entscheidungen getroffen. Diese führen zu einer Planung für die Zukunft. Der Plan wird dann als Plandaten im Data Warehouse gespeichert. Dabei können die Plandaten auch dazu führen, dass automatisiert in die operativen Systeme geschrieben wird (grauer Pfeil). Durch Planung und deren Umsetzung entstehen neue Daten in den operativen Systemen. Diese werden dann wieder in das Data Warehouse geladen und können mit den Plandaten verglichen werden. Das führt zu Entscheidungen und gegebenenfalls zu einem neuen Plan [7].
Bei der Softwareentwicklung fallen ebenfalls Artefakte in einer Vielzahl von Systemen an, und bei vielen Projekten ist die Quelltextbasis zu groß, um einfach überblickt oder analysiert zu werden. Die Vision ist daher eine integrierte, homogenisierte Sicht auf Software als multidimensionales Gebilde, durch die die verschiedenen Artefakte, die im Produktlebenszyklus auftreten, abgebildet werden können. Der Kern der Idee ist, dass Software nicht nur aus der Applikationslogik besteht, sondern aus einer Vielzahl weiterer Artefakte, wie zum Beispiel Tests. Eine multidimensionale Anordnung im Data Warehouse könnte eine integrative Komponente sein, die es erlaubt, die Software aus verschiedenen Blickwinkeln zu betrachten. Desweiteren würde es der Einsatz dieser Technologie zulassen, außer der Analyse weitere Anwendungen möglich zu machen. Der Planungsprozess in Data Warehouses, der zur integrierten Planung verwendet wird, könnte dazu dienen, Qualitätskriterien zu definieren, oder auch Weiterentwicklungen von Software zu planen und diese dann mit der erfolgten Realität zu vergleichen.
Ziel dieser Arbeit ist es daher, erste Möglichkeiten des Speicherns von Quelltext in einem Data Warehouse zu untersuchen. Diese erste Evaluation soll Hinweise über die Realisierbarkeit des Einsatzes von Data Warehouses zur Quelltextuntersuchung liefern.

3. MEHRDIMENSIONALE SOFTWARE

Software wird normalerweise unter der Nutzung von Frameworks, die vorgefertigte Funktionalität bereitstellen, erstellt. [8,9]. Software kann als ähnlich zu einem n-dimensionalen Hyperwürfel betrachtet werden. Die Ecken bzw. Kanten des Würfels sind Frameworkfunktionen. Durch diese Anschauung ist es nötig, Software nicht nur aus einer Position zu betrachten. Die Ansicht ist ähnlich zu einem Hyperwürfel und kann gedreht werden. Eine Drehung kann hierbei verschiedene Dimensionen in den Vordergrund verschieben und andere im Hintergrund verschwinden lassen. Dieser Vergleich zeigt, dass der Mensch nicht dazu fähig ist Software, die ähnlich einem Hyperwürfel ist, direkt und vollständig zu erfassen. Darstellungen sind vielmehr Projektionen in den Verständnisraum des Menschen.
Die Analogie zu einem Hyperwürfel kann auf dessen Konstruktion reduziert werden. Dabei kann ein Würfel gesehen werden, der durch weitere Dimensionen erweitert wird. Im Falle von Software wird ein Würfel durch Quelltext "befüllt", der sich im vorgegebenen Rahmen eingliedert. Dies bedeutet, dass Quelltextmodule aufgrund der Nutzung verschiedener Frameworkfunktionen verschiedenen Dimensionen zugehörig sind. Diese Zugehörigkeit ergibt sich daraus, dass die einzelnen Concerns nicht in Modulen getrennt sind, und somit ein Modul durch die Nutzung verschiedener Funktionen verschiedenen Dimensionen gleichzeitig zugehörig sein kann.
Beim Erstellen von Software werden Funktionen, die im „aktuell erstellten“ Quelltext selbst sind und nicht aus Frameworks stammen, auch genutzt. Der Programmierer erzeugt sich ein eigenes Framework für seine spezielle Aufgabe. Dieses setzt auf bestehende Funktionen von Frameworks auf oder definiert vollkommen neue Funktionen. Somit wird ein Teil des Würfelinhaltes zu neuen Dimensionen. Diese befinden sich in dem Würfelrahmen und erzeugen hierin spezialisierte Unterräume oder sie spannen Dimensionen auf, die orthogonal zu den bisherigen Dimensionen stehen. Dabei rücken diese neuen Dimensionen in den Vordergrund und bisherige Dimensionen werden verdeckt, was einer Drehung des Würfels ähnlich ist.
In Abbildung 2 wird Quelltext, der selbst in eine Frameworkdimension aufsteigt, grafisch visualisiert. Dabei sind zuerst zwei Frameworks A und B zu sehen. Der Quelltext konsumiert die Fähigkeiten eines Frameworks A und eines anderen B, womit eine Komposition von beiden Frameworks erreicht wird. Ein solches Konsumieren kann z.B. ein Funktionsaufruf oder jedes andere Nutzen der Fähigkeiten eines Frameworks sein. Die Pfeile zeigen das Konsumieren von Fähigkeiten an. Aufgrund dessen, dass neuer Quelltext den bestehenden Quelltext nutzt, steigt der zuvor erzeugte Quelltext im unteren Teil der Grafik, selbst zur eigenständigen Dimension auf. Diese wird dann von neuem Quelltext konsumiert. Dabei kann es auch sein, dass der New Code nicht mehr das originale Framework B nutzt, sondern nur den zuvor erzeugten Code und Framework A, was durch die Schraffierung angedeutet ist.
Die Idee ist, dass Frameworkfunktionen oftmals Concerns repräsentieren. Somit können die verschiedenen Concerns der Software mit Dimensionen gleichgesetzt werden.

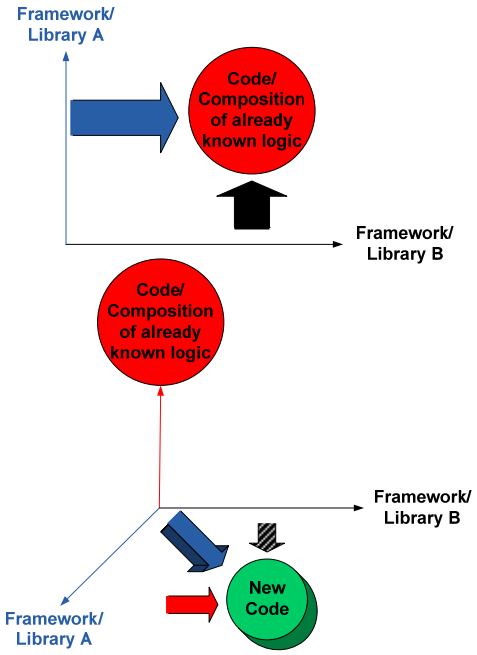


Abbildung 2: Quelltext, der zur Dimension aufsteigt

Daraus folgt eine abstrahierte Darstellung des Sachverhaltes von Abbildung 2 in Abbildung 3. Hierbei wurden die Dimensionen des Quelltextes mit C für Concern gekennzeichnet. Die Komposition durch Quelltext von verschiedenen Concerns ist durch den Verbindungsvektor VC1 dargestellt. In Abbildung 3b wird diese Dimension, beziehungsweise dieser Concern, selbst wiederum konsumiert (VC2). Der gestrichelte VC1 Vektor und dessen Parallelverschiebung zeigt, dass die neue Dimension komplett gleichberechtigt zu den „normalen“ C Dimensionen ist.

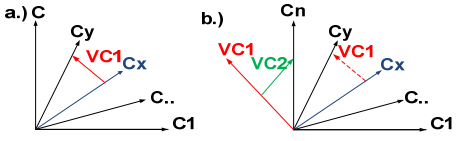


Abbildung 3: Quelltext der zur Dimension aufsteigt

Ein weiterer Umstand, den es zu beachten gilt, ist die Anordnung der Concerns im Falle von Quelltext. Dieser kann, wie beschrieben mehreren Concerns zugehörig sein. Diese können

sich zudem in verschiedenen Hierarchien befinden. Beispielsweise könnte eine Hierarchie VC2 zu VC1 und Cn sein. VC1 wiederum ist die Wurzel der Cx, Cy Hierarchie. Abstrakt dargestellt ist dies in Abbildung 4a. Dort wird die Hierarchie visualisiert, in der sich VC2 befindet. VC2 bildet dabei den Ursprung von dem aus die Hierarchie aufgespannt wird. In Abbildung 4b wird der Sachverhalt verallgemeinert gezeigt, dass Dimensionen/Concerns auch hierarchisch angeordnet sein können. Verschiedene Tiefenebenen sind als Dimensionsebenen (DE1- DEn) beschriftet.[3] Die Wurzel der Hierarchien bildet dabei das Fragment, in dem die einzelnen Concerns komponiert wurden. Ein praktisches Beispiel für Hierarchien sind Methodenaufrufe oder Vererbung.

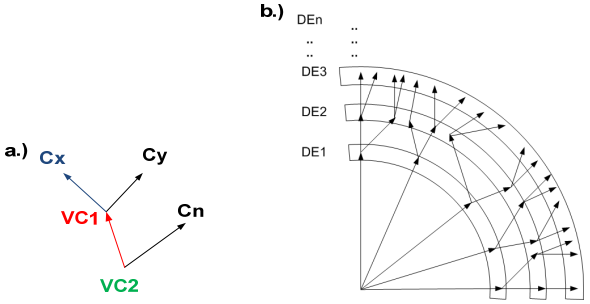


Abbildung 4: Dimensionshierarchien

Da in modernem Quelltext oftmals eine hohe Anzahl von Frameworks verwendet wird, stellt sich die Herausforderung, trotz der Analogie zwischen Frameworkfunktionen und Dimensionen, diese bei einer Umsetzung in einem Schema nicht vollständig gleichzusetzen. Dies hat den Grund, dass oftmals eine Vielzahl von Frameworks in Quelltext verwendet wird und somit eine hohe Anzahl verschiedener Dimensionen wahrscheinlich die Skalierbarkeit der Umsetzung beeinflussen würde. Folglich sollten Frameworkfunktionen als eine Dimension mit verschiedenen Elementen, den eigentlichen Funktionen der Frameworks, modelliert werden.
Ziel ist ein Modell, welches die zuvor genannten Sachverhalte in einem Data Warehouse abbildet. Dies kann genutzt werden, um Abfragen auf Quelltext zu ermöglichen. Aufgrund der Mehrdimensionalität des Ansatzes und dem Hintergrund, dass bei der Programmierung Modelle die Realität abbilden, ist der Name der Kombination, zwischen Data Warehouse und Softwarequelltext, Hypermodelling.

4. MODELLIERUNG

Data Warehouses nutzen im Normalfall eine relationale Datenquelle, um OLAP-Würfel zu füllen. Daher ist der erste Schritt ein relationales Schema, das Quelltext in einem Data Warehouse abbildet. In Data Warehouses werden zu der Abbildung von Daten in relationalen Tabellen im Normalfall Stern- oder Schneeflockenschemata verwendet [31]. Nachfolgend werden zwei relationale Modelle, inspiriert von diesen, gezeigt. Zuletzt wird die OLAP Darstellung angerissen und Problematiken des entwickelten Schemas aufgezeigt.

4.1.1 Einfaches Schema

In Abbildung 5 ist exemplarisch eine Tabellenstruktur mit Fragmenten, die in Quelltext vorkommen, an ein Sternschema angelehnt, dargestellt. In der Abbildung ist das zentrale Element die Faktentabelle. Eine Zeile in dieser repräsentiert ein Quelltextfragment, wie ein Modul. Ein solches ist zum Beispiel eine Klasse oder Funktion. Eine Klasse kann Annotationen enthalten. Oftmals werden Annotationen in Frameworks definiert und daher ist die Annotationstabelle mit der Frameworktabelle verknüpft. Die Parameter Tabelle zeigt die Möglichkeit, dass Annotationen auch Parameter besitzen können. Zusätzlich sind oftmals externe domänenspezifische Sprachen (domain specific language (DSL)) mit Quelltext assoziiert. Durch diese können zu Quelltextfragmenten verschiedene Funktionalitäten hinzu konfiguriert werden [11]. Ein bekanntes Beispiel hierfür ist die Konfiguration persistenter Klassen über eine DSL. In modernem Quelltext treten auch Aspekte [12] auf. Aspekte sind Fragmente die es ermöglichen, Funktionalität, die nach dem Objektparadigma nicht an einer Stelle kodiert werden kann, an einem zentralen Punkt zu realisieren. In diesem wird dann angegeben welche Module von dem Aspektcode (Advice) betroffen sind. Die Konfiguration der betroffenen Module erfolgt über sogenannte pointcuts. Zuletzt kann ein Modul einer Komponente und Aufgaben zugordnet werden. Beispielsweise können solche Aufgaben das Einpflegen von Änderungen in der Funktionalität darstellen.

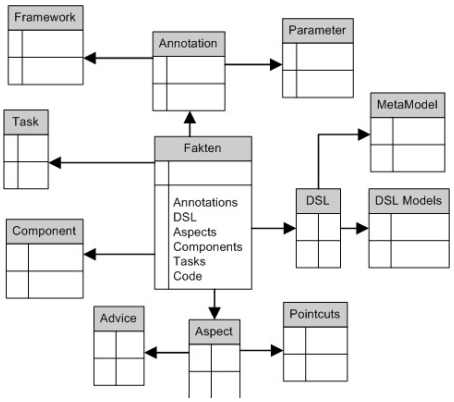


Abbildung 5: Grundlegendes relationales Quelltextschema

4.1.2 Schneeflockenschema

Das in Abschnitt 4.1.1 dargestellte Schema wurde in Abbildung 6, an ein Schneeflockenschema angelehnt, erweitert. Eine generische Zuordnung von Modulen zu Concerns wird besser ermöglicht und mehr Sachverhalte können dargestellt werden. Ebenfalls wird der Umstand beachtet, dass eine generische Darstellung der Beziehungen im Quelltext aus Abschnitt 3 dargestellt wird und verschiedene Frameworks nicht als eigenständige Tabelle realisiert sind. Vielmehr sind die Frameworkfunktionen in einer Tabelle zusammengefasst.
Wie zu sehen ist, verweist die Faktentabelle auf verschiedene Concerns, zu denen die Zugehörigkeit durch Cmembership in Prozent ausgedrückt werden kann. Als Modulgranularität wurde eine Funktion gewählt. Diese stellt eine kleine Einheit klarer Funktionalität dar und gliedert sich in eine Klasse ein. Folglich ist Aggregation von Fakten auf Klassenebene möglich. Eine Funktion kann andere Funktionen nutzen (Function_call). Hierbei kann spezifiziert werden, ob der Konsum ein Aufruf oder ein andere Nutzungsart (Usage type), wie zum Beispiel das Überschreiben einer Funktion eines Frameworks ist. Ebenfalls können Typen eines Frameworks, wie zum Beispiel durch Annotationen oder Erbschaftbeziehungen, konsumiert werden (Type_usage).
Rückverweise auf die Faktentabelle zeigen, dass Hierarchien von Funktionen möglich sind. Die Aufrufhierarchie wird durch die CallHierarchy Table dargestellt. Die EvolvedConcernCode Tabelle ermöglicht es, Funktionen/Fakten selbst als Concern zu führen. Durch diese Tabelle kann Quelltext, der sich zum Concern entwickelt (siehe Abschnitt 3), aber kein Teil eines Frameworks ist, dargestellt werden.

[3] Die Ebenen sind rein zur besseren Visualisierung dargestellt; nicht jede Dimension muss über die gleiche Anzahl von Hierarchien verfügen.

Besonders interessant ist die Issue Tabelle, die Bugs darstellt. Diese können auf zugehörige Tests verweisen. Dabei können auch Tasks oder deren Kontext mit solchen Issues verbunden sein. Ebenfalls wurde der Umstand ins Modell eingeflochten, dass Packages oftmals Layer zugeordnet werden können. Metrics, Profiling info und Author zeigen, dass weitere Informationen im Bezug auf Quelltext dargestellt werden können.

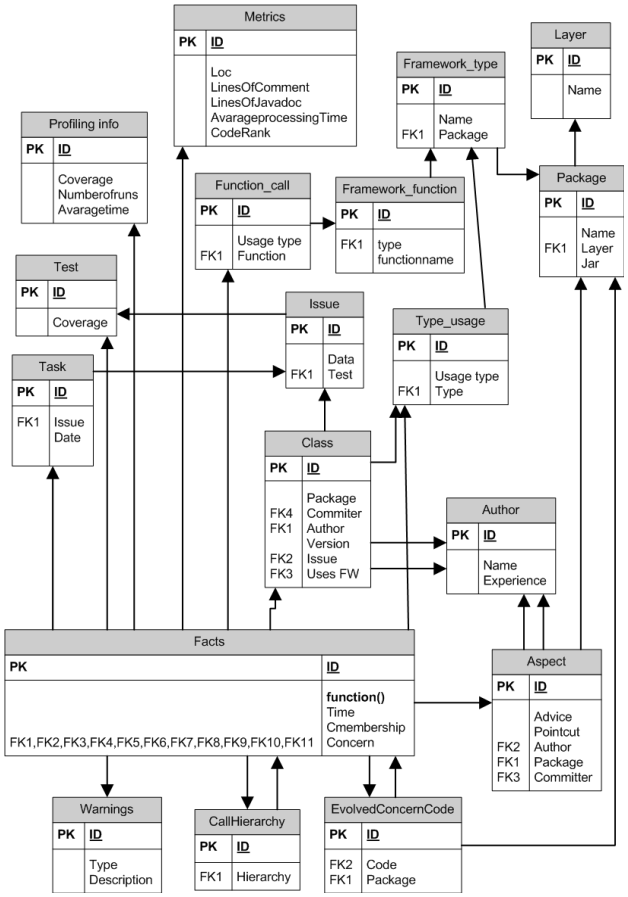


Abbildung 6: Quelltext, Schneeflockenschema

4.1.3 OLAP-Darstellung

Grundlegend zeigt das Schema aus Abschnitt 4.1.2 bei näherer Betrachtung die Einschränkung, dass eine direkte Umsetzung in einen OLAP-Würfel keine elegante Lösung darstellt. Insbesondere die rückverweisende CallHierarchy besitzt Eigenschafen einer Faktentabelle; CallHierarchy kann mehrmals gleiche Einträge besitzen, wenn eine Funktion mehrmals die gleiche Funktion aufruft. Zur Realisierung sollte somit mit der Sichtweise einer zentralen Faktentabelle gebrochen werden.

In Abbildung 7 ist aufgrund der zuvor genannten Einschränkungen daher die Idee, mehrere OLAP-Würfel zu erstellen, die einen Bezug zu den Daten eines konkreten Programmes, das analysiert werden soll, haben. Ziel ist es einen OLAP-Würfel, ähnlich A, zu erstellen, der Zuordnungen mit einem logischen Modell einer Programmiersprache enthält. Hierbei kann dieser OLAP-Würfel durch zusätzliche Daten, die im direkten Zusammenhang zu der Programmiersprachenlogik, stehen angereichert werden. Solche Daten könnten zum Beispiel Tests oder deren Ergebnisse sein. Dieser OLAP-Würfel stellt ein Abbild der Programmiersprachenstruktur dar, welches durch ein konkretes Program als Fakten „befüllt“ wird.

In einem weiteren OLAP-Würfel werden dann Fakten, die nicht der Programmiersprachenstruktur entsprechen, definiert. Jedoch besitzen die Daten zumindest einen Bezug zu dem Quelltextmodel. Im Vergleich zu OLAP-Würfel A ist OLAP-Würfel B somit keine Repräsentation der Programmstruktur, sondern vielmehr die Assoziation von Programmelementen miteinander. Ein Beispiel für die Assoziation von Programmelementen miteinander ist die zuvor beschriebene Aufrufhierarchie. Da somit beide OLAP-Würfel eine Assoziation zu dem Quelltext besitzen, können diese dann in einem Verknüpften OLAP-Würfel zusammengeführt werden. In diesem Linked Cube werden dann zusätzliche Fakten, mit denen des logischen Modells einer Programmiersprache zusammen geführt.

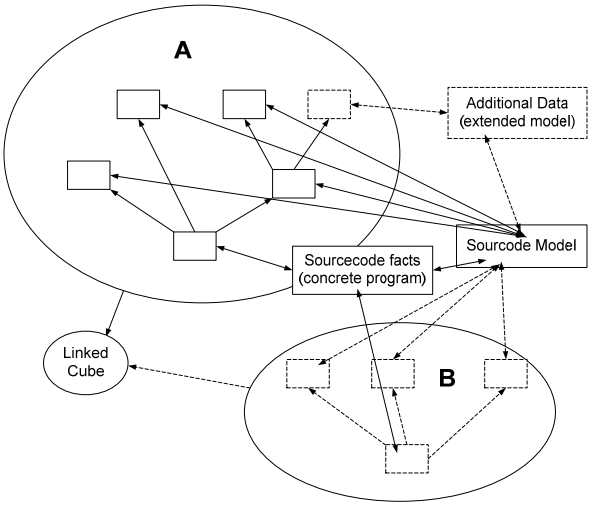


Abbildung 7: OLAP-Würfel, Zuordnung von Quelltext

Schlussendlich zeigt sich, dass für die Entwicklung eines relationalen Schemas, zuerst Fakten von der Struktur von Quelltext getrennt werden sollten. Danach können die verschiedenen Fakten in einem weiteren Modell zusammengeführt werden.

5. VERWANDTE ARBEITEN

Auf der Homepage[4] des Hypermodelling Projektes befinden sich weitere Informationen über die Idee, Quelltext in ein Data Warehouse zu laden und verwandte Forschung.

Die Gruppe der Source Code Query Tools stellt im Wesentlichen Programme dar, die Quelltext aufgrund von Abfragen untersuchen können. Diese Werkzeuge können als verwandte Arbeiten zu der Idee von Hypermodelling gesehen werden, da diesen zu Grunde liegt, Quelltext, aufgrund von Abfragen zu untersuchen. JQuery ist ein Abfrage basierter Source Code Browser für Java. Realisiert ist er als Eclipse Plugin [13]. JQuery stellt eine deklarative Abfragesprache zur Verfügung und erlaubt es, durch diese verschiedene Ansichten auf Quelltext zu erzeugen [14]. Ferret ist ein weiteres Source Code Query Tool und wurde von Brian de Alwis im Rahmen seiner Dissertation entwickelt [15, 16]. Es erlaubt es ähnlich JQuery, Abfragen an Java Quelltext zu erstellen. Ferret ist hauptsächlich dafür gedacht, den Quelltext aufgrund seiner Aufrufstruktur und Vererbung zu untersuchen. Lost ist ein Query Tool für aspekt-orientierte Programmierung [17]. Grundlegend verfolgen alle Source Code Query Tools das gleiche Ziel und besitzen ähnliche Funktionalität. JQuery erscheint von allen Tools das am besten gepflegteste und mächtigste zu sein und kann somit als bestes Vergleichsprodukt zu Hypermodelling gesehen werden. Im Vergleich zu Hypermodelling unterscheiden sich die Source Code Query Tools darin, dass diese nicht das Ziel verfolgen, Data Warehouse Technologie zur Analyse von Quelltext zu nutzen. Ebenfalls orientiert sich ihre Funktionsweise oder die Abfragemöglichkeiten nicht an Data Warehouse Technologie.

[4] http://hypermodelling.com

Martin Robillard forscht auf dem Gebiet der Concernanalyse. Sein Werkzeug, ConcernMapper [18], soll helfen, Quelltext nach verschiedenen Concerns zu unterteilen und ist auf manuelle Zuordung von Concerns zu Quelltextfragmenten ausgelegt. Der Unterschied zu den Source Code Query Tools ist, dass der Concern Mapper keine Abfragesprache oder ähnliche Mittel nützt, sondern dieser auf die rein manuelle Zuordnung von Concerns ausgelegt ist. Der größte Nachteil im Vergleich zu Hypermodelling ist, dass das Anlegen und Zuordnen der Concerns eine Mehrarbeit des Entwicklers bedeutet und nicht das Ziel verfolgt wird, Abfragen aufgrund dem Entwickler bekannter Quelltextelemente zu ermöglichen.

In [19] wird ein Verfahren beschrieben, bei dem ein Metrics Warehouse eingesetzt wird. Dieses erinnert aufgrund des Warehouses entfernt an die Idee des Hypmermodelling. Dabei wird bei dem Verfahren beschrieben, wie Projekte anhand von Metriken, gesteuert werden können. Dies bezieht sich auf Metriken aus betriebswirtschaftlichen Systemen und nicht direkt auf Metriken, die durch eine Funktion aus Quelltext berechnet werden können. Diese Metriken werden dann in ein nicht näher erläutertes Metrics Warehouse geladen, das einem Data Warehouse ähnlich ist. Der genaue Aufbau des Metrics Warehouses, die Struktur der Daten, und der Inhalt des Warehouses wird nicht näher beschrieben.

Source Code Mining ist die Anwendung von Data Mining Algorithmen auf die Quelltext, Bug Datenbanken und Versionsverwaltungssysteme [20]. Somit stellt diese Technik auch die Zielsetzung auf, Quelltext zu analysieren und dadurch Verbesserungen zu erreichen, und kann somit als ähnlich erachtet werden. Die Hauptidee hierbei ist es, Bugs aus Issue Tracking Systemen, mit deren Lösungen aus Versionsverwaltungssystemen, aufgrund der entsprechenden Quelltexte zu verknüpfen und darauf Data Mining zur Analyse zu nutzen. Muster im Quelltext werden aufgedeckt, interpretiert und Handlungen abgeleitet. Solche Analysen können beispielsweise zu Feststellungen führen, dass Problemlösungen, die an einem bestimmten Wochentag gemacht wurden, mit einer hohen Wahrscheinlichkeit wieder zurückgenommen werden [21]. Das Source Code Mining besitzt den Nachteil, dass derzeit nur Zusammenhänge zwischen Fehlern und Beziehungen zwischen einzelnen Quelltextelementen im Source Code aufgedeckt werden können. Es wird dabei kein Verfahren offenbart, wie weitere Daten in die Wissensbasis integriert werden können. Des Weiteren verfolgt Source Code Mining im Vergleich zu Hypermodelling nicht das Ziel, eine Datenbasis in Form des Ladens von Daten in ein Data Warehouse zu erschaffen, auf welcher dann Analysen ausgeführt werden können.

In [22] wird Software Intelligence beschrieben. Software Intelligence wird hier als die Anwendung von Business Intelligence Mechanismen oder Technologien auf Software beschreiben. Dies ist der hier vorgestellten Idee ähnlich. Der Unterschied liegt darin, dass Software Intelligence nicht das Ziel verfolgt, Quelltext in OLAP-Würfel zu laden und sich auf Source Code Mining konzentriert.

Die Untersuchung von Quelltext unter der Zuhilfenahme einer relationalen Datenbank zur Analyse wird im Sourcerer Projekt durchgeführt [23]. Das relationale Modell umfasst vier Tabellen. Quelltext wird in Projekte, Dateien, Kommentare und Entitäten die mit Beziehungen verknüpft sind, aufgeteilt. Solche Entitäten sind zum Beispiel Klassen, Interfaces und Methoden [24]. Im Vergleich zu Hypermodelling zeigt der Umfang des relationalen Modells einen wesentlichen Unterschied; Hypermodelling visualisiert schon in den ersten vorgestellten Modellen die verschiedenen Beziehungen in unterschiedlichen Tabellen. Zudem ist das Ziel von Hypermodelling, Data Warehouse und insbesondere OLAP-Würfel zu verwenden. Dies ermöglicht Aggregationen von Fakten, was bei Sourcerer nicht erstrebt wird. Dies ist zudem auch darin zu sehen, dass Sourcerer keine Faktentabellen nutzt, um Beziehungen im Quelltext abzubilden.

Codegenie, dass auf Sourcerer aufsetzt, ermöglicht es, Komponenten aufgrund von Testfällen aufzudecken und in ein Programm zu integrieren [25, 26]. Ähnlich zu Codegenie zeigt [27] weitere Möglichkeiten der Komponentenaufdeckung, unter der Zuhilfenahme von Codesuchmaschinen und gibt einen Überblick und Vergleich über die Aufdeckung von Komponenten. Im Vergleich zu diesen Codesuchmaschinen liegt der Fokus von Hypermodelling auf der Analyse und Abfragen aufgrund von Concerns. Eine Verwendung der Hypermodelling Idee zur Codesuche könnte jedoch eine interessante Anwendung sein.

Orthographic Software Modeling (OSM) ist der Ansatz, mehrdimensionale Navigation durch Modelle bei der Softwareentwicklung einzusetzen [30]. Ziel ist es, Quelltext und die zugehörigen Modelle in einem zentralen Modell abzubilden um daraus die verschiedenen Modellansichten dynamisch erzeugen zu können [28]. Um die Navigation durch die verschiedenen Modellansichten zu ermöglichen, werden verschiedene Dimensionen und deren Ausprägungen definiert. Zum Beispiel eine Dimension die in ihren Ausprägungen die Abstraktionsebenen festlegt oder eine Projektionsdimension, deren Ausprägung die Struktur oder die Interaktion von Elementen beschreibt. Die Auswahl einer Kombination von Ausprägungen von Dimensionen bestimmt letztendlich dann, welches Modell gezeigt wird [29]. Die mehrdimensionale Navigation und die Sichtweise von Software als ein mehrdimensionales Konstrukt, das von verschiedenen Blickwinken aus betrachtet werden kann, ähnelt Hypermodelling. Jedoch verfolgt OSM nicht das Ziel, Data Warehouse ähnliche Mechanismen zu nutzen, um dadurch Quelltext untersuchen zu können. Ferner liegen die Ziele von Hypermodelling nicht direkt in der Modellierung von Software, sondern vielmehr darin, Abfragen unter der Nutzung verschiedener Concerns zu ermöglichen. In diesem Kontext könnte natürlich auch die dynamische Erstellung von Ansichten bei OSM als Abfragen verstanden werden, und es könnte interessant sein, eine Kombination zwischen OSM und Hypermodelling zu untersuchen.

Bezogen auf die Data Warehouse Schemata zeigen [5] und [31] einen umfassenden Überblick über das Thema Data Warehouse. Hier werden die verschiedenen Schemata besprochen und praktische Beispiele aufgezeigt. Business Intelligence, wie auch die integrierte strategische Unternehmensplanung im speziellen Anwendungsfall, wird in [7] beschreiben. Ein betriebswirtschaftlich- orientierter Überblick der integrierten Unternehmensplanung stellt [6] dar. Einen herstellerneutralen Überblick über die Planung mit Data Warehouse Systemen bietet [32].

6. ZUSAMMENFASSUNG UND AUSBLICK

In diesem Papier wurde die Idee, Quelltext in ein Data Warehouse zu laden, vorgestellt. Hierbei wurde beschrieben, dass Quelltext ein mehrdimensionales Gebilde aus Concerns ist. Dabei wurde insbesondere der Umstand hervorgehoben, dass Module oftmals mehreren Concerns zugeordnet werden können. Es wurde erläutert, dass in diesem mehrdimensionalen Gebilde neue Dimensionen durch Kompositionen erschaffen werden können. Dieser Umstand zeigt eine besondere Herausforderung, im Bezug auf das Laden von Quelltext in ein Data Warehouse. Unter Beachtung dieser Rahmenbedingung wurden relationale Modelle mit Faktentabellen präsentiert, in die Quelltext geladen werden kann. Durch deren Darstellung konnte die Problematik aufgedeckt und verdeutlicht werden: Eine einzige Faktentabelle ist zur Darstellung von Quelltext nicht ausreichend. Zuletzt wurden verwandte Arbeiten präsentiert, deren Analyse zeigt, dass die Idee, Quelltext in ein Data Warehouse zu laden, bisher noch nicht durchgeführt wurde. Dennoch deu-

ten die verwandten Arbeiten darauf hin, dass der primäre Einsatzort für Abfragewerkzeuge derzeit die integrierte Entwicklungsumgebung ist.

Eine wichtige Folgerung aus den Darstellungen dieses Papiers ist, dass bei der Erstellung eines weiteren Schemas, Quelltext in verschiedene Fakten aufgeteilt werden sollte. Dies wird benötigt, um ein relationales Modell zu erstellen. Somit stellt sich für zukünftige Arbeiten die Frage, welche Mittel einer Programmiersprache als strukturell und welche als assoziativ zueinander begriffen werden können. Eine derzeitige Überlegung ist daher Quelltext als Mechanismen von Kategorisierung (strukturell) und Komposition (assoziativ) zu betrachten. Kategorisierung ist hierbei alles, das einer logischen Zuordnung dient. Zum Beispiel Klassen zu Packages. Komposition stellen sämtliche Elemente dar, die eine direkte Auswirkung auf die Funktionalität ausüben, wie Methodenaufrufe. Dennoch sind hier Diskussionspunkte offen. So ist es zum Beispiel fragwürdig, ob Ableitungen eher eine Kategorisierung oder eine Komposition darstellen. Ziel dieser Überlegungen ist es Quelltext in OLAP-Würfel zu laden, um dadurch komplexe Analysen von Quelltext zu ermöglichen.

Eine weitere Überlegung kommt aus der Betrachtung der verwandten Arbeiten, die Großteils für Eclipse umgesetzt wurden. Diese inspirieren dazu, OLAP ähnliche Abfragen direkt in der integrierten Entwicklungsumgebung zu ermöglichen. Dies kann ein besonders großer Vorteil für Entwickler sein, da diese mit solchen Hypermodelling Abfragen direkt in der Entwicklungsumgebung Quelltext untersuchen können.

7. LITERATUR

[1] E. W. Dijkstra. Selected Writings on Computing: A Personal Perspective. On the role of scientific thought. Springer.1982

[2] W.Harrison, H. Ossher. Subject-oriented programming: a critique of pure objects. OOPSLA '93. ACM. 1993

[3] Bernhard Lahres, Gregor Rayman. Objektorientierte Programmierung. Galileo Computing. 2009

[4] H. Ossher, P. Tarr. Multi-dimensional separation of concerns and the hyperspace approach. In Proceedings of the Symposium on Software Architectures and Component Technology. Kluwer. 2001.

[5] W.H. Inmon. Building the Data Warehouse. 4th ed., J.Wiley & Sons, New York. 2005.

[6] M. C. Meier, W. Sinzig, P. Mertens. Enterprise Management with SAP SEM/Business Analytics. 2nd Edition, Springer. Berlin. 2005

[7] J. M. Gómez, C. Rautenstrauch, P. Cissek. Einführung in Business Intelligence mit SAP NetWeaver 7.0. Springer.2008

[8] S. H. Kaisler. Software paradigms. John Wiley and Sons.2005

[9] W. Pree. Meta Patterns - A Means For Capturing the Essentials of Reusable Object-Oriented Design. ECOOP 94. Springer. 1994

[10] J. A. Bloch. Metadata Facility for the Java Programming Language. 2004. http://www.jcp.org/en/jsr/detail?id=175

[11] T. Stahl, M. Völter, S. Efftinge, A. Haase. Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management. Dpunkt Verlag.2007

[12] R. E. Filman, T. Elrad, S. Clarke, M. Aksit Aspect-Oriented Software Development. Addison-Wesley Professional; 1 edition. 2004

[13] JQuery a query-based code browser. homepage. The University of British ColumbiaVancouver, Canada. http://jquery.cs.ubc.ca/index.htm.

[14] K. De Volder. JQuery: A Generic Code Browser with a Declarative Configuration Language. In Practical Aspects of Declarative Languages, 8th International Symposium. Springer. 2006.

[15] B. de Alwis, G. C Murphy. Ferret: A Tool Supporting Software Exploration. The University of British Columbia Vancouver, Canada. http://www.cs.ubc.ca/~bsd/research/ferret/

[16] B. de Alwis. Supporting Conceptual Queries Over Integrated Sources of Program Information . PHD Thesis. University of British Columbia, Vancouver Canada. 2008

[17] J.-H. Pfeiffer. Complex code querying and navigation for AspectJ. OOPSLA ’05. ACM. 2005

[18] M. P. Robillard and F.Weigand-Warr. ConcernMapper: simple view-based separation of scattered concerns. In Proceedings of the 2005 OOPSLA Workshop on Eclipse technology eXchange, ACM, 2005

[19] C. R. Pandian. Software metrics: a guide to planning, analysis and application. Auerbach Publications. 2003

[20] Mining Software Archives.Lehrstuhl für Softwaretechnik Universität des Saarlandes – Informatik. Prof. Zeller. http://www.st.cs.uni-saarland.de/softevo/.

[21] T. Zimmermann, A. Zeller. When do changes induce fixes?. ICSE 05, ACM. 2005

[22] A. E. Hassan, T. Xie.Software Intelligence: Future of Mining Software Engineering Data.In Proceedings of FSE/SDP Workshop on the Future of Software Engineering Research .Santa Fe. ACM. 2010

[23] Sushil Bajracharya, Trung Ngo, Erik Linstead et. al. Sourcerer: A Search Engine for Open Source Code Supporting Structure-Based Search, OOPSLA’06. USA. ACM. 2006

[24] Sushil K. Bajracharya, Joel Ossher, Cristina V. Lopes, Sourcerer - An Infrastructure for Large-scale Collection and Analysis of Open-source Code, Third International Workshop on Academic Software Development Tools and Techniques, Belgium, ACM, 2010

[25] Otavio Lemos, Sushil Bajracharya et al.. CodeGenie: Using Test-Cases to Search and ReuseSource Code, ASE’07, 2007, Atlanta, Georgia, USA. ACM

[26] Otávio Augusto Lazzarini Lemos , Sushil Bajracharya and Joel Ossher , CodeGenie: a Tool for Test-Driven Source Code Search, OOPSLA’07, Canada. ACM, 2007

[27] O. Hummel, “Semantic Component Retrieval in Software Engineering”,Ph.D. dissertation, Faculty of Mathematics and Computer Science, University of Mannheim, 2008.

[28] C. Atkinson and D. Stoll: "Orthographic Modelling Environment", in Proceedings of Fundamental Approaches to Software Engineering (FASE'08), Hungary, Spring, 2008

[29] C. Atkinson and D. Stoll: "An Environment for the Orthographic Modeling of Workflow Components", in Proceedings of the Prozessinnovationen mit Unternehmenssoftware (PRIMIUM), Germany, 2008

[30] C. Atkinson, D. Brenner, et al.. Modeling Components and Component-Based Systems in KobrA, in A. Rausch, R. Reussner et al. The Common Component Modeling Example: Comparing Software Component Models, Springer, 2008

[31] R. Kimball, M. Ross. The data warehouse toolkit: the complete guide to dimensional modeling. John Wiley and Sons. 2002

[32] F. Navrade. Strategische Planung mit Data-warehousesystemen. Gabler Verlag. 2008.

Die probabilistische Ähnlichkeitsanfragesprache QSQL2

Sascha Saretz und Sebastian Lehrack
Brandenburgischer Technische Universität Cottbus
Institut für Informatik
Postfach 10 13 44
D-03013 Cottbus, Germany
{ssaretz, slehrack}@informatik.tu-cottbus.de

ABSTRACT

Die quantenlogik-basierte probabilistische Ähnlichkeitsanfragesprache QSQL2 soll vorgestellt werden. Dabei liegt das Hauptaugenmerk auf der Formulierung von Anfragen, welche "unsicher" sind, also nicht nur die traditionelle Boolesche Werte **wahr** und **falsch** annehmen können. QSQL2 kann Ungenauigkeiten sowohl auf Relationenebene als Eintrittswahrscheinlichkeiten, als auch auf Prädikatebene als Relevanzwahrscheinlichkeiten modellieren. Zusätzlich bietet die Sprache die Eigenschaft einer Booleschen Algebra, womit bekannte Äquivalenzen für die Anfragen nutzbar sind.

1. MOTIVATION

Im traditionellen relationalen Modell von Codd [4] sind Tupel entweder in einer Relation enthalten oder nicht. Im Gegensatz dazu können die Ansätze probabilistischer Datenbanken unpräzisen Daten verarbeiten, wobei jede mit einer Eintrittswahrscheinlichkeit annotiert ist. Es ist also *unsicher*, ob ein Tupel in einem bestimmten Datenbankzustand (*mögliche Welt*) vorkommt oder nicht [5].

Ein anderer Typ von Unsicherheiten sind Ähnlichkeitsprädikate wie "Größe≈1.80m". Sie drücken unsichere Beziehungen zwischen Tupeln aus.

Dies sind zwei unterschiedliche Arten Unsicherheit zu formalisieren. Unsere Sprache QSQL2 erlaubt es beide Arten zu nutzen, was dem Nutzer mehr Freiheiten beim Stellen von Anfragen bietet. Des Weiteren beachtet die QSQL2 die Gesetze der Booleschen Algebra, womit viele für den Nutzer sehr intuitive Äquivalenzen anwendbar sind.

Um diese erweiterten Möglichkeiten zu verstehen, betrachten wir im nächsten Abschnitt zunächst eine Klassifikation von Anfragearten.

2. ANFRAGETYPEN

Wir wollen zunächst eine Klassifikation unterschiedlicher Anfrageklassen erstellen. Mit diesen sollen semantische Unterschiede zwischen Anfragen deutlich gemacht werden. Die Entwicklung dieser Klassifikation ist in [13] zu finden.

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

Um die Klassifikation aufzubauen identifizieren wir zwei signifikante Kriterien der Ausdrucksmächtigkeit der Anfragesprache und der darunterliegenden relationalen Datenbasis:

(i) das Einbauen von Konzepten der Ungenauigkeit und Ähnlichkeit durch Ähnlichkeitsprädikate und

(ii) Tupel, welche Konfidenzwerte besitzen.

Wir benennen die Erfüllung einer dieser beiden Kriterien mit dem Term *unsicher*. Dies bedeutet, dass wir sichere oder unsichere Anfragen auf sicheren oder unsicheren relationalen Daten anwenden. Es ist dabei darauf zu achten, dass die Begriffe *sicher* und *unsicher* auch mit anderen Bedeutungen genutzt werden. In unserem Zusammenhang nutzen wir den Begriff *unsicher* für den Datenmodellierungsaspekt. Wenn ein Nutzer nicht weiß, welche die korrekte Instanz oder der richtige Wert seiner Daten ist, kann er diese mit einem Konfidenzwert annotieren, welcher die Eintrittswahrscheinlichkeit darstellt.

Indem die beiden Klassifikationsdimensionen orthogonal angewendet werden, erhält man vier Anfrageklassen. Diese Klassen werden im Folgenden kurz beschrieben.

(i) Sichere Anfragen auf sicheren Daten (`CQonCD`)
Die Klasse `CQonCD` (Certain Queries on Certain Data) enthält alle Anfragen, welche durch Boolesche Bedingungen auf deterministischen relationalen Daten erzeugt werden. Diese Anfragen können durch traditionelle relationale Anfragesprachen wie den relationalen Kalkül, die relationale Algebra und SQL gestellt werden. Die folgenden drei Klassen enthalten `CQonCD` vollständig.

(ii) Unsichere Anfragen auf sicheren Daten (`UQonCD`)
Die Klasse `UQonCD` (Uncertain Queries on Certain Data) steht für Anfragen, welche Ungenauigkeiten und Vagheit unterstützen indem Ähnlichkeitsprädikate genutzt werden können. Diese Prädikate basieren auf einer sicheren Datengrundlage. Das Evaluationsergebnis einer solchen Anfrage kann durch einen score-Wert aus dem Intervall $[0,1]$ angegeben werden, welches den Grad der Erfüllung darstellt.

(iii) Sichere Anfragen auf unsicheren Daten (`CQonUD`)
Die Anfragen der Klasse `CQonUD` sind typisch für probabilistische Datenbanken mit Possible-Worlds-Semantik (siehe Abschnitt 3.2). Diese Anfragen nutzen Boolesche Bedingungen auf unsicheren Daten mit einem Konfidenzwert aus dem Intervall $[0,1]$.

(iv) Unsichere Anfragen auf unsicheren Daten (`UQonUD`) Wenn man die Possible-Worlds-Semantik (iii) durch Bedingungen mit Ähnlichkeitsprädikaten (ii) kombiniert, erhält man eine Anfrage einer Klasse mit erweiterter Ausdruckskraft. In `UQonUD` können Ähnlichkeitsbedingungen auf Daten genutzt werden, welche nur in einem bestimmten unsicheren Datenbankzustand gegeben sind. Die Klasse `UQonUD` umfasst die ersten drei Klassen.

Wir werden sehen, dass QSQL2 eine Vielzahl von Anfragen aus allen vier Klassen auswerten kann und somit eine große Bandbreite für die Nutzung von unsicheren Anfragen bietet.

3. DATEN- UND ANFRAGEMODELL

Nun soll das grundlegende Datenmodell der Anfragesprache QSQL2 beschrieben werden. Es kombiniert zwei Wahrscheinlichkeitsarten: (i) eine Relevanzwahrscheinlichkeit gegen eine Anfrage und (ii) eine Eintrittswahrscheinlichkeit für ein Datenobjekt.

3.1 Relevanzwahrscheinlichkeit

Um die Relevanzwahrscheinlichkeit z. B. einer `UQonCD`-Anfrage auszudrücken, nutzen wir die probabilistische Interpretation eines geometrischen Retrievalmodells, welches auf dem quadrierten Kosinus-Ähnlichkeitsmaß basiert [12]. Die Hauptidee unseres Retrievalmodells ist die Anwendung von Vektorräumen, welche auch aus der Quantenmechanik oder Quantenlogik bekannt sind, um Anfrageauswertung in Datenbanken zu betreiben. Hier wollen wir eine Idee der grundlegenden Prinzipien vermitteln. Für diesen Zweck sind Zusammenhänge zwischen Konzepten aus der Anfrageauswertung und dem angewandten Retrievalmodell in Tabelle 1 dargestellt.

Das Retrievalmodell beschreibt die Auswertung eines einzelnen Tupels gegen eine gegebene Ähnlichkeitsanfrage. Wir beginnen unsere Beschreibung, indem wir uns ein Vektorraum vorstellen, welcher die Domäne für ein Tupel ist. Alle Attributwerte eines Tupels werden durch die Richtung eines entsprechenden Tupelvektors der Länge 1 ausgedrückt. Eine logik-basierte Bedingung korrespondiert zu einem spezifischen Vektorunterraum des Domänen-Vektorraums, auch *Bedingungsraum* genannt.

Das Resultat der Auswertung ist festgelegt durch den minimalen Winkel zwischen Tupelvektor und Bedingungsraum. Der quadrierte Kosinus dieses Winkels ist ein Wert aus dem Intervall $[0,1]$ und kann daher als Ähnlichkeitsmaß interpretiert werden. Wenn also ein Tupelvektor zum Bedingungsraum gehört, kann man diese Bedingung als vollständige Übereinstimmung interpretierten (mit einem Score-Wert von 1). Im Gegensatz dazu entspricht der rechte Winkel von 90° zwischen Tupelvektor und Bedingungsraum keiner Übereinstimmung, der Score-Wert ist 0.

Anfrageauswertung		**CQQL Modell**
Wertebereich $\mathrm{Dom}(t)$	$\leftrightarrow$	Vektorraum $\mathbf{H}$
angefragtes Tupel t	$\leftrightarrow$	Tupelvektor $\vec{t}$
Bedingung c	$\leftrightarrow$	Bedingungsraum $\mathbf{cs}[c]$
Auswertung $\mathrm{eval}^t(c)$	$\leftrightarrow$	quadrierter Kosinus des Winkels zwischen $\vec{t}$ und $\mathbf{cs}[c]$ $(\cos^2(\measuredangle(\vec{t}, \mathbf{cs}[c])))$

Table 1: Zusammenhänge zwischen Anfrageauswertung und dem Retrieval-Modell CQQL

In früheren Arbeiten [12, 11] entwickelten wir eine probabilistische Interpretation für unser Retrievalmodell, daher kann das geometrische Ähnlichkeitsmaß auch als Wahrscheinlichkeit der Relevanz aufgefasst werden. Aus diesem Grund kann man die folgenden bekannten Auswertungsregeln für Wahrscheinlichkeiten anwenden, wenn alle beteiligten Teilbedingungen c_1 und c_2 unabhängig sind:

$$\begin{aligned}
\mathrm{eval}^t(c) &:= \mathrm{SF}(t,c), \text{ wenn } c \text{ atomar ist}\\
\mathrm{eval}^t(c_1 \wedge c_2) &:= \mathrm{eval}^t(c_1) * \mathrm{eval}^t(c_2)\\
\mathrm{eval}^t(c_1 \vee c_2) &:= \mathrm{eval}^t(c_1) + \mathrm{eval}^t(c_2) -\\
&\qquad \mathrm{eval}^t(c_1) * \mathrm{eval}^t(c_2)\\
\mathrm{eval}^t(\neg c) &:= 1 - \mathrm{eval}^t(c).
\end{aligned}$$

Die *Berechungsfunktion* SF den Ähnlichkeitswert für atomare Ähnlichkeitsbedingungen berechnet, z. B. 'Ort ≈ Berlin'.

Um die Unabhängigkeit der Teilbedingungen zu erhalten benötigt man die folgende Einschränkung: *In einer gültigen Bedingung darf kein Attribut gegen mehr als eine Konstante in unterschiedlichen Ähnlichkeitsprädikaten angefragt werden.* Daher ist die Bedingung 'Ort≈Berlin ∧ Ort≈München' nicht in QSQL2 erlaubt. Die Ähnlichkeitsprädikate 'Ort ≈ Berlin' und 'Ort ≈ München' können somit nicht für einen festen Ort gleichzeitig zu 1 ausgewertet werden (vollständige Übereinstimmung), was auch der Intuition entspricht. Diese Einschränkung entspricht der Unabhängigkeitsannahme von Tupel-unabhängigen bzw. Block-unabhängigen probabilistischen Datenbanken, welche im folgenden Abschnitt näher erläutert werden.

3.2 Eintrittswahrscheinlichkeit

Die Possible-Worlds-Semantik wird von den meisten probabilistischen Datenbanken genutzt um Anfragen aus der Klasse `CQonUD` zu verarbeiten.

Als Grundlage dient eine Relation $R \subseteq \mathrm{Dom}(A_1) \times \ldots \times \mathrm{Dom}(A_n)$ eines Relationenschemas $\mathrm{attr}(R) = \{A_1, \ldots, A_n\}$, wobei A_i für ein Attribut steht. Dann definiert jede Tupelteilmenge von R einen eigenen Datenbankzustand, auch *Welt* von R genannt. Nehmen wir eine ein-attributige Relation $R = \{(1),(2)\}$ an. Für dieses Beispiel sind die möglichen Zustände oder möglichen Welten durch $R_{w_1} = \{(1),(2)\}$, $R_{w_2} = \{(1)\}$, $R_{w_3} = \{(2)\}$ und $R_{w_4} = \{\}$ gegeben. Eine dieser möglichen Welten repräsentiert die eine, welche in Realtität vorkommt. Allerdings ist unbekannt, welche dies genau ist. Um diese Unsicherheit zu meistern, nutzen wir ein Wahrscheinlichkeitsmaß über der Menge aller möglichen Welten, welches aus einer probabilistischen Tabelle abgeleitet ist. Wir nennen eine Welt mit einer Eintrittswahrscheinlichkeit höher als 0 eine *mögliche Welt* oder *possible world*.

Im Allgemeinen ist die Semantik der genutzten Wahrscheinlichkeitsmaße nicht vordefiniert. Um die Wahrscheinlichkeitsberechnung zu vereinfachen nutzen wir die Semantik der probabilistischen Block-unabhängigen Datenbanken [2] für QSQL2.

In probabilistischen Block-unabhängigen Datenbanken ist jedes Tupel t mit einem Ereignis $E[t]$ verknüpft, welches das

Vorkommen oder das Nichtvorhandensein eines Tupels t in der Realität ausdrückt. Insbesondere unterscheiden wir zwei Arten von Ereignissen und Tupeln. Auf der einen Seite betrachten wir *Basisereignisse* welche von *Basistupeln* abgeleitet sind, welche durch initiale probabilistische Relationen gegeben sind. Außerdem berücksichtigen wir *komplexe Ereignisse*, welche mit während der Anfrageverarbeitung erzeugen komplexen Tupeln verknüpft sind. Diese Ereignisse bestimmen die Eintrittswahrscheinlichkeit der Ergebnistupel.

Dabei sind Tupel aus einem Block disjunkt zueinander, Tupel aus unterschiedlichen Blöcken sind unabhängig zu einander. Durch diese Vereinfachung erhält man eine relativ einfache Berechnungsvorschrift für komplexe Ereignisse.

Wenn die zugrundeliegende Ereignisstruktur unabhängig ist, kann man die Wahrscheinlichkeiten eines komplexen Ereignistupels wie in [8] berechnen:

$$\begin{aligned}\Pr(E[t_1] \wedge E[t_2]) &:= \Pr(E[t_1]) * \Pr(E[t_2]) \\ \Pr(E[t_1] \vee E[t_2]) &:= \Pr(E[t_1]) + \Pr(E[t_2]) - \\ &\qquad \Pr(E[t_1] \wedge E[t_2]) \\ \Pr(\neg E[t_1]) &:= 1 - \Pr(E[t_1]).\end{aligned}$$

3.3 Kombinierter Wahrscheinlichkeitsraum

Schlussendlich kombinieren wir die eingeführten probabilistischen Modelle, um beliebige Anfragen aus der Klasse `UQonUD` verarbeiten zu können. Dies wird getan, indem die Wahrscheinlichkeitsräume, welche Relevanz- und Eintrittswahrscheinlichkeiten repräsentieren, durch einen Produktwahrscheinlichkeitsraum vereinigt werden. Die Nutzung eines Produktwahrscheinlichkeitraumes kann durch die Klassifikation der Anfrageklasse `UQonUD` gerechtfertigt werden.

Wir nehmen also zuerst ein gegebenes Tupel als Datenbasis an, welches mit einer Eintrittswahrscheinlichkeit annotiert ist. Dann wenden wir *zusätzlich* eine Ähnlichkeitsbedingung an, um eine Relevanzwahrscheinlichkeit auf dieser Datenbasis zu erzeugen. So verhindern wir das Vermischen oder Überlappen von beiden Eingabewahrscheinlichkeiten. Somit nehmen wir an, dass beide Wahrscheinlichkeitsmaße unabhängig voneinander und in den kombinierten Produktwahrscheinlichkeitsraum eingebettet sind.

4. ANFRAGEN IN QSQL2

Um Ideen zu verdeutlichen und Beispielanfragen anzugeben wollen wir ein laufendes Beispiel einführen. Es ist ein vereinfachter Verbrechenslöser, welcher an ein Beispiel vom Trio Projekt [15] angelehnt ist. Die Datenwerte sind aus [13]. Es gibt eine deterministische Tabelle *Criminals* (abgekürzt *crim*, Tabelle 2), welche ein Dossier von registrierten Kriminellen enthält. Des Weiteren gibt es eine probabilistische Tabelle *Observations* (abgekürzt *obs*, Tabelle 3) mit Zeugenaussagen und den zugehörigen Konfidenzen.

Die Datei der Kriminellen enthalt die Attribute `name`, `status`, `sex`, `age` und `height` jeder registrierten Person, wobei die Domänen für die Attribute `status` und `sex` {free, jail, parole} und {female, male} sind.

Die Aufzeichnung der Beobachtungen beinhaltet die Zeugenaussagen für ein spezielles Verbrechen, so dass jeder Zeuge nur genau eine Person mit entsprechenden Geschlecht (`obs_sex`), geschätztem Alter (`obs_age`) und geschätzter Größe (`obs_height`) sah. Jedes Aussagentupel in `obs` ist mit einem Konfidenzwert annotiert, welcher als Eintrittswahrscheinlichkeit aufgefasst werden kann. Wir nehmen an, dass Beobachtungen von unterschiedlichen Zeugen unabhängig voneinander sind (z. B. die Tupel t_6 und t_7) und dass die Aussagen eines Zeugen disjunkt sind. Zum Beispiel kann nur maximal eins der Tupel t_4, t_5 und t_6 der Zeugin *Amber* der Wahrheit entsprechen.

Criminals (crim)					
TID	name	status	sex	age	height
t_1	Bonnie	jail	female	21	1.63
t_2	Clyde	free	male	32	1.83
t_3	Al	free	male	47	1.76

Table 2: Deterministische Informationen über registrierte Kriminelle

Observation (obs)					
TID	witness	obs_sex	obs_age	obs_height	Pr
t_4	Amber	male	30	1.85	0.3
t_5	Amber	male	35	1.90	0.3
t_6	Amber	female	25	1.70	0.3
t_7	Mike	female	20	1.60	0.7
t_8	Carl	female	30	1.80	0.9

Table 3: Zeugenaussagen annotiert mit Konfidenzwerten

Anfragen bzgl. Klassifikation: Als erstes sollen Beispiele für die Anfrageklassen aus Abschnitt 2 erfolgen. Ein typisches Beispiel für ein `CQonCD`-Anfrage ist *"Bestimme alle Kriminellen, welche den Status 'free' haben"*. Diese Anfrage ist in QSQL2 und SQL gleich, da keine Ähnlichkeitsprädikate oder probabilistischen Relationen benötigt werden (Listing 1).

Eine `UQonCD`-Anfrage ist z. B. *"Bestimme alle Kriminellen, welche den Status 'free' haben und deren Altern ungefähr 30 ist"*. Listing 2 zeigt das entsprechende Anfrage in QSQL2-Syntax. Die Vagheit ("ungefähr") wird durch ein Ähnlichkeitsprädikat ($\approx$) umgesetzt.

Als ein komplexeres Beispiel betrachten wir *"Bestimme alle Kriminellen, welche möglicherweise beobachtet wurden. Dies bedeutet, dass das Alter in einem Intervall von zehn Jahren um das beobachtete Alter liegt und dass das beobachtete Geschlecht passend ist"*. Dieses Beispiel enthält eine Boolesche Bedingung, während die Relation `Observation` probabilistisch ist (Listing 3).

Als ein Beispiel für eine Anfrage aus der Klasse `UQonUD` wollen wir eine Variante der letzten `CQonUD`-Anfrage (Listing 3) betrachten: *"Bestimme alle Kriminellen, welche möglicherweise beobachtet wurden. Dies bedeutet, dass das Alter ähnlich zum beobachteten Alter ist und dass das beobachtete Geschlecht passend ist"* (Listing 4). In dieser Anfrage kommt sowohl ein Ähnlichkeitsprädikat ($\approx$), als auch eine probabilistische Relation (`Observation`) vor.

```
SELECT name FROM Criminals C
WHERE C.status = 'free'
```

Listing 1: `CQonCD`-Anfrage

```
SELECT name FROM Criminals C
WHERE C.status = 'free' and C.age ≈ 30
```

Listing 2: UQonCD-Anfrage

```
SELECT name FROM Criminals C, Observation O
WHERE C.sex = O.sex and C.age > O.obs_age-5
  and C.age < O.obs_age+5
```

Listing 3: CQonUD-Anfrage

Logische Anfragen: Ein großer Vorteil von QSQL2 ist, dass das zugrunde liegende theoretische Fundament eine Boolesche Algebra bildet, also viele bekannte mathematische Äquivalenzen wie z. B. Distributivität, Idempotenz und Absorption erfüllt sind. An dieser Stelle sollen einige dieser logischen Eigenschaften exemplarisch von QSQL2 für praxisrelevante Anfragen genutzt werden. Wie wir später noch in Abschnitt 5.4 sehen werden, erfüllen z. B. die Fuzzy-Datenbanken nicht alle diese logischen Eigenschaften, insofern sind einige der folgenden Anfragen trotz einfacher Syntax nicht selbstverständlich.

Oft macht es Sinn Implikationen der Form $A \rightarrow B$ auszudrücken, d.h. wenn die erste Aussage wahr ist, muss es die andere auch sein. Durch die bekannte Äquivalenz $A \rightarrow B \equiv \neg A \vee B$ kann man diesen Junktor auch auf Anfragen mit Relevanz- und Eintrittswahrscheinlichkeiten anwenden. Analog verhält es sich mit der Äquivalenz $A \leftrightarrow B$. Bei ihr sind im Booleschen Fall entweder beide Variablen wahr oder beide sind falsch. Durch die Umformung $A \leftrightarrow B \equiv A \rightarrow B \wedge B \rightarrow A \equiv (\neg A \vee B) \wedge (\neg B \vee A) \equiv (A \wedge B) \vee (\neg A \wedge \neg B)$ kann diese Aussage auch äquivalent in QSQL2 ausgedrückt werden.

QSQL2 bietet ebenfalls gewichtete Junktoren. So macht es manchmal Sinn den Einfluss einer Teilbedingung herauf- oder herabzusetzen. In der Sprache gibt es deshalb jeweils eine gewichtete Konjunktion, ausgedrückt durch $and[\theta_1, \theta_2]$, und eine gewichtete Disjunktion, ausgedrückt mit $or[\theta_1, \theta_2]$. Die Gewichtsvariablen θ_i sind reelle Zahlen aus dem Intervall $[0, 1]$, wobei ein Gewicht von 0 überhaupt keinen Einfluss und ein Gewicht von 1 normalen Einfluss bedeutet.

Man könnte sich vorstellen, dass die Identifizierung der Verdächtigen durch die Zeugen nicht eindeutig war, weil das Verbrechen bei Dunkelheit geschehen ist. So kann man folgende Variante der UQonUD-Anfrage in QSQL2 stellen: *"Bestimme alle Kriminellen, welche möglicherweise beobachtet wurden. Dies bedeutet, dass das Alter ähnlich zum beobachteten Alter ist und dass die Größe ähnlich zur beobachteten Größe ist. Die Relevanz des beobachteten Größe ist doppelt so hoch wie die des geschätzten Alters."* (Listing 5).

```
SELECT name FROM Criminals C, Observation O
WHERE C.sex = O.sex and C.age ≈ O.obs_age
```

Listing 4: UQonUD-Anfrage

```
SELECT name FROM Criminals C, Observation O
WHERE C.height ≈ O.obs_height and[ 1, 0.5 ]
      C.age ≈ O.obs_age
```

Listing 5: Beispiel für gewichtete Anfrage

5. VERGLEICHBARE ANSÄTZE

In den letzten Jahren wurden viele probabilistische relationale Datenbankansätze vorgeschlagen [3, 2, 7, 8, 6, 10, 1]. Sie unterstützen alle die Verarbeitung von probabilistischen relationalen Daten, d.h. Anfragen aus der Klasse CQonUD.

Neben der Berechnungskomplexität ist die Ausdruckskraft ein signifikantes Vergleichsmerkmal. Im Folgenden werden drei unterschiedliche Ansätze beschrieben, wie probabilistische Datenbanken um Ähnlichkeitsprädikate erweitert werden können.

5.1 Ähnlichkeitsprädikate als Built-In-Prädikate

Fuhr und Rölleke schlagen vor die Bewertungsfunktion eines Ähnlichkeitsprädikates durch eine separate probabilistische Relation umzusetzen [8]. Diese Relation für eine Ähnlichkeitsfunktion (SF-Relation) ersetzt das Ähnlichkeitsprädikat und wird durch ein Join in die Anfrage integriert.

Leider gibt es bei diesem Ansatz ein Problem bei der Konstruktion der Ähnlichkeitsfunktion SF. Die Funktion repräsentiert ein Ähnlichkeitsprädikat, aber bzgl. der Auswertung ist es kein unabhängiges Konzept, sondern unterliegt den selben Regeln wie alle probabilistische Relationen. So müssen die Tupel unabhängige Basisereignisse bilden, damit man geeignete Aggregationsfunktionen anwenden kann. Die Unabhängigkeit der Tupel in einer SF-Relation ist aber nicht gegeben. Fuhr und Rölleke schlagen daher vor nur Anfragen zu nutzen, in denen keine Tupel aus der selben SF-Relation kombiniert werden. Deshalb kann keine SF-Relation mehr als einmal in einer Anfrage vorkommen und Projektionen können nicht mehr beliebig genutzt werden.

5.2 Ähnlichkeitsprädikate als Wahrscheinlichkeit von Relationen

Der letzte Ansatz nutzte Ähnlichkeitsprädikate wie probabilistische Relationen, welche *während* der Anfrageauswertung eingebaut werden. Im Gegensatz dazu schlagen Dalvi und Suciu [6] vor, die Wahrscheinlichkeiten für die genutzten Ähnlichkeitsprädiakte *vor* der eigentlichen Anfrageauswertung auszuwerten. Die Ergebnisse dieser Vorberechnung werden als Eintrittswahrscheinlichkeiten den Relationen zugewiesen, auf welche die Ähnlichkeitsprädikate verweisen.

Dieser Ansatz arbeitet nur auf Anfragen mit konjunktivverknüpften Ähnlichkeitsprädiaten. Schon bei einer einfachen Disjunktion von Ähnlichkeitsprädiaten, welche sich auf unterschiedliche Relationen beziehen, ist es nicht mehr möglich, die Auswertung der disjunktiven Ähnlichkeitsbedingung aufzuspalten und hinunter in die entsprechenden Relationen zu schieben.

5.3 Ähnlichkeitsprädikate auf Attributebene

In anderen Modellen wie [1, 10] können Wahrscheinlichkeiten auch auf Attributebene modelliert werden. In diesem Fall ist es möglich, die Auswertung der Ähnlichkeitsprädikate in den abgefragten Attribut *vor* der eigentlichen An-

frageauswertung zu speichern. Wie beim letzten Ansatz aus Abschnitt 5.2 funktioniert dies nur bei konjunktiv verknüpften Ähnlichkeitsprädikaten, weil die Wahrscheinlichkeit eines Tupels konjunktiv aus den Wahrscheinlichkeiten der jeweiligen Attributwerte berechnet wird. Deshalb können nicht alle komplexen (z. B. disjunktiven) Kombinationen von Ähnlichkeitsprädikaten ausgewertet werden.

5.4 Fuzzy-Datenbanken

Fuzzy-Datenbanken wie FSQL [9] können ebenfalls unsichere Anfragen auf einer unsicheren Datengrundlage bewerkstelligen, allerdings sind sie kein probabilistisches Modell. Die entsprechenden Tupel-Konfidenzwerte werden einfach ohne Rücksicht auf die Semantik der Teilbedingungen aggregiert. Es findet also keine Überprüfung auf Korrelationen statt, was das Ergebnis verfälschen kann.

Außerdem bildet die Fuzzylogik [16] keine Boolesche Algebra, da bekannte Äquivalenzen wie Idempotenz und Distributibität nicht erfüllt sind. Aufgrund des Fehlens dieser elementaren Eigenschaft sind Fuzzy-Datenbanken für uns nicht geeignet. Einen ausführlichen Vergleich zwischen Fuzzy- und Quantenlogik wird in [14] gegeben.

Wir fassen zusammen, dass im Gegensatz zu QSQL2 die anderen Ansätze [8, 6, 1, 10, 9] nicht beliebige, logik-basierte Ähnlichkeitsbedingungen beherrschen.

6. ZUSAMMENFASSUNG

In dieser Arbeit wurde die quantenlogik-basierte probabilistische Ähnlichkeitsanfragesprache QSQL2 vorgestellt. Ihre Grundlagen wurden kurz dargelegt, ihre Syntax an Beispielen anschaulich gemacht und ihre Besonderheiten demonstriert.

Im Gegensatz zu probabilistischen Datenbanken ist die Integration und Nutzung von Ähnlichkeitsprädikaten in mehr Fällen möglich. Die zusätzlichen Eigenschaften einer Booleschen Algebra wie Idempotenz oder Distributibität ermöglichen bessere Resultate als z. B. bei Fuzzylogik-basierte Sprachen. Das mathematisch Fundament ermöglicht die Interpretation der Ergebnisse als Wahrscheinlichkeiten, was sie anschaulicher und verständlicher macht.

Danksagung: Diese Arbeit wurde durch die Förderung SCHM 1208/11 – 1 der Deutschen Forschungsgemeinschaft (DFG) unterstützt.

Literatur

[1] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom. Trio: A System for Data, Uncertainty, and Lineage. In *32nd International Conference on Very Large Data Bases. VLDB 2006 (demonstration description)*, September 2006.

[2] D. Barbara, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Trans. Knowl. Data Eng.*, 4(5):487–502, 1992.

[3] R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In P. M. Stocker, W. Kent, and P. Hammersley, editors, *VLDB*, pages 71–81. Morgan Kaufmann, 1987.

[4] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[5] N. N. Dalvi, C. Ré, and D. Suciu. Probabilistic Databases: Diamonds in the Dirt. *Commun. ACM*, 52(7):86–94, 2009.

[6] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.

[7] D. Dey and S. Sarkar. A probabilistic relational model and algebra. *ACM Trans. Database Syst.*, 21(3):339–369, 1996.

[8] N. Fuhr and T. Rölleke. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

[9] J. Galindo, A. Urrutia, and M. Piattini. *Fuzzy Databases: Modeling, Design and Implementation*. Idea Group Publishing, Hershey, USA, 2006.

[10] C. Koch. MayBMS: A system for managing large uncertain and probabilistic databases. *Managing and Mining Uncertain Data*, 2008.

[11] S. Lehrack, S. Saretz, and I. Schmitt. QSQL^p: Eine Erweiterung der probabilistischen Many-World-Semantik um Relevanzwahrscheinlichkeiten. In T. Härder, W. Lehner, B. Mitschang, H. Schöning, and H. Schwarz, editors, *BTW*, volume 180 of *LNI*, pages 494–513. GI, 2011.

[12] S. Lehrack and I. Schmitt. A Probabilistic Interpretation for a Geometric Similarity Measure. In *Proceedings of the 11th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, ECSQARU '11, June 2011.

[13] S. Lehrack and I. Schmitt. A unifying probability measure for logic-based similarity conditions on uncertain relational data. In *Proceedings of the 1st Workshop on New Trends in Similarity Search*, NTSS '11, pages 14–19, New York, NY, USA, 2011. ACM.

[14] I. Schmitt, A. Nürnberger, and S. Lehrack. On the Relation between Fuzzy and Quantum Logic. In *Views on Fuzzy Sets and Systems from Different Perspectives, chapter 5*. Springer-Verlag, 2009.

[15] J. Widom. Trio: A system for data, uncertainty, and lineage. In *Managing and Mining Uncertain Data*, pages 113–148. Springer, 2008.

[16] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965.

Informationsanbieterzentrierte Spezifikation und Generierung von Informationssystem-Apps

Jonas Pencke, David Wiesner, Hagen Höpfner und Maximilian Schirmer
Bauhaus-Universität Weimar
Bauhausstraße 11
99423 Weimar, Germany
VORNAME.NACHNAME@uni-weimar.de

Schlüsselworte

Mobile Informationssysteme, Nutzergetriebene Programmierung, App-Erzeugung

Zusammenfassung

Mobilgeräte wie z.B. Smartphones werden heutzutage nicht mehr ausschließlich zur Sprachkommunkation eingesetzt. Sie ermöglichen es, zeitnah Informationen an mobile Nutzer zu übertragen. Hierbei spielt der Aufenthaltsort der Nutzer weitestgehend keine Rolle, er/sie ist quasi jederzeit und allerorts erreichbar. Im Gegensatz zu der Einfachheit der Informationskonsumtion ist das Entwickeln von Apps nicht trivial. Hierzu ist Expertenwissen notwendig. Zahlreiche potentielle Informationsanbieter verfügen nicht über die notwendigen Kenntnisse, wenngleich ihre Informationen für zahlreiche Konsumenten interessant wären und es starke strukturelle Ähnlichkeiten zwischen mobil verfügbar gemachten Informationen gibt. Ein weiteres Problem ist, dass unterschiedliche Smartphone-Hersteller dediziert unterschiedliche Programmiersprachen benutzen. In diesem Papier präsentieren wir unseren Ansatz zur anbieterzentrierten Generierung von Apps, wobei ein Hauptaugenmerk auf der Unterstützung heterogener Zielplattformen liegt. Somit ermöglichen wir es technisch nicht versierten Informationsanbietern, ihre eigenen Apps zu erzeugen.

1. EINLEITUNG UND MOTIVATION

Der Einsatz moderner mobiler Endgeräte wie z.B. Smartphones ist nicht mehr auf reine Sprachkommunikation begrenzt. Vielmehr werden zahlreiche Bereiche der Informations- und Kommunikationstechnologien, welche noch vor wenigen Jahren Desktop-Computern vorbehalten waren, unterstützt. Leistungsfähige Geräte wie Apples iPhone oder HTCs Desire ermöglichen den Einsatz von Sofortnachrichtendienste (engl. instant messaging) und klassischen Informationssystemen sowie die Partizipation in sozialen Netzen (Facebook, MySpace, StudiVZ, etc.) somit beinahe jederzeit und allerorts. Im August 2010 veröffentliche Gartner eine Studie [11], nach der 19% aller neu verkauften Mobiltelefone derartige Smartphones sind. Neben der Tatsache, dass Smartphones den mobilen Informationszugriff ermöglichen, bieten sie auch die Möglichkeit der sogenannte Push-Notifikation. Dabei werden Neuigkeiten, Änderungen oder Hinweise direkt vom Anbieter auf das Gerät transferiert, wodurch der Benutzer nicht explizit und regelmäßig danach suchen muss. Von diesem weiteren Informationsverteilungprozess profitieren auch die Informationsanbieter, da sie somit ihre Informationskonsumenten nahezu jederzeit und direkt erreichen können.

Im Gegensatz zur Entwicklung von Softwareprodukten für Desktop-Computer ist das Programmieren von Anwendungen für Smartphones (den sogenannten Apps) an die Verwendung einer dedizierten Programmiersprache gebunden. iOS-Entwickler verwenden Objective-C. Apps für Smartphones mit Google Android werden in Java implementiert. Geräte mit Microsoft's Windows Phone 7 unterstützen C# und Apps für HPs webOS werden in HTML 5, CSS und JavaScript realisiert. Folglich bedingt der Wunsch nach einer breiten Unterstützung verschiedener Smartphones die Notwendigkeit, dass Entwickler mit mehreren dieser Programmiersprachen vertraut sind. Natürlich eröffnet dies einen neuen Markt für Programmierer, es hält jedoch zahlreiche technisch nicht versierte Informationsanbieter davon ab, ihre Information über das Medium Smartphone anzubieten. Unseres Wissens nach existieren momentan keine Lösungen, welche, vergleichbar zu den im WWW etablierten Redaktionssystemen (engl. content management systems), einen einfachen Zugang für Nicht-Programmierer bieten. Des Weiteren sind Web-basierte Ansätze, welche die Verwendung der Internet-Browser der Smartphones voraussetzen, keine Lösung des Dilemmas. Einerseits bieten diese keine Push-Notifikation, andererseits erhöht das manuelle, wiederholte (meist vergebliche) Suchen nach aktuellen Informationen, die Anzahl von energieintensiven drahtlosen Datenübertragungen.

In diesem Papier präsentieren wir unseren Ansatz, die genannten Probleme durch ein Framework zur Erzeugung von Informationssystem-Apps zu beheben. Durch die Analyse verschiedener existierender Apps haben wir herausgefunden, dass die angebotenen Informationen klassifiziert werden können. Basierend auf dieser Klassifikation haben wir eine modulare Struktur entwickelt, die als Code-Skelett für verschiedene Plattformen realisiert wurde. Mithilfe dieser Code-Skelette kann dann eine App wie folgt erzeugt werden: Zuerst wird die App mithilfe eines domänen-spezifischen, Web-basierten Konfigurationswerkzeugs parametrisiert. Die dabei spezifizierten statischen Informationen werden dem Quelltext hinzugefügt. Dynamische Informationen werden auf der Webseite des Anbieters hinterlegt, wo sie auch gewartet werden können. Anschließend wird die App automatisch kompiliert und zum Download angeboten. Neben dem Vorteil, dass auf diese Weise technisch nicht versierte Anwender wie Musiker, Politiker, etc. ohne Kenntnis einer Programmiersprache Apps erstellen können, vermeidet

GvD Workshop'11, 31.05.-03.06.2011, Obergurgl, Tirol, Austria.

unser Verfahren die potentielle Verletzung von Urheber- und Verwertungsrechten. Alle Informationen, die durch die App angeboten werden, werden auch direkt vom Anbieter bereitgestellt und befinden sich unter dessen Kontrolle[1].

Der Rest des Papieres ist wie folgt strukturiert: Abschnitt 2 beschreibt verwandte Arbeiten. Abschnitt 3 analysiert die Eigenschaften der Informationen, welche üblicherweise durch Informationssystem-Apps angeboten werden. Abschnitt 4 beinhaltet eine kurzen Einblick in unseren Evaluationsprototyp und präsentiert die Architektur unseres App-Erzeugungssystems. Abschnitt 5 fasst das Papier zusammen und gibt einen Ausblick auf Folgearbeiten.

2. VERWANDTE ARBEITEN

Die in diesem Beitrag vorgestellten Forschungsergebnisse können den Forschungsgebieten Code-Generierung (engl. code generation), mobile Informationssysteme und Mensch-Maschine-Interaktion (engl. human-computer interaction) zugeordnet werden.

Code-Generierung wird oft als Teilgebiet der modellgetriebenen Softwareentwicklung [5] angesehen. Hierbei werden domänen-spezifische Sprachen verwendet, um abstrakte Modelle von Software-Systemen zu erzeugen. Aus diesen Modellen erzeugen dann Code-Generatoren den Quelltext, entweder in Teilen oder komplett. Unser Ansatz ist es jedoch, Endbenutzern ohne technisches Wissen die Möglichkeit zu geben, Anwendungen zu erzeugen. Von diesen Benutzern kann nicht verlangt werden, formale Spezifikationen zu verstehen oder zu verwenden. Zudem unterscheiden sich die verfügbaren Smartphones verschiedener Hersteller sehr in den unterstützten Programmiersprachen, wie bereits in Abschnitt 1 erläutert. Für unseren Ansatz hätte dies bedeutet eine sehr große Anzahl von verschiedenen, formal spezifizierten Code-Generatoren und domänen-spezifischen Sprachen zu erzeugen, um eine breite Masse von Smartphones unterstützen zu können. Daher haben wir uns dazu entschlossen, ein gerätespezifisches Code-Skelett anzubieten, gemäß dem Paradigma der generativen Programmierung [2]. Unser „Code-Generator“ parametrisiert und ergänzt das Code-Skelett, welches schließlich kompiliert wird. In der Zukunft werden wir an einer formaleren Spezifikation arbeiten.

Mobile Informationssysteme machen es möglich, Informationen auf mobilen Endgeräten anzubieten. Die größte Herausforderung besteht dabei in der Reduktion des übertragenen Datenvolumens zum und vom Gerät. Die drahtlose Datenübertragung ist unverzichtbar, aber auch sehr energieintensiv [3, 1], vergleichsweise langsam [13] und (je nach Mobilfunkvertrag) teuer. Zudem mindern physikalische Effekte die Verfügbarkeit von drahtlosen Netzen [9]. Es gibt bereits verschiedene Ansätze, um das anfallende Datenvolumen zu reduzieren. Die Forschungsergebnisse reichen dabei von Caching [8, 12] über Hoarding [7] bis hin zur Replikation [4]. In unserem System begegnen wir dieser Herausforderung, indem so viele Informationen wie möglich bereits in der App enthalten sind. Nur dynamische Informationen (s. Abschnitt 3) werden an das mobile Endgerät übertragen, wo sie zudem für den späteren Zugriff zwischengespeichert werden.

Smartphones bieten meist nur sehr kleine Bildschirme und im Vergleich zu normalen Desktop-Computern grundverschiedene Möglichkeiten der Benutzerinteraktion. Entsprechend gilt es, bei Aspekten der Mensch-Maschine-Interaktion auf diese Unterschiede einzugehen [6]. Die Software-Entwicklungsumgebungen der verschiedenen Smartphone-Plattformen bieten bereits vereinheitlichte Interaktionsschemata, an die die Benutzer der jeweiligen Geräte und Plattformen gewöhnt sind. Generell bevorzugen sie eine konsistente Benutzeroberfläche über alle verwendeten Apps [10]. Deshalb haben wir uns dazu entschlossen, bestehende Benuteroberflächen-Toolkits zu verwenden.

[1]Wir nehmen an, dass der Anbieter die entsprechenden Rechte besitzt bzw. entsprechende Verträge hierfür selbst abgeschlossen hat.

3. INFORMATIONEN IN APPS

Durch die Analyse verschiedener Informationssystem-Apps haben wir zwei Klassen von präsentierten Informationen ableiten können:

Statische Informationen sind Informationen, die explizit in den Binärcode von Apps hinein kompiliert werden. Nahezu alle Apps enthalten statische Informationen wie den Namen der App, statisch verknüpfte Bilder, Informationsdialog, usw. Aktualisierungen dieser Informationen sind nur durch Neukompilierung der App möglich. Dafür ist es möglich, statische Informationen auch ohne Netzverbindung zu verwenden.

Dynamische Informationen sind Informationen, die zur Laufzeit heruntergeladen werden. Dies sind beispielsweise Nachrichten-Streams, ortsabhängige Informationen oder Informationen über Ereignisse. Zur Aktualisierung von dynamischen Informationen wird eine bestehende Netzverbindung benötigt. Durch Caching, Hoarding oder Replikation kann jedoch auch eine Offline-Nutzung zuvor heruntergeladener dynamischer Informationen ermöglicht werden. Es gibt zwei Subklassen dynamischer Informationen: (1) *Interne Informationen* sind Informationen, die zumeist vom Anbieter der App bereitgestellt werden. So werden in Zeitungs-Apps zum Beispiel aktuelle Nachrichten der zugehörigen Zeitung heruntergeladen und präsentiert. (2) *Externe Informationen* sind verknüpfte Informationen, die von Dritten bereitgestellt werden. Dies können zum Beispiel in Ereignisinformationen verknüpfte Videos bei Youtube sein, die nicht direkt vom Anbieter des Ereignisses zur Verfügung gestellt werden.

Bei der Konzeption einer App spielen auch Fragestellungen des Urheberrechts eine Rolle. In unserem Ansatz gehen wir davon aus, dass der Anbieter der App die nötigen Urheberrechte an den in einer App verwendeten statischen und internen Informationen besitzt. Ferner gehen wir davon aus, dass der Anbieter einer App bei der Verwendung von externen Informationen die Nutzungsbedingungen des jeweiligen Informationsanbieters berücksichtigt.

Generell ist davon auszugehen, dass Benutzer auf statische und dynamische Informationen nur lesenden Zugriff besitzen. Einige Apps unterstützen jedoch *interaktive* (dynamische) Informationen. So kann zum Beispiel eine App, die Informationen über Ereignisse bereitstellt, das Kommentieren von Ereignissen durch soziale Netzwerke wie Facebook, Twitter oder MySpace ermöglichen.

4. SYSTEMARCHITEKTUR UND PROOF-OF-CONCEPT

Aufgrund der besseren Verständlichkeit verzichten wir in diesem Papier auf die Trennung zwischen der Beschreibung unseres Ansatzes und der Implementierung des Proof-of-Concept-Systems namens MyBand-App. Dieses System ermöglicht es Musikern, ihre

eigenen Apps für Android- und iOS-basierte Smartphones zu erstellen. Uns ist bekannt, dass es mit FansMagnet[2] bereits einen Anbieter eines vergleichbaren Services für Musiker gibt. Jedoch werden dabei auch die Inhalte der Apps durch FansMagnet verwaltet. Aufgrund rechtlicher Bedenken, obliegt es in unserem Ansatz tatsächlich dem Informationsanbieter, dem Musiker oder dessen Beauftragten, sicherzustellen, dass alle Urheber- und Verwertungsrechtsfragen geklärt werden.

Der Proof-of-Concept stellt ein Anwendungsszenario von Bands bzw. Musikern dar, die Informationen an Ihre Fans über ein mobiles Informationssystem weitergeben möchten. Dieses Beispiel dient allerdings nur der Verdeutlichung unseres Ansatzes. Der hier beschriebene Ansatz lässt sich auf beliebige andere Informationsanbieter übertragen.

Um diese Übertragbarkeit zu gewährleisten, muss das System anpassbar und leicht erweiterbar sein. Aus diesem Grund ist unser System ist als Framework aufgebaut. Für die App-Erzeugung sind drei Komponenten verantwortlich: (1) ein Web-basiertes Konfigurationswerkzeug (MyBandServlet), (2) die Code-Skelette für die Apps und (3) die eigentliche App-Erzeugung. Der Arbeitsablauf beim Erstellen einer App ist wie folgt: Zuerst wählt der Nutzer die Zielplattform(en) und die Module, welche er/sie in die App integrieren will (vgl. Abbildung 1). Anschließend werden die App und die ausgewählten Module konfiguriert und der Erzeugungsprozess angestoßen. Nach dessen Beendigung (nach dem Kompilieren der App) kann der Nutzer seine App herunterladen.

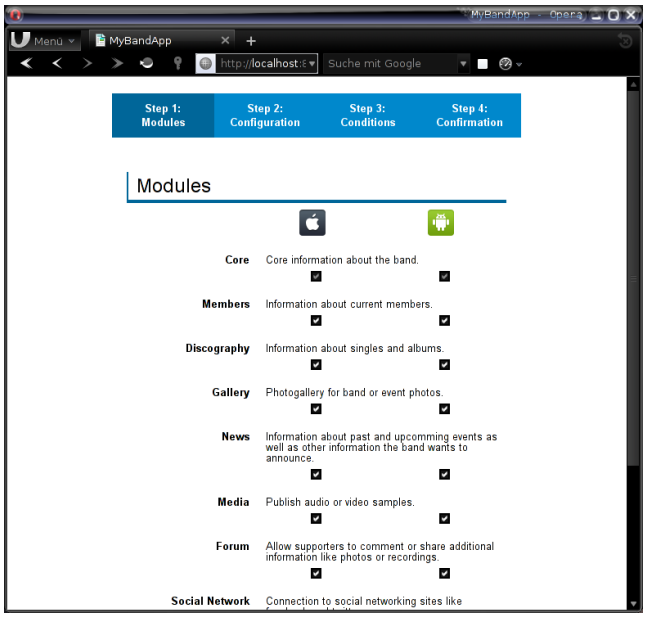


Abbildung 1: MyBand-App Web-basierte Konfiguration (Modulauswahl)

Aus einem technischeren Blickwinkel betrachtet, funktioniert unser Ansatz wie folgt: Wie Abbildung 1 verdeutlicht, sind Module die kleinsten Bausteine in unserem Erzeugungsprozess. Es gibt intern zwei Arten von Modulen: *BuilderModule* sind für die Erstellung des plattformabhängigen Binärcodes zuständig. *AppModule* kapseln die Funktionalität der App. Jedes AppModule umfasst das entsprechende Code-Skelett und eine XML-Datei, mit der die

[2]http://www.fansmagnet.com

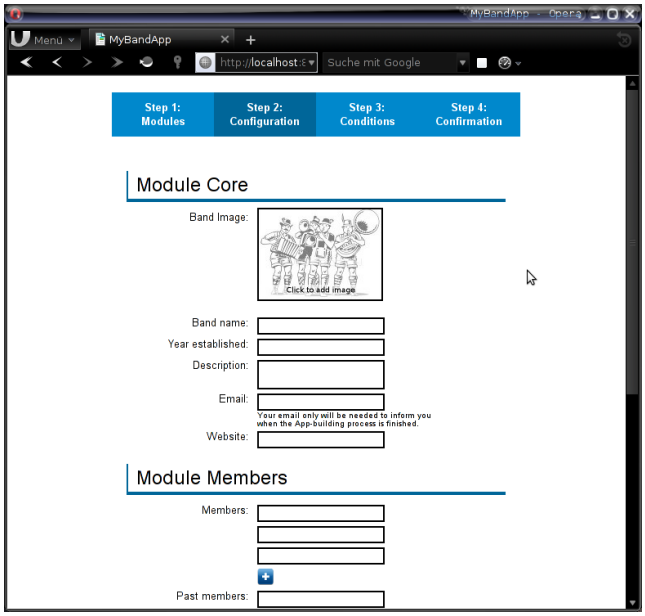


Abbildung 2: MyBand-App Web-basierte Konfiguration (Modulkonfiguration)

durch den Web-basierten Konfigurator festgelegte Modulkonfiguration (bestehend aus statischen und dynamischen Informationen) verarbeitet wird (vgl. Abbildung 2). Dies umfasst im verwendeten Anwendungsszenario unter anderem den Namen und das Bild der Band bzw. Musiker, sowie die Liste der Band-Mitglieder. Für die Diskographie werden z.B. die Datei `DiscographyModule.xml` und zwei AppModule `IPhoneDiscographyModule.jar` und `AndroidDiscographyModule.jar` genutzt. Bevor der eigentliche Erzeugungsprozess gestartet wird, analysiert das MyBand-Servlet die verfügbaren XML-Dateien, erzeugt aus den Meta-Informationen zu statischen Informationen die Formulare für den Konfigurator und generiert aus den Meta-Informationen zu dynamischen Informationen eine RSS-Vorlage[3], die zur späteren Bereitstellung der dynamischen Informationen dient. Nachdem das Konfigurationsformular durch den Nutzer abgeschickt wurde, wird pro App-Modul die `configure`-Methode, welche jedes AppModul implementiert, aufgerufen. Diese Methode modifiziert das Code-Skelett des entsprechenden AppModuls entweder direkt oder, wie in Abbildung 4 dargestellt, durch Ändern der XML-Datei basierend auf der durch den Nutzer angegebenen Konfiguration. Anschließend werden die konfigurierten AppModule an das BuilderModule übergeben, welches für das Kompilieren der App verantwortlich ist.

Die Abbildungen 5 und 6 zeigen eine iOS-App, welche beispielhaft mit dem beschriebenen MyBand-App-Ansatz erzeugt wurde. Sicherlich ist die dargestellte Oberfläche noch nicht „fancy“, die Abbildungen verdeutlichen aber, dass ein Java-basiertes Framework genutzt werden kann, um Programme in Objective-C zu erzeugen.

Abbildung 3 verdeutlicht, dass der vorgestellte Ansatz zum Ziel hat, so generisch wie möglich zu sein. Zur Unterstützungen von Apps abseits der Musiker-Domäne müssen lediglich andere AppModule und XML-Dateien bereitgestellt werden.

[3]Im Prototyp werden die RSS-Dateien händisch editiert. Ein Editor zum formularbasierten Eingeben der dynamischen Daten ist indes in Vorbereitung.

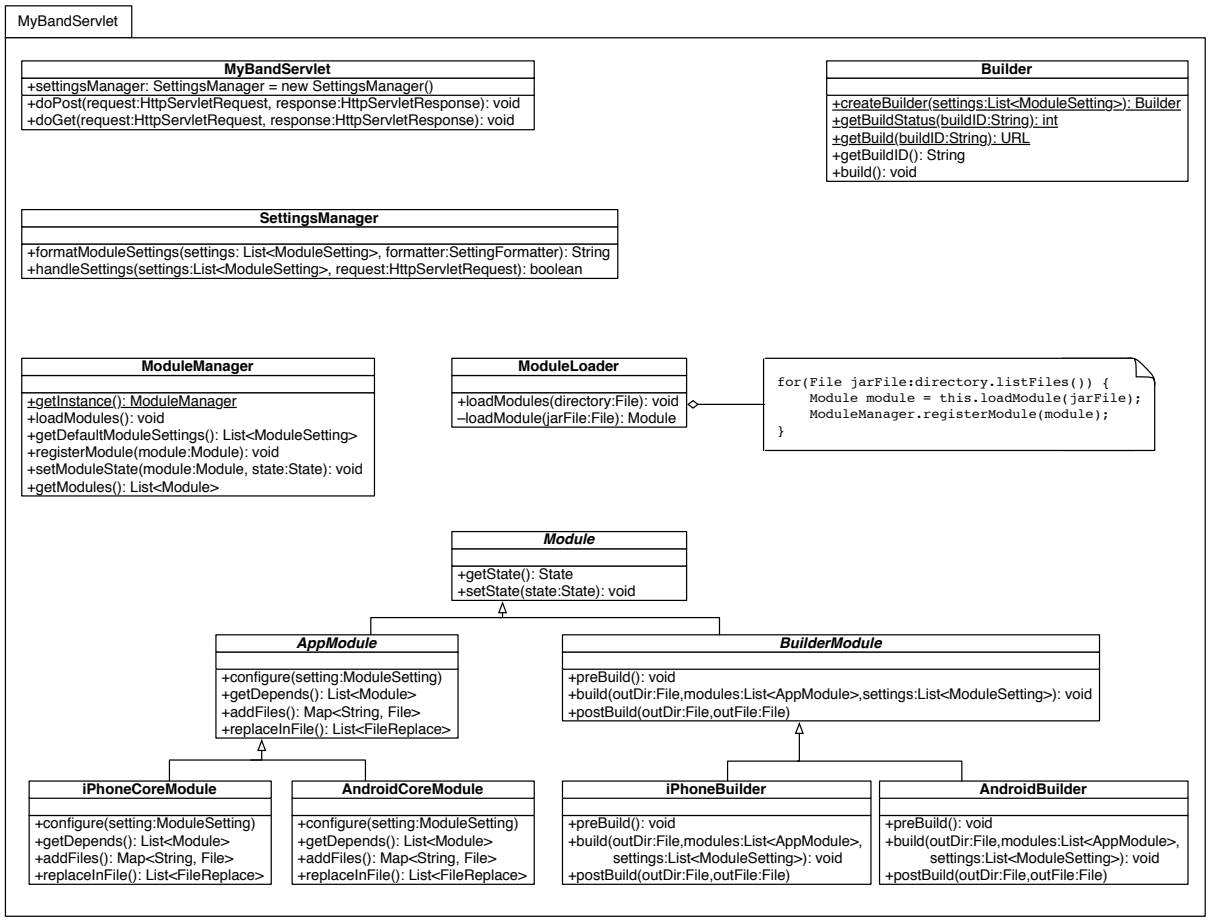


Abbildung 3: Klassendiagramm des Framework-Servlets für die MyBand-App-Konfiguration und -Erzeugung

Abbildung 4: Beispiel für die XML-basierte Transformation eines Android-Code-Skeletts

5. ZUSAMMENFASSUNG UND AUSBLICK

In diesem Beitrag haben wir erste Ideen präsentiert, die es Endbenutzern ohne technische Kenntnisse ermöglichen, Informationssystem-Apps für Smartphones verschiedener Betriebssysteme zu erstellen. Wir haben die generellen Eigenschaften von Informationen, die in solchen Apps angeboten werden, klassifiziert und diese Klassifikation für den Entwurf von Code-Skeletten verwendet. Mit Hilfe eines Web-Interfaces können Benutzer ihre App parametrisieren. Anschließend wird die App über die nativen Entwicklungswerkzeuge kompiliert und kann heruntergeladen werden. Weitere Vorzüge unseres Systems sind die native Benutzeroberfläche und die Berücksichtigung des Urheberrechts der angebotenen Informationen. Neben diesen allgemeinen Ansätzen haben wir unseren Prototypen „MyBand-App“ vorgestellt, den wir zudem einer breiten Öffentlichkeit auf der CeBIT-Messe im März 2011 in Hannover zeigen konnten.

Wir stehen erst am Anfang unserer Forschung und sehen nun verschiedene Forschungsrichtungen, die sich in Zukunft eröffnen. Zunächst werden wir interaktive Informationen berücksichtigen. Zudem werden wir die in unserem Evaluationssystem umgesetzten Ideen generalisieren, um die Erstellung von sehr unterschiedlichen mobilen Informationssystem-Apps zu unterstützen. Ferner werden wir somit die Formalisierung unseres Ansatzes beginnen, um die dem Ansatz bislang noch fehlende fundierte formale Basis zu erhalten.

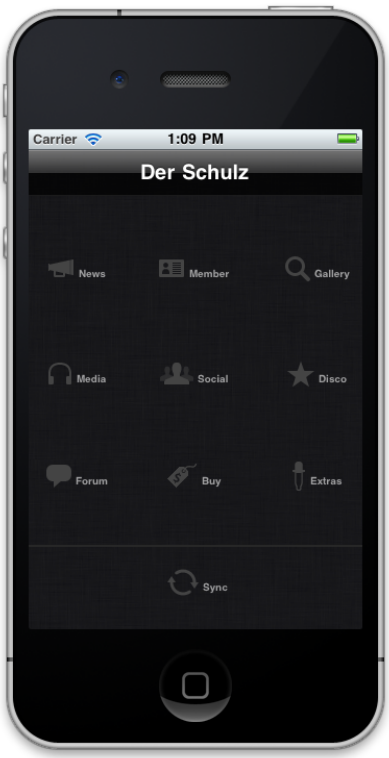


Abbildung 5: Beispiel: MyBand-App für iOS

Abbildung 6: Beispiel: MyBand-App für iOS (die Galerie)

6. LITERATURVERZEICHNIS

[1] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 280–293, New York, NY, USA, Nov. 2009. ACM.

[2] K. Czarnecki and U. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley Professional, 2000.

[3] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proceedings IEEE INFOCOM 2001, The Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Twenty years into the communications odyssey, 22-26 April 2001, Anchorage, Alaska, USA*, volume 3, pages 1548–1557, Los Alamitos, CA, USA, 2001. IEEE. available online: `http://www.sics.se/~lmfeeney/publications/Files/infocom01investigating.pdf`.

[4] H. Höpfner. Replication in Mobile Information Systems. In *Informatik bewegt*, volume P-19 of *Lecture Notes in Informatics (LNI)*, pages 590–593, Bonn, Germany, 2002. GI, Köllen Druck+Verlag GmbH.

[5] S. Kent. Model driven engineering. In M. Butler, L. Petre, and K. Sere, editors, *Integrated Formal Methods*, volume 2335 of *Lecture Notes in Computer Science*, pages 286–298. Springer Berlin / Heidelberg, 2002.

[6] J. Kjeldskov and C. Graham. A review of mobile hci research methods. In *Human-Computer Interaction with Mobile Devices and Services*, volume 2795 of *Lecture Notes in Computer Science*, pages 317–335. Springer Berlin / Heidelberg, 2003.

[7] G. H. Kuenning and G. J. Popek. Automated Hoarding for Mobile Computers. In W. M. Waite, editor, *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, pages 264–275, New York, NY, USA, 1997. ACM Press.

[8] K. C. K. Lee, H. V. Leong, and A. Si. Semantic query caching in a mobile environment. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(2):28–36, 1999.

[9] P. Nicopolitidis, A. S. Pomportsis, G. I. Papadimitriou, and M. S. Obaidat. *Wireless Networks*. John Wiley & Sons, Inc., New York, NY, USA, 2003.

[10] J. Nielsen. ipad usability: First findings from user testing. Website, May 2010. `http://www.useit.com/alertbox/ipad.html`.

[11] C. Pettey. Gartner Says Worldwide Mobile Device Sales Grew 13.8 Percent in Second Quarter of 2010, But Competition Drove Prices Down. Website, Aug. 2010. `http://www.gartner.com/it/page.jsp?id=1421013`.

[12] Q. Ren and M. H. Dunham. Semantic Caching and Query Processing. *Transactions on Knowledge and Data Engineering*, 15(1):192–210, Jan. 2003.

[13] X. Wang. *Wired and Wireless Networks*. Vdm Verlag Dr. Müller, Saarbrücken, Germany, 2007.

XQuery Framework for Interoperable Multimedia Retrieval

Mario Döller, Florian Stegmaier, Alexander Stockinger, Harald Kosch
Chair of Distributed Information Systems
University of Passau
94034 Passau, GERMANY
firstname.lastname@uni-passau.de

ABSTRACT

Multimedia retrieval relies on the underlying metadata format for effective querying of multimedia information. Most of the metadata formats are XML-based (for instance MPEG-7 or P/META). In this context, the XQuery query language is a natural choice for querying these data based on exact matches. However, XQuery lacks in expressing and evaluating multimedia specific requests (e.q., spatial, fuzzy requests). Therefore, the MPEG Query Format (MPQF), a novel XML based query language tuned for standardized multimedia requests, has been developed. Based on this, the paper introduces a MPQF aware XQuery framework which features a.) a plug-in architecture for external multimedia routines, b.) an automatic approach for MPQF to XQuery transformation and c.) an injection of information retrieval capabilities to XQuery (e.g., scoring, ranking). Besides, the framework can be adopted to any available XQuery repository and allows the retrieval in any XML based multimedia metadata format.

Categories and Subject Descriptors

H.2.4 [**Systems**]: Query Processing—*Multimedia Databases*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query Formulation*

Keywords

XQuery, MPEG Query Format, Multimedia retrieval

1. INTRODUCTION

Retrieving information in multimedia repositories is one of the major challenging tasks in the multimedia life cycle. Whenever, multimedia retrieval is discussed, one has to deal with the related metadata (formats) which are often XML based (e.g. MPEG-7[1]). In series, by investigating XML based retrieval techniques, one finally ends up by the well known and established XQuery[2] language. The XQuery language has its strengthens in expressing data centric and exact queries over XML data such as *Give me all images whose filesize > 100 kByte*, but lack the ability to express fuzzy requests common in multimedia retrieval (e.g., Query-By-Example). To fill this gap, a new multimedia query language, the MPEG Query Format [5] (MPQF) has been standardized in late 2008 by MPEG (formally known as ISO/IEC SC29 WG11). The query language addresses XML based metadata formats (e.g., MPEG-7) and combines data and information retrieval components as well as management functionalities.

In this context, the paper contributes with a XQuery framework for multimedia search that features a.) a plug-in architecture for external multimedia routines, b.) an automatic approach for MPQF to XQuery transformation and c.) an injection of information retrieval capabilities to XQuery (e.g., scoring, ranking). Besides, the framework can be adopted to any available XQuery repository and allows the retrieval in any XML based multimedia metadata format. Another benefit of adopting XQuery for multimedia retrieval is the broad diversity of available XQuery tools (databases, parsers, etc.).

The paper uses the MPEG Query Format (MPQF). The definition of the format is out of scope of this paper and has been published elsewhere. Readers not familiar with MPQF may look in [5] for detailed information. Further note, the specified transformation model presents only selected transformation rules and an extended version of this paper can be found at: `http://dimis.fim.uni-passau.de/iris/GI_Workshop_extended.pdf`.

The reminder of this paper is organized as follows: Section 2 introduces related work in the area of XQuery extensions for fuzzy retrieval. This is followed by Section 3 where the proposed MPQF to XQuery framework is described. In this context, Section 4 specifies our mapping approach for a MPQF to XQuery transformation. The specification is evaluated by a small example in Section 5. Performance analysis results are presented in Section 6. Finally, this article is concluded in Section 7.

2. RELATED WORK

As highlighted in the introduction, multimedia retrieval considers (to an high extend) multimedia metadata which is often XML-based (e.g., TV-Anytime, MPEG-7, etc.). In this context, in the past several query languages that are especially designed for XML data have been developed such as

[1] `http://mpeg.chiariglione.org/`

[2] `http://www.w3.org/TR/xquery/`

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

XIRQL [4], but the most well known representative approach is XQuery [8]. XML based query languages are strong in expressing data centric, but lack the ability to express fuzzy requests common in multimedia retrieval (e.g., Query-By-Example). There are already some approaches (e.g., VeXQuery [9]) aiming to extend XQuery in this direction. However, none of them is completely adequate for multimedia retrieval in terms of missing support for weighting of query terms (to reflect user preferences) or support for temporal or spatial retrieval etc. Further approaches that extend XQuery for fuzzy retrieval can be summarized as follows: Early works (e.g. [6]) introduced an XQuery rank operator for the evaluation of information retrieval request that target on an estimation of the *relative relevance of documents within document collections*. The integration of a vector space model and an associated *vscore* function has been presented in [7]. The *vscore* function returns the similarity degree between a query vector and a content element vector. Recently, in [3], the authors proposed a fuzzy XQuery processing technique which allows the users to use linguistic terms based user-defined functions in XQuery instances. The approach has been implemented in the native eXist XML database and provides better output than normal XQuery language execution.

3. MPQF BASED XQUERY FRAMEWORK

This section describes the overall structure and architecture of the proposed MPQF based XQuery framework.

3.1 Architecture

Figure 1 presents the overall architecture and workflow of the framework. First, the incoming MPQF request is parsed by an internal MPQF parser which establishes a visitor pattern syntax tree.

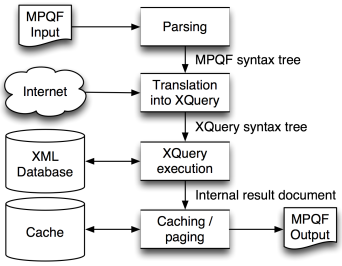


Figure 1: Overview of the system architecture

The MPQF syntax tree is forwarded to the transformation module which executes a mapping algorithm for producing an equivalent XQuery request. The plug-in system (see Subsection 3.2) supports the integration and evaluation of external information retrieval routines (e.g. query by example). After the finalization of the transformation process, a final XQuery instance is available. In series, this XQuery request is forwarded to the connected XQuery database for execution.

3.2 Plug-in system

In order to support a flexible system, the framework introduces a plug-in system. Figure 2 demonstrates where plug-ins take action in the transformation life cycle.

The framework identifies different categories of plug-ins: support for individual query types (e.g., QueryByMedia),

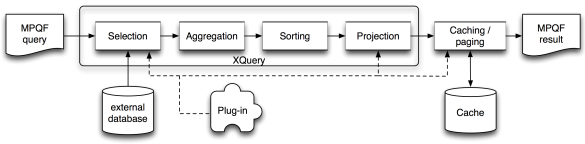


Figure 2: Overview of plug-in injection

extension of the XQuery language set (e.g., power function), data model dependent transformations (e.g., MIME type filtering) or functionality specific for the underlying XQuery database. In general, a plug-in is a Java module which receives a fixed set of input parameters and is able to manipulate the internal context of the transformation process. For instance, the plug-in module for our QueryByMedia implementation receives as input parameters the possible elements and attributes of the specified MPQF query type (e.g., *matchType* and *MediaResource*). Then, the module performs a similarity search at an external PostgreSQL[3] database where the ScalableColor features of MPEG-7 are stored. After finalization, the module responds with the results as presented by Transformation Rules 7 and 8.

4. MPQF TO XQUERY TRANSFORMATION RULES

The transformation formalism presented in this paper concentrates on the implementation of a global framework for fuzzy retrieval within XQuery based on the MPQF standard. In general, the rules distinguish between *data model depended* and *data model independent* transformation. That is, *data model depended* rules need to be adopted to the underlying metadata format as the required information can not be addressed by a given XPath[4] within the MPQF query request but need additional domain specific data.

4.1 Transformation Rules for Selection

The transformation rules for selection cover the mapping of the filtering step which is described by the *QueryCondition* element in an MPQF request. In this context, the skeleton for integrating a single MPQF condition (e.g., query type or comparison condition) into XQuery is implemented as follows:

TRANSFORMATION RULE 1 (QUERY CONDITION). *Let $\ll Score\_i \gg$ be a substitute for the transformation of an expression (e.g, comparison evaluation see Transformation Rule 6), a boolean operator (see Transformation Rule 5) or a query type which refers to external processing (see Transformation Rule 8) for e.g., information retrieval evaluation. Furthermore, let $\ll Threshold\_i \gg$ be the user given limit of the minimum similarity score for an operation (where $1 \leq i \leq n$ and $n \in \mathbb{N}$). Then the transformation of a MPQF query condition to XQuery is embedded as:*

```
let
  $scoreVar_i := <<Score_i>>,
  ...
where
  $scoreVar_i >= <<Threshold_i>>
```

Note, $\$scoreVar\_i$ is a variable keeping the score value of the evaluation.

[3] http://www.postgresql.org/
[4] http://www.w3.org/TR/xpath20/

Constitutively on the abstract transformation of query conditions to score values, a closer look to its specific substitutions has to be performed. As defined in MPQF, an individual query condition can contain comparison, string and arithmetic operations. Exemplarily for this set of operations, the contains operation (see Transformation Rule 2) and comparison operation (see Transformation Rule 3) are defined.

Transformation Rule 2 (Contains Operator).
Let $\ll Op_i \gg$ be the respective operands of a MPQF contains condition where $1 \leq i \leq 2$. Then the corresponding transformation to XQuery is defined as:

```
contains(<<Op1>>, <<Op2>>)
```

Transformation Rule 3 (Comparison Condition).
Let $\ll Operand\_i \gg$ be the respective operands of a MPQF comparison operation where $1 \leq i \leq 2$ and $\ll Operator \gg$ is the substitute of one of the defined MPQF operators (e.g., EQUAL to =). Then the transformation of a MPQF comparison condition to XQuery is defined as:

```
<<Operand_1>> <<Operator>> <<Operand_2>>
```

A special role plays the fuzzy boolean operators as they combine the results of preceding evaluations by the means of scoring functions. In this context, the fuzzy boolean operators of MPQF (OR, AND, etc.) are transformed as follows:

Transformation Rule 4 (Boolean operator).
Let $\ll scoreVar\_i \gg$ be the score value as described in Rule 1. Further, $\ll prefVar\_i \gg$ specifies the given user preference value (where i in [1 .. n] and $n \in \mathbb{N}$) for the respective operation. Then, the mapping of fuzzy boolean operators (AND, OR, XOR) follows the following interface:

```
<<OP>>
(<<scoreVar_1>>,<<prefVar_1>>,...,<<scoreVar_n>>,<<prefVar_n>>)
```

Based on the abstract definition for fuzzy boolean operators in Rule 4 one example for a fuzzy AND operator is specified in Rule 5.

Transformation Rule 5 (AND Boolean operator).
Let $\ll scoreVar\_i \gg$ and $\ll prefVar\_i \gg$ be specified as presented in Rule 4 and $scoreVar\_(N+1)$ the variable for holding the final result. Then, a mapping for the AND fuzzy boolean operator to a scoring function using the product t-norm is defined as follows:

```
$scoreVar_(N+1) := math:pow (
                   math:pow(<<scoreVar_1>>, <<prefVar_1>>)*
                   ...
                   math:pow(<<scoreVar_n>>, <<prefVar_n>>),
                   <<N>>)
```

Note, as the MPQF boolean operators rely on the fuzzy set theory, the scoring functions should cope the t-norm and t-conorm fuzzy logics [2], respectively. Further note, this Rule assumes that the XML engine provides specific mathematic libraries or is extensible in this direction. Otherwise, lookup tables containing precalculated values in the interval [0.0 .. 1.0] can be provided (e.g., has been applied as plug-in for the Berkeley DB XML).

The integration of data centric evaluations (e.g., comparison operators) which base on a true/false basis is applied by Transformation Rule 6.

Transformation Rule 6 (Expression).
Let $\ll Expression \gg$ be any expression derived by the Rules 2 and 3. Then the score of those operations is gathered by the following transformation:

```
if (<<Expression>>) then 1.0 else 0.0
```

For the evaluation of information retrieval related techniques (e.g., the QueryByMedia or SpatialQuery query type) separate external processes have to be applied. These processes filter the document set and produce document id and score value pairs which are integrated into the final XQuery request. For instance, as described in Subsection 3.2, the QueryByMedia query type can be implemented by forwarding this part of the MPQF request to a specialized similarity search engine for image retrieval by example. The result of such an evaluation is then integrated as specified by Rule 7.

Transformation Rule 7 (Plug-in Integration).
Let $qbmVar be the container variable for results of an external evaluation and $\ll anyURI\_i \gg$ an unique identifier of an XML instance document and $\ll anyScore\_i \gg$ its respective score value (where i in [1 .. n] and $n \in \mathbb{N}$). Then, the external result is integrated by the following format:

```
$qbmVar := (
  <qbm>
    <doc id='<<anyURI_1>>' score='<<anyScore_1>>'/>
    ...
    <doc id='<<anyURI_n>>' score='<<anyScore_n>>'/>
  </qbm>
)
```

Note, this approach assumes that there is a unique identifier for every document. However, parts of a description (e.g. low level features) can be swapped to specialized retrieval stores but the unique identifier remains as link between those parts.

Besides, the intermediate result set (stored in a variable) is integrated into the overall evaluation by Rule 8 supporting the access to already existing score results.

Transformation Rule 8 (Plug-in Evaluation).
Let $\ll qbmVar \gg$ be the container as specified in Rule 7. Then, access to individual score values is accomplished as follows:

```
if (exists($qbmVar/doc[@id = base-uri($doc)]))
then (number($qbmVar/doc[@id = base-uri($doc)]/@score))
else (0.0)
```

The *TargetMediaType* element of MPQF restricts the multimedia data set according to their mime type. This filtering is data model depended as no additional information (e.g. XPath to the data) is provided within the MPQF query itself. Therefore, the evaluation is embedded as follows:

Transformation Rule 9 (MIME type).
Let $\ll MIME\_type\_i \gg$ be defined as a MIME type description where i in [1 .. n] and $n \in \mathbb{N}$. Then, the filter criterion extends the XQuery where clause as follows:

```
where
  {<<MIME_type_1>> OR
  ....
   <<MIME_type_n>>} AND
```

Finally, the resulting documents are ordered by their score evaluation and stored in an internal format for further processing (see Rule 10).

Transformation Rule 10 (Ordering).
Let $scoreVarN contain the final score value after N calculation steps, then the resulting documents are ordered and preliminary stored as follows:

```
where
  ...
order by
   $scoreVarN descending
return
  <Doc score='{$scoreVarN}' id='{$id}'>{$doc}</Doc>
```

4.2 Transformation Rules for Projection

In a final step, the desired information is extracted from the filtered documents and integrated in a valid MPQF output instance. As the wanted elements are addressed as XPath expressions within an MPQF query, they can be recycled in the transformation as well. Note, the final MPQF query result is embedded in an internal proprietary format in order to support enhanced functionalities such as caching, paging, relevance feedback, etc. by the framework.

4.3 Transformation process

The so far introduced Transformation Rules (TR) describe techniques for mapping parts of a MPQF request to its equivalents in XQuery. The overall transformation process creates a rule chain during the evaluation of the query in order to map the entire MPQF request. Input of the chain is the MPQF request. Then, a post order traversal is applied which responds with a list of nodes (MPQF conditions) of the *QueryCondition* element. By parsing this list, the type of the current node is identified and the respective (set of) Transformation Rule(s) is/are accomplished. Then, the algorithm applies optional Rules for existing aggregation or sorting parts. Finally, the rest of MPQF's *OutputDescription* element is evaluated by applying the Projection rule (not shown in this paper). Finally, the output of this mapping process is an equivalent XQuery instance.

5. EXAMPLE TRANSFORMATION

For a better understanding of the defined Transformation Rules, a simple example transformation is demonstrated by the following MPQF query request (see Figure 3)[5]. The example request addresses MPEG-7 based image descriptions and selects the *title* and the *creator information* of all *JPEG* images whose file size is *greater or equal* to *500000* Bytes and where the creators *family name contains* the string *Bob*. Besides, the *threshold* of the combined score result must exceed *0.5*. Furthermore, the final result set should contain not more then *30* elements.

The processing engine of the incoming query tree applies a post order traversal to extract the internal nodes. For our example, this results on the following sequence: *Contains, GreaterThanEqual, AND, TargetMediaType*. Then, by traversing this sequence, the assigned transformation rules are executed.

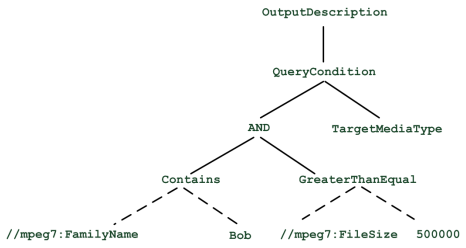


Figure 3: Example tree

In the following, Subsection 5.1 describes the used rules for the selection phase. Finally, in the project phase (see Subsection 5.2) the desired information is extracted and the entire XQuery is consolidated.

[5]Note, the request as MPQF language can be found in the extended version of this article.

5.1 Selection phase

The transformation mechanism starts by parsing all elements of the generated sequence. The first element is the *contains* condition which belongs to the set of *expressions*. and triggers the execution of Transformation Rule 2. This results in the following XQuery snippet (see Code 1).

Code 1 Transformation of the *contains* condition

```
contains($doc//mpeg7:FamilyName, 'Bob')
```

This is followed by Transformation Rule 6 which is used for the integration of expressions into XQuery (see Code 2)

Code 2 Integrating the *contains* condition

```
$scoreVar1 = if (contains($doc//mpeg7:FamilyName, 'Bob'))
          then 1.0 else 0.0
```

Similarly to the *contains* condition, the next element in the sequence is processed, namely the *GreaterThanEqual* condition. Here, in our example, the Transformation Rule 3 followed by Rule 6 are evaluated, which results in the XQuery snippet given in Code 3.

Code 3 Transformation of GreaterThanEqual

```
$scoreVar2 = if ($doc//mpeg7:FileSize >= 500000)
            then 1.0 else 0.0
```

After applying the Transformation Rules for the leaf nodes, our approach concentrates on the inner nodes (the boolean operators). The inner nodes regulate how the results of the leaf nodes are combined. For this purpose, scoring functions are assigned to the respective boolean operators according to t-norm and t-conorm rules. Our example uses the *Product* function for the AND operation in order to combine the individual score values. In this context, applying Transformation Rule 5 results in Code 4 for the *AND* condition.

The resulting score value and the threshold of an condition are integrated into the XQuery request by evaluating Transformation Rule 1. Due to space constraints in this paper, the final result for integrating the given thresholds is shown in Code 5. If no threshold is assigned, the minimum value is used (0).

As our example makes use of the *TargetMediaType* element for restricting the result set according to the file format, a data model depended filtering has to be found. In our example, we assume that the respective information is annotated in the *Content* and *FileFormat* elements of the MPEG-7 description. In assuming so, the following transformation for MIME-types (see Rule 9) has to be applied (see Code 6).

The final result of all combined transformations of the selection phase can be found in Code 7.

5.2 Projection phase

The last stage in the transformation process is the creation of a valid MPQF response and the integration of the requested information of the target data model. Our example only instantiates the *TextResult* and *Description* elements. As described beforehand, the final MPQF output description is wrapped in a proprietary format (*ResultDocu-*

Code 4 Applying Transformation Rule for scoring function

```
$scoreVar3 = math:pow(math:pow($scoreVar1, 0.5)*
          math:pow($scoreVar2, 0.5), 2)
```

Code 5 Integration of threshold values

```
$scoreVar1 >= 0.0 and
$scoreVar2 >= 0.0 and
$scoreVar3 >= 0.5
```

ment element) in order to support caching, paging, relevance feedback, etc.

6. EVALUATION

This section describes the series of experiments we performed in order to evaluate the effectiveness of our transformation approach. The tests were carried out on a subset of the CoPhiR[6] [1] data set containing MPEG-7 annotations of Flickr images. The sizes of our test data sets varied from 100 up to 10000 annotations. In order to demonstrate the transformation approach with various XML databases, the following solutions have been chosen: Saxon[7], Berkeley DB XML[8] and eXist DB[9].

The overall performance evaluation is divided into two main parts. First, the processing time of parsing and executing the Transformation Rules has been analyzed in Subsection 6.1. The final execution of the resulting XQuery instance at the mentioned databases is demonstrated in Subsection 6.2.

6.1 Transformation Evaluation

This subsection describes the set of experiments for evaluating the performance of applying the Transformation Rules. Input is a MPQF query request and output an equivalent XQuery query request. In order to receive clear differences between the used query classes a less powerful system configuration has been applied, namely an Intel Premium M 1.60 GHz CPU with 512 MB DDR2 400 main memory running Windows XP.

The complexity of the queries is divided into the following six classes in order to demonstrate the evaluation time for different compositions of Transformation Rules: queries where either aggregation or sorting is used (classes NoAgg/Sort, NoAgg/Sort Complex, Agg/NoSort), queries where both (class Agg/Sort) and queries where none (class NoAgg/NoSort) of these features have been used. Except the NoAgg/Sort Complex class, the example queries contain only one condition (e.g. EQUAL condition). The complex query class demonstrates the use of Boolean operators (AND/OR) and multiple other conditions (e.g., contains or comparison).

Finally, the sixth query class addresses the performance of the plug-in system by demonstrating a QueryByMedia query type which requires external processing. The external processing has been realized by the integration of a relational

[6] http://cophir.isti.cnr.it
[7] http://saxon.sourceforge.net
[8] http://www.oracle.com/database/berkeley-db/xml/index.html
[9] http://exist.sourceforge.net/

Code 6 Transformation of MIME type filtering

```
(
exists($doc//mpeg7:Content[@href = 'image'])
and
string($doc//mpeg7:FileFormat/mpeg7:Name/text()) = 'JPEG'
)
```

Code 7 Result of selection phase

```
$selected :=
 (for
   $doc in collection('db.dbxml')/*
  let
    $scoreVar1 = if (contains($doc//mpeg7:FamilyName, 'Bob'))
              then 1.0 else 0.0,
    $scoreVar2 = if ($doc//mpeg7:FileSize >= 500000)
              then 1.0 else 0.0,
    $scoreVar3 = math:pow(math:pow($scoreVar1, 0.5) *
              math:pow($scoreVar2, 0.5), 2),
    $id := base_uri($doc)
  where
   ((
    exists($doc//mpeg7:Content[@href = 'image'])
    and
     string($doc//mpeg7:FileFormat/mpeg7:Name/text()) = 'JPEG'
   )) and
   $scoreVar1 >= 0.0 and
   $scoreVar2 >= 0.0 and
   $scoreVar3 >= 0.5
  order by
    $scoreVar3 descending
  return
   <Doc score='{$scoreVar3}' id='{$id}'>{$doc}</Doc>),
```

PostgreSQL[10] database coping with low level features (color, texture, etc.) of images. Similarity calculation has been simplified on the basis of color features and the Euclidean distance (no index has been used). Figure 4 presents the average run time needed for the transformation of an MPQF query request to an appropriate XQuery request. The tests have been repeated 50 times. The evaluation shows that there is a slight increase of time consumption depending on the increase of query complexity. However, the maximum differences between query classes do not increase 17%.

6.2 Database comparison

The experiments have been executed on a Windows based stand-alone PC with Intel Core i7 1,6GHz (4 cores) CPU and 4 GB main memory. All databases have been tested out of the box without optimization. Similar to Subsection 6.1 six different query classes have been used during the evaluation, whereas exemplarily only two are demonstrated in this article. Note, the presented performance behavior is also valid for the rest of the tests. Figure 5 show the results of our experiments for a query where sorting has been enabled. The y-axis describes the average processing time per document (average processing time divided by the amount of documents in the database) and the x-axis shows the amount of MPEG-7 documents stored in the database. The measured overall processing time for one MPQF query consists of the transformation time applied by our module and the time needed for processing the resulting XQuery.

By evaluating the performance results, one can identify a linear scaling of the Saxon and Berkeley DB XML engines, which is stable over the increasing size of the test data set. The impact of the initial phase needed by the engines can

[10] http://www.postgresql.org/

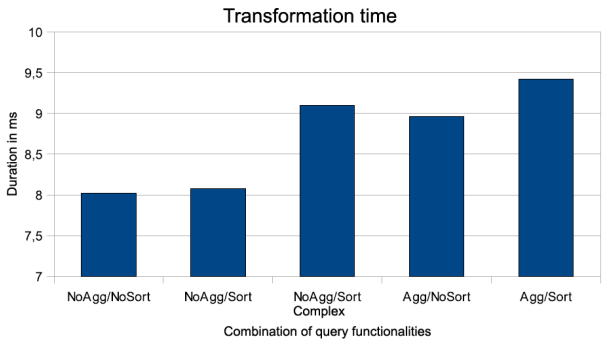


Figure 4: Performance of the transformation process

be observed by small test data sets (100 documents) and here the Berkeley DB XML is outperformed by the others. In contrast to Saxon and Berkeley DB, the eXist engine is outperformed clearly for larger test data sets as it shows a nearly quadratic scale factor.

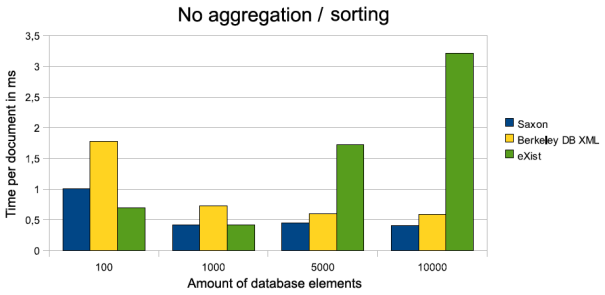


Figure 5: Comparison of database performance for XQuery with sorting

In general, the processing time for the MPQF-XQuery transformation is in average under 7% of the overall processing time for evaluating the final XQuery request and therefore negligible.

The last experiment targets on the evaluation of the plug-in injection process by the means of a QueryByMedia query type which realizes similarity search on multimedia data. As described beforehand, the test environment consists of a PostgresSQL database storing the low level features (ScalableColor of MPEG-7). As a proof of concept, similarity search is implemented by an SQL function in the target database. Of course, here is room for improvements (e.g., use of index structures or enhanced multimedia retrieval modules). Figure 6 show the results for the QueryByMedia evaluation.

7. CONCLUSIONS

This article proposed a MPQF based XQuery framework which provides a specification of a set of Transformation Rules for mapping a MPEG query format request to an equivalent XQuery request. Based on this, a framework has been developed featuring a plug-in system for external multimedia retrieval routines, a threading model for fast and scalable processing and an internal result set format enabling caching, paging and relevance feedback operations. The framework is able to connect to any available XQuery repository. By using the proposed framework, XQuery repositories can be enhanced for multimedia retrieval on any XML based multimedia metadata format in a standardized way.

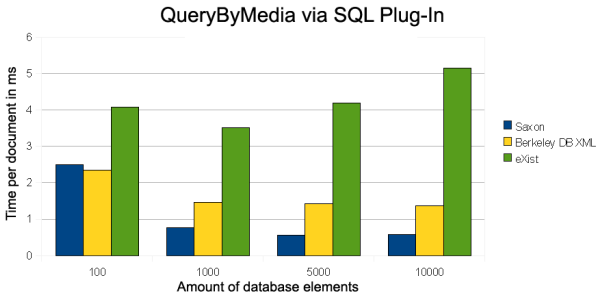


Figure 6: Comparison of database performance for XQuery with QueryByMedia plug-in

Future work will concentrate on further developments for the projection and output description part and the integration of additional plug-in elements coping for instance spatial and temporal retrieval.

8. ACKNOWLEDGMENTS

This work has been supported in part by the THESEUS Program, which is funded by the German Federal Ministry of Economics and Technology.

9. REFERENCES

[1] Paolo Bolettieri, Andrea Esuli, Fabrizio Falchi, Claudio Lucchese, Raffaele Perego, Tommaso Piccioli, and Fausto Rabitti. CoPhIR: a test collection for content-based image retrieval. *CoRR*, abs/0905.4627v2, 2009.

[2] Didier Dubois, Henri Prade, and Florence Sedes. Fuzzy Logic Techniques in Multimedia Database Querying: A Prelimiary Investigation of the Potentials. *IEEE Transaction on Knowledge and Data Engineering*, 13(3):383–392, 2001.

[3] E.J. Thomson Fredrick and G. Radhamani. Fuzzy Logic Based XQuery operations for Native XML Database Systems. *International Journal of Database Theory and Application*, 2(3):13–20, 2009.

[4] Norbert Furh and Kai Grossjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In *Proceedings of the 24th ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 172–180, New Orleans, Louisiana, USA, 2001. ACM Press.

[5] ISO/IEC. Information technology - Multimedia content description interface - Part 12: Query format. *ISO/IEC 15938-12:2008*, 2008.

[6] Ji-Hoon Kang, Chul-Soo Kim, and Eun-Jeong Ko. An XQuery engine for digital library systems. In *Proceedings of the 3rd International ACM/IEEE-CS joint conference on Digital libraries*, pages 400–400, Houston Texas, 2003.

[7] Jacques Le Maitre. Indexing and Querying Content and Structure of XML Documents According to the Vector Space Model. In *Proceedings of the IADIS International Conference WWW/Internet*, pages 353–358, Lisbon, Portugal, 2005.

[8] Priscilla Walmsley. *XQuery*. O'Reilly Media, 2007. ISBN: 978-0596006341.

[9] Ling Xue, Chao Li, Yu Wu, and Zhang Xiong. VeXQuery: an XQuery extension for MPEG-7 vector-based feature query. In *Proceedings of the International Conference on Signal-Image Technology and Internet Based Systems (IEEE/ACM SITIS'2006)*, pages 176–185, Hammamet, Tunesia, 2006. Springer-Verlag.

Workload Representation across Different Storage Architectures for Relational DBMS

Andreas Lübcke
School of Computer Science
University of Magdeburg
Magdeburg, Germany
andreas.luebcke@ovgu.de

Veit Köppen
School of Computer Science
University of Magdeburg
Magdeburg, Germany
veit.koeppen@ovgu.de

Gunter Saake
School of Computer Science
University of Magdeburg
Magdeburg, Germany
gunter.saake@ovgu.de

ABSTRACT

Database systems differ from small-scale stripped database programs for embedded devices with minimal footprint to large-scale OLAP applications for server devices. For relational database management systems, two storage architectures have been introduced: the row-oriented and the column-oriented architecture. To select the optimal architecture for a certain application, we need workload information and statistics. In this paper, we present a workload representation approach that enables us to represent workloads across different DBMSs and architectures. Our approach also supports fine granular workload analyses based on database operations.

1. INTRODUCTION

New requirements for database applications [23, 26, 27] came up in recent years. Therefore, database management system (DBMS) vendors and researchers developed new technologies, e.g., column-oriented DBMSs (*column stores*) [1, 22, 30]. New approaches are developed to satisfy the new requirements for database applications, thus the number of candidates in the decision process has also increased. Moreover, new application fields imply a more complex decision process to find the suitable DBMS for a certain use case.

We need statistics to come to a suitable design decision. These statistics have to be represented system-independent for sound and comparable decision. That implies the independence of workload representation from different storage architectures. In this paper, we introduce a new approach of workload statistics aggregation and maintenance across different DBMSs and architectures. We showed in [16] that query-based workload analyses, as described in [7], are not suitable to select the optimal storage architecture. To overcome drawbacks of query-based workload analyses, we define workload patterns based on database operations. We introduce a workload decomposition algorithm that enables us to analyze query parts. Workload patterns represent the decomposed workloads to compare the performance of database operations for column and row stores. These workload patterns contain all statistics needed for cost estimations. We simulate the statistic gathering process with a exemplary workload.

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

2. STATISTICS REPRESENTATION

To select the optimal storage architecture, we have to analyze a given workload; thus, we need to decompose this workload. We have to map single operations of a workload (at least of one query) and their optimizer statistics to evaluable patterns. Therefore, we present our pattern framework which stores all necessary statistics for subsequent performance analyses. In [18], we illustrate the procedure of our decision process regarding the storage architecture selection. Below, we outline the design of our pattern framework.

2.1 Pattern Types

To analyze the influence of single operations, we propose three patterns for operations in workload queries. The three operation patterns are ***tuple operations***, ***aggregations and groupings***, and ***join operations***. We define a number of sub-patterns for each of those three to characterize particular operations more precisely within the patterns. This way, we support analyses based on the three patterns and additionally fine granular analyses based on sub-patterns, i.e., we can determine where the majority of costs emerge within a workload (at least one query).

First, the ***tuple operation pattern*** covers all operations that process or modify tuples, e.g., selection, sort operations. We propose this pattern for performance analyses because row stores process directly on tuples in contrast to column stores that costly reconstruct tuples. We identify the following sub-patterns:

Sort/order operation: Sort/order operation creates sequences of tuples and affects all attributes of a tuple. We consider duplicate elimination as a sort operation because an internal sort is necessary to find duplicates.

Data access and tuple reconstruction: Row stores always access tuples and column stores must reconstruct tuples to access more than one column.

Projection: Projection returns a subset of tuple attribute values and causes (normally) no additional costs for query execution.

Filtering: Filtering selects tuples from tables or intermediate results based on a selection predicate, e.g., selection in WHERE-clause and HAVING-clause.

Second, we cover all column processing operations in the ***aggregation and grouping pattern***, e.g., COUNT and MIN/MAX. We propose this pattern as counterpart to the tuple operation pattern. The operations of this pattern work only on single columns except for grouping operations which can also process several columns, e.g., GROUP BY CUBE. Due to column-wise partitioned data and single column processing, column stores perform well on aggregations (cf. [16]). We identify the following sub-patterns:

Min/Max operation: The min/max operation provides the minimum/maximum value of a single attribute (column).

Sum operation: This operation provides the sum of all values in one column.

Count operation: The count operation provides the number of attribute values in a column and COUNT(*) provides only the number of key values, thus it processes a single column.

Average operation: The average operation computes all values of a single column as well as the sum operation, but it can have different characteristics, e.g., mean (avg) or median.

Group by operation: This operation merges equal values according to a certain column and results in a subset of tuples. Grouping across a number of columns is also possible.

Cube operations: The cube operation computes all feasible combination of groupings for selected dimensions. This generation requires the power set of aggregating columns, i.e., n attributes are computed by 2^n GROUP BY clauses.

Standard deviation: The standard deviation (or variance) is a statistical measure for the variability of a data set and is computed by a two pass algorithm which means two cycles.

Third, the ***join pattern*** matches all join operations of a workload. Join operations are costly tasks for DBMSs. This pattern shows differences of join techniques between column and row stores, e.g., join processing on compressed columns or on bitmaps. Within this pattern, we evaluate the different processing techniques against each other. Consequently, we define the following sub-patterns:

Vector based: The column oriented architecture naturally supports vector based join techniques while row stores have to maintain and create structures, e.g., bitmap (join) indexes [15].

Non-vector based: This pattern matches "classic" join techniques (from row stores[1]) to differentiate the performance between vector and non-vector based join, thus we can estimate effects on the join behavior by architecture.

We only propose these two sub-patterns because the join concepts, e.g., merge or nested loop join, exist for both architectures. Hence, we assume that there is no necessity to map each join concept into its own sub-pattern. Figure 1 shows all introduced patterns and their relation to each other based on our exemplary workload.

2.2 Dependencies between Patterns

Database operations are not always independent from each other. We can identify dependencies between the following patterns: join, filtering, sort/order, group/cube, and data access pattern.

Join operations innately imply tuple selections (filtering pattern). However, the tuple selection itself is part of the join operation by definition, thus we assume that an additional decomposition of join operations is not necessary. Moreover, new techniques would have to be implemented to further decompose join operations and gather the necessary statistics. Hence, the administrative cost for tuning will be noticeably increased. To a side-effect, the comparison of join techniques belonging to different architectures will be no longer possible because of system-specific decomposition.

We state that two different types of sort/order operation can occur, i.e., implicit and explicit sort. The explicit sort is caused by workload or user, thus we consider this operation in the sort/order pattern. In contrast, we do not consider the implicit sort operation in the sort/order pattern because this sort operation is caused by the optimizer, e.g., for sort-merge join. Therefore, we assign all costs of grouping to the GROUP BY (or CUBE) pattern including the sort costs to sustain comparability.

Third, tuple reconstruction is part of several operations for column stores. We add these costs to the tuple operation pattern. We sustain the comparability of operations beyond the architectures because row stores are not affected by tuple reconstructions.

We assume further workload decomposition is not meaningful because administrative costs would affect the performance of existing systems as well as the comparability of performance issues between the architectures according to certain workload parts. These impacts would disadvantageously affect the usability of our pattern framework.

[1]Some column stores also support these join techniques.

3. QUERY DECOMPOSITION

In this section, we introduce the query decomposition approach. First, we illustrate the (re-) used DBMS functionality and how we gather necessary statistics from existing systems. Second, we introduce the mapping of decomposed query parts to our established workload patterns and show a decomposition result by example. Our approach is applicable to each relational DBMS. Nevertheless, we decide to use a closed source system for the following considerations because the richness of detail of optimizer/query plan output is higher and easier to understand. More detailed information will result in more accurate recommendation.

3.1 Query Plans

A workload decomposition based on database operations is necessary to select the optimal storage architecture (cf. [16]). Therefore, we use query plans [4] which exist in each relational DBMS. On the one hand, we reuse database functionality and avoid new calculation costs for optimization. On the other hand, we make use of system optimizer estimations that are necessary for physical database design [10].

Based on query plans, we gather statistics directly from DBMS and use the optimizer cost estimations. The example in Listing 1 shows an SQL query and we transform this to a query plan in Table 1 [19]. Table 1 already offers some statistics such as number of rows, accessed bytes by the operation, or costs. Nevertheless, Table 1 shows only an excerpt of gathered statistics. All possible values for query plan statistics can be found in [20] Chapter 12.10. Hence, we are able to determine the performance of operations on a certain architecture (in our example a row store) by statistics such as CPU costs and/or I/O costs. In addition to performance evaluation by several estimated costs, we can gather further statistics from query plans which may influence performance of an operation on a certain architecture, e.g., cardinality. For column stores, the operation cardinality can indirectly affect performance if the operation processes several columns, thus column stores have to process a number of tuple reconstructions, e.g., high cardinality means many reconstructions. Thus, we use meta-data to estimate influences of data itself on the performance, e.g., we can compute the selectivity of attributes.

3.2 From Query Plans to Workload Patterns

We have to map the gathered statistics from DBMS to our workload patterns. We use a second example [21] (Listing 2 and Table 2) to simulate a minimum workload instead of a single query. In the following, we illustrate the mapping approach by using the examples in Listing 1 and 2. In our name convention, we define a unique number[2] that identifies queries of the workload within our mapping algorithm, i.e., 1.X represents query 1 (Listing 1) and equally 2.X represents query 2 (Listing 2). Furthermore, we reuse the operation IDs from query plans (Table 1 and 2) in the second hierarchy

[2]In the following considerations, we start with 1 which represents the first query.

```
1 SELECT *
2 FROM employees e JOIN departments d
3 ON e.department_id=d.department_id
4 ORDER BY last_name;
```

Listing 1: Example SQL query (14-1) [19]

ID	Operation	Name	Rows	Bytes	Cost (%CPU)	...
0	SELECT STATEMENT		106	9328	7 (29)	...
1	SORT ORDER BY		106	9328	7 (29)	...
* 2	HASH JOIN		106	9328	6 (17)	....
3	TABLE ACCESS FULL	DEPARTMENTS	27	540	2 (0)	...
4	TABLE ACCESS FULL	EMPLOYEES	107	7276	3 (0)	...

Table 1: Textual query plan of SQL example (14-1) [19]

level (for X), e.g., 1.4 is the operation with `ID 4` of query 1 (cf. Table 1). In the following, we refer the CPU cost of Table 1 and 2.

The first query (Listing 1) is decomposed into four patterns. First, we see the data access operation of the `department (ID 3)` and the `employees (ID 4)` tables in the corresponding query plan in Table 1. The total cost for the data access operations is `5`. Second, the join operation `(ID 2)` is executed with a hash join algorithm. The hash join cost is only `1` because in Table 1 costs are iteratively sum up and the costs of its children `(5)` and its own cost `(1)` are summed up to `6` for `ID 2`. Third, the sort operation `(ID 1)` implements the ORDER BY statement with cost of `1`. The total costs of all processed operations are `7` now. Fourth, the select statement `(ID 0)` represents the projection and causes no additional cost (remain `7`). Following our name convention, the identifiers from `1.0` to `1.4` represent the operations of our first query (Listing 1) in Figure 1.

We also decompose the second example (Listing 2) into four operation types (cf. Table 2). First, `IDs 3, 7, and 8` represent the data access operations and cause total costs of `14`. Second, the optimizer estimates both hash joins `(ID 2 and 6)` with no (additional) costs because their costs are only composed by the summed costs of their children (`ID 3, 4` and `ID 7, 8`). Third, the GROUP BY statement in Listing 2 is implemented by hash-based grouping operations (`ID 1` and `ID 5`). The cost of each HASH GROUP BY is `1` and the total costs of this operation type are `2`. Fourth, the projection `(ID 0)` and the sum operation represented by select statement causes again no additional costs. If the sum operation causes costs then it will be represented by a separate operation (`ID`). Following our name convention, the identifiers from `2.0` to `2.8` represent the operations of the second query (Listing 2) in Figure 1. The view (`ID 2.4`) is not represented in our workload pattern because its costs are already mapped by its child operations (`ID 2.5-2.8`).

In our examples, we summarize single operations of similar types (five for example query two). In the following, we list the five operation types and assign them to our workload patterns and their sub-patterns that we introduced in Section 2. The join operations of our example queries `ID 1.2, 2.2,` and `2.6` are assigned to the non-vector based join pattern. We assign the operations with `ID 1.3, 1.4, 2.3, 2.7,` and `2.8` to the data access sub-pattern of the tuple operation pattern. We also assign the projections (`ID 1.0` and `2.0`) and the sort operation `(ID 1.1)` to the tuple operation pattern. Finally, we assign the group by operations (`ID 2.1` and `2.5`) to the group by sub-pattern within the aggregation and grouping pattern. We present the result in Figure 1 whereby we only show ID and cost of each operation for reasons of readability. We state that the we do not need to directly extract statistics from existing systems. Our pattern framework is system independent, thus we are also able to use already extracted (or aggregated) data as well as estimated values.

3.3 Operations in Column Stores

We state that we do not need a separate decomposition algorithm for column stores, i.e., the query plan operations of column stores can be also mapped to our workload patterns. Representatively, we illustrate the mapping of C-Store/Vertica query plan operations introduced in [25] and map them to our workload patterns as follows:

Decompress: Decompress is mapped to the data access pattern. This operation decompresses data for subsequent operations in the query plan that cannot process on compressed data (cf. [1]).

Select: Select is equivalent to the selection of relational algebra with the exception that the result is represented as bitstring. Hence, we map it to the filtering pattern.

Mask: Mask process on bitstrings and returns only those values whose associated bits in the bitstring are 1. Consequently, we map mask to the filtering pattern.

Project: Projection is equivalent to the projection of relational algebra, thus this operation is mapped to the projection pattern.

Sort: This operation sorts the columns of a C-Store projection according to a (set of) sort column(s). This technique is equivalent to sort operations on projected tuples, i.e., we can map this operation to the sort/order pattern.

Aggregation Operators: These operations compute aggregations and groupings like in SQL [1], thus we directly map these operations to the corresponding sub-pattern in the aggregation & grouping pattern.

Concat: Concat combines C-Store projections sorted in the same order into a new projection. We regard this operation as tuple reconstruction and map it to the corresponding pattern.

Permute: This operation permutes the order of columns in C-Store projections according to the given order by a join index. It prevents additional replication overhead that would emerge through creation of join indexes and C-Store projections in several orders. This operation is used for joins, thus we map its cost to the join pattern.

Join: We map this operation to the join pattern and distinguish two join types. First, if tuples are already reconstructed then we process them as row stores, i.e., we map this join type to the non-vector based join pattern. Second, the join operation only processes columns that are needed to evaluate the join predicate. The join result is a set of pairs of positions in the input columns [1]. This join type can process on compressed data as well as it can use vector based join techniques, thus, we map this join type to the vector based join pattern.

Bitstring Operations: These operations (AND, OR, NOT) process bitstrings and compute a new bitstring with respect to the corresponding logical operator. These operations implement the concatenation of different selection predicates. Therefore, we map these operations to the filtering pattern.

Finally, we state that our approach can be used for each relational DBMS. Each relational DBMS is referable to the relational data model, so these DBMSs are based on the relational algebra in some manner too. Thus, we can reduce or map those operations to our workload patterns; in worst case, we have to add an architecture-specific operation for hybrid DBMSs to our pattern, e.g., tuple reconstruction for column stores. For a future (relational)

```
1 SELECT c.cust_last_name, SUM(revenue)
2 FROM customers c, v_orders o
3 WHERE c.credit_limit > 2000
4 AND o.customer_id(+) = c.customer_id
5 GROUP BY c.cust_last_name;
```

Listing 2: Example SQL query (11-9) [21]

ID	Operation	Name	Rows	Bytes	Cost (%CPU)	...
0	SELECT STATEMENT		144	4608	16 (32)	...
1	HASH GROUP BY		144	4608	16 (32)	...
* 2	HASH JOIN OUTER		663	21216	15 (27)	...
* 3	TABLE ACCESS FULL	CUSTOMERS	195	2925	6 (17)	...
4	VIEW	V_ORDERS	665	11305		...
5	HASH GROUP BY		665	15960	9 (34)	...
* 6	HASH JOIN		665	15960	8 (25)	...
* 7	TABLE ACCESS FULL	ORDERS	105	840	4 (25)	...
8	TABLE ACCESS FULL	ORDER_ITEMS	665	10640	4 (25)	...

Table 2: Textual query plan of SQL example (11-9) [21]

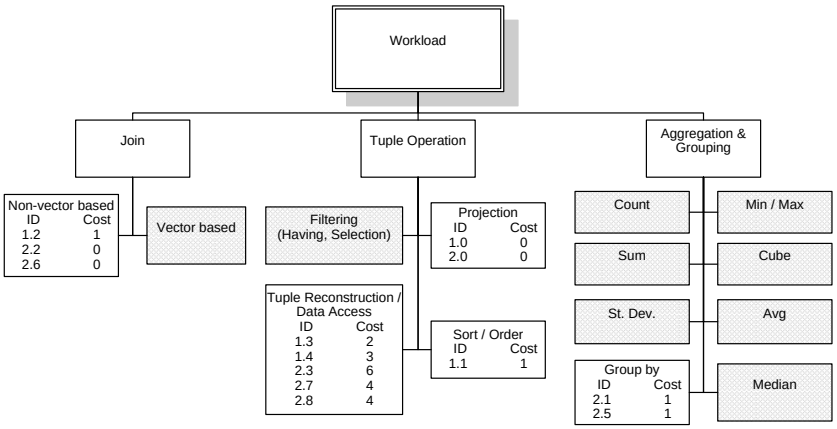


Figure 1: Workload patterns with cost of operations for the row store example workload

hybrid storage architecture, such an operation could be necessary to map the cost for conversions between row- and column-oriented structures and vice versa.

4. DEMONSTRATING EXAMPLE

We decide to simulate the workload with the standardized TPC-H benchmark (2.8.0) to show the usability of our approach. We use the DBMSs Oracle 11gR2 Enterprise Edition and Infobright ICE 3.3.1 for our experiments[3]. We run all 22 TPC-H queries and extract the optimizer statistics from the DBMSs. For reasons of clarity and comprehensibility, we only map three representative TPC-H queries namely `Q2`, `Q6`, and `Q14` to the workload patterns.

The query structure, syntax, and execution time are not sufficient to estimate the query behavior on different storage architectures. We introduced an approach based on database operations that provides analyses to find long running operations (bottlenecks). Moreover, we want to figure out reasons for the behavior, thus we have to use additional metrics. We select the I/O cost to compare the DBMSs and summarize the optimizer output in Table 3. Following our previous name convention, we define the query `IDs` according to their TPC-H query number, i.e., we map the queries with the `IDs 2, 6,` and `14`. The operations are identified by their query plan number (IDs in Table 3), thus the root operation of TPC-H query `Q2` has the `ID 2.0` in Figure 2. All values in Table 3 are given in Kbytes. The given values are input costs of each operation except the table access costs because no information on input costs to table access operations are available. Note, the granularity of Oracle's costs measurements is on the byte level whereas the measurements of ICE are on the data pack (65k) level.

In Figure 2, we present our workload patterns with I/O costs of the corresponding TPC-H queries. As we mentioned before, the projection operation causes no additional costs. Hence, the I/O costs in Table 3 and Figure 2 represent the size of final results. The stored information can be analyzed and aggregated in decision models with any necessary granularity. In our example, we only sum up all values of the data access pattern for each query to calculate the I/O costs per query in Kbytes. For these three queries, all results and intermediate results are smaller than the available main memory, thus no data has to be reread subsequently. Oracle reads 1452.133 Kbytes for query `Q2` and takes 8.14 seconds. ICE needs 41 seconds and access 2340 Kbytes. We suppose, the DBMS with minimal I/O cost performs best. Our assumption is confirmed for query `Q14`. Oracle accesses 7020.894 Kbytes and computes the query in 22.55 seconds whereas ICE computes it in 3 seconds and reads 6240 Kbytes. Nevertheless, we cannot prove our assumption for query `Q6`. Oracle (3118 Kbytes) accesses less data than ICE (5980) Kbytes but ICE (2 seconds) computes this query ten times faster than Oracle (22.64 seconds). Hence, we cannot figure out a definite correlation for our sample workload.

We state that only I/O cost is not sufficient to estimate the behavior of database operations. However, I/O cost is one important metric to describe performance behavior on different storage architectures because one of the crucial achievements of column stores is the reduction of data size (i.e., I/O cost) by aggressive compression. The I/O cost also gives an insight into necessary main memory for database operations or if operations have to access the secondary memory. Hence, we can estimate that database operations are completely computed in main memory or data have to be reread/read stepwise[4].

[3]We also wanted to evaluate our approach with the DBMSs solutions from Vertica and Sybase because both DBMSs use cost-based optimizer and we would be able to receive more expressive results. We requested the permission to use the systems for our evaluation but until now the decision is pending.

[4]We remind of the performance gap (circa 10^5) between main memory and HDDs.

Operation	Oracle			ICE		
	Q2 (8.14sec)	**Q6** (22.64sec)	**Q14** (22.55sec)	**Q2** (41sec)	**Q6** (2sec)	**Q14** (3sec)
Data Access	*ID7:*0.8;*ID12:*0.029;*ID13:*11.2; *ID15:*0.104;*ID16:*1440	*ID2:*3118	*ID3:*1620.894; *ID4:*5400	*ID4:*65;*ID5:*65;*ID6:*845;*ID7:*65*ID8:*260; *ID10:*65;*ID11:*65;*ID12:*65;*ID13:*845	*ID2:*5980	*ID3:*5980; *ID4:*260
Non-vector based join	*ID6:*202.760;*ID8:*1440;*ID9:*88.016; *ID10:*17;*ID11:*11.229		*ID2:*7020.894	*ID3:*1300;*ID9:*1040		*ID2:*6240
Sort	*ID3:*33.18;*ID5:*45.346			*ID2:*65		
Count	*ID1:*31.284			*ID1:*65		
Sum		*ID1:*3118	*ID1:*3610.173		*ID1:*5980	*ID1:*65
Projection	*ID0:*19.800	*ID0:*0.020	*ID0:*0.049	*ID0:*65	*ID0:*65	*ID0:*65

Table 3: Accessed Kbytes by query operations of TPC-H query Q2, Q6, and Q14.

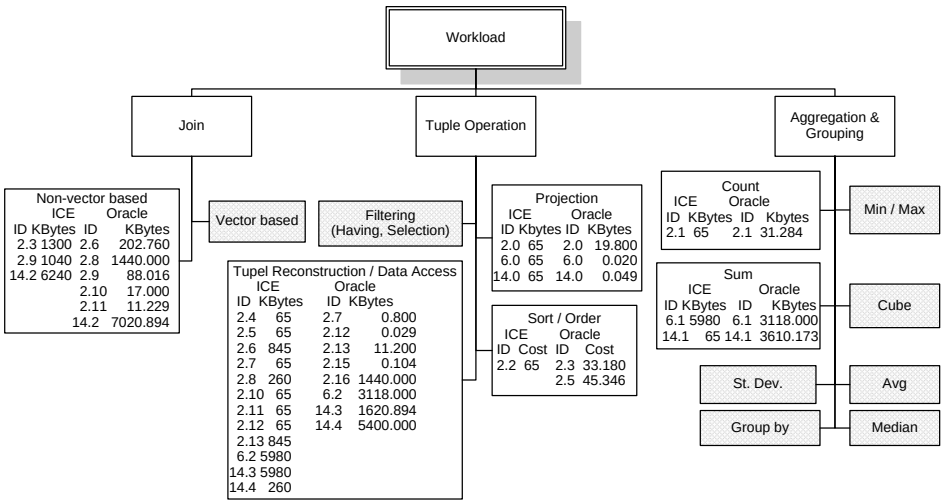


Figure 2: Workload graph with mapped I/O costs of TPC-H query Q2, Q6, and Q14.

5. RELATED WORK

Several column stores have been proposed [1, 14, 30] for OLAP applications. But all systems are pure column stores and do not support any row store functionality. Thus, a storage architecture decision between row and column store is necessary. Abadi et al. [2] compare row and column store with respect to performance on the star schema benchmark. They simulate column store architecture by indexing every single column or vertical partitioning of the schema. They show that using column store architecture in a row store is possible but the performance is poor. In this paper, we do not compare end to end performance of DBMSs or architectures. We support sound and comparable analyses based on database operations across different DBMSs with our approach. We do not discuss approaches like DSM [8], hybrid NSM/DSM schemes [9], or PAX [3] because the differences to state-of-the-art column stores have been already discussed, e.g., Harizopoulus et al. [11].

There are systems available which attempt to fill the gap between a column and a row store. C-Store [1] uses two different storage areas to overcome the update problems of column stores. A related approach brings together a column store approach and the typical row store domain of OLTP data [24]. However, we do not develop hybrid solutions that attempt to fill this gap for now.

There exist a number of design advisors which are related to our work, e.g., IBM DB2 Configuration Advisor [13]. The IBM Configuration Advisor recommends pre-configurations for databases. Zilio et al. [28, 29] introduce an approach that gathers statistics like our approach directly from DBMSs. The statistics are used to advise index and materialized view configurations. Similarly, Chaudhuri et al. [5, 6] present two approaches which illustrate the whole tuning process using constraints such as space threshold. However, these approaches operate on single systems instead of comparing two or more systems. In contrast to the mentioned approaches, our approach do not consider tune configurations, indexes, etc.

Another approach for OLAP applications is Ingres/Vectorwise which applies the Vectorwise (formerly MonetDB/X100) architecture into the Ingres product family [12]. In cooperation with Vectorwise, Ingres developes a new storage manager ColumnBM for the new Ingres/Vectorwise. However, the integration of the new architecture into the existing environment remains unclear [12].

6. CONCLUSION

In recent years, column stores have shown good results for DWH applications and often outperformed established row stores. However, new requirements arise in the DWH domain that cannot be satisfied only by column stores. The new requirements demand also for row store functionality, e.g., real-time DWHs need sufficient update processing. Thereby, the complexity of design process increases because we have to choose the optimal architecture for given applications. We showed with an experiment that workload analyses based on query structure and syntax are not sufficient to select the optimal storage architecture. Consequently, we suggested a new approach based on database operations. We introduced workload patterns which contain all workload information beyond the architectures, e.g., statistics and operation cost. We also presented a workload decomposition approach based on existing database functionality that maps operations of a given workload to our workload patterns. We illustrated the methodology of our decomposition approach using an example workload. Subsequently, we state that a separate decomposition algorithm for column stores is not needed. We stated that our presented approach is transparent

to any workload and any storage architecture based on the relational data model. In the evaluation, we proved the usability of our approach. Additionally, we demonstrate the comparability of different systems using different architectures even if the systems provide different information with respect to their query execution. The decision process can be periodically repeated, thus the storage architecture selection is not static. Moreover, our approach can be used for optimizer (decisions) in hybrid relational DBMS that has to select the storage method for parts of data.

In future work, we will investigate two strategies to implement our workload patterns in a prototype. First, we utilize a new DBS to export periodically statistics and operation costs which we map to our workload patterns. This way, we will not affect performance of analyzed systems by prediction computation. Second, we adapt existing approaches [5, 17] to automatically gather statistics, e.g., mapping statistics and workload patterns directly into a graph structure (query graph model). Additionally, aggregated or estimated values from other sources can be stored. We will perform detailed studies on OLAP, OTLP, and mixed workloads to gather expressive values for predictions.

7. REFERENCES

[1] D. J. Abadi. *Query execution in column-oriented database systems*. PhD thesis, Cambridge, MA, USA, 2008. Adviser: Madden, Samuel.

[2] D. J. Abadi, S. R. Madden, and N. Hachem. Column-stores vs. row-stores: How different are they really? In *SIGMOD '08*, pages 967–980, New York, NY, USA, 2008. ACM.

[3] A. Ailamaki, D. J. DeWitt, M. D. Hill, and M. Skounakis. Weaving relations for cache performance. In *VLDB '01*, pages 169–180, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[4] M. M. Astrahan, M. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. Gray, P. P. Griffiths, W. F. K. III, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson. System R: Relational approach to database management. *ACM TODS*, 1(2):97–137, 1976.

[5] N. Bruno and S. Chaudhuri. To tune or not to tune? A lightweight physical design alerter. In *VLDB '06*, pages 499–510. VLDB Endowment, 2006.

[6] N. Bruno and S. Chaudhuri. An online approach to physical design tuning. In *ICDE '07*, pages 826–835, 2007.

[7] S. Chaudhuri and V. Narasayya. Autoadmin "what-if" index analysis utility. In *SIGMOD '98*, pages 367–378, New York, NY, USA, 1998. ACM.

[8] G. P. Copeland and S. N. Khoshafian. A decomposition storage model. In *SIGMOD '85*, pages 268–279, New York, NY, USA, 1985. ACM.

[9] D. W. Cornell and P. S. Yu. An effective approach to vertical partitioning for physical design of relational databases. *IEEE Trans. Softw. Eng.*, 16(2):248–258, 1990.

[10] S. J. Finkelstein, M. Schkolnick, and P. Tiberio. Physical database design for relational databases. *ACM TODS*, 13(1):91–128, 1988.

[11] S. Harizopoulos, V. Liang, D. J. Abadi, and S. Madden. Performance tradeoffs in read-optimized databases. In *VLDB '06*, pages 487–498. VLDB Endowment, 2006.

[12] Ingres/Vectorwise. Ingres/VectorWise sneak preview on the Intel Xeon processor 5500 series-based platform. White Paper, September 2009.

[13] E. Kwan, S. Lightstone, K. B. Schiefer, A. J. Storm, and L. Wu. Automatic database configuration for DB2 Universal Database: Compressing years of performance expertise into seconds of execution. In *BTW '03*, pages 620–629, 2003.

[14] T. Legler, W. Lehner, and A. Ross. Data mining with the SAP NetWeaver BI Accelerator. In *VLDB '06*, pages 1059–1068. VLDB Endowment, 2006.

[15] A. Lübcke. Cost-effective usage of bitmap-indexes in DS-Systems. In *20th Workshop "Grundlagen von Datenbanken"*, pages 96–100. School of Information Technology, International University in Germany, 2008.

[16] A. Lübcke. Challenges in workload analyses for column and row stores. In *22nd Workshop "Grundlagen von Datenbanken"*, volume 581. CEUR-WS.org, 2010.

[17] A. Lübcke, I. Geist, and R. Bubke. Dynamic construction and administration of the workload graph for materialized views selection. *Int. Journal of Information Studies*, 1(3):172–181, 2009.

[18] A. Lübcke, V. Köppen, and G. Saake. A decision model to select the optimal storage architecture for relational databases. RCIS, France, MAY 2011. IEEE. to appear.

[19] Oracle Corp. Oracle Database Concepts 11g Release (11.2). 14 Memory Architecture (Part Number E10713-05), March 2010.

[20] Oracle Corp. Oracle Performance Tuning Guide 11g Release (11.2). 12 Using EXPLAIN PLAN (Part Number E10821-05), March 2010.

[21] Oracle Corp. Oracle Performance Tuning Guide 11g Release (11.2). 11 The Query Optimizer (Part Number E10821-05), March 2010.

[22] H. Plattner. A common database approach for OLTP and OLAP using an in-memory column database. In *SIGMOD '09*, pages 1–2, New York, NY, USA, 2009. ACM.

[23] R. J. Santos and J. Bernardino. Real-time data warehouse loading methodology. In *IDEAS '08*, pages 49–58, New York, NY, USA, 2008. ACM.

[24] J. Schaffner, A. Bog, J. Krüger, and A. Zeier. A hybrid row-column OLTP database architecture for operational reporting. In *BIRTE '08*, 2008.

[25] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik. C-Store: A column-oriented DBMS. In *VLDB '05*, pages 553–564. VLDB Endowment, 2005.

[26] A. A. Vaisman, A. O. Mendelzon, W. Ruaro, and S. G. Cymerman. Supporting dimension updates in an OLAP server. *Information Systems*, 29(2):165–185, 2004.

[27] Y. Zhu, L. An, and S. Liu. Data updating and query in real-time data warehouse system. In *CSSE '08*, pages 1295–1297, Washington, DC, USA, 2008. IEEE Computer Society.

[28] D. C. Zilio, J. Rao, S. Lightstone, G. M. Lohman, A. J. Storm, C. Garcia-Arellano, and S. Fadden. DB2 Design Advisor: Integrated automatic physical database design. In *VLDB '04*, pages 1087–1097. VLDB Endowment, 2004.

[29] D. C. Zilio, C. Zuzarte, S. Lightstone, W. Ma, G. M. Lohman, R. Cochrane, H. Pirahesh, L. S. Colby, J. Gryz, E. Alton, D. Liang, and G. Valentin. Recommending materialized views and indexes with IBM DB2 Design Advisor. In *ICAC '04*, pages 180–188, 2004.

[30] M. Zukowski, P. A. Boncz, N. Nes, and S. Heman. MonetDB/X100 - a DBMS in the CPU cache. *IEEE Data Eng. Bulletin*, 28(2):17–22, June 2005.

Data Locality in Graph Databases through N-Body Simulation

Dominic Pacher
Institute of Computer Science
Technikerstrasse 21a
Innsbruck Austria
dominic.pacher@uibk.ac.at

Robert Binna
Institute of Computer Science
Technikerstrasse 21a
Innsbruck Austria
robert.binna@uibk.ac.at

Günther Specht
Institute of Computer Science
Technikerstrasse 21a
Innsbruck Austria
guenther.specht@uibk.ac.at

ABSTRACT

Data locality poses a major performance requirement in graph databases, since it forms a basis for efficient caching and distribution. This vision paper presents a new approach to satisfy this requirement through n-body simulation. We describe our solution in detail and provide a theoretically complexity estimation of our method. To prove our concept, we conducted an evaluation using the DBpedia dataset data. The results are promising and show that n-body simulation is capable to improve data locality in graph databases significantly.

Categories and Subject Descriptors

H.2.4 [**Database Systems**]: Graph databases

General Terms

Locality, N-body Simulation, Graph Data, Experimentation

Keywords

Database, Graph, Simulation, Graph Database, Triple Store

1. INTRODUCTION

Recently the demand to manage high amounts of linked data increased substantially. This development has its origin in data, generated by social as well as linked knowledge networks like Wikipedia [1]. In addition, all of today's imperative programming languages work on graph oriented (object) memory systems, because they are easy to understand and can be efficiently processed in main memory. Moreover, graph oriented memory systems provide means to easily formulate complex recursive behavior and data structures. Usually these data structures need to be stored persistently in some kind of external database.
Beside the exact internal concept, this (graph) database has to support query, update and remove operations of single nodes or complete sub graphs as fast as possible. Clearly this requirement influences all of the different sub components of a graph database and can be fulfilled through improvements on many different levels. However, there is no other property, which has as much influence on the performance and scalability of the overall system as *data locality*. In terms of graphs this means that any node stored has to be also physical near to its linked nodes in the memory. This seems to be a straightforward requirement, but it's hard to fulfill practically. In theory, a graph describes a multidimensional data structure, which has to be managed by the computer. Unfortunately, since memory systems work on a fixed one-dimensional memory layout, this cannot be done directly. The common solution to this problem is to define a mapping from multidimensional data to less (one) dimensional space. Although it's not a problem to find any kind of mapping, it's hard to preserve data locality at the same time. Therefore data locality isn't assured directly (Figure 1) and databases try to speed up operations using additional indexes or in the case of main memory systems, by providing cheap jumps through random access memory.

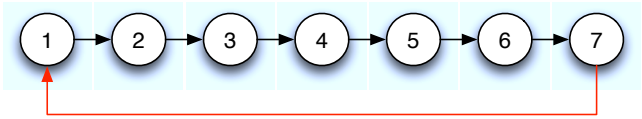


Figure 1: A two dimensional structure gets mapped to one-dimensional space. Since no locality is preserved, large jumps (6 nodes) appear in the data (red).

Despite the fact that this solutions work out quite well for the problem, they are always tied to additional costs and remaining limitations and don't solve the actual problem. For example, additional indices need space and have to be updated on every change. Main memory systems work well on one core and one computer. But since, frequent jumps between the cores memory or even worse, between computers, are orders of magnitudes costlier than jumps within main memory of one single thread, it's hard to distribute them properly.

To come up with a new approach to improve this situation, this paper suggests building a graph database whose nodes are aligned in memory by a n-body simulation system. Inspired by real world physics laws, links will be simulated as springs causing nodes to arrange themselves automatically. As a result, when the state of lowest energy is reached, a

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

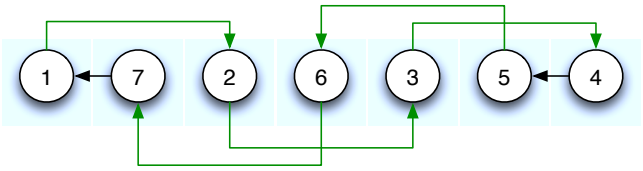


Figure 2: Better solution of Figure 1. Locality is preserved and the maximum jump length is reduced to two nodes (green)

maximum of data locality is provided at the same time (Figure 2). In addition, n-body simulation systems are known to be highly distributable and computational feasible [16].

Consequently the aim of this paper is to show through experimentation, that such a simulation will optimally place graph nodes in memory achieving improved data locality on global scale.

The remainder of this paper is structured as follows. Section 2 describes related papers in more detail. In section 3 we present our new method by a short introduction to n-body simulation idea, as well as some adjustments we had to make. To prove that our concept is feasible, we performed some preliminary evaluations which results are discussed in section 4. Section 5 sums up with an conclusion and future works.

1.1 Related Work

Although there is, at best of our knowledge, no related approach solving the data locality problem of graph database using n-body simulation, papers exists which make use of this method for related problems.

The idea of n-body simulation to support graph alignment has been already proposed in the 80s [14] and constantly improved [19]. However, these algorithms try to find an optimal layout for graph nodes, which is a far more complicated problem than preserving locality as it includes additional requirements like finding *aesthetic pleasing* solutions. Fortunately, this is clearly not an affordance for graph databases.

Plenty of systems were developed in the RDF research area trying to optimize storing and querying graphs. These *graph stores* can be separated into three groups of stores, which

- reside completely in memory (In-Memory Store)
- are based on a relational database (Relational Triple Store)
- use their own implementation (Native Triple Store)

To the group of *In-Memory Stores* belong GRIN [17] and Brahms [12], which mainly try to solve special purpose queries through dedicated indices. Also SpiderStore [6] operates in memory completely. However it makes no special assumptions about queries.

Jena SDB [18] and Virtuoso RDF Views [10] are part of the second group using a traditional row oriented *relational model*. Mapping graph data to the relational model tend to result in one big table with three columns: source node, edge, destination node (or in RDF terms subject, predicate object) and billions of rows. As the mapping of this table to a common row oriented store is inefficient [2] and [15] applied a column oriented relational model.

Part of the third group, the *native implementations*, are the adapted Jena TDB [18] (in contrast to SDB), Virtuoso [10], YARS [11] and RDF-3X [13]. Where the last two approaches make excessive use of indices to speed up the query execution process. Though RDF-3X achieved new query speed records, this approach is heavily optimized on read operations. As a consequence on any data update all indices have to be changed accordingly. In contrast, BitMap [3] uses a completely different design using compressed bit-matrix structure.

Finally Sesame [8] provides storage engines of all tree groups. The performance of these systems have been evaluated in [13] [6] and through the Berlin Benchmark [7].

Consequently there is no system yet using n-body simulation to improve data locality and it's interesting if such an approach is able to improve the overall performance of graph databases.

2. THE METHOD

In contrast to existing methods to store graph data we suggest an algorithm, which achieves a high degree of data locality. This algorithm is based on the idea, that link length don't come for free, making longer links to more distant data locations more expensive than shorter links. With this additional costing factor c, an optimal solution for the locality problem in databases can be defined as achieving the global minimum of the sum of this costs overall nodes n:

$$C_{all} = \min \sum_{i=0}^{n} c_i$$

This optimization process becomes quickly unsolvable using analytically methods, therefore a common n-body simulation approach is applied. Every edge is seen as a physical spring between two data nodes. Springs will add distance depended forces F_l to the connected links causing them to approach each other:

$$F_l = F_c * D(l)$$

Where F_c is the force constant and $D(l)$ a distance function of linked node l. This distance function can be for example a linear function returning the distance to the linked node l or an exponential function causing forces to increase exponentially with the distance.

Since a node is influenced by all its linked nodes, all forces F_l have to summed up to achieve the final overall force F_n:

$$F_n = \sum_{i=0}^{n} F_i$$

Now we can calculate the acceleration of the current node $n$using its mass m_n:

$$a_n = F_i / m_n$$

In our prototype we set m_n to 1 but for later implementations this parameter may represent a ideal way to reduce the movement of big nodes using the number of links as mass. This would cause big nodes to be moved less often. Finally we can use a_n to calculate the change of velocity

$$\Delta v_n = a_n * s$$

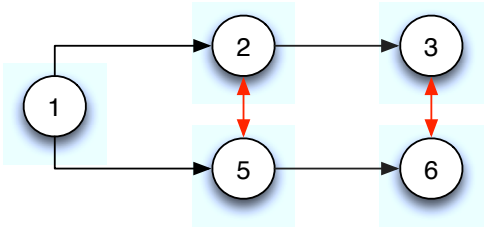


Figure 3: Nodes 2 / 5 and 3 / 6 have to be stored to the same position in one-dimensional space.

where s describes the used step size. For the sake of simplicity our prototype used a step size of 1. The simulation can now be formulated in three steps:

1. Calculate v_n for all n.
2. Change v_n according to Δv_n and calculate new position.
3. Check if there is any movement . If yes then goto 1.
4. Simulation finished.

2.1 Adjustments

N-body simulation methods have been used widely and very successfully in many fields of physics over the past decades. However some adjustments are necessary to make the approach useful for locality calculations.
As memory of all modern computers is accessed through discrete addresses, the simulation has to take this into account and have to operate on integers entirely. This approach has two advantages. In the first place it avoids the introduction of additional repulsion forces to keep nodes at a minimum distance to each other and secondly, the calculations can be done with faster integer calculations.
As mentioned previously, graph data is naturally multi dimensional, which stands in direct contrast to the one-dimensional memory space. Because of that, nodes may have found a final position, which is already be claimed by another node. Therefore, a priority function has to be defined to solve this problem, preserving that the node wins which leads to less energy in the overall system. This can be accomplished by using the nodes overall force as priority value.
An example of this problem can be found in Figures 3 and 4 where nodes 2/5 and 3/6 claiming the same position. Figure 4 also shows, that preserving locality comes at cost of the link length of other nodes.

2.2 Complexity Estimation

Generally the complexity of n-body simulations can be estimated as $\mathcal{O}(n * m * s_{num})$ where n describes the number of nodes, m the number of links per node and s the number of simulation steps needed until energy equilibrium. Consequently for a complete graph, where $n = m$ the complexity raises to $\mathcal{O}(n^2 * s_{num})$. This wouldn't be feasible for high amounts of data. Fortunately [4] showed for gravity simulations, which can be reduced to a fully connected graph, that complexity can be reduced to $\mathcal{O}(n * log(n) * s_{num})$, using a supporting tree structure and aggregation for distant nodes. Although this is the worst-case estimation, it is very unlikely to happen for real data where the number of links per node should always be significantly smaller than the number of

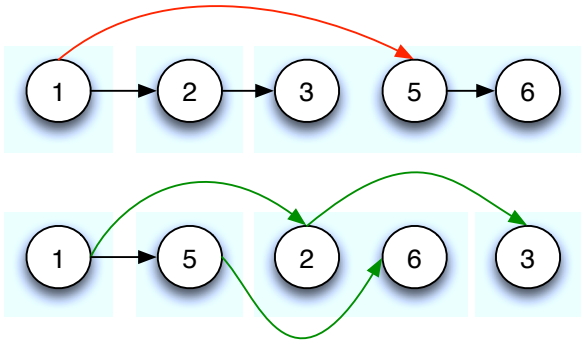


Figure 4: Mapping of Figure 3 to one dimension. The upper Figure shows a non optimal solution with jump size of three. The lower Figure shows a better solution with a jump size of two.

nodes. Indeed for a complete graph, the n-body simulation cannot improve any locality, because energy equilibrium is already reached and the simulation would terminate after the first step. Assuming that $n >> m$ the next important factor is s_{num}, which depends on the dataset as well as the used step size s. Consequently an increased step size would lead to less simulation steps. However too large steps sizes also increase the computational error between steps and will lead to an unstable simulation eventually. Fortunately, a variety of algorithms exist to minimize this approximation error, using one step or multistep methods as well as methods with variable step size at need [9].
Finally, if simulated once, we assume that the majority of data locations will remain stable and won't have to be recalculated on every data update on global scale. This estimation and in addition, that existing implementations deal with about 10 billion elements [16], let us believe that a large scale simulation of graph data is feasible.

3. PRELIMINARY EVALUATION

To prove the suggested concept we implemented a prototype and made some preliminary evaluations. To get realistic results we chose a subset (first 200 000 triples = 110,205 nodes) of the DBpedia dataset. All tests were conducted on a single machine (Mac Pro Intel Xeon 2,26 GHz) using a simple single threaded process with 200 MiB of dedicated ram.
To get an visual impression how effective our method increases data locality, we visualize every data element as a pixel in an image. As we are working on a one-dimensional space, all values are simply wrapped at the end of image width to create a two dimensional image. Every pixel position corresponds to the actual position of a data node in the data space.
In Figure 5 the color of this pixel represents the maximum distance of a node to its linked neighbor nodes in the data space. Using a maximum value of $n/2$ (value red) this Figure shows the development over time until energy equilibrium is reached. At $t = 0$ the data is scattered randomly in the

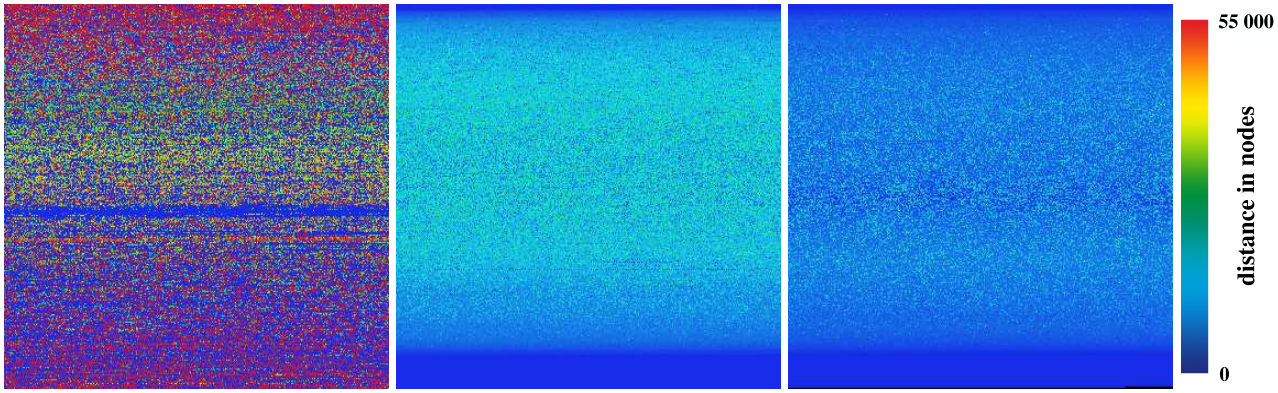


Figure 5: Complete data space of 110205 nodes. Each pixel represents one node. The color key on the right describes the maximum distance of a node's link. From left to right the image shows the data space at at t=0, t = 0.5 and t = 1.

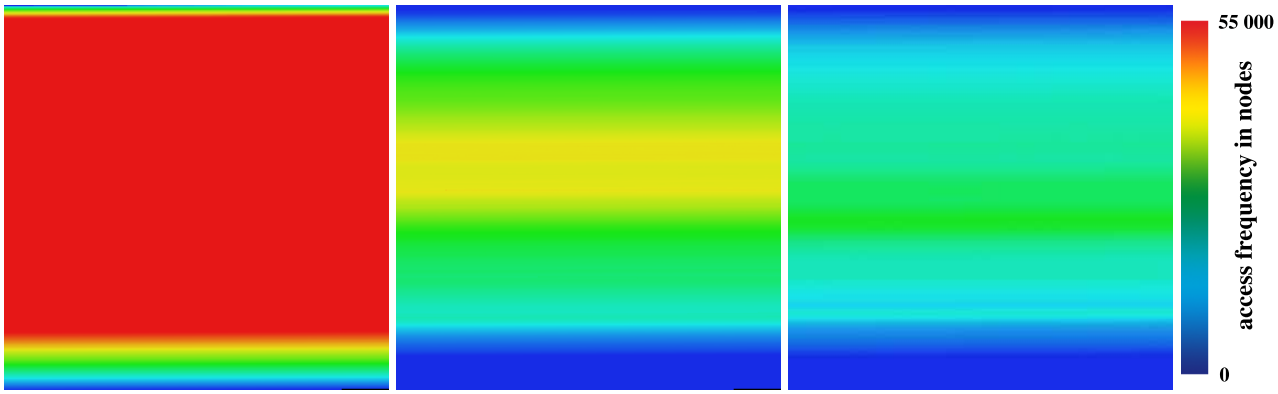


Figure 6: Series of access heat maps showing high access zones which should not be separated. Red positions are accessed by 55,000, dark blue nodes by less than 1,000 nodes. From left to right the image shows the data space at at t=0, t = 0.5 and t = 1.

data space. There are plenty of nodes, which have to jump through the whole data space to access their linked nodes. At $t = 0.5$ the data distribution has improved already, but can be further enhanced until a state of minimum energy (no more movement of nodes) is reached ($t = 1$). The final result shows that all nodes have now arrived at a position, where they can access their most distant linked node with a minimum of locality change.

To get a better impression of the exact numbers, we created a histogram (Figure 7) for the same sample data. For better understanding, be aware of the logarithmic scaling on the vertical axis. Furthermore we moved the mid-term frame $t = 0.5$ to quarter time $t = 0.25$ to get a better impression of the progress over time. Similar to Figure 5, at $t = 0$ one can observe an almost equal distribution of nodes along the complete range of possible distances. As an important matter of fact, there is already a peak of values having very close linked nodes on the very left side ($distance < 1,000$) of the diagram. Since it's often the case that new nodes are introduced followed by their direct neighbors, the input file itself can be seen as origin of this peak. Of course this issue can only have a positive impact on local data locality and not on global scale, as we want to achieve. During the simulation over $t = 0.25$ to $t = 1.0$ one can observe a significant reduction of distances to about one quarter of the originally data space. In addition, the histogram points out that global data locality comes also at cost of local data locality, which is showed by the reduction of the red peak at $t = 0$ mentioned before to the blue one at $t = 1.0$. The cause of this reduction can be seen again in the mapping of multi dimensional data to less (in our case one) dimensional space, where different data nodes claim the same position (Figures 3 and 4).

Furthermore there are some small peaks (two nodes) remaining near distance $50,000$. These nodes couldn't be aligned very well. Although we always expected problems with nodes that are highly linked to different other nodes and can't be further optimized by means of location, this appears not to be the case here, as the number of links of the worst-case node where only about 300. Unfortunately, this remains a rather unsatisfied situation and has to be further investigated in future. However, as shown in our sample data set, these nodes can be considered as very rare (overall 8 nodes out of $110,025$ until distance of $26,000$).

These problems apart, it's most important that the global data locality improved substantially. As previously seen in Figure 5, the histogram shows that most nodes are far less distant to their linked nodes than at the start of simulation.

Figure 7: Histogram of the occurrence of nodes and their maximum distances to linked nodes. For better understanding the time frames were changed to t=0, t = 0.25 and t = 1.

In particular, the link length was reduced to 1/4 in the worst and to 1/10 in the average case.
Based on this data it should be possible to make efficient decisions how graph data can be separated in certain chunks, to be distributed on different cores as well as on different computers. To gather more insight into this, we used an access heat map. To create this map all data positions lying between a node and all it's linked nodes are incremented by one. Of course in main memory we are able to randomly access every position at same speed, but in a distributed environment this model fits very well. When this is done for all nodes and their respective linked nodes, every position marks the number of accesses needed to visit every neighbor node within data space (Figure 6). This image gives an impression, where the data space can be separated best, choosing less dense (blue in the Figure) zones.

4. CONCLUSION AND FUTURE WORK

The aim of this paper was to show that a n-body simulation can improve graph data locality significantly. After a introduction to our suggested method, we evaluated a experimental prototype using partial data of the DBpedia dataset [5]. As a result, we were able to restrict jumps to about 1/4 of the whole data space in the worst-case and to 1/10 for the average case. Although these are very promising results, there is plenty of work remaining.
We theoretically showed that our n-body approach should scale well into millions of graph nodes. However, our prototype is currently not optimized for very large data sets like the complete DBpedia dataset, consisting of about 100 million triples. Hence our goal for future works will be to optimize the simulation by improving the algorithm and finding a way to distribute the simulation on many cores and computers. As a result of this development, we hope to provide practically evidence that our method is working on large real world graphs preserving computational feasibility.

5. REFERENCES

[1] *Wikipedia Free Encyclopedia*. http://wikipedia.com, apr 2011.
[2] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. *Scalable semantic web data management using vertical partitioning*. VLDB Endowment, sep 2007.
[3] M. Atre, V. Chaoji, M. J. Zaki, and J. A. Hendler. *Matrix Bit loaded: a scalable lightweight join query processor for RDF data*. ACM, apr 2010.
[4] J. Barnes and P. Hut. A hierarchical O(N*log(N)) force-calculation algorithm. *nature*, 324(4):446–449, 1986.
[5] C. Becker. *RDF Store Benchmarks with DBpedia*. www4.wiwiss.fu-berlin.de, 2011.
[6] R. Binna, W. Gassler, E. Zangerle, and D. Pacher. SpiderStore: Exploiting Main Memory for Efficient RDF Graph Representation and Fast Querying. 2010.
[7] C. Bizer. The berlin sparql benchmark. *Int J Semantic Web Inf Syst*, 2009.
[8] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF schema. In *Semantic Web - Iswc 2002*, pages 54–68, Aidministrator Nederland BV, Amersfoort, Netherlands, 2002. Aidministrator Nederland BV, Amersfoort, Netherlands.
[9] J. Butcher. *Numerical Methods for Ordinary Differential Equations* . Wiley, 2 edition, jun 2008.
[10] O. Erling and I. Mikhailov. Rdf support in the virtuoso dbms. In T. Pellegrini, S. Auer, K. Tochtermann, and S. Schaffert, editors, *Networked Knowledge - Networked Media*, volume 221 of *Studies in Computational Intelligence*, pages 7–24. Springer Berlin / Heidelberg, 2009.
[11] A. Harth, J. Umbrich, and A. Hogan. YARS2: A federated repository for querying graph structured

data from the web. *The Semantic Web*, 2007.

[12] M. Janik and K. Kochut. BRAHMS: A WorkBench RDF Store and High Performance Memory System for Semantic Association Discovery. In *Fourth International Semantic Web Conference*, pages 431–445. Springer, 2005.

[13] T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. *The VLDB Journal — The International Journal on Very Large Data Bases*, 19(1):91–113, feb 2010.

[14] E. Peter. A Heuristic for Graph Drawing. *Congressus Numerantium*, 42:149–160, nov 1984.

[15] L. Sidirourgos, R. Goncalves, M. Kersten, N. Nes, and S. Manegold. Column-store support for rdf data management: not all swans are white. *Proc. VLDB Endow.*, 1:1553–1563, August 2008.

[16] V. Springel, S. D. M. White, A. Jenkins, C. S. Frenk, N. Yoshida, L. Gao, J. Navarro, R. Thacker, D. Croton, J. Helly, J. A. Peacock, S. Cole, P. Thomas, H. Couchman, A. Evrard, J. o. r. Colberg, and F. Pearce. Simulations of the formation, evolution and clustering of galaxies and quasars. *nature*, 435(7042):629–636, jun 2005.

[17] O. Udrea, A. Pugliese, and V. S. Subrahmanian. Grin: a graph based rdf index. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, pages 1465–1470. AAAI Press, 2007.

[18] K. Wilkinson, C. Sayers, and H. Kuno. Efficient RDF storage and retrieval in Jena2. In *Proceedings of SWDB*, 2003.

[19] V. Zabinako and P. Rusakovs. Development and Implementation of Partial Hybrid Algorithm for Graphs Visualization. *Scientific Proceedings of Riga Technical University*, 5(34):192–203, jul 2008.

SpiderStore: A Native Main Memory Approach for Graph Storage

Robert Binna, Wolfgang Gassler, Eva Zangerle, Dominic Pacher, Günther Specht
Databases and Information Systems, Institute of Computer Science
University of Innsbruck, Austria
{firstname.lastname}@uibk.ac.at

ABSTRACT

The ever increasing amount of linked open data results in a demand for high performance graph databases. In this paper we therefore introduce a memory layout which is tailored to the storage of large RDF data sets in main memory. We present the memory layout *SpiderStore*. This layout features a node centric design which is in contrast to the prevailing systems using triple focused approaches. The benefit of this design is a native mapping between the nodes of a graph onto memory locations connected to each other. Based on this native mapping an addressing schema which facilitates relative addressing together with a snapshot mechanism is presented. Finally a performance evaluation, which demonstrates the capabilities, of the *SpiderStore* memory layout is performed using an RDF-data set consisting of about 190 mio triples.

Categories and Subject Descriptors

H.2.2 [**Database Management**]: Physical Design; H.3.2 [**Information Storage and Retrieval**]: Information Storage; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Performance, Algorithms, Design, Experimentation

Keywords

RDF, Main Memory, Database, RDF Store, Triple Store, SPARQL, Addressing Scheme

1. INTRODUCTION

Due to the increasing significance of linked open data and publicly available SPARQL-endpoints, the need for high performance graph databases has increased. To meet those requirements several approaches for storing and retrieving large RDF (Resource Description Framework) graphs have been developed. Despite the highly connected nature of these graphs, the main approaches proposed in this context are facilitating technologies originating from relational databases. Even though these represent major and robust technologies, they were not tailored for the scenario of storing graph based structures. At the same time the ever increasing capacities of main memory and the increasing numbers of cores have lead to an architectural shift in the development of databases and information systems by moving from hard disk to main memory as the primary storage device. Whereas these architectural changes lead to enormous performance improvements, when implementing graph-operations like graph traversals they still have to be implemented through costly self join operations. Despite the possibility of supporting these operations with appropriate index structures they still take $\mathcal{O}(log(n))$ where n denotes the number of index entries. Therefore we present the SpiderStore storage concept as an in-memory storage approach, which allow to process edges in $\mathcal{O}(1)$. In contrast to previous work [3] the storage layout and space estimations are captured in more detail. In addition, a new relative addressing scheme is introduced. The successive sections are structured as follows. Chapter 2 deals with the memory layout and space estimations. Chapter 3 explains the relative addressing scheme used for faster restarts and for snapshot generation. In chapter 4 we present an evaluation of the presented technology using the YAGO2 [8] data set. Chapter 5 discusses the related work in the field of RDF-databases. Finally Chapter 6 draws a conclusion and makes forecasts for possible future work.

2. MEMORY LAYOUT

This section represents a detailed description over the *SpiderStore* memory layout. The aim of this memory layout is to provide an in-memory storage of graphs, where the basic operation of navigating between two vertices can be done in $\mathcal{O}(1)$. Therefore, the node is the core component of the layout. This is in contrast to the concept favored by most triple stores where the the triple represent the atomic building block. To realize this concept of a node centric layout, two factors have to be fulfilled. The first is that all edges belonging to a node need to be stored in a single place, which allows to navigate back and forth along all edges. The second factor is that there need to be a direct connection between those nodes that can be resolved within a single operation. These requirements can be fulfilled by an in-memory storage layout, which is designed as follows. Each node has to be located at a unique location within

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

memory. At each of these locations the information about ingoing and outgoing edges is stored and grouped by the respective edge label. The information about edges itself is not stored directly but implicitly through pointers. Therefore, the traditional pointer structures, which themselves can be seen as graphs connecting different memory locations are used to represent arbitrary graphs.

Each node contains a set of ingoing and a set of outgoing edges (pointers). These pointers are grouped by the predicates labeling the corresponding edges. Grouping is done by the address of the predicates. This address is equal to the address of the property within the predicate index, which stores a list of all subjects occuring together with a specific property.
Beside the raw pointer data, implicit statistical information is stored. This is the case for the number of nodes, the number of predicates and the number of strings. Furthermore, for each subject the number of predicates and for each subject/predicate combination the number of objects is stored. The same information is stored the other way around (the number of predicates for an object and the number of predicate/subject combinations for an object).

To illustrate this layout, Figure 1 visualizes the memory layout of a short example. The main node emphasized in this example represents the category of a `wordnet_scientist`. It can be seen from this example that two different types of edges exist: ingoing and outgoing edges. Both types themselves group their destination pointers by the predicate node. In Figue 1, an outgoing edge with the property ⟨`hasLabel`⟩ and an incoming edge with the property ⟨`type`⟩ is featured. To simplify matters, URIs are abbreviated and marked with angle bracket while literals are put in quotes. As it can be seen in the example, all nodes independent of their type (URI, literals, ...) share a uniform node layout. For example the node `"scientist"` has the same structure as the node ⟨`wordner_scientist`⟩. The facts described in this example are that ⟨`Einstein`⟩ is of ⟨`type`⟩ ⟨`wordner_scientist`⟩ and that this category has a label with the name `"scientist"`. The triple notation of the example in Figure 1 is shown in the listing below:

```
...
⟨Einstein⟩ ⟨type⟩ ⟨wordner_scientist⟩
⟨wordner_scientist⟩ ⟨hasLabel⟩ "scientist"
...
```

2.1 Space Estimations

Given that SpiderStore is a main memory based graph store, space consumption becomes a crucial factor. Therefore we introduce and discuss a formula that can be used to calculate the expected amount of memory needed for a specific set of data. To describe a specific data set we use the following variables. The variable `#nodes` represents the total number of distinct nodes within a graph. A node can either be a URL or a character string and can be be used as subject, predicate or object. The second variable used for calculating the expected space consumption is the total number of triples or facts `#notriples`. The space consumption is then calculated as follows:

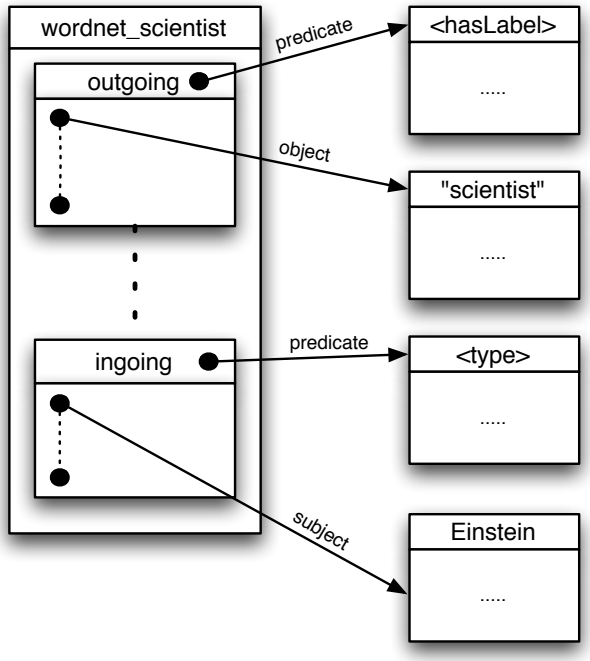


Figure 1: Memory Layout - Example

$$m = (\#nodes * (5 + degree * 2) + \#edges * 3) * sizeof(pointer) + sizeof(dictionary)$$

This formula consists of three parts, i.e. the size of the data dictionary, the fraction influenced by the number of nodes and the fraction influenced by the number of edges. The part of this formula that depends on the number of nodes can be interpreted as follows. For each node a data structure is stored consisting of the number of incoming edges and a link to the table containing the incoming edges as well as the number of outgoing edges and a link to the table containing the outgoing edges. Furthermore a pointer to the corresponding entry within the dictionary table is stored within each node. The term $degree * 2$ can be explained by the pointers which group the edges by their corresponding predicates. For each predicate there exist a pointer to the edges itself and a counter, which is used for statistics. Because all edges are bidirectional, the estimation is quite accurate even though it does not take the number of distinct subjects or objects into account.
The part of the formula, which depends on the number of edges, can be derived of the following facts. For each edge the destination is stored on both ends. Furthermore, an additional entry is stored within the predicate index which allows to start the query execution not only at subject or object nodes but as well at predicate nodes.
As an example the YAGO2 [8] data set used throughout the evaluation consists of 194,35 million triples and of 28,74 million distinct nodes. The dictionary size in the case of the YAGO2 data set is roughly about 1.1 gigabytes. In this example 1.2 would be an appropriate value for the variable *degree*. This number can be derived by counting the number

of all distinct subject-predicate combinations and dividing it by the total number of nodes. By apply these values to the formula above a space consumption of about 7,4 gigabytes might be expected. In comparison to the real footprint, which is about 7.2 gigabytes this value is quite accurate. Hence knowing the basic facts about a data set allows to establish an adequate estimation about the expected memory footprint.

3. RELATIVE ADDRESSES

Using absolute pointer addresses for the location of nodes enables the navigation between two nodes in $\mathcal{O}(1)$ as it has been explained in the section before. The drawback of this concept is that storing a persistent state of a whole graph database can become quite complex. The reason for this is that the database can only be dumped either by serializing the whole graph and later deserializing the whole graph or by dumping the whole memory snapshot and swizzling the pointers to their correct representation at load time.
Another alternative to these approaches is to arrange the whole database content based on an offset address. Hence all memory locations have to be converted into relative addresses based on this offset. Resolving such relative pointers yields in an overhead, which is negligible compared to the total amount of time needed to fetch an arbitrary junk of memory. The time for fetching an arbitrary chunk of memory can vary between some cpu cycles, when accessing memory already located within L1-cache, up to some hundreds cpu cycles, when accessing a random chunk of memory which is not available in the cache hierarchy yet.
In the context of SpiderStore we decided to use relative addressing. This has two advantages. The first advantage is that the general architecture of SpiderStore is still applicable and the overhead introduced through relative addressing is insignificant in the SpiderStore concept as it has been explained before. The main advantage of this addressing scheme is that database restarts can be executed within a few milliseconds. This is possible by facilitating the Unix memory mapping techniques which does not dismiss the mapped pages unless another executable allocates large amounts of memory. Furthermore this concept allows to facilitate the copy on write approaches used by the HyPer project [10]. This approach benefits from the operating system's memory management, which would allow different processes to share large memory segments, while preserving an isolated view on the data.

Due to the lack of a customized memory allocator, the SpiderStore snapshot mechanism is currently implemented as a read only approach. Each snapshot is split up into five parts. One part is responsible for the storage of the node data structures, another for the indexes between node identifiers and node addresses. One more file stores the node identifiers. The other files are responsible for the storage and indexing of predicate nodes and for the storage of the edge information. The separation into several files prevents memory fragmentation and leads to shorter "addresses" for nodes, predicates and strings. For example all entries within the node, predicate or index files have uniform sizes and can therefore be seen as simple array structures.

4. EVALUATION

As a platform for the evaluation a server equipped with two Intel Xeon L5520 Quad Core CPUs, 2.27 GHz, Linux kernel 2.6.18, CentOS, 64-bit architecture and 96 GB main memory was used.

4.1 DataSet

For the evaluation we used the YAGO2 [8] data set. The YAGO2 data set is the successor of the YAGO [16] data set and represents a large semantic knowledge base which is generated on the basis of Wikipedia, WordNet and GeoNames. The data set consist of of 194,350,853 triples (98 predicates, 28,744,214 unique subjects and objects). The queries executed on this data set are derived from the queries on the YAGO data set used in the benchmark presenting the RDF-3X approach [12].

4.2 Evaluated Systems

For the evaluation of *SpiderStore* we used Virtuoso [6] in version 6.1.3, RDF-3X [12] in version 0.3.5 and Jena TDB [19] in version 0.8.9. The decision for these systems was taken to the best of our knowledge. Even though *SpiderStore* is the only main memory system tested, the decision for choosing the other systems is accurate. The reason for this is that those systems are assumed to be the currently fastest systems available and that the benchmark results are measured with warm caches on a system where the whole database would be able to fit into main memory. All systems were granted a maximum of 40 GB to ensure that sufficient space is available.

4.3 Evaluation Results

The test results are separated into two parts. The first part compares the bulk import times of the different systems. The bulk import time is specified as the time needed to load the data, to create the indexes and to ensure that a persistent state of the database is written to disk. A summary of the load times can be seen in Table 1. As can be seen, *SpiderStore* is significantly faster than any of the other systems. The reason for this is that *SpiderStore*, due to its implicit statistics, does not need to explicitly create statistics or indexes.

System	Load Time
SpiderStore	1:09:18
Jena	1:36:35
RDF-3X	1:21:12
Virtuoso	3:32:16

Table 1: Import Times (in hours, minutes and seconds)

The second part of the evaluation compares the query execution for each query on the YAGO2 data set. Queries with an execution time over 15 minutes without producing any output are marked with "DNF". For the calculation of the geometric mean, a query runtime of 15 minutes is assumed for each cell which is marked with "DNF". Due to large result sets for query C-1 this limit is set to 60 minutes. The results of this evaluation are shown in Table 2.

Query	A1	A2	B1	B2	B3	C1	C2	C3	geom. mean
SpiderStore	**0.0016**	0.2084	0.3688	**0.0119**	0.5143	DNF	16,5464	319.0703	0.4521
Jena	55.203	142.155	DNF	DNF	DNF	DNF	DNF	DNF	578.3126
RDF-3X	0.0496	**0.0524**	**0.0471**	0.0936	**0.0482**	**657.2100**	**0.2056**	**2.4741**	0.3414
Virtuoso	0.766	0.127	0.71	0.46	3.223	2197,672	2.401	36.474	3,4420
#results	3	5	274	128	166	6,876,673	5313	35811	

Table 2: Query Runtimes on the YAGO2 data set (in seconds).

Considering the average execution time, RDF-3X performs better than all other stores tested. While *SpiderStore* in the average case is the second fastest system, two queries exist where *SpiderStore* outperforms the other stores. In the case of those queries, the coarse heuristics were able to generate a highly efficient execution plan. On the other side considering the queries C1-C3 *SpiderStore* has significant problems to arrange and optimize the execution order to obtain better results. Regarding query C1 even though intermediate results where present the total execution time did exceed the time limit of one hour. The reason for the performance shortcomings in the case of those queries is that the selectivity estimations based on the triple selectivity used for the generation of the execution plan can in some cases produce misleading execution orders. This is due to the fact that the cardinality of properties is not considered in the current algorithm. Regarding the other evaluated systems Jena TDB seem to have severe problems when executing queries on such large knowledge bases, because only two queries were able to determine the results within the given time frame. Whereas Virtuoso can be considered as the third fastest system, which has a stable performance without any negative outliers.

5. RELATED WORK

Several approaches exist for storing graph based data in databases. In the particular case of this paper we focus on RDF-stores because *SpiderStore*, the developed system, can be considered as part of this category. Hence we give a short overview about the different approaches available for storing RDF data.

For storing RDF-data, two approaches are prevailing. On the one hand the approach of mapping the RDF-data onto relational schema exists while on the other hand the approach of native RDF-stores exist.

The mapping of RDF-data onto relational databases is done either by facilitating a large triple table, where the columns correspond to the RDF atoms subject, predicate and object or by clustering the triples according to their predicate into several tables. The latter approach is called property tables. Both approaches are less than perfect because both suffer from severe performance drawbacks imposed by the architecture of relational databases. For example in the property tables approach the number of tables is equal to the number of properties in the worst case. Several approaches which extend these two main approaches when mapping RDF-data onto relational database have been developed and are benchmarked in [17]. Beside mappings to traditional relational database systems, mappings which make use of column oriented databases exist [1, 15]. In the case of native stores two major categories exist: (i) systems which have a main memory architecture and (ii) systems which use secondary memory storage as their primary storage layer. Systems falling into the first category are for example Brahms [9], Grin [18], Swift-OWLIM [11] or BitMat [2] as well as our system *SpiderStore* [3]. While Brahms is highly optimized for association finding and Grin for answering long path queries the goal of *SpiderStore* is to provide efficient query processing for arbitrary SPARQL queries. Swift-OWLIM is also a general purpose RDF-store, which has a strong emphasis on OWL-reasoning. Bitmat on the other hand represents a lightweight index structure which uses bitmap indexes to store space efficient projections of the three dimensional triple space. In contrast to the main memory based systems YARS2 [7], RDF-3X [12] can be considered as native systems of type (ii). Both systems make heavy use of index structures. While YARS facilitates six index structures for subject, predicate, object and the context, RDF-3X [12] generates index structures for all possible combinations and orderings of subject, predicate and object. Beside these huge number of index structures, RDF-3x makes heavy use of statistical information and has a highly sophisticated query execution engine, which is described in [13]. While such an enormous effort results in a good query performance, the management of these specific data structures can become quite complex. Neumann *et al.* therefore describe in [14] how a query centric RDF-engine can be extended to provide full-fledged support for updates, versioning and transactions. Beside these systems which can be clearly dedicated to either the category of memory-native, secondary memory-native or relational based systems several semantic web frameworks exist, which provide storage engines fitting in several or all of these categories. Examples of such frameworks are Sesame [5], Jena [19] and Virtuoso [6]. For all the systems described in this section several benchmarks exist [4, 12], which extensively compare those systems.

6. CONCLUSION AND FUTURE WORK

In this paper we presented the *SpiderStore* memory layout, which is able to store arbitrary graphs. The node centric layout has been discussed in detail and a formula for the estimation of the space consumption was described. An enhancement to the basic layout introducing a relative addressing schema was presented. Finally our experiments showed that a node centric layout is able to perform arbitrary SPARQL-queries on knowledge bases of up to 190 mio nodes with a performance comparable to highly sophisticated RDF-stores. This promises further performance improvements because the query optimisation approach, which has been developed in [3] is rather simple. Therefore future

work on *SpiderStore* will emphasize on the query execution engine to achieve excellent performance results on a scale of up to a billion triple.

7. REFERENCES

[1] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Scalable semantic web data management using vertical partitioning. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 411–422. VLDB Endowment, 2007.

[2] M. Atre, V. Chaoji, M. J. Zaki, and J. A. Hendler. Matrix "bit" loaded: a scalable lightweight join query processor for rdf data. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 41–50, New York, NY, USA, 2010. ACM.

[3] R. Binna, W. Gassler, E. Zangerle, D. Pacher, and G. Specht. Spiderstore: Exploiting main memory for efficient rdf graph representation and fast querying. *In Proceedings of the 1st International Workshop on Semantic Data Management (Sem-Data) at the 36th International Conference on Very Large Databases, Singapore*, Jan 2010.

[4] C. Bizer and A. Schultz. The berlin SPARQL benchmark. *International Journal On Semantic Web and Information Systems*, 5(1), 2009.

[5] J. Broekstra, A. Kampman, and F. Van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF schema. *The Semantic WebâĂŤISWC 2002*, pages 54–68, 2002.

[6] O. Erling and I. Mikhailov. RDF Support in the Virtuoso DBMS. *Networked Knowledge-Networked Media*, pages 7–24.

[7] A. Harth, J. Umbrich, A. Hogan, and S. Decker. YARS2: A federated repository for querying graph structured data from the web. *The Semantic Web*, pages 211–224.

[8] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: a spatially and temporally enhanced knowledge base from wikipedia. Research Report MPI-I-2010-5-007, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, November 2010.

[9] M. Janik and K. Kochut. Brahms: A workbench RDF store and high performance memory system for semantic association discovery. *The Semantic Web–ISWC 2005*, pages 431–445.

[10] A. Kemper and T. Neumann. Hyper: Hybrid OLTP & OLAP high performance database system. Technical Report TU-I1010, TU Munich, Institute of Computer Science, Germany, May 2010.

[11] A. Kiryakov, D. Ognyanov, and D. Manov. Owlim–a pragmatic semantic repository for owl. *Web Information Systems Engineering–WISE 2005 Workshops*, pages 182–192, Jan 2005.

[12] T. Neumann and G. Weikum. RDF-3X: a RISC-style engine for RDF. *Proceedings of the VLDB Endowment*, 1(1):647–659, 2008.

[13] T. Neumann and G. Weikum. Scalable join processing on very large rdf graphs. In *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*, pages 627–640, New York, NY, USA, 2009. ACM.

[14] T. Neumann and G. Weikum. x-rdf-3x: fast querying, high update rates, and consistency for rdf databases. *Proceedings of the VLDB Endowment*, 3(1-2), Jan 2010.

[15] L. Sidirourgos, R. Goncalves, M. Kersten, N. Nes, and S. Manegold. Column-store support for RDF data management: not all swans are white. *Proceedings of the VLDB Endowment*, 1(2):1553–1563, 2008.

[16] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA, 2007. ACM Press.

[17] Y. Theoharis, V. Christophides, and G. Karvounarakis. Benchmarking database representations of RDF/S stores. *The Semantic Web–ISWC 2005*, pages 685–701, 2005.

[18] O. Udrea, A. Pugliese, and V. Subrahmanian. GRIN: A graph based RDF index. In *Proceedings of the National Conference on Articial Intelligence*, volume 22, page 1465. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[19] K. Wilkinson, C. Sayers, H. Kuno, D. Reynolds, et al. Efficient RDF storage and retrieval in Jena2. In *Proceedings of SWDB*, volume 3, pages 7–8. Citeseer, 2003.

APPENDIX

A. QUERIES

A.1 YAGO data set

prefix rdfs:⟨http://www.w3.org/2000/01/rdf-schema#⟩
prefix xsd:⟨http://www.w3.org/2001/XMLSchema#⟩
prefix owl:⟨http://www.w3.org/2002/07/owl#⟩
prefix rdf:⟨http://www.w3.org/1999/02/22-rdf-syntax-ns#⟩
prefix yago:⟨http://www.mpii.de/yago/resource/⟩

A1: SELECT ?GivenName ?FamilyName WHERE { ?p yago:hasGivenName ?GivenName. ?p yago:hasFamilyName ?FamilyName. ?p rdf:type ?scientist. ?scientist rdfs:label "scientist". ?p yago:wasBornIn ?city. ?city yago:isLocatedIn ?switzerland. ?switzerland yago:hasPreferredName "Switzerland". ?p yago:hasAcademicAdvisor ?a. ?a yago:wasBornIn ?city2. ?city2 yago:isLocatedIn ?germany. ?germany yago:hasPreferredName "Germany". }

A2: SELECT ?name WHERE { ?a yago:hasPreferredName ?name. ?a rdf:type ?actor. 1 ?actor rdfs:label "actor". ?a yago:actedIn ?m1. ?m1 rdf:type ?movie. ?movie rdfs:label "movie". ?m1 yago:hasWikipediaCategory "German films". ?a yago:directed ?m2. ?m2 rdf:type ?movie2. ?movie2 rdfs:label "movie". ?m2 yago:hasWikipediaCategory "Canadian films". }

B1: SELECT ?name1 ?name2 WHERE { ?a1 yago:hasPreferredName ?name1. ?a2 yago:hasPreferredName ?name2. ?a1 rdf:type yago:wikicategory_English_actors. ?a2 rdf:type yago:wikicategory_English_actors. ?a1 yago:actedIn ?movie. ?a2 yago:actedIn ?movie. FILTER (?a1 != ?a2) }

B2: SELECT ?name1 ?name2 WHERE { ?p1 yago:hasPreferredName ?name1. ?p2 yago:hasPreferredName ?name2. ?p1 yago:isMarriedTo ?p2. ?p1 yago:wasBornIn ?city. ?p2 yago:wasBornIn ?city. }

B3: SELECT distinct ?name1 ?name2 WHERE { ?p1 yago:hasFamilyName ?name1. ?p2 yago:hasFamilyName ?name2. ?p1 rdf:type ?scientist1. ?p2 rdf:type ?scientist2. ?scientist1 rdfs:label "scientist". ?scientist2 rdfs:label "scientist". ?p1 yago:hasWonPrize ?award. ?p2 yago:hasWonPrize ?award. ?p1 yago:wasBornIn ?city. ?p2 yago:wasBornIn ?city. FILTER (?p1 != ?p2) }

C1: SELECT DISTINCT ?name1 ?name2 WHERE { ?p1 yago:hasFamilyName ?name1. ?p2 yago:hasFamilyName ?name2. ?p1 rdf:type ?scientist. ?p2 rdf:type ?scientist. ?scientist rdfs:label "scientist". ?p1 ?c ?city. ?p2 ?c2 ?city. ?city rdf:type ?cityType. ?cityType rdfs:label "city". }

C2: SELECT DISTINCT ?name WHERE { ?p yago:hasPreferredName ?name. ?p ?any1 ?c1. ?p ?any2 ?c2. ?c1 rdf:type ?city . ?c2 rdf:type ?city2. ?city2 rdfs:label "city". ?city rdfs:label "city". ?c1 yago:isCalled "London". ?c2 yago:isCalled "Paris". }

C3: SELECT ?p1 ?predicate ?p2 WHERE { ?p1 ?anypredicate1 ?city1. ?city1 yago:isCalled "Paris". ?p1 ?predicate ?p2. ?p2 ?anypredicate2 ?city2. ?city2 yago:isCalled "Hong Kong". }

Kriterien für Datenpersistenz bei Enterprise Data Warehouse Systemen auf In-Memory Datenbanken

Thorsten Winsemann
Otto-von-Guericke-Universität Magdeburg
Kanalstraße 18
D-22085 Hamburg
+49(0)160/90819410
thorsten.winsemann@t-online.de

Veit Köppen
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2
D-39106 Magdeburg
+49(0)391/67-19351
veit.koeppen@ovgu.de

ABSTRACT
Persistente Datenhaltung über mehrere Schichten innerhalb eines Enterprise Data Warehouse Systems ist notwendig, um den dort vorhandenen, sehr großen Datenbestand nutzen zu können, z.B. für Reporting und Analyse. Die Pflege und Wartung solcher meist redundanten Daten ist jedoch sehr komplex und erfordert einen hohen Aufwand an Zeit und Ressourcen. Neueste In-Memory-Technologien ermöglichen gute Performanz beim Datenzugriff, so dass sich die Frage stellt, welche Daten aus welchem Grund bzw. für welchen Zweck überhaupt noch persistent abgelegt werden müssen – und wie sich dies effizient entscheiden lässt. In diesem Papier präsentieren wir eine Übersicht von Gründen für Datenpersistenz, welche als Entscheidungsgrundlage bei der Problematik dient, Daten in Enterprise Data Warehouses auf In-Memory Datenbanken zu speichern.

Kategorien und Themenbeschreibung
H.2.7 [**Database Management**]: Datenbank-Administration – *Data Warehouse und Repository.*

Allgemeine Begriffe
Management, Design.

Schlüsselwörter
Enterprise Data Warehouse, Persistenz, In-Memory Datenbank.

1. EINLEITUNG
Heutige Data Warehouse Systeme (DWS) sind gekennzeichnet durch sehr große Datenvolumina [1]. Der Aufbau und Betrieb solcher Systeme erfordert hohe Anforderungen an die Datenbereitstellung, insbesondere hinsichtlich Performanz, Datengranularität, -flexibilität und -aktualität. Außerdem erfordern solche Einschränkungen die Speicherung zusätzlicher Daten. Verdichtungsebenen werden verwendet, um die Geschwindigkeit des Datenzugriffs zu verbessern, z.B. bei Reporting und Analyse. Diese Datenredundanz wiederum erfordert einen hohen Aufwand an Zeit und Ressourcen, um Datenkonsistenz zu gewährleisten Gleichzeitig wird eine zeitnahe Datenverfügbarkeit eingeschränkt. Neueste Ankündigungen versprechen auf In-Memory Datenbanken (IMDB) basierende Anwendungen, die auf größte Datenbestände – ohne zusätzliche Verdichtungsebenen – performant zugreifen können [2,3].

23rd GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011-03.06.2011, Obergurgl, Austria.

Die Verbesserung der Datenzugriffsgeschwindigkeit ist oftmals der Hauptgrund zusätzlicher Datenhaltung. Setzt man voraus, dass dies in einer IMDB weniger wichtig ist, so kommt die Frage auf: Wieviel Persistenz, d.h. nicht-flüchtige Datenspeicherung, ist in IMDB-basierten DWS überhaupt noch notwendig? Dies gilt insbesondere für Enterprise Data Warehouses (EDW). Ist es möglich, jede Art von Analyseanfrage direkt auf dem Rohdatenbestand abzusetzen, welcher „on-the-fly“ transformiert wird? Oder gibt es dennoch gewichtige Gründe der Datenspeicherung? Um diese Fragen zu beantworten, erläutern wir Persistenzgründe in EDW-Systemen und potentielle Konflikte zwischen Datenspeicherung und -verwendung. Zudem definieren wir Indikatoren zur Entscheidungsunterstüzung, ob Daten gespeichert werden sollen oder nicht.

Abschnitt 2 erläutert einführend die Besonderheiten von Enterprise Data Warehouses und einer Schichtenarchitektur. In Abschnitt 3 führen wir Gründe der Datenpersistenz in heutigen EDW auf und beschreiben mögliche Konflikte, welche aufgrund der Anforderungen der Datennutzung und der hierfür notwendigen Aufwände entstehen. Abschnitt 4 erläutert Datenpersistenz auf IMDB-basierten EDW sowie entscheidungsunterstützende Faktoren. Abschnitt 5 fasst die Teile zusammen und gibt einem Ausblick auf zukünftige Arbeiten.

2. EINE SCHICHTENARCHITEKTUR FÜR ENTERPRISE DATA WAREHOUSES
Ein Enterprise Data Warehouse ist ein Business Data Warehouse [4], stellt also entscheidungsunterstützende Informationen für das Management in allen Geschäftsbereichen zur Verfügung. Darüber hinaus stellen EDW eine wichtige Datenbasis für eine Vielzahl von Anwendungen dar, wie zum Beispiel Business Intelligence (BI), Customer Relationship Management (CRM) und die Planung. Innerhalb einer umfassenden Systemlandschaft stellen EDW-Systeme die „Single Source of Truth“ (vgl. [3]) für alle analyse-relevanten Daten des Unternehmens dar. Das heißt, sie ermöglichen eine allgemein gültige Sicht auf einen zentralen, harmonisierten, validen und konsistenten Datenbestand. Ein EDW integriert sehr große Datenbestände aus einer Vielzahl unterschiedlicher Quellsysteme des Konzerns – oftmals weltweit, so dass Daten verschiedener Zeitzonen zusammengeführt werden müssen. Dies erfordert eine fortlaufende Datenverfügbarkeit mit gleichzeitigem Datenladen und -zugriff. Zudem gibt es weitere Anforderungen an den Datenbestand: Ad-hoc-Berichte, „near-real-time“ Verfügbarkeit und Anwendungen, wie beispielsweise CRM, mit einem Bedarf an detaillierten historischen Daten. Ein sich ändernder Informationsbedarf muss schnell und flexibel

gedeckt werden können. Zudem wird ein umfassendes Berechtigungskonzept zur Sicherung sensibler Daten vorausgesetzt. Somit sind verschiedene Gründe von Datenpersistenz spezifisch in einem EDW.

Persistenz in einem Data Warehouse ist eng verbunden mit dessen Architektur. Eine allgemeine Referenzarchitektur (vgl. z.B. [5,6,7,8]) definiert drei Bereiche, welche die drei Arten der Datenverarbeitung darstellen: Datenbeschaffung in der „Staging Area", Datenbearbeitung in der Basisdatenbank, Datenbereitstellung im Data-Mart-Bereich. In diesem eher groben Modell ist Datenspeicherung in jedem Bereich implizit [9]. Die in [10] vorgestellte Schichtenarchitektur (Abb. 1) entwickelt diesen Ansatz hinsichtlich der bereits erwähnten Anforderungen an ein EDW weiter. Die Schichten werden zweckbestimmter; jede der fünf Schichten repräsentiert einen Bereich, in dem der Wert der Daten hinsichtlich ihrer Verwendung gesteigert wird, wenn dies notwendig ist. Eine Schicht bedeutet aber nicht zwangsläufig Datenspeicherung. Wird beispielsweise ein Datenbestand nach der Harmonisierung gespeichert und ist bereits für Analysezwecke verwendbar, so muss er nicht auf die oberste Schicht „durchgereicht" und dort nochmals gespeichert werden. In einem ersten Schritt muss entschieden werden, in welchem Format die Daten wo zu speichern sind. Deshalb ist zunächst der Zweck der Datenverwendung als Grund der Datenspeicherung zu ermitteln.

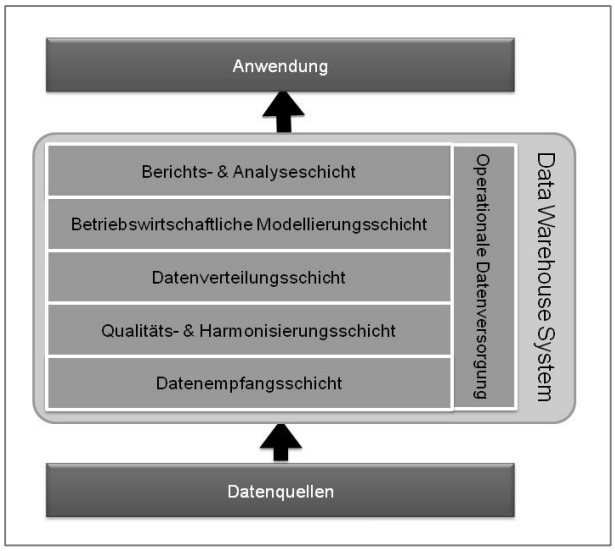


Abb. 1. Schichtenarchitektur für EDW (nach [10])

Die in Abb. 1 dargestellte Schichtenarchitektur für EDW unterteilt sich in die folgenden Bereiche:
Die *Datenempfangsschicht* stellt den „Posteingang" des EDW dar; extrahierte Daten werden ohne oder mit geringen Modifikationen entgegengenommen und abgelegt.
Innerhalb der *Qualitäts- & Harmonisierungschicht* werden die Daten technisch und semantisch integriert. Das beinhaltet Dublettenerkennung, Aspekte der Informationsintegration (vgl. [11]) etc., und entspricht der Transformation des ETL-Prozesses.
Die *Datenverteilungsschicht* enthält harmonisierte und integrierte Unternehmensdaten ohne betriebswirtschaftliche Logik und bildet somit die einheitliche Datenbasis für alle Anwendungen.
In der *Betriebswirtschaftlichen Modellierungsschicht* werden Daten hinsichtlich der Geschäftsanforderungen transformiert; zum Beispiel werden Finanz- mit Logistikdaten verknüpft.
In der *Berichts- & Analyseschicht* werden Daten hauptsächlich verwendungsbezogen transformiert, um performante Zugriffe zum Beispiel beim Reporting oder der Analyse zu gewährleisten.
Innerhalb der *Operationalen Datenversorgung* werden Daten für sehr spezielle Anwendungsfälle und Anforderungen zur Verfügung gestellt, zum Beispiel bei „near real-time Reporting".

Obwohl die Grenzen fliessend sind, können die fünf Schichten den drei Bereichen der Datenverarbeitung wie folgt zugeordnet werden: Datenbeschaffung in der Datenempfangs- und der Qualitäts- & Harmonisierungsschicht, Datenbearbeitung in der Datenverteilungs- und der Betriebswirtschaftlichen Modellierungsschicht, sowie Datenbereitstellung in den Data Marts der Berichts- & Analyseschicht.

3. GRÜNDE FÜR DATENPERSISTENZ

Zwei Gründe von Datenpersistenz im Data Warehouse werden hauptsächlich genannt: Speicherung der bereits transformierten Daten in der Basisdatenbank und Speicherung redundanter, aggregierter Daten im Data-Mart-Bereich. Darüber hinaus gibt es allerdings noch eine Vielzahl von Persistenzgründen im EDW. Hierzu zählen technische Einschränkungen, Governance-Bestimmungen des Unternehmens und Gesetze, Vereinfachung der Datenhandhabung oder ein subjektives Sicherheitsbedürfnis. Sofern uns bekannt, werden Gründe für Datenpersistenz in der Literatur nur selten erwähnt; zudem vermissen wir eine vollständige Auflistung, wie im Folgenden beschrieben.

Entkopplung des Quellsystems: Zur Entlastung des Quellsystems werden Daten direkt nach ihrer erfolgreichen Extraktion im Eingangsbereich des DW gespeichert; hierbei werden die Daten nicht oder nur in geringem Maße verändert (z.B. werden Herkunftsmerkmal oder Zeitstempel angefügt).
Datenverfügbarkeit: Oftmals sind Daten nicht mehr oder nur in einem veränderten Zustand verfügbar; hierzu zählen z.B. Daten aus dem Internet, aus Dateien, welche regelmäßig überschrieben werden, oder aus Altsystemen. Zudem können Netzwerkprobleme dazu führen, dass auf Daten nicht zugegriffen werden kann. Die Speicherung im Warehouse garantiert die Datenverfügbarkeit.
Komplexe Transformationen: Aufgrund ihrer Komplexität sind einige Transformationen sehr zeit- und ressourcenaufwendig, so dass die Daten gespeichert werden, um ein wiederholtes Transformieren zu vermeiden.
Abhängige Transformationen: Unter „abhängige Transformation" verstehen wir solche, deren Durchführung den Zugriff auf weitere Daten erfordert; z.B. erfordert die Verteilung eines Bonus' auf die einzelnen Mitarbeiter die Gesamtanzahl der Mitarbeiter. Diese notwendigen Daten werden im DW gespeichert, um das korrekte Durchlaufen der Transformation zu gewährleisten.
Veränderte Transformationsregeln: Regeln können geändert werden. Besitzen die Daten kein Zeitmerkmal und werden die Transformationen nicht „historisiert", so ist eine identische Transformation nicht mehr möglich.
Aufwendige Datenwiederherstellung: Sind Daten nicht mehr im DWS verfügbar (z.B. weil sie archiviert sind), ist eine Wiederherstellung aufwendig, so dass sie gespeichert werden.
Datenzugriffsgeschwindigkeit: Die redundante Speicherung von Daten in Verdichtungsebenen oder materialisierten Sichten zum Zwecke der Performanzverbesserung beim Datenzugriff stellt einen der häufigsten Gründe für die Einführung einer weiteren Persistenzebene dar.
„En-bloc Datenversorgung": Üblicherweise fliessen neue Daten, aus verschiedenen, gegebenenfalls weltweiten Quellen, zeitlich

verteilt in ein EDW. Nachdem diese syntaktisch und semantisch integiert wurden, werden sie zwischengespeichert und erst zu einem bestimmten Zeitpunkt in die Datenbasis des Warehouse gespielt. Hierdurch wird ein zeitlich definierter, konstanter und in sich plausibler Datenbestand für die darauf aufsetzenden Anwendungen gewährleistet.

Konstante Datenbasis: Einige, auf Daten des DWS aufbauende Applikationen, wie beispielsweise Planung, erfordern eine konstante Datenbasis, welche sich während der Benutzung nicht ändern darf und deswegen separiert gespeichert wird.

„Single Version of Truth“: Transformierte Daten werden nach unternehmensweit gültigen Definitionen, aber ohne spezielle Geschäftslogik gespeichert. Hierdurch wird ein einheitlicher, vergleichbarer Datenbestand geschaffen, auf den die jeweiligen Geschäftsbereiche und Anwendungen zugreifen können [10].

„Corporate Data Memory“: Alle ins EDW extrahierten Daten werden ohne oder nur mit minimaler Veränderung (z.B. durch Anfügen eines Herkunftsmerkmals) gespeichert, um eine größtmögliche Autarkie und Flexibilität von Datenquellen zu ermöglichen. So können Datenbestände (wieder-)hergestellt werden, ohne auf die Quellsysteme zuzugreifen, in denen die Daten möglicherweise schon gelöscht wurden oder nicht mehr zum Zugriff bereitstehen (vgl. [10]).

Komplex-abweichende Daten: Zu integrierende Daten können in Syntax und Semantik sehr von der im EDW üblichen abweichen; eine (zumeist schrittweise) Eingliederung erfolgt erst nach vorheriger Speicherung.

Data-Lineage: Daten in Berichten oder Analysen sind häufig Ergebnis mehrstufiger Transformationsprozesse. Um eine Rückverfolgung zu den Urprungsdaten zu erleichtern oder zu ermöglichen, etwa zur Validierung, können gespeicherte Zwischenergebnisse erforderlich sein (vgl. [12]).

Komplexe Berechtigungen: Anstatt der Definition und Erstellung komplexer Benutzerberechtigungen (z.B. auf Merkmale oder auf Feldinhalte), werden bestimmte Data-Marts mit den Daten erstellt und die Berechtigungen auf dem Data-Mart vergeben.

„Informationsgewährleistung“: Viele EDW haben zu gewährleisten, dass die Daten den Benutzern in einem bestimmten Zeitraum (oftmals sogar 24 Stunden pro Tag) zur Verfügung stehen und für die Anwendungen genutzt werden können. Hierfür werden in der Regel besonders kritische Datenbestände zusätzlich gespeichert.

Corporate Governance: Daten werden gemäß den Compliance-Vorgaben des jeweiligen Unternehmens (Corporate Governance) gespeichert; z.B., um eine aufgrund bestimmter Daten getroffene Managemententscheidung auch im Nachhinein beurteilen zu können.

Gesetze und Bestimmungen: Zudem gibt es auch Gesetze und Bestimmungen, die eine Datenspeicherung begründen; für Deutschland existieren solche beispielsweise im Finanzbereich (Handelsgesetzbuch u.a., [13]) und bei der Produkthaftung [14].

Subjektive Sicherheit: Letzlich kann das sujektive Bedürfnis an Sicherheit ein Grund für Datenspeicherung sein.

Persistenz beinhaltet häufig redundante Datenhaltung, da sowohl Quell- als auch transformierte Zieldaten gespeichert werden; ausschließliches Speichern der Zieldaten bedeutet in aller Regel Datenverlust. Hieraus entstehen hohe Anforderungen, nicht nur an die Hardware (Speicherplatz etc.), sondern auch an die Datenpflege, um etwa die Datenbestände konsistent zu halten.

Der Betrieb produktiver DWS führt zwangsläufig zu Konflikten zwischen den Anforderungen der Datennutzung, wie z.B. Performanz bei Reporting und Analyse, und dem Aufwand an Zeit und Ressourcen, die notwendigen Voraussetzungen hierfür zu schaffen. Im Folgenden beschreiben wir diese Anforderungen und ihre Konsequenzen kurz.

Wie bereits erwähnt, stellt ein EDW häufig die Datenbasis für verschiedene Anwendungen dar; das hohe Datenvolumen resultiert aus den unterschiedlichen Anforderungen dieser Applikationen an die Daten. Der Bedarf an Detailinformationen erfordert viele Daten feinster Granularität. Der Bedarf an historischen Informationen erfordert eine lange Historisierung der Daten. Schließlich wird eine große Bandbreite an Daten gesammelt, beispielsweise für Data-Mining-Szenarios. Dieser große Datenbestand muss für seine Verwendung aufbereitet werden; z.B. ist eine gute Berichtsperformanz sicherzustellen.

Eine hohe Geschwindigkeit beim Datenzugriff wird zumeist durch ein reduziertes Datenvolumen erreicht – durch den Aufbau materialisierter Sichten oder Verdichtungsebenen. Ein einfaches Beispiel hierfür ist die Verdichtung tagesgenauer Daten auf Monat, mit einem Faktor von etwa 30. Pflege und Verwaltung solcher Redundanzen erfordert nicht nur Speicherplatz, sondern auch zusätzlichen Aufwand, die Daten aktuell und konsistent zu halten (vgl. [15]). Da diese Aufwände Zeit kosten, ist die Verfügbarkeit der Daten eingeschränkt. Außerdem beschränken die vordefinierten Datenbestände die Flexibilität der Daten hinsichtlich geänderter und neuer Nutzungsanforderungen.

Ein komplexer Staging-Prozess mit mehreren Schichten persistenter Daten ist einer schnellen Datenverfügbarkeit gegensätzlich. Dies ist insbesondere auch bei Konzepten für „Near-Realtime Reporting“ zu beachten [16].

4. PERSISTENZ BEI IN-MEMORY

EDW-Architekturen basieren gewöhnlich auf relationalen Datenbanken (RDBMS) mit Stern-, Snowflake- oder Galaxy-Schema als Grundlage der Datenmodellierung; siehe z.B. [15,17]. Solche Modelle ermöglichen gute Performanz bei On-line Analytical Processing. Große Datenbestände müssen aber auch hier mittels materialisierter Sichten und Verdichtungsebenen reduziert werden – mit den bereits beschriebenen Konsequenzen.

Spaltenbasierte Datenbanken (vgl. [18,19]) werden aufgrund ihrer Vorteile bei der Datenkomprimierung und dem Lesezugriff [20,21] im Data Warehousing genutzt (z.B. [22,23]). Seit einigen Jahren wird spaltenbasierte In-Memory-Technologie in kommerziellen Data Warehouse Produkten verwendet (z.B. „SAP NetWeaver® Business Warehouse Accelerator“ [24,25], „ParAccel Analytic Database™“ [26]), um verbesserte Antwortzeiten beim Zugriff auf sehr große Datenbeständen zu erzielen. Solche Technologien erlauben das Laden und Abfragen von Datenvolumina im Teradatenbereich mit guter Performanz. Es wurden bereits Installationen angekündigt, die On-Line Transactional und Analytical Processing in einem System mit bis zu 50 TB Daten im Hauptspeicher ermöglichen [27]. In diesem Bereich ist SanssouciDB als ein erstes Produkt zu nennen [3].

Diese technologischen Veränderungen führen zu der Frage, in welchem Maße Datenpersistenz in IMDB-basierten EDW-Systemen noch notwendig ist. Es wird suggeriert, dass bei In-Memory-Technologie keine Daten zusätzlich zu den gespeicherten Urspungsdaten persistent gehalten werden müssen. Alle abgeleiteten Daten, insbesondere die für Analysezwecke aggregierten oder verdichteten, werden „on-the-fly“ ermittelt und zur Verfügung gestellt [3,27]. Dies gilt jedoch nur für einige der o.g. Gründe, wie im folgenden deutlich wird. In diesem Zusammenhang fokussieren wir uns auf Datenbanken, die ACID-fähig sind, inklusive Dauerhaftigkeit (z.B. SanssouciDB, solidDB

von IBM und TimesTen von Oracle [3,28,29]). Persistenz ist hier zu unterscheiden von volatiler Speicherung, bei der die Daten in flüchtigem Speicher gehalten werden und verloren gehen, wenn das System heruntergefahren wird oder abstürzt.

4.1 Notwendigkeit der Datenpersistenz

Eine Entscheidung für Datenpersistenz kann nicht ausschließlich nach einem kostenbasierten Vergleich von „Plattenplatz und Kosten des Updates versus Geschwindigkeitsgewinn der Analyse" getroffen werden. Zunächst ist der Grund der Datenspeicherung (s. Abschnitt 3) zu berücksichtigen. Im RDBMS-basierten DWS ist diese Überlegung weniger ausgeprägt, da die geringere Leistungsfähigkeit der Datenbank und der daraus resultierende Bedarf an aggregierten Daten das Speichern begründet. Um die Notwendigkeit von Datenpersistenz zu ermitteln, führen wir eine Einteilung dieser Gründe ein: die Speicherung der Daten ist nur unterstützend, essentiell oder sogar verpflichtend.

Tab. 1. Persistenzgründe, nach Notwendigkeit gruppiert

Grund/Zweck	Notwendigk.	Gruppierung
Gesetze und Bestimmungen	Verpflichtend	-
Corporate Governance	Verpflichtend	-
Datenverfügbarkeit	Verpflichtend	-
Veränderte Transformationsregeln	Verpflichtend	-
Abhängige Transformationen	Verpflichtend	-
Quellsystem-Entkopplung	Essentiell	Aufwand
Aufwendige Datenwiederherstellung	Essentiell	Aufwand
Komplexe Transformationen	Essentiell	Aufwand
Konstante Datenbasis	Essentiell	Aufwand
"En-bloc Datenversorgung"	Essentiell	Vereinfachung
Komplex-abweichende Daten	Essentiell	Vereinfachung
Data-Lineage	Essentiell	Vereinfachung
Komplexe Berechtigungen	Essentiell	Vereinfachung
„Single Version of Truth"	Essentiell	Design
„Corporate Data Memory"	Essentiell	Design
„Informationsgewähr"	Essentiell	Sicherheit
Zugriffsgeschwindigkeit	Essentiell	Performanz
Subjektive Sicherheit	Unterstützend	-

Verpflichtend zu speichern sind Daten aufgrund von Gesetzen und Bestimmungen sowie Regeln der Corporate Governance. Zudem gilt dies für Daten, welche nicht wieder hergestellt werden können, weil sie nicht mehr oder nur verändert zur Verfügung stehen oder aufgrund geänderter Transformation nicht mehr erstellt werden können. Auch Daten, die bei der Transformation anderer Daten benötigt werden, sind zu speichern, wenn eine gleichzeitige Verfügbarkeit nicht gewährleistet werden kann.
Essentielle Datenpersistenz kann in bestimmte Gruppen unterteilt werden: Zum einen Daten, deren Wiederherstellung nur mit sehr hohem *Aufwand* (an Zeit und Ressourcen) möglich ist, wie z.B. archivierte oder komplex transformierte Daten. Hierbei ist „sehr hoch" allerdings subjektiv und näher zu untersuchen. Eine zweite Gruppe sind Daten, die gespeichert werden, um den Betrieb des Warehouse oder einzelner Anwendungen zu *vereinfachen*; hierzu zählen speziell abgelegte Plandaten oder Data-Marts mit Berechtigungen für besondere Benutzer. Drittens begründet sich Persistenz mit spezieller Konzeption (*Design*) des EDW: „Single Version of Truth", „Corporate Data Memory" zählen u.a. hierzu. *Sicherheit*, etwa zur Gewährleistung der Datenverfügbarkeit, stellt eine weitere Gruppe dar. Letztlich ist Datenspeicherung für eine hohe *Performanz* ein Grund; oftmals der, dem das größte redundante Datenvolumen zugrunde liegt.
Daten, deren Speicherung *unterstützend* ist, beinhalten solche, die wegen subjektiver Sicherheitsüberlegungen abgelegt werden.
Eine komplette Auflistung der Persistenzgründe, gruppiert nach Notwendigkeiten, zeigt Tab. 1.

Abb. 2 zeigt ein vereinfachtes Entscheidungsdiagramm für Datenpersistenz, in dem z.B. unscharfe Begriffe wie „aufwendig", „komplex" und „häufig" abhängig von der Domäne spezifiziert werden müssen. Die ersten drei Abfragen betreffen verpflichtende Gründe, d.h. die Daten sind – auch in IMDB-basierten EDW – zu speichern. Bei den aus anderen Gründen gespeicherten Daten sind die Entscheidungsgrundlagen sehr vielfältig. Stellen die Daten eine „Single Version of Truth" dar oder umfasst das EDW-Design ein „Corporate Data Memory", so sind diese Daten zu speichern. Ist hingegen eine komplexe Reproduktion oder Transformation Grund des Speicherns, so müssen z.B. Zugriffshäufigkeit und Sicherstellung der Verfügbarkeit in Betracht gezogen werden, um entscheiden zu können.

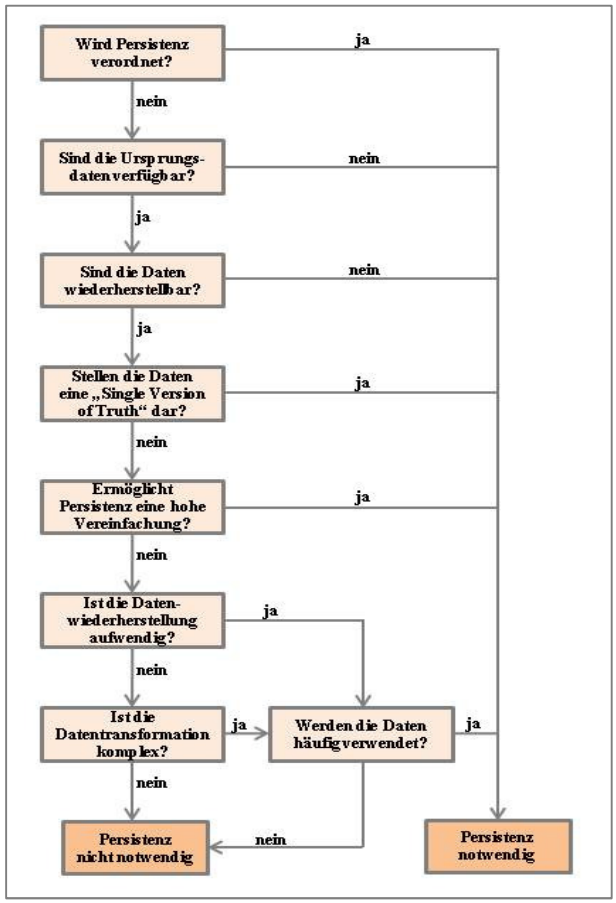


Abb. 2. Entscheidungsdiagramm „Datenpersistenz"

4.2 Bewertung der Persistenz in IMDBs

Alle nicht-verpflichtend gespeicherten Daten sind Gegenstand der Betrachtung bei der Frage nach Persistenz in IMDB-basierten EDW. Insbesondere betrifft dies Daten, die zur Verbesserung der Zugriffsperformanz oder aufgrund komplexer Transformation redundant abgelegt werden. Das bedeutet aber nicht, dass allein die Geschwindigkeit der Datenverarbeitung in einem solchen System jede Art zusätzlicher Speicherung überflüssig machen wird. Dies gilt beispielsweise für die „En-bloc Datenversorgung“ oder beim Aufbau einer konstanten Datenbasis für Planungsläufe. IMDB-Snapshot-Mechanismen, wie in [30] erläutert, halten den Datenbestand zumeist nicht über die benötigte Zeit von Stunden oder Tagen konstant. Hier kommt es nicht auf eine schnelle Versorgung mit neuen Daten an, sondern auf die Herstellung eines über einen definierten Zeitraum unveränderten Datenbestands. Zeitstempelverfahren in In-Memory-Konzepten [3,30] können ein Lösungsszenario sein. Für die Ersetzung eines „Corporate Data Memory“ jedoch sind diese Verfahren nicht geeignet, wenn Daten verschiedener Quellsysteme integriert werden, was insbesondere für ein EDW gilt. Auch werden Persistenzgründe wie komplexe Berechtigungen oder Data-Lineage weiterhin gültig bleiben.

Die Erfahrung zeigt, dass technische Beschränkungen meist früher als erwartet eintreten, so dass die Systemressourcen für die an sie gestellten Aufgaben nicht mehr ausreichen werden. Die Möglichkeit, auf sehr viele Daten mit sehr hoher Performanz zuzugreifen, wird neue Bedürfnisse wecken. Es werden neue Anforderungen aufkommen und die Datenmengen zunehmen. Aufgrund dessen ist auch bei IMDB-basierten Systemen zu betrachten, ob wiederholte, gleichartige Zugriffe und Bearbeitung von Daten „on-the-fly“ nicht durch Vorhalten der Daten im benötigten Format günstiger ist. Dies gilt insbesondere für Datenbestände, auf die häufig zugegriffen wird und die sich nicht oder nur wenig ändern, wie beispielsweise die geschlossenen Jahres-, Quartals- oder Monatsabschlüsse der Finanzbuchhaltung.

Eine weitere Frage in diesem Zusammenhang ist das Datenformat, in dem gespeichert wird, d.h. auf welcher Transformationsstufe die Speicherung optimal ist. Hierbei ist das Format zu ermitteln, welches eine möglichst flexible Verwendung der Daten bei einer größtmöglichen Vermeidung wiederholter, gleichartiger Transformationen darstellt. Dies kann durch kostenbasierte Laufzeitmessungen geschehen, wie folgendes Beispiel erläutert:

Gegeben sei ein Rohdatenbestand (R), der über eine mehrstufige Transformation (Tn; n={1,2,3}) für Analysen (A) abgefragt wird. Zu vergleichen ist, ob es effizienter ist, die Daten nach den einzelnen Transformationen persistent zu speichern (P), sie volatil zu halten (V), oder sie jeweils „on-the-fly” neu zu ermitteln:

(1) R ➔ T1 + P ➔ T2 + P ➔ T3 + A
(2) R ➔ T1 + P ➔ T2 + V ➔ T3 + A
(3) R ➔ T1 + P ➔ T2 ➔ T3 + A
(4) R ➔ T1 + V ➔ T2 ➔ T3 + A
(5) R ➔ T1 ➔ T2 ➔ T3 + A

Weitere Indikatoren, die hier betrachtet werden müssen, sind:

Datenvolumen: Ist die Datenmenge so groß, dass die zur Nutzung notwendige, oftmals sehr komplexe Aufbereitung überhaupt bzw. in einer akzeptablen Zeit „on-the-fly“ durchgeführt werden kann?

Häufigkeit der Datennutzung: Wird auf die Daten so häufig zugegriffen, dass der Nutzen einer zusätzlichen Materialisierung deren Kosten aufwiegt?

Häufigkeit von Datenänderungen: Wird ein Datenbestand so oft geändert (durch Update, Insert, Delete), dass der Aufwand, z.B. für die Konsistenzsicherung der abgeleiteten Verdichtungsebenen, geringer ist als der Geschwindigkeitsgewinn der Anwendung? Und: Wie aufwendig sind diese Änderungen?

Untersuchungen dieser Art sind auch bei RDBMS-basierten DWS valide. Hierbei lässt die Leistungsfähigkeit einer IMDB jedoch als Ergebnis erwarten, dass Transformationen eher „on-the-fly“ als mit redundanter Persistenz durchgeführt werden.

Einige IMDB ermöglichen die Festlegung unterschiedlicher Kriterien zur Dauerhaftigkeit, z.B. durch Definition temporärer Tabellen [29,30]. Hierdurch können Daten, die nicht verpflichtend zu speichern sind, nur in flüchtigem Speicher gehalten werden. Da ein Herunterfahren oder Absturz der Datenbank relativ selten geschieht, sind die Wartungskosten für solche Daten gering. Ein beispielhafter Anwendungsfall hierfür ist die Ermittlung von RFM-Attributen (Recency, Frequency, Monetary) zur Kundenkategorisierung im CRM-Umfeld [31]. Die Emittlung (s. Abb. 3) basiert auf Kundenstamm- und Transaktionsdaten (Kassenbons, Aufträge, Fakturen) und umfasst Selektionen, Kalkulationen, Währungsumrechnungen, Look-Ups zu komplexen Steuerungsdaten etc. Die berechneten Attribute werden zeitnah aktualisiert benötigt, sowohl im DWS, als auch im CRM-System. Zu berücksichtigen ist, dass es sich hierbei um oft sehr große Mengen an Daten handelt, mehrere Millionen Kunden mit jeweils einer zweistelligen Anzahl Transaktionen. Diese Datenbestände ändern sich häufig, so dass auch die RFM-Attribute laufend aktualisiert werden müssen. Da die Ermittlung der Attribute reproduzierbar ist, kann das Vorhalten dieser Daten ausschließlich im flüchtigen Speicher einer Persistierung vorzuziehen sein.

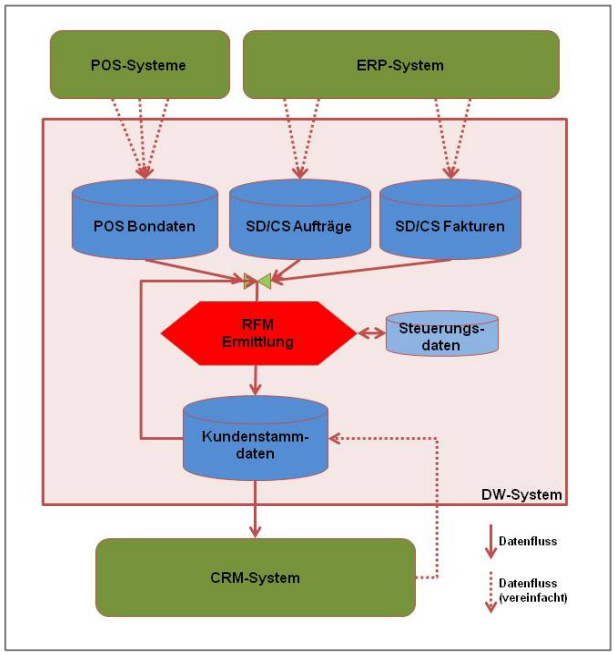


Abb. 3. Ermittlung von RFM-Attributen

Festzuhalten bleibt, dass in einem IMDB-basierten EDW viele Daten nicht mehr gespeichert werden müssen, die in einem RDBMS-basierten aufgrund von Performanzgewinn redundant zu halten sind. Höhere Zugriffsgeschwindigkeiten werden es ermöglichen, Daten „on-the-fly“ für die Nutzung aufzubereiten, insbesondere solche mit relativ einfacher Transformationslogik, wie z.B. Aggregation, Joins etc. Eine Vielzahl materialisierter Sichten wird zu virtuellen Sichten.

5. FAZIT UND AUSBLICK

Enterprise Data Warehouses sind komplexe Systeme mit speziellen Anforderungen an Datenbestand und Datenhaltung, für die eine Architektur dedizierter, zweckbestimmter Schichten geeignet ist. Die Notwendigkeit von Datenpersistenz in solchen Systemen kann nur durch den Zweck der Daten begründet werden. Diese Sichtweise wird bei IMDB-basierten EDW noch entscheidender. Wir beschreiben Gründe der Datenpersistenz und unterteilen sie in verpflichtende, essentielle und unterstützende. Darauf aufbauend nähern wir uns der Entscheidungsfindung, ob Daten in solchen Systemen gespeichert werden.

Persistente Datenhaltung wird es auch in EDW-Systemen auf IMDB geben. Ein großer Anteil heutiger persistierter Daten wird allerdings nur flüchtig gespeichert oder „on-the-fly" berechnet. Zudem wird die Frage aufkommen nach dem Format, in dem die Daten abgelegt werden. Die Antwort hierauf wird nicht einfach zu ermitteln sein; es handelt sich hierbei vielmehr um eine multidimensionale Gewichtung verschiedener Faktoren, wie: Aufwand für Transformation, Speicherung und Updating, Anzahl und Zeit von Datenabfrage und -aktualisierung.

Zukünftige Arbeiten werden eine detaillierte Aufstellung von Persistenzgründen mit ausführlichen Beispielen umfassen. Darüber hinaus werden Indikatoren definiert und beschrieben werden, die die Entscheidungsfindung für/gegen Datenpersistenz unterstützen. Dies umfasst sowohl messbare, wie beispielsweise Vergleiche von Laufzeiten und Wartungsaufwänden zwischen Datenbeständen in verschiedenen Speicherzuständen, als auch nicht-messbare Indikatoren. So wird ermittelt, ob Entscheidungen durch Berechnungen getroffen oder hierdurch zumindest unterstützt werden können.

6. DANKSAGUNG

Diese Arbeit wird teilweise unterstützt vom Bundesministerium für Bildung und Forschung (BMBF) innerhalb des ViERforES-II-Projekts (Nr. 01IM10002B).

7. LITERATUR

[1] R. Winter: „Why Are Data Warehouses Growing So Fast?"; www.b-eye-network.com/print/7188 {03.05.2011}; 2008.

[2] H. Plattner et al.: „ETL-less Zero Redundancy System and Method for Reporting OLTP Data" (US 2009/0240663 A1); US Patent Application Publication; 2009.

[3] H. Plattner, A. Zeier: „In-Memory Data Management"; Springer-Verlag, Berlin; 2011.

[4] B.A. Devlin, P.T. Murphy: „An architecture for a business and information system"; in: IBM Systems Journal 27(1), S.60-80; 1988.

[5] V. Poe: „Building a data warehouse for decision support"; Prentice Hall PTR, Upper Saddle River; 1996.

[6] H. Muksch, W. Behme (Hrsg.): „Das Data Warehouse-Konzept"; Gabler-Verlag, Wiesbaden, 4.Auflage; 2000.

[7] P. Gluchowski; P. Chamoni: „Entwicklungslinien und Architekturkonzepte des On-Line Analytical Processing"; in: Analytische Informationssysteme, Springer-Verlag, 3.Auflage, S.143-176; 2006.

[8] T. Zeh: „Referenzmodell für die Architektur von Data-Warehouse-Systemen (Referenzarchitektur)"; www.tzeh.de/doc/gse-ra.ppt {03.05.2011}; 2008.

[9] B.A. Devlin: „Business Integrated Insight (BI²)"; www.9sight.com/bi2_white_paper.pdf {03.05.2011}; 2009.

[10] SAP: „PDEBW1 - Layered Scalable Architecture (LSA) for BW"; Schulungsunterlagen, SAP AG; 2009.

[11] U. Leser, F. Naumann: „Informationsintegration"; dpunkt-Verlag, Heidelberg; 2007.

[12] Y. Cui, J. Widom: „Lineage Tracing for General Data Warehouse Transformations"; in: The VLDB Journal 12(1), S.41-58; 2003.

[13] §§239,257 HGB (Stand: 01.03.2011); §25a KWG (Stand: 01.03.2011); §147 AO (Stand: 08.12.2010).

[14] §13 ProdHaftG (Stand: 19.07.2002).

[15] W. Lehner: „Datenbanktechnologie für Data-Warehouse-Systeme"; dpunkt-Verlag, Heidelberg; 2003.

[16] J. Langseth: „Real-Time Data Warehouses: Challenges and Solutions"; on: www.dssresources.com {03.05.2011}; 2004.

[17] R. Kimball, M. Ross: „The Data Warehouse Toolkit"; Wiley Publishing Inc., Indianapolis, 2.Auflage; 2002.

[18] G.P. Copeland, S.N. Khoshafian: „A Decomposition Storage Model"; in: SIGMOD`85, S.268-279; 1985.

[19] M.J. Turner et al.: „A DBMS for large statistical databases"; in: 5th VLDB`79, S.319-327; 1979.

[20] D.J. Abadi et al.: „Integrating Compression and Execution in Column-Oriented Database Systems"; in: SIGMOD`06, S.671-682; 2006.

[21] D.J. Abadi: „Query Execution in Column-Oriented Database Systems"; Dissertation, MIT; 2008.

[22] M. Stonebraker et al.: „C-Store: A Column-oriented DBMS"; in: 31st VLDB`05, S.553-564; 2005.

[23] D. Slezak et al.: „Brighthouse: An Analytic Data Warehouse for Ad-hoc Queries"; in: PVLDB 1(2), S.1337-1345; 2008.

[24] T. Legler et al.: „Data Mining with the SAP NetWeaver BI Accelerator"; in: 32nd VLDB`06, S.1059-1068; 2006.

[25] J.A. Ross: „SAP NetWeaver® BI Accelerator"; Galileo Press Inc., Boston; 2009.

[26] ParAccel: „PARACCEL ANALYTIC DATABASE™"; www.paraccel.com/wp-content/uploads/2010/07/PA_DS.pdf {03.05.2011}; 2011.

[27] H. Plattner: „A Common Database Approach for OLTP and OLAP Using an In-Memory Column Database"; in: SIGMOD`09, S. 1-2; 2009.

[28] IBM: IBM solidDB™; www.ibm.com/software/data/soliddb {03.05.2011}; 2010.

[29] Oracle: „Extreme Performance Using Oracle TimesTen In-Memory Database"; www.oracle.com/technetwork/database/timesten/overview/wp-timesten-tech-132016.pdf {03.05.2011}; 2009.

[30] A. Kemper, T. Neumann: „HyPer: Hybrid OLTP&OLAP High PERformance Database System"; www3.in.tum.de/research/projects/HyPer/HyperTechReport.pdf {03.05.2011}; 2010.

[31] J. Stafford: „RFM: A Precursor of Data Mining"; www.b-eye-network.com/view/10256 {03.05.2011}; 2009

Ein Verfahren zur automatischen Erstellung eines visuellen Wörterbuchs für die Bildsuche

Magdalena Rischka
Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
D-40225 Düsseldorf, Deutschland
rischka@cs.uni-duesseldorf.de

ZUSAMMENFASSUNG

Das Internet bietet eine enorme Anzahl an Bildern. Bildsuchmaschinen stehen vor der Herausforderung Bilder effektiv und effizient zu erschließen. Die klassischen Arten der Bildsuche, die stichwort- und die inhaltsbasierte Bildsuche, haben Nachteile. Ein Retrieval-Modell, welches die Vorteile beider Sucharten integriert und die Nachteile ausschließt, ist die auf einem visuellen Wörterbuch basierende Bildsuche. Ein visuelles Wörterbuch ist dabei eine Menge von Stichwort-zu-visueller-Beschreibung Beziehungen. Wir präsentieren ein Verfahren zur automatischen Erstellung eines visuellen Wörterbuchs aus einer Trainingsmenge von annotierten Bildern. Dabei werden verschiedene Modelle von visuellen Beschreibungen untersucht und anschließend evaluiert. Wir zeigen, dass eine kompakte visuelle Beschreibung existiert, die verglichen mit multiple-Instanzen visuellen Beschreibungen bessere Retrieval-Ergebnisse liefert und gleichzeitig die Anfragezeit drastisch senkt.

Schlüsselwörter

image search, visual dictionary, visual words, visual phrases

1. EINLEITUNG

Das heutige World Wide Web stellt einen großen und ständig wachsenden Datenbestand von Bildern dar und bildet somit eine gute Basis für die Suche nach gewünschten Bildern. Es gibt zwei klassische Arten der Bildsuche: die stichwortbasierte und die inhaltsbasierte Bildsuche. Die stichwortbasierte Bildsuche basiert auf Annotationen und Metadaten der Bilder. Die Anfrageformulierung erfolgt textuell, somit schnell und unkompliziert. Bei der Verarbeitung der Anfrage sucht das System nach Bildern, die, grob gesagt, die Stichwörter aus der Anfrage beinhalten. Einen Nachteil hat diese Suchart jedoch: der Erfolg der Suche hängt von der Qualität der Annotationen und Metadaten der Bilder ab. Je nachdem, ob Bilder manuell vom Benutzer oder automatisch mit Hilfe eines Algorithmus annotiert wurden, weisen diese unterschiedliche Schwächen auf, z.B. die Subjektivität des Beschreibenden, abstrakte Formulierungen oder falsche Stichwortzuordnungen, sowie Unvollständigkeit der Beschreibung. Aufgrund dieses Nachteils versucht man heutzutage, fern von den Annotationen, auf das Bild selbst einzugehen und somit den Inhalt des Bildes zu erschließen. Die inhaltsbasierte Bildsuche basiert demnach auf visuellen Eigenschaften des Bildes, z.B. bzgl. der Farbe, der Textur, Form usw. Eine Anfrage wird mittels einem Beispielbild gestellt, das Retrieval-System sucht dann nach Bildern, die dem Anfragebild ähnlich sind, bezogen auf den, dem System zugrundeliegenden Deskriptor und das Ähnlichkeitsmaß. Der Nachteil dieser Suchart betrifft die Anfrageformulierung mittels dem Anfragebild - ein Anfragebild liegt dem Benutzer in der Regel nicht vor, dieses wird schließlich gesucht. Gewünscht ist daher ein Retrieval-System, welches die Vorteile beider Sucharten integriert, d.h. eine textuelle Anfrageformulierung mit einer inhaltsbasierten Bildsuche kombiniert. Eine Lösung ist das Modell des visuellen Wörterbuchs als eine Menge von Stichwort-zu-visueller-Beschreibung Beziehungen. Bei der Bildsuche auf der Basis des visuellen Wörterbuchs wird nun eine Anfrage textuell gestellt, dann die Stichwörter aus der Anfrage in dem visuellen Wörterbuch nachgeschlagen und deren Übersetzung, d.h. eine visuelle Beschreibung des Stichwortes, für die anschließende inhaltsbasierte Bildsuche verwendet. Die Entwicklung eines Verfahrens zur automatischen Erstellung eines visuellen Wörterbuchs ist Gegenstand dieses Papers. Wir geben zunächst einen Überblick über verwandte Arbeiten, beschreiben dann das entwickelte Verfahren, evaluieren visuelle Beschreibungen und schließen mit einer Schlussfolgerung und einem Ausblick.

2. VERWANDTE ARBEITEN

In der Literatur existieren zwei weitverbreitete Definitionen des Begriffs *visuelles Wörterbuch*. Die erste Definition beschreibt das Konzept der Zuordnungen von Stichwort zu visueller Beschreibung, die zweite betrifft die Quantisierung des Deskriptor-Raums in Partitionen, sogenannte *visuelle Wörter*. Jeder Deskriptor wird dann mit seinem zugehörigen visuellen Wort repräsentiert. Alle Partitionen bilden das visuelle Wörterbuch. Oft werden beide Konzepte kombiniert [1, 4]. [1] verwendet eine gut vorbereitete Trainingsmenge, SCD und HTD (MPEG-7 Standard) Deskriptoren und beschreibt ein Stichwort mit einer konstanten Anzahl von visuellen Wörtern. [4] entwickelt ein visuelles Wörterbuch auf der Grundlage von SIFT-Deskriptoren und daraus abgelei-

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

teten visuellen Wörtern und stellt jedes Stichwort mit einer Gaußschen Mischverteilung dar. Das Konzept der visuellen Wörter wird mit der Idee der *visuellen Phrase* als ein Paar adjazenter visueller Wörter erweitert. Basierend auf SIFT wird in [6] das Modell der visuellen Phrase untersucht und dabei die Verbesserung des Retrievals nachgewiesen. Wir verwenden den Begriff des visuellen Wörterbuchs um die erste Definition auszudrücken. Falls das Konzept der zweiten Definition und ihre Erweiterung gemeint ist, sprechen wir von visuellen Wörtern und visuellen Phrasen.

3. DAS VERFAHREN ZUR ERSTELLUNG EINES VISUELLEN WÖRTERBUCHS

In diesem Kapitel präsentieren wir das entwickelte Verfahren zur automatischen Erstellung eines visuellen Wörterbuchs aus einer Trainingsmenge von annotierten Bildern. Das Verfahren basiert auf der Idee, die Trainingsbilder einmal bzgl. der Ähnlichkeit ihrer Annotationen und einmal bzgl. ihrer visuellen Ähnlichkeit zu gruppieren, dann die Trainingsbilder, die bzgl. der beiden Aspekte zueinander ähnlich sind, d.h. bzgl. beider Aspekte zusammen gruppiert wurden, aufzusuchen und aus diesen schließlich Korrelationen zwischen Stichwörtern und visuellen Bildmerkmalen abzuleiten.

3.1 Anforderungen an das visuelle Wörterbuch

Das visuelle Wörterbuch kann man sich wie ein herkömmliches Wörterbuch vorstellen, welches aus einer Menge von Einträgen besteht. In dem visuellen Wörterbuch sollen Objekte und visuelle Zusammenhänge, wie z.B. Tiere, Gegenstände, Gebäude, Logos, Symbole, etc. verwaltet werden. Jeder Eintrag ist ein Paar aus einem Stichwort, der das Objekt benennt und einer dazugehörigen visuellen Beschreibung des Objektes. Stichwörter sollen in der Grundform vorliegen - wir sprechen dann von Termen -, und es soll die Polysemie der Terme unterstützt werden. Eine visuelle Beschreibung stellt eine Einheit dar, die für die inhaltsbasierte Bildsuche verwendet wird. Diese soll nur die für dieses Objekt relevanten visuellen Charakteristika erfassen, die allen Perspektiven und Erscheinungsformen des Objektes gemeinsam sind. Zudem soll diese aus Effizienzgründen kompakt, sowie zu der Repräsentation der Bilder kompatibel sein.

3.2 Das konzeptuelle Modell des Verfahrens

Das konzeptuelle Modell des Verfahrens ist in Abbildung 1 dargestellt. Grundlage zum Erlernen des visuellen Wörterbuchs bildet die Trainingsmenge von annotierten Bildern, die beliebig und ohne zusätzliche Vorbearbeitung gewählt werden kann. Ausgehend von dieser werden zunächst einmal zwei Ziele verfolgt: die Gruppierung von ähnlichen Bildern auf der Basis der semantischen Ähnlichkeit ihrer Annotationen und die Gruppierung von ähnlichen Bildern bezüglich ihrer visuellen Ähnlichkeit. Dazu werden die Annotationen sowie die Bilder unabhängig voneinander in eine interne Repräsentation überführt und auf der Basis eines definierten Ähnlichkeitsmaßes gruppiert. Aus den beiden Gruppierungen wird dann das visuelle Wörterbuch erstellt. Dazu wird zunächst einmal das Vokabular für das visuelle Wörterbuch bestimmt. Für jeden Term des Vokabulars werden Trainingsbilder ermittelt, die diesen Term in der Annotation enthalten und bzgl. der Ähnlichkeit von Annotationen und der visuellen Bildmerkmale ähnlich sind. Es findet also ein Abgleich der Gruppierungen statt. Aus den ermittelten Trainingsbildern eines Terms wird schließlich die visuelle Beschreibung des Terms gelernt und zusammen mit dem Term als ein Eintrag in dem visuellen Wörterbuch abgespeichert. Wir erhalten das visuelle Wörterbuch aus Stichwort-zu-visueller-Beschreibung Einträgen.

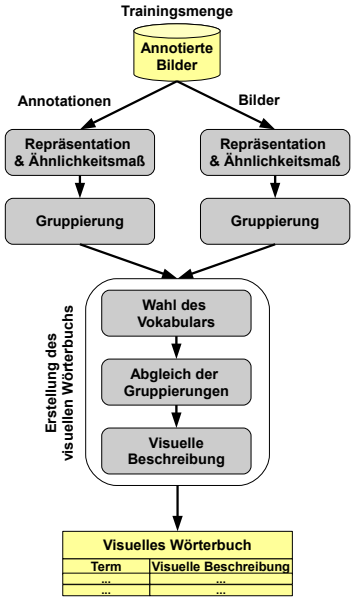


Abbildung 1: Das konzeptuelle Modell des Verfahrens

3.3 Repräsentation und Ähnlichkeitsmaß von/ für Annotationen und Bilder

Für einen semantischen Vergleich müssen Annotationen und Bilder in eine interne Darstellung überführt werden.

Wir bereiten auf und bereinigen zuerst die Annotationen, erstellen dann einen Index mit dem Indexvokabular und leiten daraus für jede Annotation einen Annotationsvektor gemäß der *tf-idf* Gewichtung. Als Ähnlichkeitsmaß wählen wir das Kosinusmaß.

Als Grundlage für die Repräsentation von Bildern wählen wir *Scale Invariant Feature Transform* (SIFT)[2], da es in der Literatur als eins der robustesten Features gilt. Eine auf rohen SIFT-Features basierende Bilddarstellung ist schwer zu handhaben und aus Gründen der Effizienz ungeeignet. Um alle Bilder einheitlich zu repräsentieren wenden wir daher die Technik der visuellen Wörter an. Mit dem Clusteringalgorithmus K-Means basierend auf der Euklidischen Distanz wird der 128-dimensionale Deskriptor-Raum der SIFT-Keypoints in 1000 Partitionen, die visuellen Wörter, zerlegt. Jedem Deskriptor wird gemäß dem Nächsten-Nachbar-Prinzip das entsprechende visuelle Wort zugeordnet. Ein Bild wird schließlich mit einem Histogramm der visuellen Wörter dargestellt, indem das i-te Bin die Vorkommenshäufigkeit des i-ten visuellen Wortes in dem Bild misst. Weiterhin verwenden wir auch das Konzept der visuellen Phrase für die Bilddarstellung. Eine visuelle Phrase vp_{ij} ist ein nichtgeordnetes Paar (Menge) von zwei visuellen Wörtern vw_i, vw_j. Für die Eigenschaft der räumlichen

Nähe übernehmen wir die in [5] definierte Bedingung. In einem Bild liegt eine visuelle Phrase vp_{ij} vor, falls in dem Bild zwei Keypoints kp_a, kp_b existieren und für diese folgendes gilt: das visuelle Wort von kp_a ist vw_i und von kp_b ist vw_j und die Euklidische Distanz *distanz* zwischen den (x, y) Positionen der Keypoints erfüllt die Bedingung:

$$distanz(kp_a, kp_b) < s_a \cdot \lambda \qquad \text{oder} \qquad distanz(kp_a, kp_b) < s_b \cdot \lambda \tag{1}$$

wobei s_a und s_b die Skalierung der Keypoints und λ ein Parameter ist, welcher das Auftreten der visuellen Wörter Paare kontrolliert. Den experimentellen Ergebnissen aus [5] folgend setzen wir $\lambda = 4$. Analog zu visuellen Wörtern erstellen wir auch für visuelle Phrasen ein Histogramm, welches das Vorkommen der 500.500 visuellen Phrasen in einem Bild zählt. In [6] wurde gezeigt, dass Retrieval-Systeme, die auf beiden Bilddarstellungen, der visuellen Wörter und der visuellen Phrasen, basieren, die besten Ergebnisse liefern. Wir folgen dieser Erkenntnis und repräsentieren jedes Trainingsbild mit zwei Histogrammen, der visuellen Wörter und der visuellen Phrasen:

$$b = \left(aHist^{VW}, aHist^{VP}\right) \tag{2}$$

Für die Bestimmung der Ähnlichkeit zweier Bilder verwenden wir ein Ähnlichkeitsmaß, das auf dem *Histogrammschnitt hs* zweier Histogramme basiert:

$$hs(nHist_i^X, nHist_j^X) = \sum_{l=1}^{k} min(nHist_i^X[l], nHist_j^X[l]) \tag{3}$$

wobei $nHist^X$ die normalisierte Version des absoluten Histogramms $aHist^X$ darstellt. Die Ähnlichkeit zweier Bilder b_i und b_j ergibt sich dann mit:

$$\begin{aligned} \ddot{a}hnlichkeit(b_i, b_j) = (1-\alpha) \cdot hs(nHist_i^{VW}, nHist_j^{VW}) \\ + \alpha \cdot hs(nHist_i^{VP}, nHist_j^{VP}) \end{aligned} \tag{4}$$

Für den Wert des Gewichts α orientieren wir uns an dem Paper [6], in welchem der Einfluß unterschiedlicher Gewichtswerte auf die Retrieval-Resultate untersucht wird. Es zeigt sich, dass das Optimum bei dem Wert $\alpha = 0.75$ liegt.

3.4 Gruppierung von Annotationen und von Bildern

Für die Gruppierung der Annotationen wenden wir den in [3] vorgeschlagenen Clusteringalgorithmus *Clustering by Committee* (CBC) an.

Die Gruppierung von ähnlichen Bildern bedeutet, Bilder, die dasselbe Objekt beinhalten, in eine Gruppe zu fassen. Da wir von nicht vorbearbeiteten Trainingsbildern ausgehen, liegen diese Bilder also in der Regel etwas „verschmutzt“ vor, d.h. sie beinhalten neben dem Hauptobjekt ggf. noch andere irrelevante Objekte oder einen Hintergrund. Dadurch kann es leicht zu dem Problem kommen, dass zwei Bilder, die wir intuitiv nicht gruppiert hätten, weil diese unterschiedliche Hauptobjekte haben, trotzdem einen höheren Ähnlichkeitswert haben, als zwei Bilder, die dem menschlichen Empfinden nach ähnlich sind. Bei der Wahl eines Gruppierungsverfahrens müssen wir diese Problematik einbeziehen. Clusteringverfahren, die die Trainingsbilder in Partitionen zerlegen, sind nicht geeignet, es könnte nämlich passieren, dass Bilder aufgrund für uns falsch erscheinenden Gemeinsamkeiten, wie dem Hintergrund, zusammengefasst und dann bzgl. des relevanten Objektes nicht mehr gruppiert werden. Am besten wäre, man hätte visuelle Beschreibungen von den, in der Trainingsmenge enthaltenen Objekten und würde diese als Clusterzentren nehmen, um die Trainingsbilder anhand dieser Clusterzentren überlappend zu gruppieren. Die visuellen Beschreibungen sind aber genau das was wir suchen. Hätten wir solche Beschreibungen, dann wäre die Gruppierung hier überflüssig. Man kann trotzdem versuchen solche visuellen Beschreibungen zu simulieren, indem man das was den Trainingsbildern gemeinsam ist, extrahiert. Sind zwei Bilder bzgl. einem Objekt ähnlich und teilen damit die Charakteristika des Objektes, dann müssen die gemeinsamen Charakteristika auch in der kompakten Bilddarstellung der visuellen Wörter und der visuellen Phrasen verankert sein, nämlich als Durchschnitt der visuellen Wörter und visuellen Phrasen Histogramme der beteiligten Bilder. Der Durchschnitt *ds* zweier Bilder b_i, b_j ist wie folgt definiert:

$$\begin{aligned} ds(b_i, b_j) &= \left(ds^{VW}(b_i, b_j), ds^{VP}(b_i, b_j)\right) \qquad \text{mit} \\ ds^X(b_i, b_j) &= (m(1), ..., m(k)) \qquad \text{und} \\ m(l) &= min(aHist_i^X[l], aHist_j^X[l]) \end{aligned} \tag{5}$$

wobei bei $X = VW$ ist $k = 1000$ und $X = VP$ ist $k = 500.500$. Für die Gruppierung der Trainingsbilder berechnen wir die Durchschnitte der ähnlichsten Bilder, betrachten diese als Pseudo-Objekte und damit als Centroide, und clustern die Trainingsbilder gemäß einem Schwellwert überlappend an diese Durchschnitte. Wir erhalten eine Menge von Gruppen visuell ähnlicher Bilder $\left\{G_l^{visuell} \mid 1 \leq l \leq n\right\}$.

3.5 Wahl des visuellen Wörterbuch Vokabulars

Als nächstes müssen wir klären, welche Stichwörter in das visuelle Wörterbuch aufgenommen werden. Als Stichwörter kommen natürlich nur Terme aus dem Indexvokabular in Frage. Die Übernahme aller Terme als Stichwörter ist jedoch nicht sinnvoll, denn nicht alle Terme bzw. die den Termen zugrundeliegenden Wörter beschreiben Objekte oder beinhalten einen visuellen Aspekt. Wir betrachten daher die Gruppen, die wir durch das Clustering von Annotationen erhalten haben. Wir nehmen an, dass innerhalb einer Annotationsgruppe die Terme, die in den meisten Annotationen vorkommen, etwas mit dem visuellen Inhalt der zugehörigen Bilder zu tun haben müssen. Für jede Annotationsgruppe werden daher diese hochfrequenten Terme bestimmt. Dazu wird zunächst der Term mit der höchsten Annotationshäufigkeit ermittelt und dann noch weitere, deren Annotationshäufigkeit größer ist als 0.7 mal die maximale Häufigkeit. Die Vereinigung der so erhaltenen Terme bildet dann das Vokabular des visuellen Wörterbuchs, also im Grunde die Einträge. Um die Forderung nach der Unterstützung der Polysemie von Termen zu realisieren, werden die betroffenen Terme mehrmals, nur mit unterschiedlichem Kontext, in dem visuellen Wörterbuch aufgeführt. Der jeweilige Kontext eines Terms ergibt sich aus der Gruppe, genauer aus den anderen Termen der Gruppe, zu der der Term gehört. Als Kontext wird der Centroid der Gruppe verwendet. Wir erhalten somit eine Seite des visuellen Wörterbuchs, nämlich eine Menge von Einträgen, die jeweils ein Objekt repräsentieren und aus einem Term und seinem Kontextvektor bestehen.

3.6 Abgleich der Gruppierungen

Für jeden Eintrag des visuellen Wörterbuchs muss nun eine Menge von Trainingsbildern bestimmt werden, aus der die visuelle Beschreibung des Terms gelernt werden soll. Das bedeutet, es müssen die Bilder bestimmt werden, die sowohl bzgl. des Terms als auch visuell bzgl. des beinhaltenden Objekts ähnlich sind. Dazu werden Bilder, die diesen Term in der Annotation enthalten, aus der Annotationsgruppe des Terms genommen und es wird daraus eine Gruppe $G^{eintrag}$ gebildet. Diese Gruppe wird dann mit jeder Gruppe $G_l^{visuell}$ visuell ähnlicher Bilder abgeglichen. Beim Abgleich wird der Mengendurchschnitt jeweils zweier Gruppen gebildet, indem die Bilder übernommen werden, die in der Gruppe $G^{eintrag}$ und in der Gruppe $G_l^{visuell}$ vorkommen. Der resultierende Mengendurchschnitt zweier Gruppen muss mindestens zwei Bilder beinhalten, sonst können keine gemeinsamen Charakteristika gelernt werden. Als Resultat des Abgleichs erhalten wir wiederum, ggf. überlappende, Gruppen von Bildern. Die Bilder innerhalb einer solchen Gruppe sind nun visuell als auch bzgl. des Terms und seinem Kontext ähnlich. Jeder Eintrag des visuellen Wörterbuchs besteht nun aus einem Term, seinem Kontextvektor und der Menge der Bildgruppen aus welcher eine visuelle Beschreibung im nächsten Schritt hergeleitet wird.

3.7 Visuelle Beschreibungen

Als nächstes muss die rechte Seite des visuellen Wörterbuchs, die Seite der visuellen Beschreibungen, bestimmt werden. Wir betrachten einen Eintrag, also einen Term, des visuellen Wörterbuchs und die ihm zugehörige, im letzten Abschnitt bestimmte Menge von Bildgruppen. Es gibt mehrere Möglichkeiten aus der Menge der Bildgruppen eine visuelle Beschreibung abzuleiten. Im Folgenden stellen wir einige Arten von visuellen Beschreibungen in der Reihenfolge der eigenen Entwicklung und Untersuchung vor.

3.7.1 Alle Bilder

Die erste und einfachste Methode eine visuelle Beschreibung anzugeben ist, die Bildgruppen zu vereinigen und die so erhaltene Menge an Trainingsbildern als Repräsentation des Terms zu verwenden. Bei der Bildsuche zu diesem Term finden dann mehrere inhaltsbasierte Bildsuchen statt, indem jedes dieser Trainingsbilder als Anfragebild verwendet wird. Bei dieser multiple-Instanzen visuellen Beschreibung erhalten wir jedoch zunächst für jedes Anfragebild ein Ranking von Bildern als Ergebnis. Es stellt sich also die Frage, wie das Endergebnis aus den Ergebnissen der einzelnen Anfragen berechnet werden soll. Für die Angabe des Endergebnisses werden drei Strategien untersucht.

Bei der ersten Strategie wird das beste Resultat als Endergebnis ausgegeben. Dazu wird die Güte der einzelnen Ergebnisse mittels einem Qualitätsmaß berechnet. Eine solche Berechnung erfordert allerdings zu wissen, welche Bilder des Ergebnisrankings für den Anfrageterm relevant und welche irrelevant sind. Dafür müssten die Bilder in der Bilddatenbank kategorisiert oder mit Termen versehen sein. Von diesem Fall kann man in der Realität jedoch nicht ausgehen. Diese Strategie ist auf einer Bilddatenbank also praktisch nicht anwendbar, lediglich auf einer vorbereiteten Testmenge. Aus Gründen des Performance Vergleichs wird diese trotzdem aufgeführt und untersucht.
(*AlleBilder-BesterScore*)

Jedes Bild aus der Bilddatenbank hat für jedes Anfragebild der visuellen Beschreibung, also in jedem der einzelnen Ergebnisse, eine Rankingposition und einen Ähnlichkeitswert zum Anfragebild. Bei der zweiten Strategie wird für jedes Bild aus der Bilddatenbank der maximale Ähnlichkeitswert aus seinen Ähnlichkeitswerten zu allen Anfragebildern ausgewählt, die Bilder dann entsprechend ihrem maximalen Ähnlichkeitswert sortiert und als Endergebnis ausgegeben (*AlleBilder-MaxÄhnlichkeit*).

Eine dritte Lösung zur Bestimmung des Endergebnisses ist, für jedes Bild aus der Bilddatenbank das arithmetische Mittel ihrer Rankingpositionen aus den einzelnen Ranking-Ergebnissen zu berechnen, dann die Bilder bezüglich diesem arithmetischen Mittel aufsteigend zu sortieren und dieses Ranking als Endergebnis auszugeben.
(*AlleBilder-DurchschnittsRank*)

3.7.2 Durchschnitte

Bei der letzten visuellen Beschreibung werden nicht wirklich Charakteristika des Objektes gelernt, diese stellt also keine visuelle Beschreibung in unserem gewünschten Sinne dar. Wir gehen davon aus, dass die Ähnlichkeit zweier ähnlicher Bilder auf einer gemeinsamen Teilmenge der visuellen Wörter und visuellen Phrasen basiert. Wir extrahieren daher die Gemeinsamkeiten zweier ähnlicher Bilder, indem wir den Durchschnitt ihrer Histogramme gemäß der Formel 5 bilden. Für jede Bildgruppe aus der Menge der Bildgruppen werden paarweise Durchschnitte der Trainingsbilder aus der Bildgruppe berechnet. Die visuelle Beschreibung besteht dann aus allen gebildeten Durchschnitten, d.h. jeder Durchschnitt dient bei der inhaltsbasierten Bildsuche als ein Anfragebild und es finden mehrere Anfragen statt.

Wie bei der ersten visuellen Beschreibung, erhalten wir auch hier eine Menge von einzelnen Ergebnissen und müssen diese zu einem Endergebnis berechnen. Wir wenden dazu die drei beschriebenen Strategien an (*Durchschnitte-BesterScore*, *Durchschnitte-MaxÄhnlichkeit*, *Durchschnitte-DurchschnittsRank*).

3.7.3 Bestes Bild

Die bisher vorgestellten visuellen Beschreibungen sind problematisch: sie bestehen aus mehreren Anfrageinstanzen und weisen daher eine zeitaufwändige Anfrageverarbeitung auf. Eine kompakte Darstellung der visuellen Beschreibung, d.h. eine Darstellung, die aus nur einer Anfrageinstanz besteht, wäre von Vorteil. Eine einfache Lösung wiederum ist, das beste Trainingsbild aus den Trainingsbildern eines Eintrags als visuelle Beschreibung zu wählen. Um das beste Trainingsbild zu bestimmen, vereinigen wir die Bildgruppen und stellen mit jedem Bild aus der Vereinigung eine Anfrage an die ganze Trainingsmenge. Mit einem Qualitätsmaß wird jedes Anfrageergebnis bewertet und das Anfragebild mit der besten Güte, d.h. mit dem höchsten Score des Ergebnisses für die visuelle Beschreibung übernommen (*BestesBild*). Das gewählte Trainingsbild kann jedoch ein lokales Optimum darstellen und in der Suche auf der Bilddatenbank versagen. Weiterhin zeigt sich auch hier das Problem, dass keine Charakteristika von Objekten aus den ähnlichen Trainingsbildern gelernt werden.

3.7.4 Durchschnitte kompakt - Anzahl

Um eine kompakte Darstellung der visuellen Beschreibung zu erhalten, die die gemeinsamen Charakteristika des Objektes ausdrückt, kommen wir auf das Konzept der Durch-

schnitte zurück. Wie in der zweiten visuellen Beschreibung beschrieben, bilden wir zunächst Durchschnitte der paarweisen Trainingsbilder pro jede Bildgruppe. Wir nehmen an, dass visuelle Wörter und visuelle Phrasen, die in den meisten Durchschnitten auftreten, für das Objekt relevanter sind, als die, die seltener vorkommen. Wir erstellen daher eine visuelle Beschreibung aus zwei Histogrammen, der visuellen Wörter und der visuellen Phrasen, und zählen für jedes visuelle Wort und jede visuelle Phrase, in wievielen Durchschnitten es vorkommt. Diese absolute Durchschnitts-Frequenz bildet dann den Wert des jeweiligen visuellen Wortes oder der visuellen Phrase in den Histogrammen (*DurchschnitteKompakt-Anzahl*).

3.7.5 Durchschnitte kompakt - Summe

Um die Wichtigkeit jedes visuellen Wortes und jeder visuellen Phrase innerhalb eines Durchschnitts zu betonen, wird anstatt der Anzahl der Durchschnitte eine Summe der Durchschnitte gebildet. Genaugenommen werden wieder zwei Histogramme der visuellen Wörter und visuellen Phrasen erstellt und jedes Bin des Histogramms ist die Summe der entsprechenden Bins der Histogramme aller Durchschnitte. (*DurchschnitteKompakt-Summe*)

3.7.6 Durchschnitte kompakt - Gewichtete Summe

Der nächsten visuellen Beschreibung liegt die folgende Frage zugrunde: gibt es visuelle Phrasen, die für ein Objekt spezifisch sind, d.h. ist der Anteil der Bilder zu einem Term und einer visuellen Phrase an allen Bildern, die diese visuelle Phrase beinhalten, besonders hoch? Wir berechnen für jede visuelle Phrase vp und dem zugrundeliegenden Term t des Eintrags das Gewicht:

$$g^{VP}(t, vp) := \frac{\#B(t, vp)}{\#B(vp)} \tag{6}$$

mit $B(t, vp)$ stellt die Menge aller Trainingsbilder zu dem Term t, d.h. die Vereinigung der Bilder aus den Bildgruppen zu t, die die visuelle Phrase vp beinhalten, dar. Wir übernehmen die zuvor definierte visuelle Beschreibung *Durchschnitte-Kompakt-Summe* und gewichten den Häufigkeitswert jeder visuellen Phrase vp mit $g^{VP}(t, vp)$. (*DurchschnitteKompakt-GewichteteSumme*)

3.7.7 Durchschnitte kompakt - TFIDF

Für die folgende visuelle Beschreibung übernehmen wir die Idee der tf-idf Gewichtung für Dokumentvektoren. Mit Hilfe der inversen Dokumenthäufigkeit eines Terms, hier inverse Bildhäufigkeit eines visuellen Wortes oder einer visuellen Phrase, wollen wir die Häufigkeiten der visuellen Wörter und visuellen Phrasen, die in sehr vielen Trainingsbildern vorkommen, schwächer, und die die seltener vorkommen, stärker gewichten. Analog zum Text-Retrieval bilden wir also eine Summe aller Durchschnitte, wie in der visuellen Beschreibung *DurchschnitteKompakt-Summe* beschrieben, berechnen dann für jedes visuelle Wort vw und jede visuelle Phrase vp das idf Gewicht:

$$g^{IDF}(vw) := log\frac{\#B}{\#B(vw)} \tag{7}$$

wobei B ist die Menge aller Trainingsbilder und $B(vw)$ die Menge der Trainingsbilder, die das visuelle Wort vw beinhalten. Analog für vp. Die aus der Summe der Durchschnitte entstandenen Histogramme werden dann mit diesen Gewichten multipliziert: jedes Bin zu einem visuellem Wort vw mit $g^{IDF}(vw)$ und jedes Bin zu einer visuelle Phrase vp mit $g^{IDF}(vp)$. (*DurchschnitteKompakt-TFIDF*)

3.7.8 Durchschnitte kompakt - Gewichtetes TFIDF

Die Gewichte aus den beiden letzten visuellen Beschreibungen werden im Folgenden kombiniert. Wir erstellen wieder die Summe aller Durchschnitte und gewichten dann jeden Häufigkeitswert des jeweiligen visuellen Wortes vw mit $g^{IDF}(vw)$, und jeden Häufigkeitswert einer visuellen Phrase vp mit dem kombinierten Gewicht:

$$g^{VP-IDF}(t, vp) := \frac{\#B(t, vp)}{\#B(vp)} \cdot log\frac{\#B}{\#B(vp)} \tag{8}$$

(*DurchschnitteKompakt-GewichtetesTFIDF*)

4. EVALUATION

4.1 Trainings- und Testmenge

Für die Test- und Trainingsmenge werden Bilder und Annotationen zu 50 Objekten aus dem World Wide Web gesammelt. Für jedes der 50 Terme werden jeweils 10 Trainingsbilder und ca. 30 Testbilder heruntergeladen. Als Objekte werden Tiere, Früchte, Gegenstände, Gebäude und Symbole gewählt. Fast alle Bilder liegen in einer Auflösung von ca. 400×400 Pixel vor.

4.2 Testdurchführung

Die vorgestellten visuellen Beschreibungen werden hinsichtlich der Qualität des Retrievals und der Anfrageeffizienz analysiert, um so aus den daraus gewonnenen Ergebnissen und Erkenntnissen die beste für das visuelle Wörterbuch auswählen zu können. Dazu wird für jede visuelle Beschreibung zuerst ein visuelles Wörterbuch aus der Trainingsmenge gelernt und dieses dann in der Anwendung der Bildsuche eingelesen. Für jeden Eintrag des visuellen Wörterbuchs, also jeden Term (im jeweiligen Kontext), wird eine Anfrage auf der Testmenge durchgeführt, dabei die Anfragezeit gemessen und schließlich aus dem erhaltenen Ranking-Ergebnis die Güte des Ergebnisses mit dem Maß *Score*, der im Folgenden erläutert wird, berechnet. Um die visuellen Beschreibungen letztlich miteinander vergleichen zu können, wird für jede visuelle Beschreibung, also jedes Wörterbuch, das arithmetische Mittel der Anfragezeiten und der Scores über allen Einträgen gebildet.

4.3 Bewertungsmaß

In [6] wird für die Evaluation des Retrieval-Systems ein Maß *Score* benutzt. Score bewertet die Top-20 zurückgegebenen Bilder, indem jedes relevante Bild entsprechend des Intervalls, in dem seine Rankingposition liegt, gewichtet wird, die Gewichte aller relevanten Bilder summiert und schließlich auf den Bereich $[0, 1]$ normalisiert werden. Die Autoren des Papers begründen, dass die meisten Benutzer nur die ersten beiden Ergebnisseiten, mit jeweils 10 Bildern pro Seite, betrachten und daher nur die Top-20 der zurückgegebenen Bilder zu einer Anfrage die relevantesten für den Benutzer sind. Wir stimmen mit der Argumentation überein und übernehmen dieses Maß für die Qualitätsbewertung der visuellen Beschreibungen.

4.4 Testergebnisse

Als Testergebnis erhalten wir die zwei Diagramme in Abbildung 2. Das obere Diagramm stellt den durchschnittlichen Score und das untere die durchschnittliche Anfragezeit für jede visuelle Beschreibung dar. Die besten durchschnittlichen Scores erreichen die visuellen Beschreibungen *AlleBilder-BesterScore*, *Durchschnitte-BesterScore*, die aus multiplen Instanzen und der Endergebnis-Strategie *BesterScore* bestehen. Dabei sieht man, dass die auf Durchschnitten basierende visuelle Beschreibung ein besseres Retrieval-Ergebnis liefert, die durchschnittliche Anfragezeit sich gleichzeitig aber verdoppelt. Wie bereits erwähnt ist diese Endergebnis-Strategie nur ein theoretisches Modell. Die zwei praktisch realisierbaren Endergebnis-Strategien verhalten sich je nach visueller Beschreibung unterschiedlich: *MaxÄhnlichkeit* schneidet bei *AlleBilder* besser und bei *Durchschnitte* schlechter ab als *DurchschnittsRank*. Diese multiple-Instanzen visuellen Beschreibungen mit den Strategien *MaxÄhnlichkeit* und *DurchschnittsRank* werden jedoch von den eine-Instanz, auf Durchschnitten basierenden visuellen Beschreibungen bzgl. dem durchschnittlichen Score deutlich übertroffen. Von den besten multiple-Instanzen visuellen Beschreibung *AlleBilder-MaxÄhnlichkeit*, *Durchschnitte-DurchschnittsRank* zu den besten eine-Instanz, *DurchschnitteKompakt-GewichteteSumme* und *DurchschnitteKompakt-TFIDF* haben wir einen Zuwachs des durchschnittlichen Scores von 0.07 und die Anfragezeit sinkt dabei drastisch um das 9- bzw. 17-fache. Die eine-Instanz, auf Durchschnitten basierenden visuellen Beschreibungen weisen einen deutlich besseren, um ca. 0.12 höheren, durchschnittlichen Score gegenüber *BestesBild* auf, sind untereinander mit Unterschieden von bis 0.02 aber relativ ähnlich. Die besten unter ihnen, *DurchschnitteKompakt-GewichteteSumme* und *DurchschnitteKompakt-TFIDF* liefern zudem den besten durchschnittlichen Score unter allen praktisch realisierbaren visuellen Beschreibungen. *DurchschnitteKompakt-TFIDF* maximiert den durchschnittlichen Score und minimiert gleichzeitig die Anfragezeit, ist daher am besten für das visuelle Wörterbuch geeignet.

5. SCHLUSSFOLGERUNG UND AUSBLICK

Von den gestellten Anforderungen an das visuelle Wörterbuch werden die Grundform der Terme mit dem Stemming-Schritt in der Aufbereitungsphase und die Polysemie mit dem CBC Clustering, dem Kontextvektor und den damit verbundenen Mehreinträgen eines Terms, realisiert. Mit der erwähnten besten visuellen Beschreibung ist das Erfassen der Charakteristika des Objektes mit dem Konzept der Durchschnitte, die Kompaktheit und Effizienz mit der eine-Instanz Darstellung und die Kompatibilität zu der Bilddatenbank mit den Histogrammen der visuellen Wörter und Phrasen erfüllt. Zukünftig, um die Anfragezeiten der eine-Instanz visuellen Beschreibungen von ca. 12 Sekunden weiter zu reduzieren, kann man geeignete effiziente Indexstrukturen und Algorithmen für die Bildsuche untersuchen und einsetzen. Um die Qualität des Retrievals weiter zu verbessern, könnte man versuchen auch Farbeigenschaften und ihre Relevanz für Objekte miteinzubeziehen, d.h. diese für die visuelle Beschreibung zu lernen und in der Bildsuche einzusetzen.

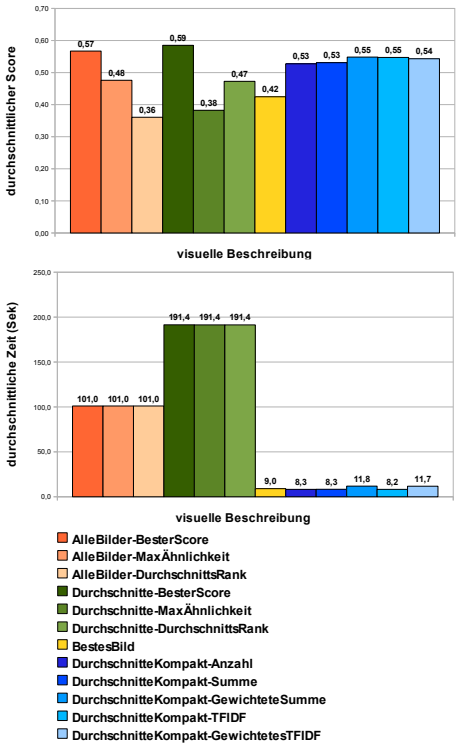


Abbildung 2: Durchschnittlicher Score und durchschnittliche Anfragezeit der visuellen Beschreibungen

6. LITERATUR

[1] C. Hentschel, S. Stober, A. Nürnberger, and M. Detyniecki. Adaptive multimedial retrieval: Retrieval, user, and semantics. chapter Automatic Image Annotation Using a Visual Dictionary Based on Reliable Image Segmentation, pages 45–56. Springer-Verlag, Berlin, Heidelberg, 2008.

[2] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60:91–110, November 2004.

[3] P. A. Pantel. *Clustering by Committee*. PhD thesis, University of Alberta, 2003.

[4] M. Wang, K. Yang, X.-S. Hua, and H.-J. Zhang. Visual tag dictionary: interpreting tags with visual words. In *Proceedings of the 1st workshop on Web-scale multimedia corpus*, WSMC '09, pages 1–8, New York, NY, USA, 2009. ACM.

[5] S. Zhang, Q. Tian, G. Hua, Q. Huang, and S. Li. Descriptive visual words and visual phrases for image applications. In *Proceedings of the seventeen ACM international conference on Multimedia*, MM '09, pages 75–84, New York, NY, USA, 2009. ACM.

[6] Q.-F. Zheng and W. Gao. Constructing visual phrases for effective and efficient object-based image retrieval. *ACM Trans. Multimedia Comput. Commun. Appl.*, 5:7:1–7:19, October 2008.

A feedback guided interface for elastic computing

Sebastian Schönherr[1,2,*], Lukas Forer[1,2,*], Hansi Weißensteiner[1,2], Florian Kronenberg[2], Günther Specht[1], Anita Kloss-Brandstätter[2]

[*] contributed equally

[1]Databases and Information Systems
Institute of Computer Science
University of Innsbruck, Austria
sebastian.schoenherr@uibk.ac.at

[2]Division of Genetic Epidemiology
Department of Medical Genetics, Molecular and Clinical Pharmacology
Innsbruck Medical University, Austria
lukas.forer@i-med.ac.at

ABSTRACT

Computer Science plays an important role in today's Genetics. New sequencing methods produce an enormous amount of data, pushing genetic laboratories to storage and computational limits. New approaches are needed to eliminate these shortcomings and provide possibilities to reproduce current solutions and algorithms in the area of Bioinformatics. In this paper a system is proposed which simplifies the access to computational resources and associated computational models of cluster architectures, assists end users in executing and monitoring developed algorithms via a web interface and provides an interface to add future developments or any kind of programs. We demonstrate on existing algorithms how an integretation can be done with little effort, making it especially useful for the evaluation and simplified usage of current algorithms.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Distributed System, Experimentation, Application

Keywords

Bioinformatics, Hadoop, MapReduce, Cloud computing

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

1. INTRODUCTION

In recent years Computer Science became an essential part in the field of Genetics. Especially through the advent of Next Generation Sequencing (NGS), whereby a human genome (3 billion base pairs/chromosome set) can be sequenced in acceptable time, the amount of data is growing significantly, exceeding all known dimensions in Genetics. Figure 1 shows a comparison between the reducing DNA sequencing costs and Moore's law. Moore's law is used as a reference to show that computer hardware can currently not keeping pace with the progress in DNA sequencing. Furthermore, the amount of complete sequenced individuals is growing exponentially from year to year [11], making new models necessary. For instance, to store the data of *one* complete human DNA (Deoxyribonucleic acid) in raw format with 30-times coverage, 30 terabytes of data is produced.

In the area of *Copy Number Variations*, a possible cause for many complex genetic disorders, high throughput algorithms are needed to process and analyze several hundred gigabytes of raw input data [16] [6], yielding to a wall time of up to one week for a typical study size [18]. This remarkable increase of data and time causes genetic departments to consider new ways of importing and storing data as well as improving performance of current algorithms.

Cluster architectures in connection with associated models have the potential to solve this issue, but especially for small departments often gainless and unaffordable. Using clusters on demand, also referred to *Infrastructure as a Service* (IaaS), builds therefore a good opportunity to circle these issues. To capitalize the full potential of IaaS, a combination with distribution models like MapReduce [5] is for specific applications both possible and obvious. Several isolated applications [9], [10], [14] already exist using a distributed approach for storing data and processing algorithms. But since no general system is given to execute those solutions, an evaluation and reproducibility is often not feasible. Scientists need to setup a cluster on their own or using a provided remote cluster architecture to evaluate a published algorithm, being both time wasting and insecure for sensitive data.

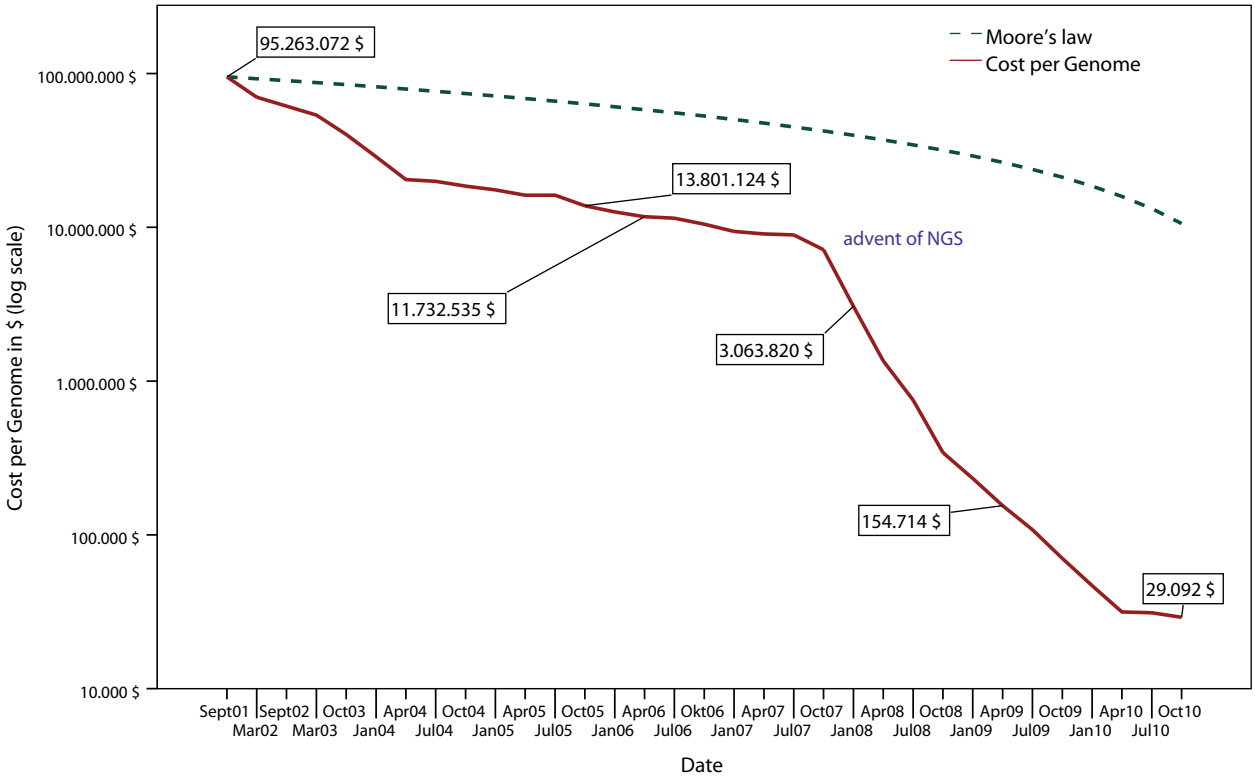


Figure 1: Comparision of DNA sequencing cost with Moore's law; Data from [17]

In this paper we present the idea to build an integrated system for scientists in the area of Bioinformatics to (1) get access to distributed cluster architectures and execute existing algorithms, (2) build maintainable and reproducible workflows and (3) provide an interface to add future developments or any kind of programs to the system without detailed IT knowledge. The reminder of this paper is structured as follows: Section 2 gives an overview of the related work. In section 3 the architecture of our suggested system is explained in more detail with potential case studies in section 4. Section 5 shows necessary future work and the paper ends with a conclusion in section 6.

2. RELATED WORK

Cluster solutions guided by a web-interface to execute distributed algorithms like Myrna [9], CrossBow [10] or CloudBurst [13] already exist. Unfortunately, the user must login to the Amazon Web Services (AWS) console to monitor the progress of executed jobs or to shutdown the cluster after execution. Additionaly, a data storage in S3 buckets is often required and a custom web interface needs to be implemented for every single approach.
Galaxy [7] is a software system which facilitates the creation, execution and maintainability of pipelines in a fast and user friendly way. The platform itself executes the scripts and the user has the possibility to monitor the progress. Galaxy's extension CloudMan [1] provides the possibility to install and execute Galaxy on Amazon EC2 (Elastic Compute Cloud). However, the user needs to start the master node manually by using the AWS console and Galaxy does not provide a native support of Hadoop programs, executes modules step by step and distributes only whole jobs among the cluster.

3. ARCHITECTURE

A modular architecture is suggested in Figure 2, separating the process of instantiate and set up a cluster (*Cloudgene*) from the process of monitor and run a program (*EMI*). Based on open source frameworks like Apache Hadoop [2] and Apache Whirr [3], we implemented a prototype to verify our approach. The user utilizes Cloudgene to set up a cluster architecture to his needs through XML configuration files. This allows adding new algorithms dynamically without digging into Cloudgene to deep. A fully operable and customized cluster is then provided, including all necessary user data. In a subsequent step EMI (Elastic MapReduce Interface) is launched on the master node of the cluster. EMI can be seen as an abstraction of the underlying system architecture from the end user, lies on top of the integrated programs and allows the user to communicate and interact with the cluster as well as receive feedback of currently executed workflows (see Figure 3). EMI can be disabled in case a program already includes an interface by its own, yielding to the most general approach to execute any kind of developed solution. Both parts can be operated separately via configuration files with clear defined input and output variables.

3.1 Cloudgene

Amazon provides with its EC2 the currently most developed service for public clouds in the area of IaaS. Cloudgene supports besides EC2 also Rackspace [12] to provide access to cluster infrastructure. As mentioned in the introduction

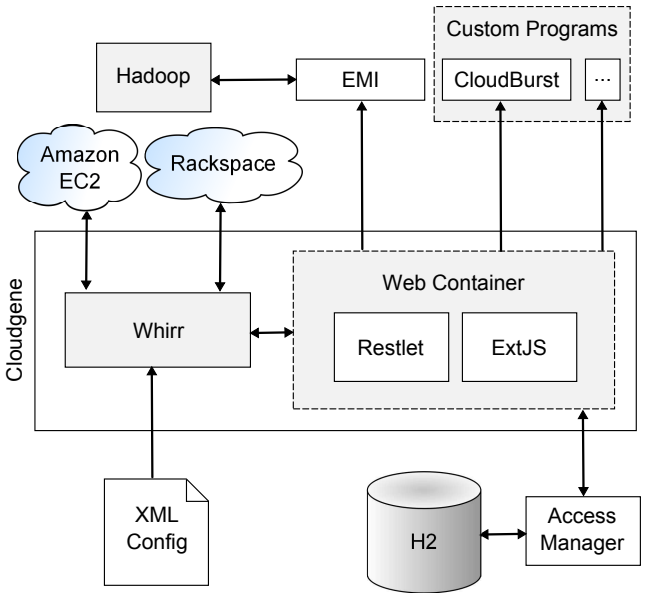


Figure 2: Architecture of the suggested system including Cloudgene and EMI

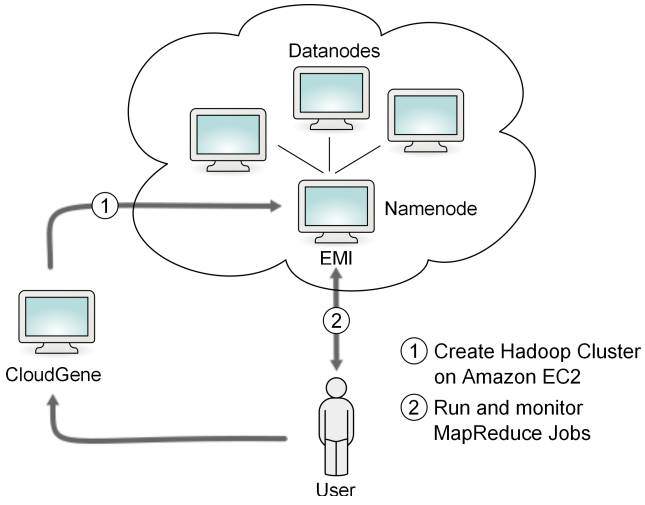


Figure 3: Workflow of the system including Cloudgene and EMI

a combination with MapReduce is useful: In this paradigm, the master node chops up data into chunks and distributes it over all active worker nodes (*map step*). Subsequently, the master node reassigns coherent map results to worker nodes (*sort and shuffle*) to calculate the final result (*reduce step*). For this project Apache Hadoop's implementation of MapReduce and its distributed file system (HDFS) are used. Using Whirr as a connector, Cloudgene is able to instance a full working EC2 or Rackspace cluster for end users with various defined properties and copies the necessary program data and configuration files to the cluster. Examples for defined variables could be the desired image, amount and kind of instances, HDFS options, MapReduce properties and the user's SSH public key. Amazon already provides several predefined images for all sorts of use cases, which can be be used with Cloudgene (e.g. http://www.cloudbiolinux.com). Cloudgene takes over the customization of predefined images and installs services like MapReduce, in our case included in Cloudera's distribution of Apache Hadoop [4]. The cluster configuration is defined in an XML-based file format, including all necessary information for a successful cluster boot. Cloudgene routinely checks if new configurations are added and offers the possibility to execute newly defined programs. Since EC2 is using a pay-per-use model, end users must provide their Amazon Access ID and Secret Key, which is transferred via Cloudgene to Amazon in a secure way. Alternatively, Cloudgene can also be launched on every machine having Java installed, eliminating the transfer via our server. Cloudgene solves one important issue and gives genetic departments access to computational power and storage. A still unresolved problem is the lack of a graphical user interface to control jobs deriving from command line based applications. Especially the need of putting enormous amount of local data into HDFS has to be considered. To overcome these shortcomings, a user interface (EMI) was designed.

3.2 Efficient MapReduce Interface (EMI)

Running Hadoop MapReduce programs on a cluster requires the execution of several non-trivial steps: First, the user must upload all input data to the master node, copy the data into the proprietary HDFS, run the Hadoop MapReduce job, export the results from the filesystem and finally download them to the local workstation. For researchers without expertise in Computer Science these tasks turns out to be very challenging. For this purpose we developed EMI which facilitates the execution, monitoring and evaluation of MapReduce jobs. A web interface, which runs on the master node of the cluster, enables the execution of jobs through well-structured wizards and setting all required parameters step by step. As several studies have shown, reproducibility of data analysis is one of the greatest problems in biomedical publications [15]. For this purpose the execution of a MapReduce job with its parameters and input data is logged, thus a fast comparison of experiments with different settings is possible. Moreover, the user always has the full control over an execution of each job and can monitor its current progress and status. All running jobs are listed whereby the progress of the map and reduce phase are displayed separately. Since using resources from Amazon costs money, EMI informs the user about the uptime of the cluster and the number of rented instances (Figure 4).
The modular architecture enables a fast integration of any Hadoop job which could be normally executed through the

command line. A simple and clear XML configuration file describes the input and output parameters of the program and contains other relevant information that are necessary to start the job (see Section 4). In addition to this file, a zip archive file exists which contains all software relevant data (e.g. jar file, meta data, configuration files). With those files, EMI automatically generates a web interface in which the possibility to set each defined parameter through wizards and to run the defined job by a single click is provided. As mentioned earlier, all input data must be put into the robust and fault-tolerant HDFS. As this process is very time-intensive an error prone, EMI supports the user by providing a wizard which enables the import of data from different sources (FTP, HTTP, Amazon S3 buckets or local file uploads). In addition, files defined as output parameters can be exported and downloaded as a zip archive or can be uploaded to Amazon S3 or FTP servers. EMI supports a multi-user mode whereby all data by a certain user are password protected and executed jobs are scheduled through a queue system. Overall, EMI is fully independent from Cloudgene and can be installed on a local Hadoop cluster too.

4. CASE STUDIES

In this section we explain how new programs can be integrated into Cloudgene and EMI. Based on two different biomedical software solutions we demonstrate the diversity and simplicity of our approach.

4.1 CloudBurst

CloudBurst is a parallel read-mapping algorithm to map NGS data to the human genome and other reference genomes [13]. It is implemented as a MapReduce program using Hadoop and can be executed with the following command:

```
hadoop jar emi/cloudburst/CloudBurst.jar \
  reference_genome reads results 36 36 3 0 1 240 \
  48 24 24 128 16
```

In order to execute CloudBurst we create a configuration file for Cloudgene which starts a Hadoop cluster on Amazon EC2 with a standard Ubuntu Linux with open Hadoop ports 50030 and 50070. The corresponding XML has the following structure:

```
<cloudgene>
  <name>CloudBurst</name>
  <options>
    <option name="provider" value="amazon-aws"/>
    <option name="image" value="default"/>
    <option name="service" value="hadoop"/>
    <option name="emi" value="true"/>
    <option name="ports" value="50030 50070"/>
  </options>
</cloudgene>
```

As CloudBurst has no graphical user interface, we install EMI on the Amazon EC2 cluster and use it for user interactions. For this purpose the command above with its arguments must be translated into the following configuration file:

```
<emi>
  <program>
    <name>CloudBurst</name>
    <command>
       hadoop jar emi/cloudburst/CloudBurst.jar \
          $input1 $input2 $output1 36 36 3 0 1 240 \
          48 24 24 128 16
    </command>
    <input>
      <param id="1" type="hdfs">
        <name>Reference Genome</name>
        <default>data/cloudburst/s_suis.br</default>
      </param>
    </input>
    <input>
      <param id="2" type="hdfs">
        <name>Reads</name>
        <default>data/cloudburst/100k.br</default>
      </param>
    </input>
    <output>
      <param id="1" type="hdfs" merge="true">
        <name>Results</name>
        <default>data/cloudburst/results</default>
      </param>
    </output>
  </program>
</emi>
```

After the XML file is uploaded to the Cloudgene server, the user starts a web browser to (1) login to Cloudgene, (2) start up a cluster preconfigured with CloudBurst and (3) run and monitor jobs with EMI (Figure 4).

Compared to a standard manual approach, this eliminates error-prone and time-consuming tasks such as (1) setting up a cluster and connecting via the command line onto the master node, (2) uploading and importing data into HDFS, (3) exporting final results from HDFS and downloading them and (4) executing and reproducing MapReduce jobs with different configurations via a web interface. This shows, that an easy integration can be done using a simple XML configuration, supporting and guiding researchers as far as possible.

4.2 HaploGrep

HaploGrep is a reliable algorithm implemented in a web application to determine the haplogroup affiliation of thousands of mitochondrial DNA (mtDNA) profiles genotyped for the entire mtDNA or any part of it [8]. As HaploGrep provides its own web interface we do not need to install EMI. Since it does not use the Hadoop service either, we note this option in the configuration as well. HaploGrep listens on the ports 80 (http) and 443 (https), therefore this ports are marked as open. The configuration file for Cloudgene with all requirements looks as follows:

```
<cloudgene>
  <name>Haplogrep</name>
  <options>
    <option name="provider" value="amazon-aws"/>
    <option name="image" value="default"/>
    <option name="service" value="none"/>
    <option name="emi" value="false"/>
    <option name="ports" value="80 443"/>
```

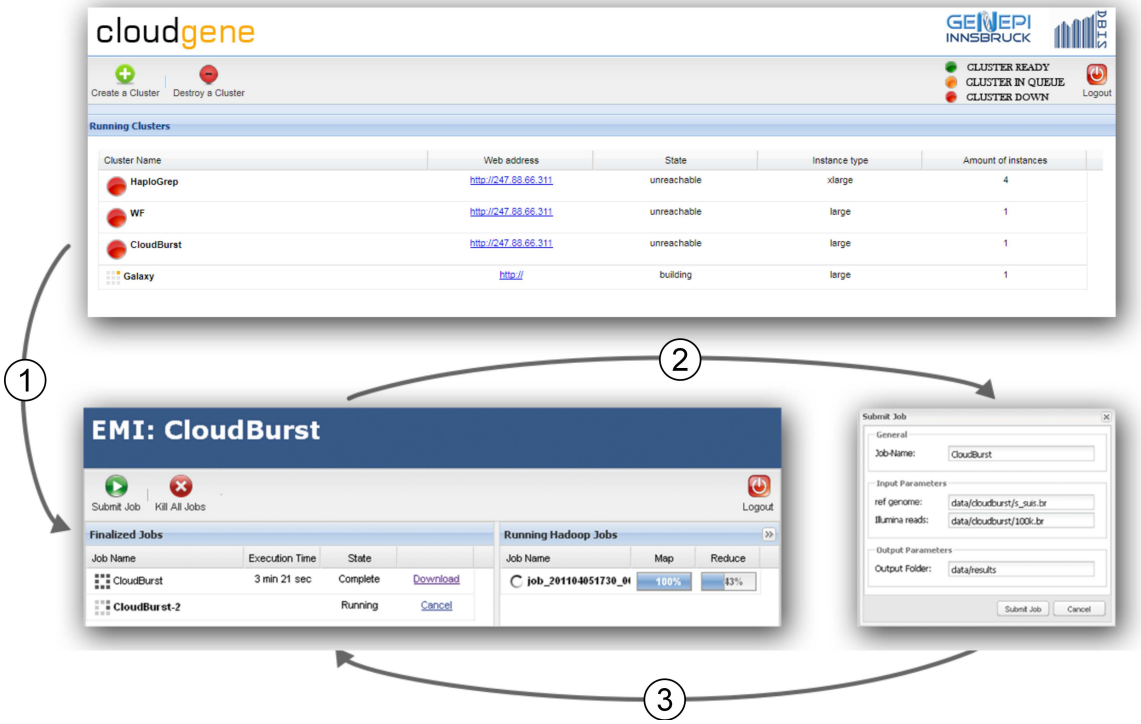


Figure 4: Workflow based on CloudBurst

```
  </options>
</cloudgene>
```

After the cluster setup is finalized, Cloudgene returns a web address which points to the installed instance of HaploGrep.

5. FUTURE WORK

One of the biggest advantages of IaaS is the changable amount of needed datanodes on demand. Thus, the next version of Cloudgene is conceived to provide functions for adding and removing instances during runtime. Currently, clusters started with Cloudgene are not data persistent which yields to a data loss after a shutdown is fulfilled. For this purpose we plan to store all results on persistent Amazon EBS volumes. Furthermore, a simple user interface for Hadoop is not only useful for the end user but also for developers. It supports them during the whole prototyping and testing process of novel MapReduce algorithms by highlighting performance bottlenecks. Thus, we plan to implement time measurements of the map, reduce and shuffle phase and to visualize them in an intuitive chart. Additionally, Hadoop plans in its next generation approach to support alternate programming paradigms to MapReduce, what is particularly important for applications (e.g. K-Means) where custom frameworks out-perform MapReduce by an order of magnitude.

6. CONCLUSION

We presented a software system for running and maintaining elastic computer clusters. Our approach combines the individual steps of setting up a cluster into a user-friendly system. Its modular architecture enables a fast integration of any Hadoop job which could be only executed through the command line. By hiding the low-level informatics, it is the ideal system for researchers without deeper knowledge in Computer Science. Moreover, our system is not constricted to the life sciences and can be used in nearly every application range. Overall, it is a first approach in order to narrow the gap between cloud-computing and usability.

7. ACKNOWLEDGMENTS

Sebastian Schönherr was supported by a scholarship from the University of Innsbruck (Doktoratsstipendium aus der Nachwuchsförderung, MIP10/2009/3). Hansi Weißensteiner was supported by a scholarship from the Autonomous Province of Bozen/Bolzano (South Tyrol). The project was supported by the Amazon Research Grant. We thank the Whirr Mailinglist especially Tom White and Andrei Savu for their assistance.

8. REFERENCES

[1] E. Afgan, D. Baker, N. Coraor, B. Chapman, A. Nekrutenko, and J. Taylor. Galaxy CloudMan: delivering cloud compute clusters. *BMC Bioinformatics*, 11 Suppl 12:S4, 2010.

[2] Apache Hadoop. http://hadoop.apache.org.

[3] Apache Whirr. http://incubator.apache.org/whirr/.

[4] Cloudera. http://www.cloudera.com/.

[5] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. In *OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*, pages

10–10, Berkeley, CA, USA, 2004. USENIX Association.

[6] L. Forer, S. Schönherr, H. Weißensteiner, F. Haider, T. Kluckner, C. Gieger, H. E. Wichmann, G. Specht, F. Kronenberg, and A. Kloss-Brandstätter. CONAN: copy number variation analysis software for genome-wide association studies. *BMC Bioinformatics*, 11:318, 2010.

[7] J. Goecks, A. Nekrutenko, J. Taylor, E. Afgan, G. Ananda, D. Baker, D. Blankenberg, R. Chakrabarty, N. Coraor, J. Goecks, G. Von Kuster, R. Lazarus, K. Li, A. Nekrutenko, J. Taylor, and K. Vincent. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol.*, 11:R86, 2010.

[8] A. Kloss-Brandstättter, D. Pacher, S. Schönherr, H. Weißensteiner, R. Binna, G. Specht, and F. Kronenberg. HaploGrep: a fast and reliable algorithm for automatic classification of mitochondrial DNA haplogroups. *Hum. Mutat.*, 32:25–32, Jan 2011.

[9] B. Langmead, K. D. Hansen, and J. T. Leek. Cloud-scale RNA-sequencing differential expression analysis with Myrna. *Genome Biol.*, 11:R83, 2010.

[10] B. Langmead, M. C. Schatz, J. Lin, M. Pop, and S. L. Salzberg. Searching for SNPs with cloud computing. *Genome Biol.*, 10:R134, 2009.

[11] R. E. Mills et al. Mapping copy number variation by population-scale genome sequencing. *Nature*, 470:59–65, Feb 2011.

[12] Rackspace. http://www.rackspace.com.

[13] M. C. Schatz. CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics*, 25:1363–1369, Jun 2009.

[14] M. C. Schatz. The missing graphical user interface for genomics. *Genome Biol.*, 11:128, 2010.

[15] L. Shi et al. The balance of reproducibility, sensitivity, and specificity of lists of differentially expressed genes in microarray studies. *BMC Bioinformatics*, 9 Suppl 9:S10, 2008.

[16] K. Wang, M. Li, D. Hadley, R. Liu, J. Glessner, S. F. A. Grant, H. Hakonarson, and M. Bucan. PennCNV: An integrated hidden Markov model designed for high-resolution copy number variation detection in whole-genome SNP genotyping data. *Genome Research*, 17(11):1665–1674, Nov. 2007.

[17] Wetterstrand, K. A. DNA Sequencing Costs: Data from the NHGRI Large-Scale Genome Sequencing Program Available: http://www.genome.gov/sequencingcosts; Accessed 04/11/11.

[18] H. E. Wichmann, C. Gieger, and T. Illig. KORA-gen–resource for population genetics, controls and a broad spectrum of disease phenotypes. *Gesundheitswesen*, 67 Suppl 1:26–30, Aug 2005.