School of Information Technology

Proceedings of the 20. GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken)

May 13 – May 16, 2008, Apolda, Germany

Hagen Höpfner (IU Bruchsal, Germany)
Friederike Klan (Uni Jena, Germany)

Technical Report 01/2008

Bruchsal, April 2008

ISSN 1861-9231 (print version)
ISSN 1861-924X (online version)

Dr. Hagen Höpfner is Assistant Professor for Databases and Information Systems at the International University in Germany.
Friederike Klan is research assistant at the Friedrich-Schiller-University of Jena.

Contact: hagen.hopfner@i-u.de

International University in Germany
School of Information Technology
Campus 3
D-76646 Bruchsal
Germany

http://www.i-u.de

Vorwort

Die Workshop-Serie „Grundlagen von Datenbanken“ des GI-Arbeitskreises „Grundlagen von Informationssystemen“ im Fachbereich „Datenbanken und Informationssysteme (DBIS)“ soll die Kommunikation zwischen Wissenschaftler/-innen im deutschsprachigen Raum fördern, die sich grundlagenorientiert mit Datenbanken und Informationssystemen beschäftigen. Sie ist insbesondere als Forum für Nachwuchswissenschaftler/-innen gedacht, die aktuelle, sich eventuell noch in der Entwicklung befindende Arbeiten in einem größeren Forum vorstellen und diskutieren wollen. Im Gegensatz zur häufig anonymen Atmosphäre großer Konferenzen bieten die mehrtägigen Workshops (fast) in Klausur die Gelegenheit zu einem sehr viel intensiveren Kontakt zu den übrigen Teilnehmenden. Auch in diesem Jahr konnten mit Prof. Dr. Kai-Uwe Sattler (TU Ilmenau), Prof. Dr. Bernhard Seeger (Uni Marburg) und Prof. Dr. Wolf-Tilo Balke (TU Braunschweig) zusätzlich zum Programm aus eingereichten Papieren drei hochkarätige Wissenschaftler als eingeladene Vortragende gewonnen werden.

Austragungsort des 2008er GVDB-Workshops und damit auch Autragungsort zum 20. Geburtstag der Workshopserie war Apolda. Diese am Ostrand des Thüringer Beckens gelegene Stadt ist für ihre über 200-jährige Tradition des Glockengießens bekannt und wird daher auch als „Glockenstadt“ bezeichnet. Unter anderem wurde hier die größte Glocke des Kölner Doms gegossen. Darüber hinaus verfügt Apolda über ein sehenswertes Schloss, sowie ein Kunsthaus, welches jährlich zahlreiche Besucher mit Werken weltbekannter Künstler anlockt. Es gab also neben dem rein wissenschaftlichen Programm, wie in den vergangenen Jahren, wieder einiges zu entdecken.

Organisiert wurde der Workshop 2008 von Dr. Hagen Höpfner (International University in Germany) und Friederike Klan (Friedrich-Schiller-Universität Jena).

Jena und Bruchsal, im April 2008

Friederike Klan
Hagen Höpfner

Organisations- und Programmkomitee

Hagen Höpfner (International University in Germany)
Friederike Klan (Friedrich-Schiller-Universität Jena)
Birgitta König-Ries (Friedrich-Schiller-Universität Jena)
Eike Schallehn (Otto-von-Guericke Universität Magdeburg)

Inhaltsverzeichnis

Eingeladene Vorträge

Eingereichte Beiträge

Eingeladene Vorträge

Skyline Queries for Preference-Based Information Systems

Wolf-Tilo Balke

Chair for for Databases and Information Systems,
Technical University of Braunschweig, Germany
balke@ifis.cs.tu-bs.de

Cooperative database retrieval is a challenging problem. In today's information systems instead of just retrieving all objects from databases that exactly match a user's query, often a set of objects best matching the query attributes is desired. Starting from a set of individual user preferences for each attribute, the query can be relaxed step by step until a satisfying result set can be returned. This avoids the notorious empty result effect of over-specified queries, while still respecting the user wishes. The prime paradigm for this kind of retrieval are skyline queries, where a Pareto optimal result set with respect to (partial-order) attribute preferences is returned. The price for this cooperative behavior is however usually a large result set size.

In a decision scenario, like e.g., product search in e-commerce applications, users thus have to choose from a large set of alternatives where each alternative can be described with respect to a list of predefined attributes. The task is to pick the alternative having the highest value with respect to the user's personal preferences among all alternatives. For example, if one is looking for a new car, each alternative is a car offer described in terms of price, top speed, make, and many further attributes. Preferences may consist of simple statements like "I want a fast car", "I want to pay as little as possible, and definitely at most € 30.000", or "I don't mind whether the car color is red or blue, but it should not be black." Since there usually is no brand-new Ferrari for free, the decision one is faced with is often getting complicated.

Multi-criteria optimization is a discipline that deals with problems like the one described above. Unfortunately, although the discipline is several decades old now, there seem to be no method available that provides both high result quality and easiness of use. Only too often the proposed methods work on complex utility functions that cannot be provided by the user in an intuitive way. An important goal of current database research is therefore to develop methods that determine 'optimal' choice selections and return reasonably small sets of alternatives to the user.

Autonomie und Vorhersagbarkeit in DBMS

Eingeladener Vortrag

Kai-Uwe Sattler
Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
FG Datenbanken und Informationssysteme
Postfach 100 565, 98684 Ilmenau
kus@tu-ilmenau.de

Zusammenfassung

Datenbanksysteme sind seit vielen Jahren ein unverzichtbarer Bestandteil von IT-Infrastrukturen in allen Bereichen. Die ursprünglich exponierte Rolle mt dedizierten Systemen und Administratoren verändert sich jedoch zunehmend in Richtung einer Black-Box-Komponente innerhalb komplexer Software-Stacks. Die Bandbreite reicht hierbei von vorkonfigurierten Paketen wie LAMP über datenbankbasierte Appliances bis hin zum Cloud Storage. Wesentliche Anforderungen in diesem Kontext sind einerseits ein geringer Aufwand im Betrieb, andererseits aber auch die Einhaltung von konkreten Dienstgüten wie Antwortzeit oder Durchsatz. Einen wichtigen Beitrag können hier Techniken zum autonomen Management (Self-*-Techniken) von Datenbanksystemen leisten, die Tuning- bzw. Optimierungsaufgaben automatisieren. Im Beitrag werden ausgehend von ausgewählten Anwendungsszenarien und damit verbundenen Anforderungen Lösungen aus dem Bereich Self Tuning und Vorhersagbarkeit skizziert sowie Zielstellungen für weitere Forschungsarbeiten diskutiert.

Mehrdimensionale Indexstrukturen: Die Geister, die ich rief, ...

Eingeladener Vortrag

Bernhard Seeger
Department of Mathematics and Computer Science
Philipps-University Marburg
Hans-Meerwein-Str.
35032 Marburg
seeger@informatik.uni-marburg.de

Zusammenfassung

Vor etwa 25 Jahren war eine der zentralen Forschungsfragestellungen, wie dynamische, eindimensionale Indexstrukturen für mehrdimensionale Daten so erweitert werden können, dass die bekannten Leistungseigenschaften eindimensionaler Strukturen erhalten bleiben. Im Zuge dessen wurden Hash-Verfahren, wie das Grid-File, und diverse Baumstrukturen, wie der R-Baum, entwickelt. Primäres Ziel war es dabei Suchoperationen, wie z. B. die Fensteranfrage, für zweidimensionale Geo-Daten effizient zu unterstützen. Im Zuge dessen arbeitete ich in der Arbeitsgruppe von Professor Kriegel maßgeblich an der Entwicklung von Hash-basierten Zugriffstrukturen mit. Man sprach damals von ordnungserhaltenden Hash-Verfahren; eigentlich hätte der Begriff Interpolationsverfahren besser gepasst. Auf Grund der Suche nach experimentellen Vergleichsstrukturen beschäftigten wir uns schließlich mit dem R-Baum und stellten dabei fest, dass die bis dahin verwendeten Algorithmen verbessert werden können. Niemand von uns dachte wirklich daran, dass wir mit dem R*-Baum eine solch populäre Referenzstruktur entwickelten, die in vielen anderen Forschungsprojekten nicht nur für Geo-Daten, sondern auch für die Verwaltung hochdimensionaler Punktdaten verwendet wurde.

In diesem Vortrag möchte ich die Entwicklung zum Thema mehrdimensionaler Indexstrukturen aus heutiger Sicht kritisch beleuchten. Ein erster Kritikpunkt, der mich Anfang der 90er Jahre bereits bewegt hat, über den Tellerrand von Datenbanksystemen zu schauen, sind die in den meisten Arbeiten getroffenen (einfachen) Modellannahmen über Externspeichersysteme. Diese Systeme sind in den letzten Jahren hochgradig optimiert worden, was dazu führt, dass das einfache Zählen von I/Os (wahlweise und sequentiell) zur Leistungsbestimmung nicht mehr ausreichend ist. Darüber hinaus gibt es bis heute nur sehr wenige Ansätze für verteilte mehrdimensionale Indexstrukturen. Zweitens, ist es bis heute nicht gelungen, mehrdimensionale Indexstrukturen in ähnlicher Weise wie B+-Bäume in bestehende DBMS zu integrieren. Es ist deshalb abzuwägen, ob eine Problemtransformation auf B+-Bäume nicht generell gegenüber der Neuimplementierung von R-Bäumen zu bevorzugen ist. Drittens, und das mag wirklich überraschen, gibt es bis heute noch nicht einmal ein Benchmark oder wenigstens eine Menge standardisierter Daten- und Anfragedateien, um die Performance der verschiedenen Strukturen experimentell zu untersuchen.

Trotz der oben genannten Kritikpunkte ist das Gebiet der Indexstrukturen auch in Zukunft von großer Bedeutung. Zum einen wird die schiere Flut von Daten, die mittels Hard- und Software-Sensoren erfasst werden, so groß, dass der Zugriff auf die Daten nur noch mit Hilfe von geeigneten Indexen zu bewältigen ist. Hierbei spielt das effiziente Laden und Bereinigen der Indexe eine essentielle Rolle. Hohe Update-Raten müssen bei den Indexstrukturen unterstützt werden, um Suchoperation auf sich bewegende Objekte möglichst akkurat zu beantworten. Zum anderen, ergeben sich auch neue Problemstellungen, die den Bedarf an Indexstrukturen für spezifische Datentypen erforderlich machen. Graphen, Zeitreihen und Web-Daten sind Beispiele, bei denen ich derzeit einen hohen Bedarf an neuen Indexierungstechniken sehe. Trotz der jahrelangen Abstinenz lassen mich die Forschungsgeister im Bereich Indexstrukturen nicht los, so dass ich mich zukünftig wieder stärker dafür engagieren werde.

Eingereichte Beiträge

Towards a Syntactic and Partially Semantic Optimization of Database Queries for Indexing Purposes

Hagen Höpfner
International University in Germany
Campus 3; 76646 Bruchsal; Germany

hoepfner@acm.org

Abstract

Database queries are typically used for retrieving data of database systems. Therefore, each query is transformed into a query tree representing the particular operators that need to be performed by the database management system. However, various research areas like semantic caches or update relevancy checks require query based indexing. Queries form indexes that, on the one hand, keep the queries themselves and, on the other hand, assign them to the corresponding objects. In semantic caches such an object is a relation or a (materialized) view. Reusable cached data can then be found by analysing the queries. Unfortunately, most work in such areas did not research the efficient creation and maintenance of query indexes. In this paper we discuss first steps towards optimizing them based on the well know query trees.

1 Introduction and Motivation

Nowadays ubiquitous, nomadic, and pervasive computing are no longer visions but realized in various scenarios. Devices become smaller and easier to carry around and wireless links allow to connect to world wide available information anytime and everywhere. Data processing with lightweight and implicitly impermanently connected devices always results in redundant data management. Hoarding [10], replication [4] or semantic caching [11] are appropriate techniques discussed in the literature. Some part of transmitted data is stored on the mobile device and reused locally. Thus, not only the data themselves but also the information about how this data have been computed must be managed efficiently on the mobile device. This issue becomes obvious in the case of semantic caches. In order to reuse parts of semantically cached data, algorithms have to compare cached queries to a given new query in order to find overlaps. In other words: queries are used for indexing the cached data. Furthermore, in [8] we have shown, that cache coherency aspects must be handled on the information systems' server. Therefore, queries have to be stored on the server as well as on the mobile clients.

Normally database queries are used for retrieving, defining, and modifying data and meta data managed by a database management system (DBMS). In the (object) relational world the descriptive structured query language (SQL) became a quasi standard. Unfortunately, SQL allows to formulate semantically equivalent queries in a syntactically different way. Even easy examples (see Example 1) illustrate problems resulting from the full variety of SQL. Both queries Q_1 and Q_2 result in the same relation but would be represented differently in a query index as they are syntactically different.

Example 1: *Semantically but not syntactically equivalent SQL queries*
Q_1 : `SELECT * FROM TABLE1 AS T1 WHERE B=5 AND A=4;`
Q_2 : `SELECT * FROM (SELECT * FROM TABLE1 WHERE A=4 and B=5) AS T1;`

In order to answer retrieval queries, which we focus on in this paper, SQL's select statements are first translated into query trees by the DBMS' query processor. A query tree contains the relational algebra operators and is the starting point of the query optimization process. In the following paper we adapt the idea of transforming SQL into a query tree but do not optimize it regarding execution time. We generate a less descriptive, semantically equivalent query representation from the tree to support the indexing. The two queries Q_1 and Q_2 will be transformed to the same index entry and thus represented only once.

The reminder of the paper is structured as follows: Section 2 introduces the concept of query tries and specifies the (relational complete) set of operators that we support at the moment. Section 3 discusses our approach to unify the query representation. Section 4 concludes the paper and gives an outlook on future research.

2 Queries and Query Trees

As mentioned in Section 1 we assume that database queries are posted in form of SQL-statements. In this paper we focus on the relational complete set of algebra operators that is described in the standard database literature such as [3]. This set[1] consists of selection σ, projection π (`SELECT DISTINCT`), set union $\cup$, set difference $-$, and Cartesian product $\times$. In addition to this we allow renaming of attributes as well as renaming of relations ρ, and set intersection $\cap$. For the time being we ignore more enhanced operators like aggregate functions or groupings.

The difference between set and multi-set semantics, which might be specified in the SQL-query, is marked by an enhanced relational projection operator π^a (`SELECT`). We implicitly assume multi-set semantics for the Cartesian product, the set operators and the selection operator. If necessary, duplicate elimination is done by the projection operation that is mandatory for each query. There exist more detailed studies on multi-set relational algebra [2, 5] illustrating that multi-set relational algebras have nearly the same properties as set based approaches. Therefore, we do not discuss them in more detail but use them for syntactical purposes only. Assuming that the "*" means all attributes of relations used by a query, translating the two SQL queries from Example 1 could result in the relation algebra expression $Q_1 = \pi^a_*(\rho_{\texttt{T1}}(\sigma_{B=5 \wedge A=4}(\texttt{TABLE1})))$ and $Q_2 = \pi^a_*(\rho_{\texttt{T1}}(\pi^a(\sigma_{A=4 \wedge B=5}(\texttt{TABLE1}))))$. Since we explicitely parse an SQL-query we can guarantee the following structural properties of the resulting relational algebra expressions. A query q can have the following recursive structure[2]:

- $q : \{\pi|\pi^a\}([\sigma]([\rho](R)))$
- $q : \{\pi|\pi^a\}([\sigma](\rho(q)))$
- $q : \{\pi|\pi^a\}([\sigma](cp))$
- $cp : \{[\rho](R)|\rho(q)\} \times \{[\rho](S)|\rho(q)|cp\}$
- $q : \{\pi|\pi^a\}(q\{\cup| - |\cap\}q)$

The dbms' internal processing of queries in database management systems is based on query trees. Leave nodes represent the relations, inner nodes store query operators. Since the result of each relational algebra operator is a relation, queries can be processed bottom-up following the edges in the tree. After performing the root operation, which is based on the results of all other node operations' results, the query is completely answered. Sub-queries are boxed and form subtrees. The two example queries and a query $Q_3 = \pi^a_B(\sigma_{\texttt{TABLE2}.B>\texttt{TABLE1}.A}(\texttt{TABLE1} \times \texttt{TABLE2}))$ corresponding to the SQL expression `SELECT DISTINCT B FROM TABLE1, TABLE2 WHERE TABLE2.B>TABLE1.A` would lead to the query trees in Figure 1.

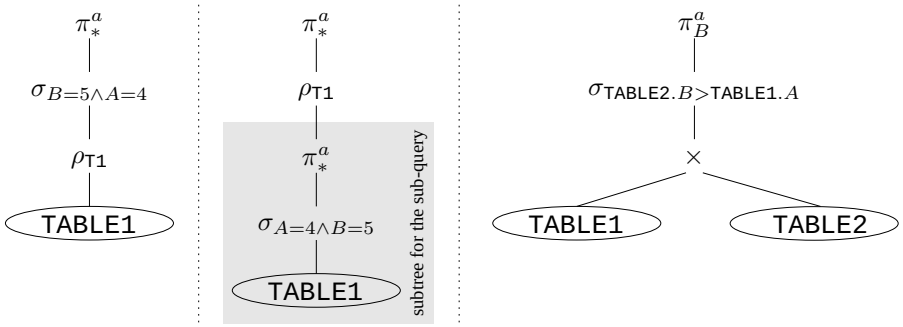

Figure 1: Query trees for the example queries (Q_1 left, Q_2 middle, Q_3 right)

Following [3] we start with canonical (unoptimized) query tree that – per sub-query – represents select and join conditions in one selection node on top of Cartesian products.

3 Query Normalization

In this section we describe our approach towards less complex query representations for indexing semantic caches. Therefore, we introduce two steps to rewrite the canonical query tree: (1) The removal of unnecessary sub-trees and (2) the equivalence transformation of the predicate expressions.

[1] We are aware of the fact that there are other relation complete sets. Due to the completeness property they can be substituted by each other.
[2] Operators in square brackets encapsiolate optional operators. R and S are relations. Braces mark alternatives.

Depending on the intended use of the query index, these steps can be implemented in the semantic cache on the mobile client, on a mobility supporting middle-ware [7], or in the back-end DBMS (if it stores posted queries instead of using a middle-ware).

3.1 Sub-Tree Optimization

As Figure 1 shows, queries with a different sub-tree structure can have the same semantics. In order to assign equivalent queries with the same entry in a semantic cache, the sub-tree structure has to be normalized. Therefore, we reduce the number of unnecessary sub-trees in our first query rewriting step. The resulting query is semantically equivalent to the initial one. Since this paper describes our ongoing work, there are still cases where two different but semantically equivalent queries result in different representation after normalization. Thus, there is high potential for future research.

Different sub-trees can handle duplicates differently, i.e., one sub-tree might follow a multi-set semantics while another one removes duplicates. Thus, we have to consider duplicate elimination during the tree transformation.

3.1.1 π^a-π-Optimization

Before starting the tree transformation we make use of the fact that duplicate elimination at the (sub-)root node of a certain sub-tree implizitely holds for it's sub-trees, too. In other words, all sub-query of a `SELECT DISTINCT`-query can be written as `SELECT DISTINCT`-sub-queries. The proof of this assumption can be found in [6]. The authors of this paper state that `SELECT DISTINCT` D `FROM` $R \equiv$ `SELECT` D `FROM` R `GROUPBY` D and that groupings can be pushed down and pulled up under certain conditions. Hence, we traverse the query tree and replace all π^a operators below a π-operator by the appropriate duplicate eliminating projection.

3.1.2 Sub-Queries without Set Operations and Cartesian Products

(Sub-)queries consisting of nested queries of the form $Q : Q_{out}(Q_{in_1}(Q_{in_2} \ldots (Q_{inmost})))$ can be reduced if the inner queries contain *neither* set-operations *nor* Cartesian products. Following the formal query representation in Section 2, the inmost sub-query Q_{inmost} must have the following structure $Q_{inmost} : \{\pi|\pi^a\}([\sigma]([\rho](R)))$. The handling of duplicates in nested sub-queries without Cartesian products is comparable to our π^a-π-optimization: the projections in all sub-queries Q_* can be unified to π if at least one sub-query removes duplicates. Otherwise, all projections in Q_* are π^a. For simplification purposes we use in the following discussion π only. However, the same transformations can be applied for π^a. Remember that SQL requires the renaming of sub-queries. As shown in Section 2 the inmost operator of the outer query has to be a ρ. In our canonical query tree structure, ρ is the parent operator of the projection node of an inner sub-query.

Our sub-tree optimization starts at the inmost nested sub-query and is used bottom-up until a set operation, a Cartesian product or the root-node is reached. We combine a boxed sub-query with its nesting query by merging the projections and selection predicates, and by unifying all relation names and attribute names which have been renamed in the nesting query. Therefore, we have to distinguish two cases:

Case 1: Outer (sub-)query uses projection and renaming: $\overbrace{\pi_{pl_o}(\rho_{al_o}}^{\text{outer sub-query}}(Q_{inmost}))$ π_{pl_o} overwrites the projection of the inner query *but* is based on the renaming ρ_{al_o}. Furthermore, ρ_{al_o} overwrites a potential renaming within Q_{inmost}. So, both queries are merged as follows:

1. Remove the projection operator of the inner sub-query if it is less restrictive (i.e., it covers more attributes) than the projection operator of the outer sub-query. Otherwise, replace the outer projection operator by the inner one.

2. If the inner query does neither contain a renaming nor a selection, both queries have been merged: Stop rewriting.

3. If the inner query contains a selection σ_{sc_i}, change the order of σ_{sc_i} and ρ_{al_o}.

   If the inner query contains a renaming $\rho_{al_i}(R)$, merge the set of aliases in the inner ($al_i = RA_i(AA_i)$) and outer ($al_o = RA_o(AA_o)$) query, i.e.,

   (a) Merge and reduce the attribute-name-aliases set: $AA_m = reduce(AA_i \cup AA_o)$.

   (b) For all relations in nested renaming sequences $R \rightarrow RA_i \rightarrow RA_o$: Replace $\rho_{al_i}(R)$ by $\rho_{al_m}(R)$ where $al_m = RA_o(AA_m)$.

(c) For all inner attribute aliases $al^i \leftarrow a$ with $a \in R$ and $al^i \in AA_i$ that are overwritten by the outer renaming $al^o \leftarrow al^i$, $al^o \in AA_o$: Replace $al^i \leftarrow a$ by $al^o \leftarrow a$.

(d) Remove $al^o \leftarrow al^i$ from $AA_i \cup AA_o$.

(e) Replace the outer renaming ρ_{al_o} by the merged renaming ρ_{al_m}, and remove ρ_{al_i}.

4. If the inner query contains a selection σ_{sc_i}, adapt the selection conditions sc_i to the new aliases, i.e.,

(a) Replace all relation names of R and RA_i in sc_i by RA_o.

(b) Replace an attribute name a in sc_i by al with $al \leftarrow a \in AA_m$, if it has been renamed by $\rho_{RA_m(AA_m)}$.

The function *reduce* replaces transitive aliases similar to the table aliases above. Note that Step 3 is not an equivalence transformation, because ρ_{al_o} might rename attributes or relations needed for σ_{sc_i}. Step 4 corrects this.

Case 2: Outer (sub-)query uses projection, selection and renaming: $\overbrace{\pi_{pl_o}(\sigma_{sc_o}(\rho_{al_o}}^{\text{outer sub-query}}(Q_{inmost})))$ The only difference to Case 1 is that the outer selection σ_{sc_o} and the inner selection σ_{sc_i} have to be merged after performing the other steps. Therefore, σ_{sc_i} is removed from from the sub-tree and and σ_{sc_o} is replaced by a new selection σ_{sc_m} where $sc_m = sc_o \wedge sc_i$ holds. Example 2 illustrates the usage of the steps for optimizing Q_2:

Example 2: *Sub-query optimization*
The initial tree representation of Q_2 is $\pi^a_* - \rho_{\texttt{T1}} - \boxed{\pi^a_* - \sigma_{A=4 \wedge B=5} - \texttt{TABLE1}}$. In the first step the outer projection is replaced by the inner one. The resulting tree is: $\pi^a_* - \rho_{\texttt{T1}} - \boxed{\sigma_{A=4 \wedge B=5} - \texttt{TABLE1}}$. The remaining inner sub-query contains a selection. Therefore, we have to continue with step 3. The result of step 3 is: $\pi^a_* - \boxed{\sigma_{A=4 \wedge B=5} - \rho_{\texttt{T1}} - \texttt{TABLE1}}$. Since the inner sub-query does not contain a renaming, $\rho_{\texttt{T1}}$ becomes the final renaming operator. However, $\sigma_{A=4 \wedge B=5}$ does not use the renamed relation name and the attributes were not renamed. Therefore, the select condition does not have to be changed.

3.2 Where-Condition Optimization in all Sub-Trees

Our second optimization step targets at selection and join predicates. As semantically equivalent predicates can be expressed in numerous ways, the usage of query representations for indexing calls for normalization of predicates.

It is well-known that the equivalency problem of logical expressions is undecidable in the general case. However, when considering SQL, where-conditions are (first-order) predicate logic expressions. Since we focus on the relational complete relational algebra operator set which does not contain quantifiers, our where-conditions correspond to propositional logic expressions. Thus, due to the recursive definition of the term expression and due to the well known properties of comparison operators it is possible to conduct a large number of equivalence transformations on such expressions. We propose to optimize the predicates in two phases:

Phase 1: Lexicographical sorting One kind of difference between two predicates can result from the order of conjuncts. For example, the where-conditions of $Q_1 : B = 5 \wedge A = 4$ and $Q_2 : A = 4 \wedge B = 5$ are equivalent, because they are satisfied independent from the order of the conjuncts. Thus, our first optimization phase orders the conjuncts lexicographically. Obviously, the following rules hold: $B < A \equiv A > B$, $B > A \equiv A < B$, $B \leq A \equiv A \leq B$, $B \geq A \equiv A \leq B$, $B = A \equiv A = B$, and $B \neq A \equiv A \neq B$. In particular, we rewrite all attribute-attribute-comparisons[3] (AAC) so that that the first attribute is lexicographically smaller than the second one.

Example 3: *Ordering of conjuncts*
Given the AAC `TABLE2.B>TABLE1.A` of Q_3 we force a lexicographic order by exchanging the attributes and substituting the comparison operator. The result is `TABLE1.A<TABLE2.B`.

[3] e.g. from join-conditions

Phase 2: Term minimization In the second phase we also analyze the attribute-value-comparisons[4] (AVC). AVCs might form an unnecessarily complex propositional logic expression. There exist algorithms for transforming an arbitrary logical expression to a conjunctive normal form [14] and to minimize this normal formed expression [13, 9, 12, 1]. However, all those algorithms find *one* minimal expression, but cannot exclude that there are other expressions which are minimal as well. Hence, it is not possible to guarantee that semantic equivalent expressions are reduced to the same minimal expression. However, for the time being we take the algorithmically found term and order it lexicographically. More specifically, we represent each where-condition in a minimal conjunctive normal form where the atomic terms of each conjunct are ordered lexicographically. After ordering the conjuncts, we order the entire expression lexicographically. The resulting example query trees are shown in Figure 2 (changes are boxed).

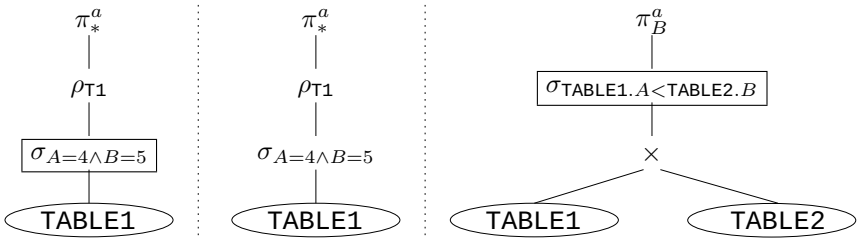

Figure 2: Query trees for the example queries (Q_1 top left, Q_2 top right, Q_3 bottom) after the where-condition optimization

4 Conclusions, Summary and Outlook

In this paper we presented first ideas towards finding a syntactically equivalent representation of semantically equivalent database queries. After introducing the concept of reusing the initial query tree for this purpose we explained some operator rewritings that are mostly based on reformulating the syntax but also the semantic of the query. However, this needs to be researched in more detail. For example, in the integer domain a select condition $A > 5 \wedge A \neq 4$ is equivalent to $A > 4$. In order to handle this kind of equivalence one has to know the domains of the attributes. Thus, we will have to consider the meta data here. Operator semantic is a key issue for future research. Obviously, `SELECT DISTINCT * FROM A` and `SELECT * FROM A UNION SELECT * FROM A` are semantically equivalent as both simply remove duplicates from `A` but syntactically different.

Furthermore, we did not discuss table and attribute name aliases in much detail in this paper. The example queries Q_1 and Q_2 would result in the same extension without using the renaming operator. Hence, on must think about the cases where renaming operations do not have any influence on the extensional query result. Especially table aliases might be removable by analysing the query only. Attribute name aliases require a deeper look at the meta data since they are relation specific.

References

[1] Nripendra N. Biswas. Computer aided minimization procedure for boolean functions. In *Proceedings of the 21st conference on Design automation, Albuquerque, New Mexico, United States*, pages 699–702, Piscataway, NJ, USA, 1984. IEEE Press.

[2] Umeshwar Dayal, Nathan Goodman, and Randy H. Katz. An extended relational algebra with control over duplicate elimination. In *Proceedings of the 1st ACM SIGACT-SIGMOD symposium on Principles of database systems*, pages 117–123, New York, NY, USA, December 1982. ACM Press.

[3] Ramez Elmasri and Shamakant B. Navathe. *Fundamentals of Database Systems*. Addison Wesley, Boston, 5th edition, 2007.

[4] Jim Gray, Pat Helland, Patrick O'Neil, and Dennis Shasha. The Dangers of Replication and a Solution. *SIGMOD Record*, 25(2):173–182, June 1996.

[5] Paul W.P.J. Grefen and Rolf A. de By. A multi-set extended relational algebra: a formal approach to apractical issue. In *Proceedings of the 10th International Conference on Data Engineering, February 14-18, 1994, Houston, Texas, USA*, pages 80–88, Washington, DC, USA, March 1994. IEEE Computer Society.

[6] Ashish Gupta, Venky Harinarayan, and Dallan Quass. Aggregate-Query Processing in Data Warehousing Environments. In Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio, editors, *Proceedings of the 21st Conferencer on Very Large Data Bases (VLDB '95), September 11-15, 1995, Zurich, Switzerland*, pages 358–369, San Francisco, USA, 1995. Morgan Kaufmann Publishings, Inc. available online: `http://www.vldb.org/conf/1995/P358.PDF`.

[7] Hagen Höpfner. Query Based Client Indexing in Client/Server Information Systems. *Journal of Computer Science*, 3(10):773–779, December 2007.

[8] Hagen Höpfner. *Relevanz von Änderungen für Datenbestände mobiler Clients*. VDM Verlag Dr. Müller, Saarbrücken, 2007. in German.

[9] Maurice Karnaugh. The Map Method for Synthesis of Combinational Logic Circuits. *Transactions of American Institute of Electrical Engineers*, 72(7):593–599., 1953.

[10] Geoffrey H. Kuenning and Gerald J. Popek. Automated Hoarding for Mobile Computers. *ACM SIGOPS Operating Systems Review*, 31(5):264–275, December 1997.

[11] Ken C. K. Lee, Hong Va Leong, and Antonio Si. Semantic query caching in a mobile environment. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(2):28–36, 1999.

[12] Edward J. McCluskey. Minimization of Boolean Functions. *Bell System Techical Journal*, 35(5):1417–1444, November 1956.

[13] Willard Van Orman Quine. The problem of Simplifying Truth functions. *American Mathematics Monthly*, 59(8):521–531, October 1952.

[14] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, USA, 2nd edition, December 2002.

[4] resulting from select-conditions

WISSLOG - Semantik-basierte Logistik-Services in unternehmensübergreifenden Seehafen-Umgebungen

Stefan Audersch, Guntram Flach, Matthias Rust
Zentrum für Graphische Datenverarbeitung e.V., Rostock
Joachim-Jungius-Str. 11, 18059 Rostock
{audersch, gf, mrust}@rostock.zgdv.de

Abstract: Durch die Nutzung von Wissens-Technologien lassen sich komplexe Logistikinformationssysteme innerhalb der inner- und überbetrieblichen Kommunikation effizienter und flexibler gestalten. Die Bearbeitung der Prozesse setzt dabei meist die Zusammenarbeit zwischen verschiedenen Dienstleistern, Anwendern sowie die Einbeziehung von Fachwissen voraus. Workflow-, Web-Services- und Semantic-Web-Technologien bilden eine mögliche Grundlage zur Konzeptionierung eines universellen Ansatzes zur Schaffung interoperabler Logistik-Services, die semantisch gesteuert die Integrations- und Wissensprozesse im Sinne "intelligenter" Services realisieren. Ausgehend von der Workflow-Verarbeitung sowie der Verwendung von Metadaten, Ontologien und Regel-Systemen werden unterschiedliche Wissensmanagement-Ansätze für die Nutzung von semantischen Informationen entwickelt, die eine Unterstützung unternehmensübergreifender Logistikprozesse ermöglichen. Die Ansätze werden am Beispiel „Seehafen-Logistik“ vorgestellt.

1 Einleitung

Im Vorhaben WISSLOG werden Konzepte, Verfahren und Werkzeuge erarbeitet, die im hochvolantilen Umfeld der Logistik eine integrative Kombination der Schwerpunkte Flexibilität, Wissen und Ressourcen ermöglichen. Die Strukturierung der Forschungsthemen orientiert sich dabei in starkem Maße an den Defiziten und Problemstellungen derzeitiger Wertschöpfungsketten [Sch06, SF+05]. Der Kopplung von neuartigen Logistik-Ansätzen mit adäquaten Wissensmanagement- und Prozesssteuerungs-Strategien kommt demnach zunehmende Bedeutung zu.

WISSLOG wird seit Ende 2006 als Kooperationsprojekt mit der Scheller Systemtechnik GmbH[1] im Rahmen der PROINNOII-Initiative (BMWi) entwickelt. Die Evaluierung der entwickelten Lösungskomponenten für die Bereiche Service-orientierte Container-Disposition und adaptive Lager-/Schiffsliegeplatz-Planung und -Optimierung erfolgt unter Berücksichtigung der Anforderungen im Seehafens Wismar[2]. Eine Übertragung der entwickelten Konzepte und Methoden auf den Seehafen Rostock[3] ist darüber hinaus vorgesehen. Die Abbildung 1 zeigt die für die Problemstellung relevanten Prozesse des Schiffsdurchlaufes und des nebenläufigen Warenflusses mit den zugehörigen Teilprozessschritten, Akteuren und zugehörigen Informationssystemen. Problematisch ist hierbei die teilweise fehlende Integration und Anbindung von unternehmensübergreifenden Informationssystemen und die mangelnde adaptive Planungs-, Optimierungs- und Entscheidungsunterstützung bei der Reaktion auf prozessrelevante Ereignisse.

Zur Lösung dieser Aufgaben besteht die Notwendigkeit, die Daten sowie die unternehmensübergreifenden Planungs- und Optimierungsprozesse bzgl. der verschiedenen Informationsquellen zu integrieren und in interoperable, wissensbasierte Logistikprozesse einzubetten. Voraussetzung für eine intelligente Zusammenführung und dynamische, adaptive Prozessteuerung sowie Assistenz-Unterstützung ist eine maschinenverständliche Semantik der Informationsqellen und Teilprozesse. Grundlage hierfür bietet eine gemeinsame domänenspezifische

[1] www.scheller.de
[2] www.hafen-wismar.de
[3] www.rostock-port.de

Seehafen-Logistik-Ontologie, die unter anderem eine gemeinsame Terminologie abbildet sowie weitere ontologische Strukturen (Güter-Klassifikation) und Regelsysteme enthält.

Ausgehend vom geschilderten Anwendungsszenario und den genannten Anforderungen wird im nächsten Abschnitt die Architektur kurz vorgestellt, bevor darauf folgend einige Realisierungsaspekte genauer dargestellt werden.

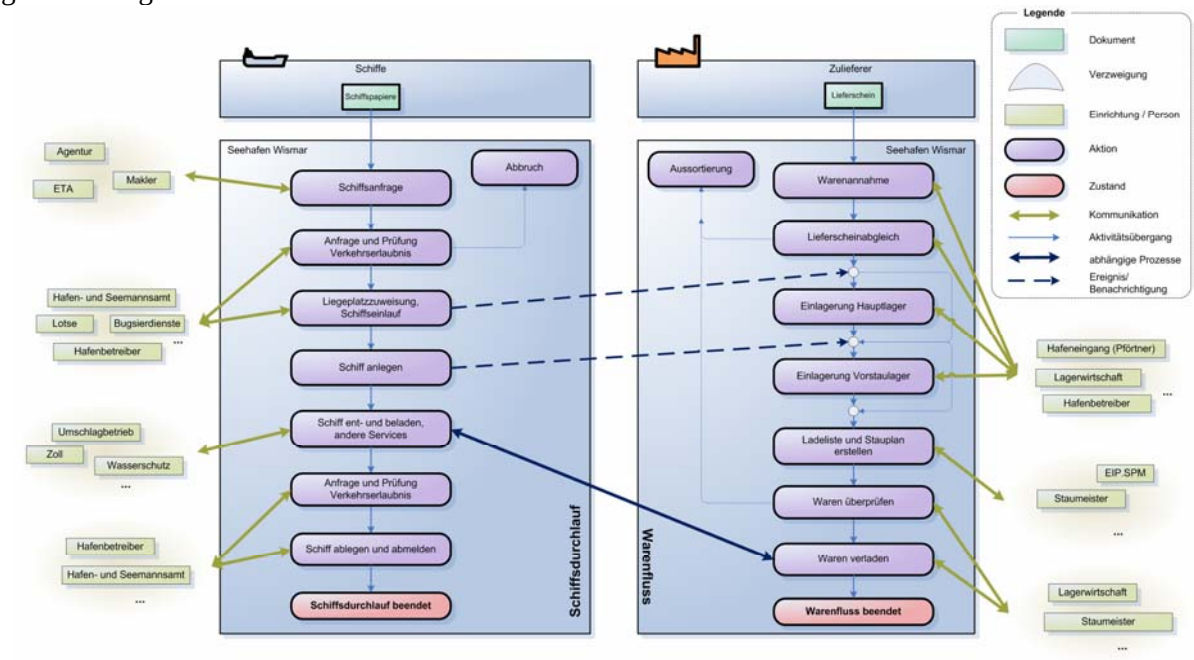

Abbildung 1: Schiffsdurchlauf und Warenfluss

2 System-Architektur

Im Rahmen des Projektes wurde die in Abbildung 2 dargestellte Systemarchitektur entwickelt. Zentrale Komponente ist der *Controller*, der für die Steuerung und Kontrolle umgebender Systeme und Dienste verantwortlich ist. Die Planung und Optimierung von logistischen Aktivitäten wird vom Controller auf Grundlage eines *mathematischen Modells* durchgeführt. Dieses Optimierungsmodell muss jeweils an das konkrete Anwendungsszenario und die verfügbaren Informationen und Systeme angepasst werden. Der *Event Handler* unterstützt die Adaptivität des Systems und ist dafür verantwortlich, Kerngrößen innerhalb des Optimierungsmodells in Abhängigkeit von den umgebenden Systemen und Wissensquellen (AIS[4], HIS-MV[5], SPM[6]) anzupassen und ggf. durch einen erneuten Planungsvorgang eine neue optimale Aufgabenverteilung zu finden (z. B. bei Abweichung der erwarteten Schiffsankunftszeit, beim Eintreffen von RFID-events zur Gut-Lokalisierung).

Die Resultate des Plans werden vom *Auftrags Handler* verwaltet, der in Zusammenarbeit mit dem *Workflow-System* und der Logistik-Ontologie aus dem *Metadaten Repository* die Ausführung einzelner Tätigkeiten und logistischer Dienstleistungen veranlasst und kontrolliert. Das *Metadaten Repository* verwaltet alle Metadaten und ermöglicht den Zugriff auf diese Daten für alle anderen Komponenten. Das Repository beinhaltet Metadaten, die für die Abarbeitung des Workflows, der Planung-/Optimierung und zur Entscheidungsunterstützung (Assistenz) notwendig sind. Hierzu gehören unter anderem eine Güter-Ontologie, domänenspezifische Ontologien (Seehafen) sowie eine Ontologie, die die Organisations- und Serviceeinheiten des Seehafens Wismar abbildet (Logistik-Ontologie). Darüber hinaus umfasst das Metadaten Repository weitere Metadaten zu angebundenen Informationssystemen (SPM, AIS, HIS-MV) und deren Schnittstellen.

[4] Automatic Identification System
[5] Hafeninformationssystem Mecklenburg-Vorpommern
[6] Stock Processing Management

Für die manuelle Kontrolle und Einflussnahme auf die Kernprozesse durch Dispatcher ist ein Web-Interface vorgesehen, das direkt mit den Kernkomponenten des Controllers verbunden ist und durch die Einbettung von WebGIS-Komponenten zusätzliche Assistenz für die Entscheidungsfindung bietet.

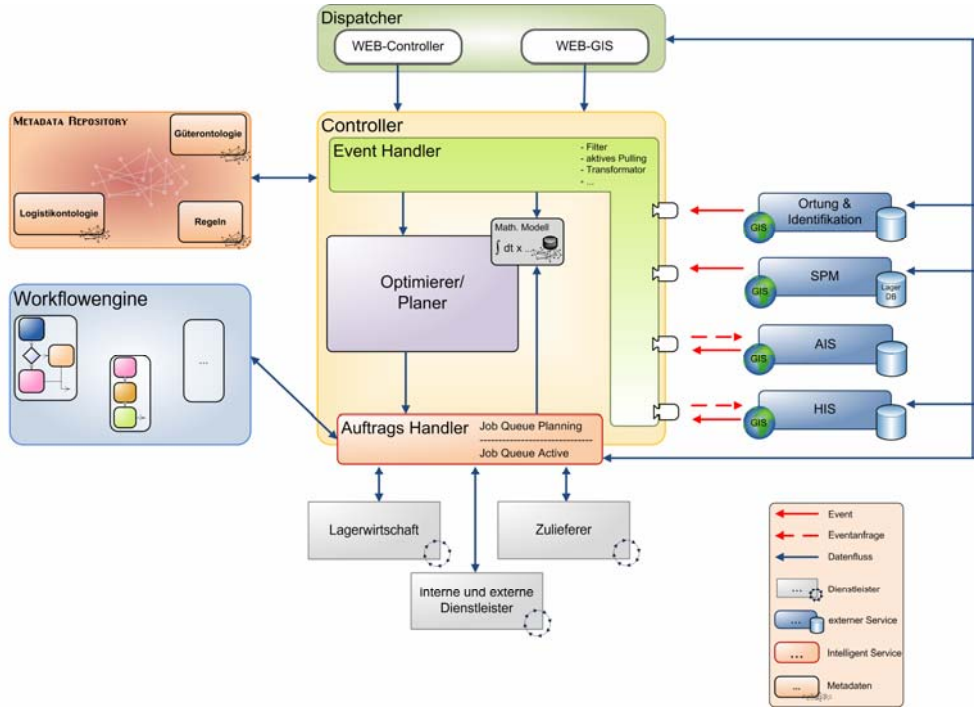

Abbildung 1: WISSLOG-Systemarchitektur

3 Realisierungsaspekte

Event-Handling

Die Informationssysteme sind, wie auf der Abbildung 3 zu sehen, durch Konnektoren an den Event Handler angebunden. Für beide Event-Strategien (1.+2.) hält der Konnektor eine Transformator- und Filterfunktion zur Verfügung. Bei Eintreffen einer Nachricht werden zunächst durch den Transformator die service-spezifschen Formatierungen in ein internes, einheitliches und temporäres XML-Format überführt. Der Filter verwirft die Informationen, die für Planungs- und Optimierungsprozesse vollkommen irrelevant sind und weiterhin die Daten, die sich seit den letzten Anfragen bzw. der letzten Entgegennahme von Nachrichten nicht verändert haben.

Das interne Problem-Modell ist eine Sammlung relevanter Daten über die aktuellen Inhalte der umgebenden Informationssysteme (3.) und zusätzlichem Wissen aus dem Metadata Repository (4.). Dieses interne Modell ist in einer XML-Variante von PDDL (Planning Domain Description Language, Siehe Abschnitt 3.2) verfasst. Dadurch kann eine schnelle Transformation und Weiterverarbeitung erfolgen. Die Haltung der Daten in einem internen Modell (XPDDL) ermöglicht das schnelle Ausführen neuer Planungs- und Optimierungsprozesse unter Nutzung aktueller Daten. Zusätzlich können Informationen verschiedener Systeme bequem zentral abgeglichen werden.

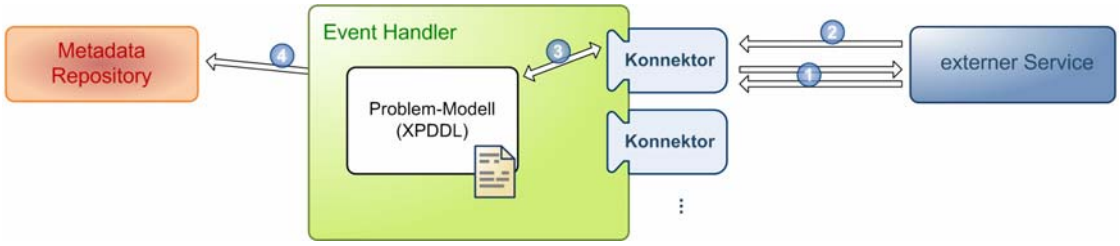

Abbildung 3: Aufbau und Funktionsweise des Event-Handlers

Planungs- und Optimierungsprozess

Der zentrale Optimierungs- und Planungsprozess wurde unter Anwendung der Planning Domain Description Language PDDL [PDD08, AF+08] abgebildet. Ein PDDL-Planer erwartet zwei Eingabestrukturen. Die Domain-Beschreibung bildet die im Optimierungsmodell vorkommenden Objektklassen, mögliche Eigenschaften, Beziehungen und Aktivitäten ab. Die eigentliche Problem-Beschreibung verweist auf die Domain-Beschreibung, instanziiert konkrete Objekte und deren Eigenschaften. Die Erzeugung der Problem-Beschreibung findet bei WISSLOG in enger Kooperation mit dem Event und dem Auftrags-Handler statt. Durch Einbindung des Ontologie-Repositories werden semantische Informationen in den Planungsprozess einbezogen (z.B. Lagereinschränkungen aufgrund der Güterklassen). Eine Neuplanung kann dabei jederzeit mit einer geänderten Problembeschreibung veranlasst werden. Das Ergebnis des Planungsprozesses ist eine Aktivitätenliste, die an den Auftragshandler übergeben wird. Der Vorteil in der Verwendung von PDDL liegt in der Eignung der Sprache für die Abbildung unterschiedlicher Optimierungszenarien und der Verfügbarkeit mehrerer PDDL-kompatibler Planer.

Semantische Workflowsteuerung

Das Ergebnis eines Planungsvorgangs ist eine Liste durchzuführender Aktivitäten. Dabei muss zwischen „nur“ geplanten und bereits in Ausführung befindlichen Aktivitäten unterschieden werden. Durch die Workflowkomponente wird je Aktivität ein komplexer Workflow initiiert, der u.a. die Auftragsvergabe an Dienstleister und die Erfolgsüberwachung abbildet. An relevanten Punkten wird das mathematische Optimierungsmodell an den aktuelle Workflowstatus angepasst, da die in Ausführung befindlichen Aktivitäten nicht mehr neu geplant werden dürfen. Die Verwendung von BPEL unterstützt die vorliegende WebService-basierte Architektur.

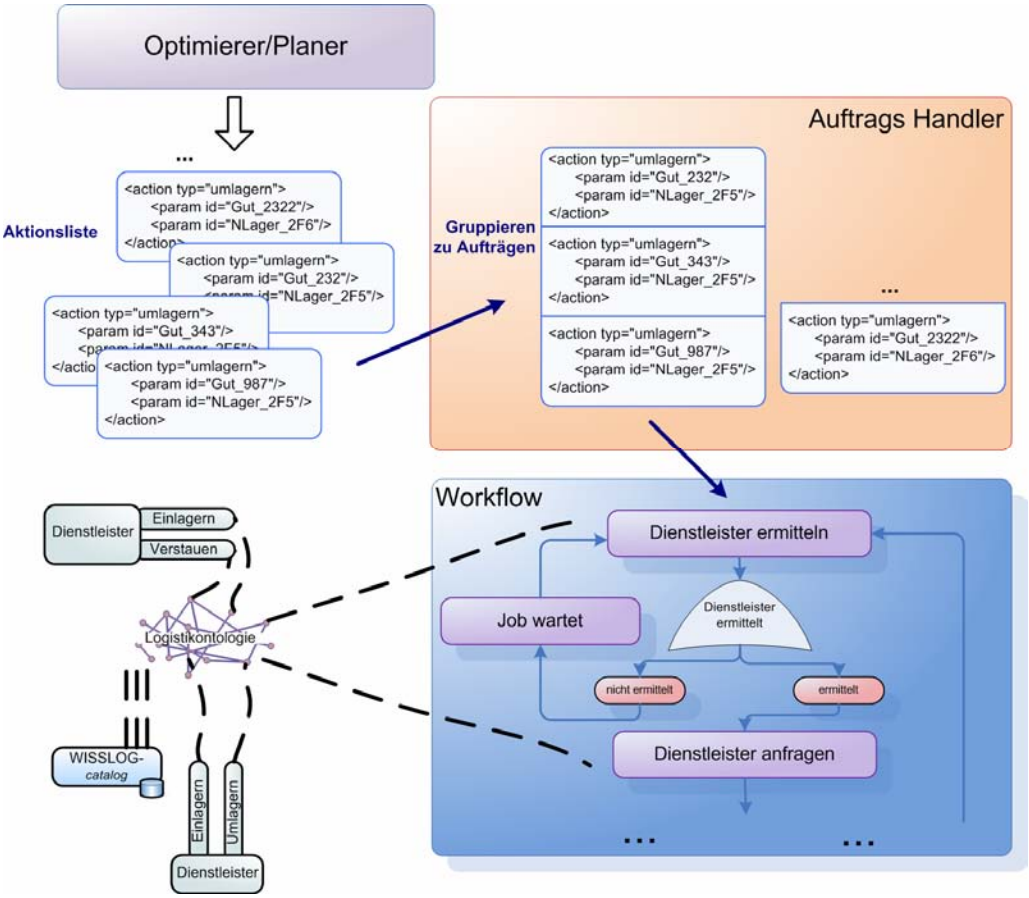

Abbildung 4: Auftrags-Handler

4 Zusammenfassung und Ausblick

Das entwickelte WISSLOG-Framework dient der flexiblen, wissensbasierten Assistenzunterstützung sowie der adaptiven Planungs- und Optimierungssteuerung in unternehmensübergreifenden Logistik-Umgebungen. Ausgehend von der Workflow-Verarbeitung sowie der Verwendung von Metadaten, Ontologien und Regel-Systemen wurden unterschiedliche Ansätze für die Nutzung von semantischen Informationen entwickelt, um eine Unterstützung wissensintensiver Logistik-Services zu ermöglichen. Ein Schwerpunkt dabei ist die Realisierung einer kontextbezogenen Assistenz-Funktionalität, die bei komplexen Entscheidungs- und Dispositionsprozessen die Arbeit der Dispatcher unterstützt oder ereignisgesteuert parametrisierte Dienste-Kompositions- bzw. Auftrags-Prozesse zur Laufzeit auslöst. Neben der Evaluierung des derzeitigen wissensbasierten Optimierungs-Ansatzes wird in zukünftigen Arbeiten der Einsatz von RFID-Technologie [BIT08] für unterschiedliche Aspekte der Ereignissteuerung und nachgelagerten Aufgaben-Delegation untersucht sowie eine Kopplung mit der EIP-Plattform der Scheller Systemtechnik GmbH vorgenommen..

Literaturverzeichnis

[AF+08] Audersch, S., Flach, G., Rust, M., Hünemörder, U.: Wissensbasierte Planung und Optimierung in unternehmensübergreifenden Seehafen-Logistik-Umgebungen. In: 13. Magdeburger Logistik-Tagung – Netzwerklogistik, 2008

[BIT05] BITKOM – Telekommunikation und neue Medien: White Paper RFID Technologie, Systeme und Anwendungen, 2005

[PDD08] International Planning Competition: PDDL 3.0 - Planning Domain Definition Language, Verfügbar unter: http://zeus.ing.unibs.it/ipc-5/

[Sch06] Schwänzl, S.: Systematik zur Gestaltung von Supply Chains unter Einfluss des Wissensmanagement. IN: 12. Magdeburger Logistik-Tagung – Sicherung von Prozessketten, 2006

[SF+05] Scholz-Reiter, B., Freitag, M., Rekersbrink, H., Wenning, B.: Auf dem Weg der Selbststeuerung in der Logistik. In: 11. Magdeburger Logistik Tagung – Intelligente Logistikprozesse, 2005

Anforderungen Mobiler Informationssysteme im Umfeld Ubiquärer Umgebungen

Andre Peters, Andreas Heuer
Universität Rostock, Institut für Informatik
Lehrstuhl Datenbanken und Informationssysteme
{ap,heuer}@informatik.uni-rostock.de

1. April 2008

Zusammenfassung

Im Rahmen des Forschungsprojektes **MuSAMA** (Multimodal Smart Appliance Ensembles for Mobile Applications) der Deutschen Forschungsgemeinschaft beschäftigen wir uns mit der Fragestellung, inwieweit ubiquäre Intelligenz unserer zukünftigen Umwelt durch dynamische Ensembles gebildet wird. Ein Schwerpunkt dieser Forschung obliegt der Nutzbarkeit mobiler Informationssysteme in spontan vernetzten Umgebungen. Schon heute dienen mobile Geräte als Informationsquelle in jeder Lebenslage, zu jeder Zeit, an jedem Ort. Im Rahmen der Tätigkeit wird untersucht, inwieweit lokales Wissen, repräsentiert durch mobile Datenbestände, aber auch durch sensorisch erfasste Informationen, kontextsensitiv verwaltet, verbreitet und angefragt werden kann. Um diesen Anforderungen gerecht zu werden, bilden Dienste die Grundlage eines robusten und skalierbaren Frameworks.

1 Einleitung und Motivation

Im Graduiertenkolleg MuSAMA forschen wir insbesondere auf dem Gebiet der kooperativen Alltagsgegenstände, die ohne Einfluss des Menschen sinnvoll miteinander aggieren, um den Nutzer in seiner Handlung zu unterstützen. Besonders die Dynamik eines derartigen Ensembles steht im Mittelpunkt meiner Forschungsarbeit, die im Folgenden dargestellt wird. Damit ein Geräteensemble autonom auf die aktuellen Gegebenheiten reagieren kann, bedarf es Verfahren, die es ermöglichen, die Fähigkeiten und das Wissen einzelner Kommunikationspartner vermitteln zu können bzw. es in kooperativer Weise zu nutzen. Aus informationstechnischer Sicht bedarf es also einerseits einheitlicher Schnittstellen, die eingehende Anfragen in mobilen Szenarien verarbeiten können, und andererseits Techniken zur Aufbereitung (Indexierung) komplexer lokaler Datenbestände, deren Struktur über Dateien hinausgehen. Des weiteren werden Techniken konzipiert, die eine Vermittlung dieser Datenbestände in Peer-to-Peer ähnlichen Strukturen erlauben. Ein möglicher Ansatz zur Realisierung dieses Aufgabenspektrums können Dienste darstellen. Wie in (4) erläutert, so glauben auch wir, dass eine dienstorientierte Architektur eine „[...] geeignete Grundlage zur gemeinschaftlichen Nutzung von Resourcen in Ad-hoc-Netzen [...]“ sein kann. Ziel der Forschungsarbeit sind Strategien zur Lösung der genannten Herausforderungen, deren Effizienz und Effektivität mittels synthetischer Test und in realen Anwendungsszenarien nachzuweisen ist.

2 Service Discovery in MANETs

Mobile-ad-hoc-networks (MANETs) sind Kollektionen mobiler Geräte, die nicht über eine zentrale Verwaltungseinheit verfügen. Die Nutzbarkeit derartiger Strukturen aus einer anwendungsorientierten Sicht wird sehr diskutiert. Wie in (9) glauben auch wir, dass MANETs nützlich sind

„[...] if the members cooperate by efficiently searching and using services and resources offered by other devices.“ Komplexe Probleme oder Anfragen können anhand der aggregativen Nutzung von Diensten in einer kooperativen Art gelöst bzw. beantwortet werden. Die Suche nach derartigen Diensten ist dementsprechend ein sehr wichtiger Schritt. Service-Discovery benötigt oft sehr viel Zeit und verursacht einen hohen Netzwerk Verkehr. In Bluetooth Netzen z.B. bedarf die Suche folgender Schritte:

- Suche nach weiteren Bluetooth Knoten im Scan Bereich.
- Anfrage jedes einzelnen Gerätes nach angebotenen Diensten.
- Auswahl geeigneter Dienstgeber.

Service-Advertisments können benutzt werden, um Dienste nach außen zu repräsentieren, aber besonders in mobilen ad-hoc Netzwerken gibt es keine zentrale Einheit, die derartige Angebote verarbeiten bzw. speichern kann. In (9) wird eine Ring basierende Struktur vorgeschlagen, die versucht, physikalisch enge Knoten, die semantisch ähnliche Informationen (Dienste) anbieten, in einem Ring anzuordnen. Ein ausgezeichneter Knoten, der Service Access Point (SAP), sammelt alle Dienstinformationen dieses Ringes und dient hier sozusagen als Informationsquelle nach außen. Des Weiteren wird vorgschlagen auch SAPs in Ringen anzuordnen. Diese Technik hat den Vorteil, dass Service Anfragen nun von zentralen Einheiten beantwortet werden. Eine Auswahl geeigneter Dienstgeber ist somit in einer effizienten Art und Weise möglich. Wir sehen hier aber zwei Probleme:

- Einerseits bedarf dieser Lösungsvorschlag der sehr hohen Verfügbarkeit der ausgezeichneten Knoten. Gerade dies ist aber in spontanen Ensembles nicht gegeben.
- Andererseits werden zur semantischen Zuordnung der Geräte Ähnlichkeitsbegriffe benötigt, deren Bedarf in dieser Arbeit zwar aufgegriffen wird, aber kein Lösungsvorschlag geäußert wurde.

Auch wenn dieses Verfahren Schwächen offenbart, so zeigt sich hier wie auch in weiteren Arbeiten (3; 6; 1; 8) der Bedarf, lokale Dienste bzw. Informationen zu indexieren um einen effizienten Zugriff zu gewährleisten. Oft werden standartisierte Information-Retrieval Techniken benutzt (z.B. Terme in invertierten Listen zur Beschreibung der Metainformationen vorhandener Dienste). Aber auch Overlay-Strukturen (z.B. Content Adressable Networks (CAN) (12)), die versuchen, Wissen unabhängig von der physikalischen Netzwerk Struktur darzustellen, stehen im Fokus der Betrachtungen.

2.1 Terme als Grundlage der Definition von Ähnlichkeitsbegriffen in domainspezifischen Umgebungen

In unserem Ansatz versuchen wir die Vorzüge beider oben genannter Techniken zu kombinieren. Einerseits werden Terme benutzt, um Metainformationen wie Dienstbeschreibungen oder Diensttypen zu indexieren bzw. vergleichbar (ranking) zu machen. Des Weiteren müssen semantische Dienstinformationen wie Inputs/Outputs, Preconditions/Postconditions (11) ebenso im Index aufgenommen werden. Ausgehend von derartigen Indexstrukturen können nun Overlay Strukturen zur Dienst-Verwaltung (unabhängig von der darunterliegenden Netzwerkstruktur) formiert werden. Wie bereits aufgegriffen, bedarf es der Definition von Ähnlichkeiten, um Overlaystrukturen effizient zu nutzen. Semantische Ähnlichkeiten werden normalerweise durch Typ-Beziehungen gemessen wie in (11; 7). Die Ähnlichkeit von Diensten, die sich nicht in einer Typ-Subtyp Beziehung befinden, wird dabei normalerweise ignoriert, da derartige Dienste die gegebene Anfrage (spezifiziert durch Ein-/Ausgabetypen) nicht exakt beantworten können. Um dieser Situation entgegenzuwirken beschreiben wir einen Mechanismus der es erlaubt auch hier Ranking Entscheidungen treffen zu können.

2.2 Ontology basierendes Ranking

In domainspezifischen Umgebungen können Ontologien benutzt werden, um lokale Informationen oder Dienste zu ranken. Wie in (2) können baumartige Ontologie-Strukturen (OG) (Abbildung 1) oder Typ-Ontologien benutzt werden, um Dienste bzw. semantische Dienstinforma-

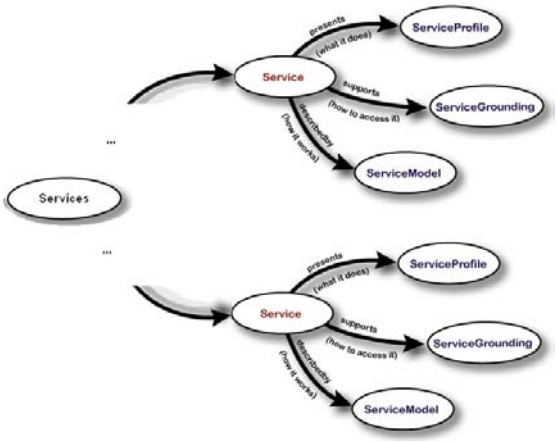

Abbildung 1: Ontology Baum

tionen(Inputs/Outputs) zu repräsentieren. Metainformationen wie Terme (vgl. 2.1) können nun auf diese Baumstrukturen gemappt werden, so dass jeder Term eine oder mehrere Knoten IDs (ni) der Form $1, \ldots, 1.X.Y.Z.$ zugewiesen bekommt. Falls ein Knoten mehrere Vorgänger hat, so können auch ihm mehrere IDs zugewiesen werden. Dienste lokal zu repräsentieren bedeutet nun:

- Beschreibung der Metainformationen durch Terme bzw. Beschreibung der semantischen Informationen durch Typen
- Mapping der Terme/Typen auf einen Ontologie Baum

Invertierte Listen oder spektrale Bloomfilter (5) können benutzt werden, um diese Indexinformationen zu speichern. Tabelle 1 zeigt derartige erweiterte invertierte Listen. Spektrale Bloomfilter werden nun nicht zur Zählung der Termhäufigkeit benutzt, sondern zur Zählung von Knoten IDs wie in Abbildung 2.

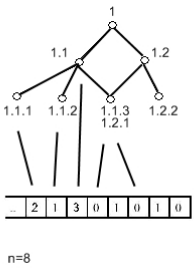

Abbildung 2: Bildung von Hash Funktionen für spektrale Bloomfilter anhand von Ontolgie Graphen (OG)

Sei n die Anzahl der Knoten in OG, so kann n als Arraygröße für einen spektralen Bloomfilter B benutzt werden. Falls Kreise existieren, muss n erhöht werden. Eine mögliche Hash-Funktion für B kann nun durch eine Linearisierung von OG, z.B. durch Nutzung der Inorder/Postorder Traversierung, gewonnen werden. Abbildung 2 zeigt diesen Prozeß. Abschließend müssen die lokalen Informationen noch innerhalb des Netzwerkes z.B. mittels Multicast verteilt werden.

Term	Paare von { ServiceID,{KnotenIds}}
abc	{12,{1.1.6,1.5.2.6}},{17,{1.1.6.5}}

Tabelle 1: Invertierte Liste für Ontologien

2.2.1 Ranking Prozeß

Werden Ähnlichkeiten anhand von Graphen berechnet, so stellen wir drei Fakten in den Vordergrund:

- d: Abstand zwischen zwei Knoten
- l: Level des ersten Eltern Knotens (der Knoten muss eine Elternknoten beider anderen Knoten sein und sein Level muss maximal sein)
- $maxl$: Das Maximallevel des Graphen
- a: Skalierungsfaktor

Baumähnliche Ontologien vermitteln ebenso den Fakt, dass Knoten, die tiefer innerhalb des Graphen angesiedelt sind (höheres Level), detailierter sind als Knoten die oberhalb angesiedelt sind. Basierend auf diesem Wissen kann ein Ranking nun folgendermaßen durchgeführt werden:

$$rank(n_1, n_2) \;=\; \frac{a * l}{d \cdot maxl}, a = 2.0 \tag{1}$$

Diese Berechnung ist sehr effizient aber auch sehr mächtig. Knoten die in einer Typ-Subtype beziehung stehen (Parent-Child) erhalten hier den Ranking Wert 1. Ähnliche Versuche werden in (10) beschrieben:

$$rank(n_1, n_2) \;=\; e^{(-0.2*d)} * \frac{(e^{(0.6*l)} - e^{(-0.6*l)})}{(e^{(0.6*l)} + e^{(-0.6*l)})} \tag{2}$$

Abbildung 3 vergleicht diese Ansätze. Es ist zu erkennen, dass die Verfahren auf ähnliche Weise reagieren, wenn l sich $maxl$ annähert. Aufgrund der Einfachheit des vorgestellten Lösungsansatzes kann dieser sehr gut auf mobilen Endgeräten mit beschränkter Rechenleistung zum Einsatz kommen.

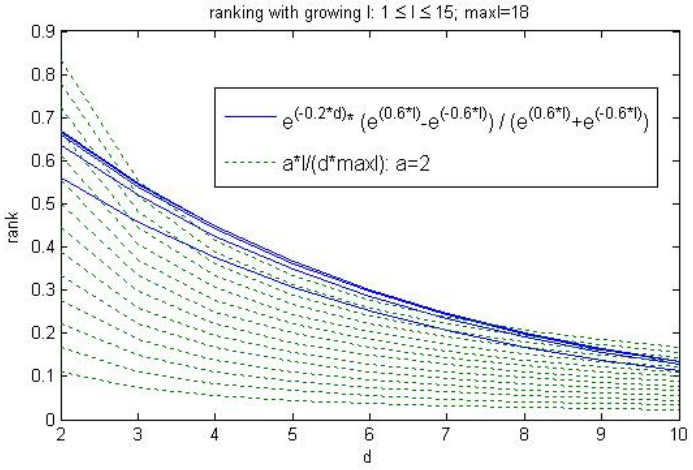

Abbildung 3: Vergleich zweier Graphbasierender Ranking Mechanismen

3 Zusammenfassung

Der Schwerpunkt der Arbeit liegt in der Verwaltung und Vermittlung von Diensten bzw. Informationen in spontan vernetzten mobilen Umgebungen als Basis eines intelligenten adaptiven Informationssystems. Im Rahmen der Dissertation beschreiben bisherige Tätigkeiten das Umfeld des Ähnlichkeitsbegriffes in domainspezifischen Umgebungen. Ontologien wurden als Mittel dargelegt, um Ranking Entscheidungen treffen zu können, die dem Aufbau von Overlay Strukturen zur Vermittlung lokaler Dienste hilfreich sind. In der weiteren Arbeit muss die Effektivität bzw. Effizienz dieser Techniken, z.B. mittels synthetischer Tests, in ubiquären Umgebungen in denen Geräte miteinander kooperativ agieren untersucht werden. Hierzu bedarf es der Konzeption und Umsetzung eines Frameworks zur intelligenten Dienstvermittlung z.B. in Bluetooth Netzen. Bisherige Arbeiten dienen als Grundlage zur Realisierung eines derartigen Frameworks.

Literatur

[1] Adjie-Winoto, W. ; Schwartz, E. ; Balakrishnan, H. ; Lilleyr, J. : The design and implementation of an intentional naming system. In: *17th ACM Symposium on Operating Systems Principles (SOSP 99),Published as Operating Systems Review*, 1999

[2] Burstein, M. ; Hobbs, J. ; Lassila, O. ; Mcdermott, D. ; Mcilraith, S. ; Narayanan, S. ; Paolucci, M. ; Parsia, B. ; Payne, T. ; Sirin, E. ; Srinivasan, N. ; Sycara, K. ; Martin, D. (Hrsg.): *OWL-S: Semantic Markup for Web Services.* `http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/`. Version: 22 November 2004, November 2004

[3] Crespo, A. ; Garcia-Molina, H. : Routing Indices For Peer-to-Peer Systems. In: *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2002

[4] Domagalski, R. ; König-Ries, B. : Möglichkeiten der Anfragebearbeitung in mobilen Ad-hoc-Netzwerken. In: *Grundlagen und Anwendungen mobiler Informationstechnologie*, 2004, S. 41–52

[5] Eisenhardt, M. ; Müller, W. ; Henrich, A. : Spektrale Bloom-Filter für Peer-to-Peer Information Retrieval. In: *GI Jahrestagung (2)*, 2004, S. 44–48

[6] Joseph, S. : NeuroGrid: Semantically Routing Queries in Peer-to-Peer Networks. In: *In Proceedings of the International Workshop on Peer-to-Peer Computing*, 2002

[7] Klein, M. ; König-Ries, B. : Multi-Layer Clusters in Ad-Hoc Networks - An Approach to Service Discovery. In: *Proceedings of the First International Workshop on Peer-to-Peer Computing (Co-Located with Networking 2002).* Pisa, Italy, May 2002, S. 187–201

[8] Klein, M. ; König-Ries, B. ; Obreiter, P. : Lanes – A Lightweight Overlay for Service Discovery in Mobile Ad Hoc Network. In: *3rd Workshop on Applications and Services in Wireless Networks (ASWN2003). Berne, Swiss*, 2003

[9] Klein, M. ; König-Ries, B. ; Obreiter, P. : Service Rings – A Semantical Overlay for Service Discovery in Ad Hoc Networks. In: *The Sixth International Workshop on Network-Based Information Systems (NBIS2003), Workshop at DEXA 2003, Prague, Czech Republic*, 2003

[10] Li, Y. ; Bandar, Z. A. ; McLean, D. : An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources. In: *IEEE Transactions on Knowledge and Data Engineering* 15 (2003), Nr. 4, S. 871–882. – ISSN 1041–4347

[11] Mokhtar, S. B. ; Kaul, A. ; Georgantas, N. ; Issarny, V. : Efficient Semantic Service Discovery in Pervasive Computing Environments. In: *Middleware 2006*, 2006, S. 240–259

[12] Ratnasamy, S. ; Francis, P. ; Handley, M. ; Karp, R. ; Schenker, S. : A scalable content-addressable network. In: *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications.* New York, NY, USA : ACM, 2001. – ISBN 1–58113–411–8, S. 161–172

RADIX – Steuerung und Verwaltung heterogener Dokument- und Prozessstrukturen in ebenenübergreifenden GeoGovernment-Umgebungen

Stefan Audersch, Guntram Flach
Zentrum für Graphische Datenverarbeitung e.V.
Joachim-Jungius-Str. 11, 18059 Rostock
{stefan.audersch, guntram.flach}@rostock.zgdv.de

Jan-Christian Kuhr, Jan Pretzel
GECKO mbH
Deutsche-Med-Platz 2, 18057 Rostock
{jku, jpr}@gecko.de

Abstract: In diesem Beitrag wird ein universeller, Web Service-orientierter Ansatz vorgestellt, der verschiedene Technologien aus den Bereichen GeoGovernment, Workflow Management und Wissensmanagement miteinander verknüpft, um eine flexible, semantikgestützte Workflowsteuerung und Dienstekomposition in organisationsübergreifenden GeoGovernment-Umgebungen zu realisieren. Die Ansätze werden am Beispiel Bauleitplanung vorgestellt und wurden gemeinsam mit der Hansestadt Rostock und weiteren Partnern im Rahmen des RADIX-Projektes konzeptioniert.

1 Einleitung

Die flexible Automatisierung von institutions- und ebenenübergreifenden eGovernmentprozessen stellt eine neue Herausforderung für den Einsatz von IT-Technologien hinsichtlich Komplexität, Integration, Rechtsverbindlichkeit und Prozess-Steuerung dar. SOA-, OGC- und Semantic Web-Technologien bilden eine mögliche Grundlage zur Konzeptionierung eines universellen Ansatzes zur Schaffung interoperabler GeoGovernment-Services, die semantisch gesteuert die Integrations- und Wissensprozesse im Sinne dynamischer Verwaltungs-Services realisieren.
Das am ZGDV Rostock entwickelte VESUV-Framework [AFS07] [Fr05] [FPR06] stellt grundlegende Funktionen für die Ämter-übergreifende Kommunikation bereit (Assistenz, Workflow-Steuerung) und wurde in vorausgehenden Arbeiten im Anwendungskontext der Hansestadt Rostock entwickelt.
Ausgehend von der Workflow-Verarbeitung sowie der Verwendung von Metadaten, Ontologien und GIS-Systemen werden unterschiedliche Ansätze für die Nutzung von semantischen Informationen entwickelt, um eine Unterstützung der Verwaltungsverfahren zu ermöglichen. Ein Schwerpunkt dabei ist die adäquate Integration von geograpischen Informationen in ein prozessgesteuertes Dokumentenmanagement auf der Basis von XPlanung [ErR06] [DeK07]. Dabei sollen die geographischen Daten ebenfalls wie Dokumente aufgefasst werden. Die technologische Umsetzung basiert auf Standards wie XPlanung, BPEL, OGC (KML) sowie W3C (RDF, OWL).
Die Ansätze werden am Beispiel Bauleitplanung vorgestellt und wurden gemeinsam mit der Hansestadt Rostock und weiteren Partnern im Rahmen des RADIX-Projektes konzeptioniert.

2 Anwendungsszenario und Anforderungen

Die Erstellung von Bauleitplänen erfordert das Zusammenwirken verschiedener Akteure (Planer – Kommune, Kommune – Landkreis, Planer – Landkreis, Landkreis – Land). Notwendig sind ein verlustfreier Datenaustausch zwischen den verschiedenen Planungsebenen und den unterschiedlichen öffentlichen und privaten Planungsakteuren während des Planungsprozesses sowie die Bereitstellung unterschiedlicher Services im Verwaltungshandeln "Planen und Bauen". Der anvisierte RADIX-Ansatz soll helfen, den Planungsprozess horizontal zwischen benachbarten, aneinandergrenzenden Städten effizient aufeinander abzustimmen. Abbildung 1 zeigt den für die Bauleitplanung in Rostock

zugrunde liegenden Verwaltungsprozess. Ziel des Projektes ist es, der Wirtschaft (z.B. zur Unterstützung der regionalen Wirtschaftsförderung oder zur Unterstützung des Standortmarketings), den beteiligten Planungsakteuren (potentielle Investoren, Immobilienwirtschaft, Planungsbüros), anderen Fachbehörden sowie sonstigen Trägern öffentlicher Belange (TÖBs) die digitalen Bauleitpläne, begleitende Dokumente (Stellungnahmen) oder Metainformationen durch ein geeignetes Frontend, als standardisierte Datensätze oder als Web-Service zur Verfügung zu stellen. Das Anwendungsszenario „Bauleitplanung“ ist für die Umsetzung leittragend, jedoch sollen auch andere Verwaltungsprozesse durch das System unterstützt werden.

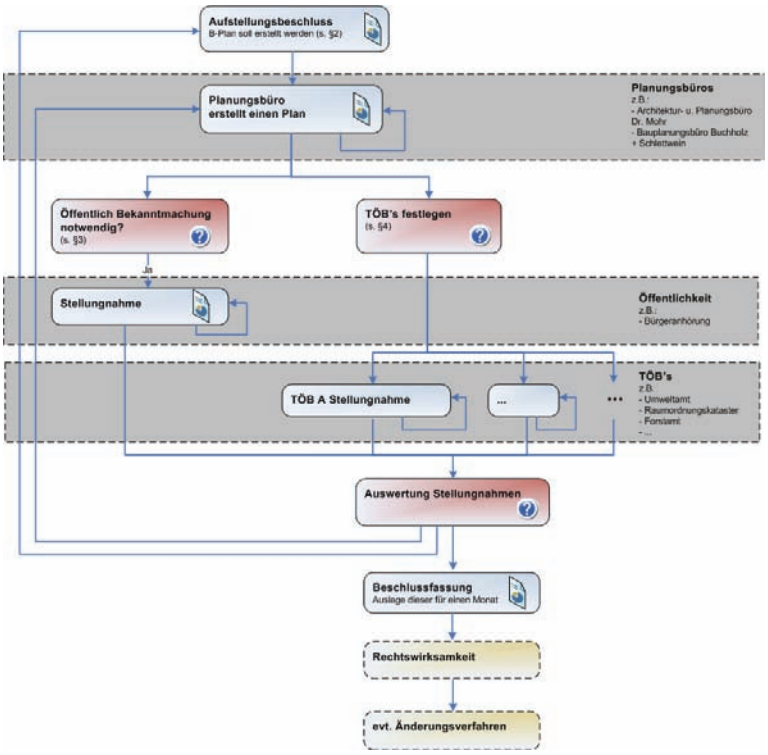

Abbildung 1: Prozess „Bauleitplanung“

Untersucht man verschiedene ämterübergreifende bzw. ebenenübergreifende Verwaltungsprozesse, so ergeben sich verschiedene Gemeinsamkeiten. Die Prozesse sind zumeist dokumentengetrieben, bei denen während der Prozessabarbeitung eine Vielzahl von Dokumenten von verschiedenen Ämtern und Institutionen erstellt, eingebunden oder aktualisiert werden. Neben dem Umgang mit Dokumenten werden an entsprechenden Prozessschritten Entscheidungen getroffen, die den weiteren Prozessverlauf beeinflussen. Allgemein lassen sich aus der Analyse der verschiedenen Verwaltungsprozesse die folgenden Prozessschritte identifizieren:

- Erstellen bzw. Hinzufügen neuer Dokumente: Bei der Erstellung neuer Dokumente werden oftmals Informationen aus vorhandenen Dokumenten verwendet. So werden z.B. die Anschrift eines Antragstellers oder auch Punkte aus einer Stellungnahme in einen Bescheid übernommen.
- Änderung von Dokumenten: Vorhandene Dokumente werden innerhalb des Verwaltungsprozesses inhaltlich überarbeitet. Im Rahmen der Bauleitplanung wird z.B. der Plan iterativ erstellt, wobei ein Planungsbüro einen Planentwurf anfertigt und dieser Stand dann immer wieder mit den Anforderungen des Amtes für Stadtplanung abgeglichen wird.
- Treffen von Entscheidungen: Innerhalb einiger Prozessschritte ist es notwendig, Entscheidungen zu treffen. Hierbei handelt es sich in der Regel um die Auswahl von bestimmten vordefinierten Optionen, wie z.B. die Festlegung, welche Fachabteilungen für einen Bebauungsplan eine Stellungnahme anzufertigen haben. Die Entscheidungen haben oftmals Einfluss auf den weiteren Prozessverlauf. Die Auswahl solcher Optionen kann nicht unbedingt automatisch erfolgen, jedoch kann der Fachanwender bei der Entscheidung durch entsprechende Hilfestellungen unterstützt werden.

- Anzeigen und Ausdrucken von Dokumenten: Im Rahmen des Verwaltungsprozesses greifen verschiedene Institutionen auf die eingestellten Dokumente zu, um diese zu betrachten bzw. auszudrucken.
- Zugriff auf externe Systeme: In bestimmten Prozessschritten ist es notwendig, mit anderen externen Fachanwendungen zu kommunizieren. So kann z.B. ein Bezahlvorgang initiiert oder ein endgültiger Bebauungsplan in einem externen GIS-System abgelegt werden.

Das System soll dem Anwender eine adäquate Arbeitsgrundlage bieten, die es ihm erlaubt, mit den Dokumenten des Verfahrens zu arbeiten und Entscheidungen zu treffen. Dabei erhält der Fachanwender eine optimale Sicht auf den Gesamtprozess mit allen Dokumenten und sonstigen Informationen. Unterstützt werden verschiedene Dokumentformate, wie z.B. Text-, Graphik- oder Geodaten. Die einzelnen Aufgaben innerhalb des Verwaltungsprozesses werden durch einen Workflow gesteuert, der jedoch auch die Flexibilität bietet, die speziell bei Sonderfällen benötigt wird. Allgemein lassen sich die folgenden Anforderungen an das Gesamtsystem stellen:

- Bei den Verfahren handelt es sich um dokumentgetriebene Prozesse. Es soll möglich sein, mit einer sehr hohen Flexibilität Dokumente einzufügen, zu ändern oder darauf zuzugreifen. Das heißt, dass Dokument nicht nur zu einem bestimmten Prozessschritt in das System eingestellt werden können, sondern bereits im Vorfeld oder zu einem späteren Zeitpunkt.
- Bei der Bearbeitung der Verwaltungsaufgaben liegt ein grundlegender Prozess (Workflow) dahinter. Dieser basiert auf gesetzlichen Grundlagen und lässt sich eindeutig spezifizieren und technisch umsetzen. Dabei ist jedoch zu beachten, dass eine Vielzahl von Sonderfällen existiert und dass die Bearbeitung sich nicht unbedingt an einen statisch definierten Prozess hält. Daher sollte sich die Bearbeitung des Verfahrens zwar an der Prozessstruktur orientieren, dem Fachanwender aber auch die mögliche Flexibilität bieten, um jeweilige Sonderfälle bearbeiten zu können.
- Der Prozessverlauf eines Verwaltungsverfahrens hat gegebenfalls Einfluss auf weitere Entscheidungsprozesse. Es ist daher notwendig, den Prozess transparent für den Fachanwender darzustellen mit den Informationen, welche Schritte bereits abgeschlossen wurden und welche Punkte noch offen stehen.
- Um verschiedene Verwaltungsverfahren abbilden zu können, ist es notwendig, unterschiedliche Dokumenttypen zu unterstützen. Hierbei ist neben typischen Text- und Graphikformaten speziell für den Bereich Bauleitplanung auch die Unterstützung von geographischen Daten notwendig. Dabei ist zu beachten, dass unterschiedliche Formate (GML, Shape, KML, MapInfo, XPlanung) existieren und eine Betrachtung der Geodaten in der Regel mit der Integration in das jeweilige GIS-System einhergeht.
- Im Rahmen von Verwaltungsprozessen spielen Rechte bzw. Sichten (Zweckbindung) und Signaturen (beglaubigte Ersteller) eine wichtige Rolle und müssen demzufolge in dem System abgebildet werden.
- Bei der Bearbeitung von Verwaltungsprozessen werden an verschiedenen Stellen Entscheidungen vorgenommen. Fachanwender können durch geeignete Hilfestellungen unterstützt werden. Grundlage für die Entscheidungsunterstützung können Informationen aus dem Dokumentenbestand, der Prozessverlauf oder Daten von externen Systemen sein.
- Das System soll die Bearbeitung von Verwaltungsprozessen ämter- bzw. ebenenübergreifend unterstützen. Hierfür sind geeignete Schnittstellen z.B. in Form von Web Services umzusetzen. Für eine übergreifende Arbeit mit dem System bietet sich zudem ein webbasierter Zugang in Form eines Webfrontends an.

3 Lösungsansatz

Ziel des RADIX-Projektes ist es, ein System zu entwickeln, welches die genannten Anforderungen adäquat umsetzt, um die Fachanwender beim Prozess der Bauleitplanung zu unterstützen. Um ebenfalls andere Verwaltungsprozesse zu unterstützen, fließen deren Anforderungen zusätzlich in die Entwicklung mit ein.
Im Nachfolgenden werden verschiedene Lösungsaspekte erläutert, die eine Grundlage für das Gesamtsystem darstellen.

Datenstruktur

Grundlage für die Verwaltung der im Prozess anfallenden Dokumente ist eine geeignete Struktur zur Speicherung der Daten. Berücksichtigte Konzepte hierfür sind die Unterstützung verschiedener Formate, Versionierung, Derivate, Nachvollziehbarkeit, Beziehungen, Archivierbarkeit und weitere. Abgebildet werden diese in einem XML-Dokument, welches jeweils die Dokumente und Metadaten eines Verwaltungsprozesses beinhaltet. Die Struktur des XML-Dokumentes wurde an XDomea und XPlanung angelehnt. Grundlegende Erweiterungen sind:

- Dokumentcontainer, die eine Zusammenfassung mehrerer Primärdokumente erlauben, wie es z.B. beim Bauleitplan notwendig ist
- Prozessschritte, die eine Nachvollziehbarkeit des Verwaltungsprozessablaufes bzgl. des Workflows erlauben,
- Ordnerstrukturen, die eine Einsortierung von Dokumenten in mehrere Ordnerstrukturen für eine adäquate Darstellung des Dokumentenbestandes erlauben, sowie
- die Unterstützung weiterer Geodatenformaten neben XPlanGML.

Aufgabe des DocumentServer (siehe Abbildung 2) sind die geeignete Speicherung der Daten sowie die Bereitstellung von Änderungsfunktionen (Native, XUpdate), Anfragefunktionen (XQuery, XPath) und Transformationsfunktion (XSLT, Native).

Workflow

Der Workflow strukturiert und steuert den Verwaltungsprozess. Eine detaillierte Spezifikation des Workflows (siehe auch Abbildung 1) erfolgt auf der Grundlage von CPN Tools [RWL03]. Bei der iterativen Analyse mit den Fachanwendern wird das BRITNeY [RWL03] zur Animation und Simulation des Workflows herangezogen. Die Umsetzung erfolgt in BPEL.

Architektur

Zentraler Kern des Systems ist der ApplicationServer. Die hier abgebildeten Businessobjekte und die darauf aufbauende Logik erlauben u.a. die Steuerung und Überwachung des Verwaltungsprozesses, bieten Funktionalitäten und Daten für die Entscheidungsunterstützung und ermöglichen den Zugriff auf die Verfahrensdokumente. Direkt daran angebunden ist der WebServer, der die Schnittstelle zum Fachanwender in Form eines geeigneten GUI abbildet. Hierüber hat der Anwender entsprechend seinen Rechten und Kompetenzen Einsicht auf die verschiedenen Verwaltungsprozesse und kann durch Aktionen den Prozess vorantreiben. Erfolgt die Arbeit nicht über den WebServer, so bietet der ApplicationServer Schnittstellen in Form von WebServices und JMS. Die Steuerung der Prozesse erfolgt über die Workflowengine. Für deren Kopplung mit externen Systemen wird das JCoupling-Framework [AVD05] verwendet. Zugriff, Speicherung und Manipulation die Verfahrensdokumente bildet der DocumentServer ab.

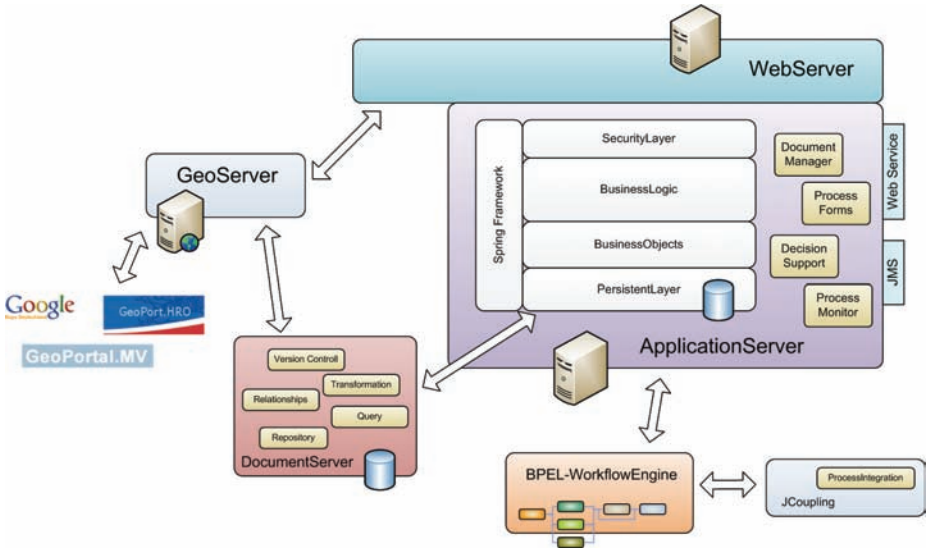

Abbildung 2: Architektur

Integration geographischer Daten

Im Gegensatz zu allen anderen Dokumenten werden geographische Dokumente (z.B. Bebauungsplan) in der Regel im Kontext zu anderen geographischen Daten (z.B. Orthofoto oder ALK) angezeigt. Hierfür ist es notwendig, spezielle Geodienste (WMS, WFS) in das System zu integrieren, um eine einfache Anbindung an unterschiedliche GIS-Systeme, wie sie z.B. von den unterschiedlichen Ämtern und Institutionen verwendet werden, zu ermöglichen. Dadurch können beispielweise beim Bauleitplanverfahren die Pläne direkt in den externen GIS-Anwendungen angezeigt oder über den WebServer des Systems im Kontext zu externen Kartendaten dargestellt werden.

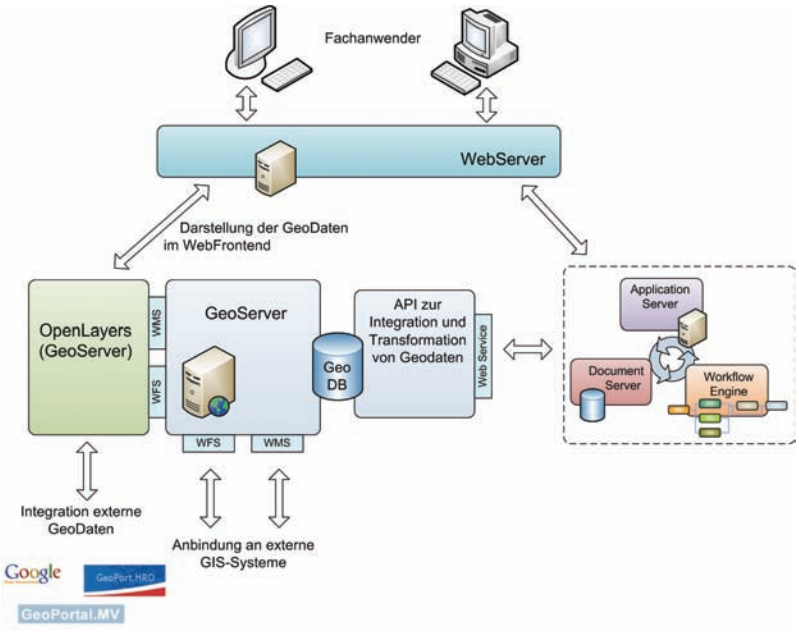

Abbildung 3: Architektur zur Integration von geographischen Daten

Literaturverzeichnis

[AFS08] Audersch, S., Flach, G., Schultz, J.: Digitale Assistenz in komplexen, ämterübergreifenden Verwaltungsprozessen, Multikonferenz Wirtschaftsinformatik, München, 2008

[AFS07] Audersch, S., Flach, G., Schultz, J.: Wissensbasierte Assistenz in prozessorientierten eGovernment-Umgebungen, In Proc.: Professionelles Wissensmanagement - Erfahrungen und Visionen, Potsdam, 2007

[AVD05] Aldred, L., van der Aalst, W., Dumas, M., ter Hofstede, A.H.M.: On the Notion of Coupling in Communication Middleware. In Proceedings On the Move to Meaningful Internet Systems - 7th International Symposium on Distributed Objects and Applications (DOA) 3761, Agia Napa, Cyprus, 2005

[DeK07] Debold, J., König, H.: Integriertes Baileitplanverfahren nach dem Modell XPlanung in Nordwestmecklenburg, 2. Rostocker E-Government Forum, Rostock-Warnemünde, 2007

[ErR06] Eichhorn, T., Raser, F.: Spezifikationsbericht – XPlanung, Media@Komm-Transfer (BMWI), 2006

[FPR06] Flach, G.; Pakulat, A.; Rust, M.: Ontologie-getriebene Metadaten-Syndikation zur Unterstützung von GeoGovernment-Anwendungen. Symposium für Angewandte GeoInformatik, Salzburg, 2006.

[Fr05] Franz, A.: Semantik-gestützte Workflow-Steuerung und Dienste-Komposition in organisationsübergreifenden eGovernment-Umgebungen. Diplomarbeit, Universität Rostock, 2005.

[RWL03] Ratzer A., Wells L., Lassen H.M., Laursen M., Qvortrup J.F., Stissing M.S., Westergaard M., Christensen S., Jensen K.: CPN Tools for Editing, Simulating, and Analysing Coloured Petri Net. In Proceedings 24th International Conference, ICATPN 2003 Eindhoven, 2003

An XML Database as Filesystem in Userspace

Alexander Holupirek Marc H. Scholl

University of Konstanz
Dept. of Computer & Information Science
Box D 188, 78457 Konstanz, Germany
holupire|scholl@inf.uni-konstanz.de

Abstract

More and more data is stored and converted to XML, and with the upcoming ability of databases to handle semi-structured data efficiently it seems reasonable to rethink database-supported filesystems. This paper describes an early prototype of a filesystem implementation in userspace using a database with XML/XQuery support as back-end. Traditionally files are roughly classified as either text or binary. We add XML as a third type and expose formerly hidden content of files to the system and the user with both, its structure and content. The implementation establishes a link between database and operating system and allows the use of XML processing languages, such as XPath and XQuery, on the data. Since the database is mounted as a conventional filesystem by the operating system kernel, access via the established (virtual) filesystem interface as well as database enhanced access to the same data is provided. The implementation uses MonetDB/XQuery, a well-known relational database system with XQuery and XQuery Update Facility (XQUF) support, which integrates the Pathfinder XQuery compiler [2] and the MonetDB kernel [1].

1 Motivation

Since the beginning of database management systems there is a desire to store any data in a database and have it ready to be queried. Several industrial and research efforts, such as WinFS or the Be Filesystem have been undertaken to push the filesystem into a database. None made it to technical production quality. Offshoots, like Microsoft's Instant Search or Apple's Spotlight Architecture, however, can be found in any of the recent operating system variants, and a users' demand for products helping to find relevant content can be derived from the increasing popularity of Desktop Search Engines, such as Google's Desktop Search or Yahoo's X1. While these tools offer a smarter way to access personal information stored in the filesystem, the keyword driven search approach, as it is used by today's search engines, is inherently limited. An additional support of database query languages would be preferable.

2 Relational XML storage as jump start for database filesystems

Despite the fact that several database-driven filesystem attempts have already failed, the advent of XML brought some significant enhancements to RDBMS which inspired us to dare another attempt. A major problem of storing files in a DBMS (apart from using BLOBs) has been the basic necessity to provide a schema first. With an unmanageable amount of file formats this is an impossible mission. Schema-oblivious storage techniques rendered it possible for XML data to be stored in the database without previous knowledge of its interior structure[1].

[1] An additional XML Schema specification for the file type may be of advantage to formulate queries against the document, but is not mandatory.

Since more and more applications store their own data as XML (OOXML, OpenDocument-Format), those files are already prepared to be handled with database technology. From our point of view, these documents are nothing but serialized database instances. In consequence, they are not only stored as plain text, but directly shredded[2] into the DBMS. Legacy application may still process them conventionally by serializing them to their textual representation, however they are ready to be processed using XPath/XQuery in direct communication with the database. Applications may access their data in a more granular way with direct support from the RDBMS.

Imposed by the tree-based XML model relational storage and processing techniques for hierarchically structured data evolved and RDBMSs have learned to handle tree-shaped data. This is of direct benefit as the hierarchic nature of filesystem organization can now consistently be mapped to the relational storage and leverage the associated algorithms (an elaborate discussion of relational XML storage and algorithms can be found in [13, Chap. 2]).

In the following we describe an early implementation stage of how a database with XML/XQuery support can be used as a userlevel filesystem. The paper is structured as follows: The implementation is using the MonetDB/XQuery database system and the Filesystem in USErspace (FUSE) framework. These will be introduced shortly in Section 3 and 4. Section 5 describes our system architecture. After a short description of what we like to demonstrate during the workshop, we finish with a summary.

3 MonetDB/XQuery and the Pathfinder XQuery compiler

Pathfinder is an XQuery compiler implementation backed by relational database kernels. It is based on the transformation of XML document collections into relational tables [4] and emits relational algebra plans which consequently can be executed on any RDBMS [6]. In combination with the MonetDB kernel [1] it results in the open source MonetDB/XQuery system, which has proven to be one of the fastest and most scalable XQuery implementations available today [3]. RDBMS incorporate the knowledge and research efforts of years. They have proven to store and query large data sets efficiently and can be considered as mature technology. Using the power of relational database technology in combination with the recent findings in the domain of semi-structured data processing make such systems a promising choice for the implementation of a userlevel filesystem with query capabilities. Besides, the Pathfinder XQuery compiler appears particularly well suited due to the following aspects:

Scalability. It is able to handle huge amounts (up to 10GB) of XML data in an efficient manner.

Targets multiple back-ends. The Pathfinder XQuery compiler has already been enhanced by a code generator that emits SQL. This code generator targets off-the-shelf relational database systems (e.g., DB2) and turns them into efficient and scalable XQuery processors [6, 5].

Integrates a XML information retrieval system. PF/Tijah [8] is a text search system which is integrated with the Pathfinder compiler. It includes out-of-the-box solutions for common tasks such as index creation and result ranking. We expect that to be of great benefit for the stored textual and XML files.

4 Filesystems in USErspace (FUSE)

FUSE is a framework for implementing filesystems outside of the kernel in a separate protection domain in a user process. It was first implemented for and integrated into the Linux operation system kernel [12]. There are reimplementations for the Mac OS X [11], FreeBSD [7], and NetBSD [10, 9] kernels. The FUSE library interface closely resembles the in-kernel virtual file

[2] A terminus technicus used to indicate the conversation of XML in its textual representation to an internal format used by the database. Read it as “import”.

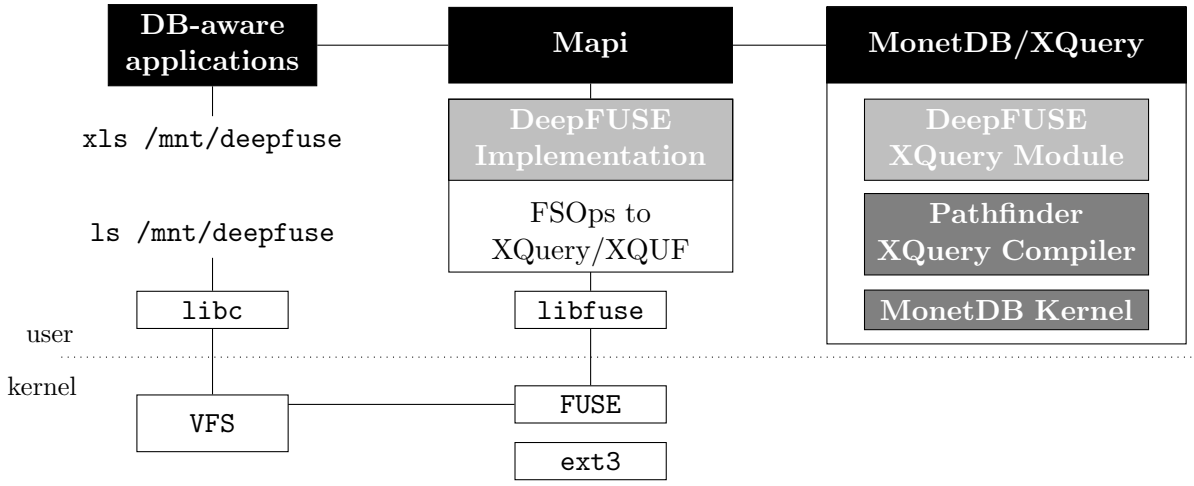

Figure 1: Establishing a link between OS and DBMS by implementing the DBMS as filesystem in userspace. Conventional as well as database supported access to the filesystem data is achieved.

system interface. The userlevel implementations are able to register function callbacks which get executed once a corresponding request is issued by the OS kernel. The FUSE kernel module and the FUSE library communicate via a special file descriptor: `/dev/fuse`. This file can be opened multiple times, and the obtained file descriptor is passed to the mount syscall, to match up the descriptor with the mounted filesystem. Our implementation is build on top of the `libfuse` library as depicted in Figure 1.

5 System Architecture

From a user's perspective, the system provides two access paths to the filesystem. Conventional/ legacy access for any application can be achieved as exemplified by the `ls` command. The (virtual) filesystem operations relevant for the listing of directory contents are looped back into userspace and captured by the functions registered with the callback interface of the `libfuse` library. The DeepFUSE implementation[3] is responsible for translating the filesystem operations into according XQuery/XQUF requests. The MonetDB/XQuery server offers a well-defined and easy to use API (encapsulated in the Mapi library) to leverage its functionality. DeepFUSE communicates as a client with a running server using a database driver and a textual protocol. The most frequently used functions called by the FUSE implementation have been encapsulated in an XQuery Module. This is most advantageous as it allows MonetDB/XQuery to prepare a query plan for those functions. A much faster execution is the consequence.

The userlevel filesystem implementation expects to operate on a filesystem XML representation valid against a W3C XML Schema Definition. A DeepFUSE XML instance is, of course, a (possibly empty) collection of files. Following the UNIX tradition there are block and character special, directory, fifo, symbolic link, socket and regular file types. Filesystem metadata, the `stat` information (access time, protection mode, file size ...) and any information relevant to operate a traditional filesystem is placed in the `http://www.deepfs.org/2008/fs` namespace. This namespace encapsulates the information needed to operate the database as a filesystem in userspace. It is stored in a separate XML collection inside the MonetDB/XQuery system.

The second access path (as depicted with the `xls /mnt/deepfuse` command) goes along

[3]We call it DeepFUSE, because operations on the file hierarchy do not necessarily end at the file level, but can continue on the file's inherent structure. The navigation along the file hierarchy in our implementation is basically the same as the navigation inside the XML files.

the conventional database interface. The filesystem data is stored in the database and functions implemented in the XQuery module are ready to be used. However, this is not the crucial point. We expect applications which use XML as their storage format to query their data for partial content or to only update "dirty" nodes instead of reading, processing and writing back complete XML files in their textual format as it is the case when stored in filesystems.

6 Demonstration during the Workshop

During the workshop we would like to give a demonstration of the system showing how filesystem operations propagate into requests in the DBMS. In the process we will work on the shell as a common UNIX user and issue some basic commands. Simultaneously, we will trace the resulting calls in the VFS and examine the corresponding generated queries.

7 Summary

While filesystems provide an easy and well-understood interface to the data, they lack important and demanded features like the ability to query the data. Applications such as desktop search engines or personal information management tools often use a keyword-based search approach which undeniably provides a user-friendly information discovery mechanism. There is no need to know the structure or format of the data let alone a query language. Ideally however, all retrieval strategies, i.e., with no, partial or full knowledge about structure, format and content of the data, should be supported. Existing tools usually index plain text content and thus have started to break the file content out of its black box. That means, while traditional filesystems do not care about the content of a file, the supporting userlevel applications which provide the missing search and retrieval functionality take it into account. A consistent further development is to provide means to expose the inherent file structure to allow content and structure queries.

RDBMSs are the right choice to query vast amounts of structured data. Data stored in the filesystem, however, is very heterogenous. With the rise of XML, the database community has been challenged by semi-structured data processing, enhancing their field of activity. Ongoing research efforts made RDBMSs, such as MonetDB/XQuery, capable of operating on tree-shaped data. As more and more applications save their data as XML, those files are ready to be directly saved into a DBMS. To go through with the concept, the file hierarchy tree along with its metadata is to be stored in the database in the same fashion.

We contribute an implementation that establishes the link between OS and DBMS. The DBMS is finally mounted as a filesystem by the OS and offers its data to arbitrary applications via the established (virtual) filesystem interface as well as through its database interface. As such we demonstrate the possibility of providing legacy filesystem access while storing the data in the database and have it ready to be queried.

8 Acknowledgment

This research is supported by the German Research Council (DFG) Research Training Group GK-1042 "Explorative Analysis and Visualization of Large Information Spaces".

References

[1] Peter A. Boncz. *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*. Ph.d. thesis, Universiteit van Amsterdam, Amsterdam, The Netherlands, May 2002.

[2] Peter A. Boncz, Torsten Grust, Maurice van Keulen, Stefan Manegold, Jan Rittinger, and Jens Teubner. Pathfinder: XQuery - The Relational Way. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *VLDB*, pages 1322–1325. ACM, 2005.

[3] Peter A. Boncz, Torsten Grust, Maurice van Keulen, Stefan Manegold, Jan Rittinger, and Jens Teubner. MonetDB/XQuery: A Fast XQuery Processor Powered by a Relational Engine. In Surajit Chaudhuri, Vagelis Hristidis, and Neoklis Polyzotis, editors, *SIGMOD Conference*, pages 479–490. ACM, 2006.

[4] Torsten Grust. Accelerating XPath Location Steps. In Michael J. Franklin, Bongki Moon, and Anastassia Ailamaki, editors, *SIGMOD Conference*, pages 109–120. ACM, 2002.

[5] Torsten Grust, Manuel Mayr, Jan Rittinger, Sherif Sakr, and Jens Teubner. A SQL:1999 Code Generator for the Pathfinder XQuery Compiler. In Chee Yong Chan, Beng Chin Ooi, and Aoying Zhou, editors, *SIGMOD Conference*, pages 1162–1164. ACM, 2007.

[6] Torsten Grust, Sherif Sakr, and Jens Teubner. XQuery on SQL Hosts. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *VLDB*, pages 252–263. Morgan Kaufmann, 2004.

[7] Csaba Henk. FUSE for FreeBSD. `http://fuse4bsd.creo.hu/`.

[8] Djoerd Hiemstra, Henning Rode, Roel van Os, and Jan Flokstra. PF/Tijah: text search in an XML database system. In *Proceedings of the 2nd International Workshop on Open Source Information Retrieval (OSIR)*, 2006.

[9] Antti Kantee. puffs - Pass-to-Userspace Framework File System. In *Proceedings of the Asia BSD Conference (AsiaBSDCon) 2007*, 2007.

[10] Antti Kantee and Alistair Crooks. ReFUSE: Userspace FUSE Reimplementation Using puffs. In *Proceedings of the 6th European BSD Conference (EuroBSDCon)*, 2007.

[11] Amit Singh. A FUSE-Compliant File System Implementation Mechanism for Mac OS X. `http://code.google.com/p/macfuse/`.

[12] Miklós Szeredi. Filesystem in USErspace. `http://fuse.sourceforge.net/`.

[13] Jens Teubner. *Pathfinder: XQuery Compilation Techniques for Relational Database Targets*. Ph.d. thesis, Technische Universität München, Munich, Germany, Oct 2006.

Implementierung einer RDF-Datenstromverarbeitung mit SPARQL

André Bolles, Marco Grawunder

Universität Oldenburg
Department für Informatik, Abteilung Informationssysteme
[andre.bolles|marco.grawunder]@uni-oldenburg.de

Zusammenfassung

Diese Arbeit betrachtet die Verarbeitung von RDF-Datenströmen. Im Gegensatz dazu betrachten viele andere Arbeiten lediglich relationale oder XML-Datenströme (vgl. [ABW03, Krä07] u. a.). Zur Verarbeitung von Anfragen auf RDF-Datenströmen ist ein Entwurf zur Erweiterung bestehender Anfragekonzepte notwendig. Mit Stream-SPARQL wurde in Oldenburg eine Möglichkeit geschaffen, basierend auf der W3C-Empfehlung SPARQL, Anfragen auf RDF-Datenströmen zu verarbeiten. Hierzu wurde SPARQL in vier Schritten erweitert: Spracherweiterung, Definition der Semantik von Algebra-Operatoren auf Datenströmen, Definition möglicher Operatorrealisierungen und Erzeugung von Anfrageplänen aus SPARQL-Datenstromanfragen. Die Umsetzung dieser SPARQL-Erweiterung erfolgte dabei im Odysseus-Framework, einem Datenstrommanagementsystem der Abteilung Informationssysteme der Universität Oldenburg. Diese Arbeit stellt die Struktur der SPARQL-Erweiterung in Odysseus vor und erläutert, wie dardurch die Verarbeitung von RDF-Datenströmen ermöglicht wird.

1 Einleitung

Datenströme spielen heute in vielen Anwendungen eine Rolle. Beispielsweise werden moderne Windenergieanalagen (WEA) mit Sensoren ausgestattet, die über Datenströme Informationen über den Zustand einer WEA bereitstellen. Datenströme sind potentiell unendlich. Daher werden in der Regel nur Ausschnitte eines Datenstroms in Form von Fenstern betrachtet und Datenstromelemente werden mit Gültigkeiten, bspw. über Zeitintervalle (vgl. [Krä07]), versehen. Der Standard 61400-25 der IEC [Int06] bietet eine Möglichkeit zur standardisierten Überwachung und Steuerung von WEA. Ähnlich wie der IEC Standard 61970 könnte dieser Standard zukünftig auch in RDF modelliert werden (vgl. [UG07]). Daher untersuchen wir Möglichkeiten zur Verarbeitung von RDF-Datenströmen (vgl. [BGJ08]). Diese Arbeit beschreibt die Umsetzung einer Erweiterung der W3C Empfehlung SPARQL für RDF-Datenströme im **O**ldenburger **Dy**naQue**s**t Data**s**tream Q**u**ery **S**ystem (Odysseus).

Der Aufbau dieser Arbeit ist an die einzelnen Schritte zur SPARQL-Erweiterung angelehnt. Der folgende Abschnitt 2 betrachtet die SPARQL-Spracherweiterung. Daran anschließend wird auf logische Algebra-Operatoren eingegangen, bevor im Abschnitt 4 die Realisierung dieser Algebra-Operatoren in einer physischen Algebra beschrieben wird. Der Abschnitt 5 erläutert die Erzeugung von ausführbaren Anfrageplänen, die aus SPARQL-Datenstromanfragen gewonnen werden können. Diese Arbeit schließt mit einer Darstellung verwandter Arbeiten in Abschnitt 6 und einem Ausblick auf weitere Forschungsvorhaben in Abschnitt 7 ab.

2 SPARQL-Spracherweiterung

Im ersten Schritt wurde die eigentliche Anfragesprache SPARQL so erweitert, dass auch die Referenzierung von Datenströmen in einer Anfrage möglich wird. Außerdem können in Anlehnung an den SQL:2003 Standard[1] Fenster auf diesen Datenströmen ausgedrückt werden. Das folgende Listing 1 zeigt ein Beispiel einer SPARQL-Datenstromanfrage:

[1] http://savage.net.au/SQL/sql-2003-2.bnf.html#window%20clause

```
SELECT ?x ?y ?z
FROM STREAM <http://iec.org/td.rdf>
    WINDOW RANGE 200 SLIDE
WHERE { ?x wtur:StrCnt ?y .
        { ?x wtur:StopCnt ?z .
          WINDOW RANGE 450 FIXED } }
```

Listing 1: Beispiel einer SPARQL-Datenstrom-Anfrage

Die Verarbeitung einer solchen Anfrage geschieht in einem ersten Schritt durch das Parsen mit anschließender Überführung in einen logischen Anfrageplan. Mit ARQ [Sea07] existiert bereits eine Referenzimplementierung von SPARQL, in der die SPARQL-Grammatik mit JavaCC [2] umgesetzt wird. Auf Basis dieser Referenzimplementierung wurde mit unserer Arbeit das Parsen von SPARQL-Datenstromanfragen ermöglicht. Dazu haben wir die JavaCC-basierte Grammatik so erweitert, dass die neuen SPARQL-Datenstrom-Konstrukte mit einem von JavaCC erzeugten Parser erkannt werden. Außerdem erweiterten wir die Klassenstruktur von ARQ, um eine Repräsentation einer SPARQL-Datenstromanfrage im Hauptspeicher vorhalten zu können. Die folgende Abbildung 1 zeigt diese erweiterte Klassenstruktur der ARQ-Engine.

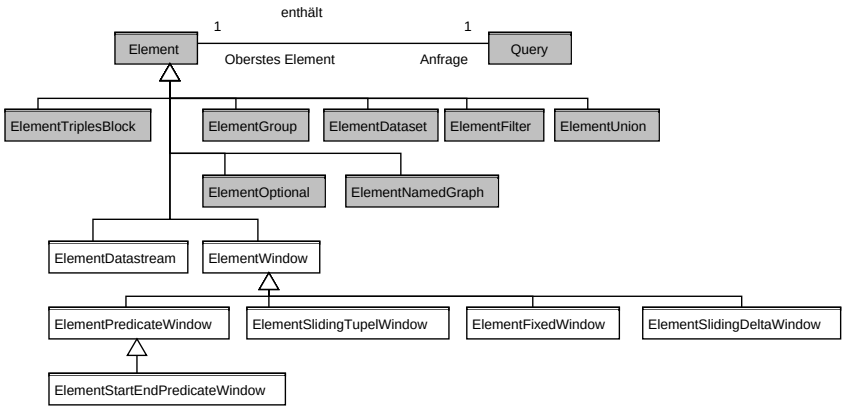

Abbildung 1: Erweiterte Klassenstruktur in ARQ

In dieser Abbildung sind sowohl die Klassen zu erkennen, die bereits von ARQ zur Übersetzung gewöhnlicher SPARQL-Anfragen benötigt werden (grau hinterlegt) als auch die neu entworfenen Klassen zur Übersetzung von SPARQL-Datenstromanfragen. Insbesondere wurden Klassen für die Repräsentation verschiedener Fenstertypen entworfen, die auf Datenströmen definiert werden können. Objekte der von `Element` abgeleiteten Klassen werden vom Parser beim übersetzen erzeugt und zu einem Abstract Syntax Tree (AST) zusammengeführt. So kann bspw. ein Objekt der Klasse `ElementGroup`, welches ein Basic-Graph-Pattern (BGP) in einer SPARQL-Anfrage repräsentiert, als Kindknoten ein Objekt der Klasse `ElementFixedWindow` erhalten, welches ein festes Zeitfenster innerhalb eines BGP repräsentiert (vgl. Listing 1). So entsteht beim Parsen einer SPARQL-Datenstromanfrage-Zeichenkette bereits eine erste logische Repräsentation der Anfrage mit grundlegenden Metainformationen. Um jedoch auch die Semantik einzelner Operatoren zu beschreiben, ist die Erzeugung eines logischen Anfrageplans notwendig, worauf der nächste Abschnitt eingeht.

[2] https://javacc.dev.java.net/

3 Logische Operatoren

Zur Definition der Semantik von SPARQL-Datenstromanfragen haben wir uns an dem allgemeinen Vorgehen bei der Entwicklung von Anfrageverarbeitungssystemen orientiert und eine logische Algebra analog zu [Krä07] definiert. Neben der Semantik stellen die logischen Algebra-Operatoren einfache Metainformationen wie die Fenstergröße bereit und erlauben die Berechnung von Ausgabeschemata, um diese nicht während der Anfrageverarbeitung auf der physischen Ebene mitführen zu müssen. Diese Schemaberechnung ist in Abb. 2 dargestellt.

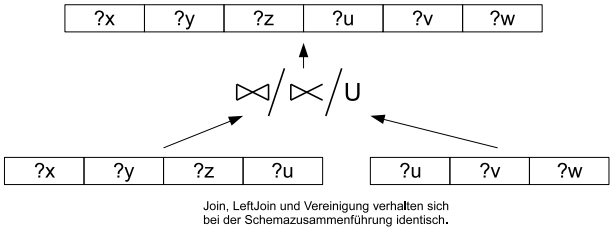

Abbildung 2: Schemamitführung in Odysseus

In dieser Abbildung ist zu erkennen, dass bei einem Join, LeftJoin und der Vereinigung bereits im Vorfeld der Anfrageverarbeitung das Ausgabeschema dieser Operatoren bestimmt wird, indem zunächst die Attribute der linken Eingabe in das Ausgabeschema übernommen und dann die fehlenden Attribute der rechten Eingabe angehängt werden. Dabei entsprechen in diesem Beispiel die Attribute den Variablen, die in einer SPARQL-Anfrage spezifiziert und durch ein Basic-Graph-Pattern-Matching in sogenannten SPARQL-Lösungen zusammengefasst werden (vgl. [PS08]). Die Schemazusammenführung wird in SPARQL auch bei der Vereinigung benötigt, da hier, anders als in SQL, auch Datenströme mit unterschiedlichen Schemata vereinigt werden können. Die in Abbildung 3 dargestellte Klassenstruktur für logische Anfragepläne ermöglicht eine solche Vorausberechnung der Ausgabeschemata einzelner Operatoren.

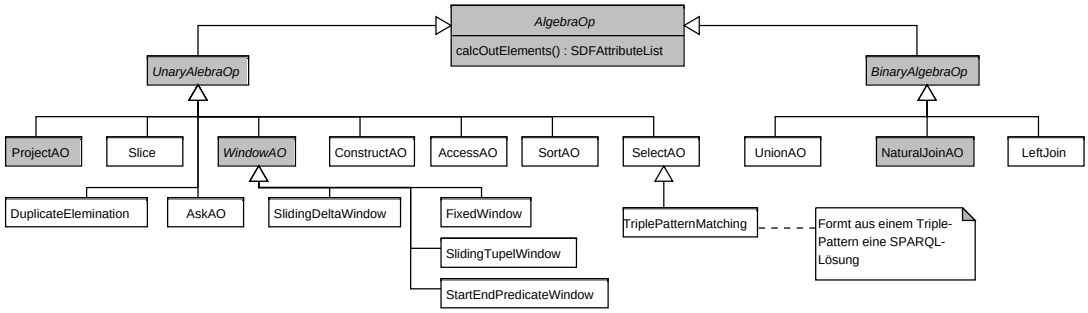

Abbildung 3: Klassenstruktur für logische Anfragepläne

Alle Klassen sind von einer abstrakten Klasse `AlgebraOp` abgeleitet und müssen damit eine Methode `calcOutElements` bereitstellen. Mit Hilfe dieser Methode ist es möglich die Ausgabeschmata einzelner Operatoren in Form einer `SDFAttributeList` zu bestimmen (Source Description Framework, s. [Gra05])[3]. Neben der Berechnung von Ausgabeschemata können logische Anfragepläne dazu dienen Optimierungen durchzuführen, ohne verschiedene Operatorrealisierungen berücksichtigen zu müssen. Um dennoch eine Äquivalenz zu physisch ausführbaren Anfragepläne zu gewährleisten, haben wir Transformationsfunktionen zwischen logischen und physischen Anfragepläne analog zu [Krä07] definiert. Für Details sei daher auf [Krä07] verwiesen.

[3]Von Odysseus bereitgestellte Klassen sind grau hinterlegt

4 Physische Operatoren

Während auf der logischen Ebene die Semantik einzelner Operatoren definiert wird, enthalten physische Operatoren die eigentlichen Realisierungen. Dabei arbeiten diese Operatoren auf physischen Datenströmen, in denen die Gültigkeit von Elementen über Zeitintervalle ausgedrückt wird (Intervallansatz, vgl. [Krä07]). Die Abbildung 4 zeigt die Klassenstruktur für physische SPARQL-Datenstrom-Operatoren in Odysseus.

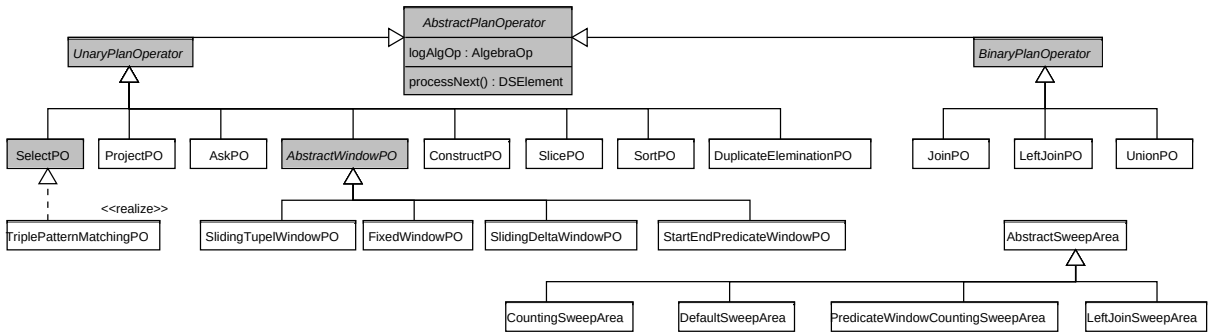

Abbildung 4: Klassenstruktur für physische Anfragepläne

Alle Klassen erben von einer abstrakten Klasse `AbstractPlanOperator`. Die eigentliche Realisierung der Operatoren wird jeweils in der Methode `processNext` implementiert[4]. Die *SweepAreas* [Krä07] werden von zustandsbehafteten Operatoren benötigt. Sie enthalten jeweils die aktuell gültigen Elemente. Je nach Operator werden evtl. besondere Implementierungen benötigt, so dass eine Klassenhierarchie für SweepAreas entwickelt wurde. Die Metadaten erhalten physische Operatoren aus ihren logischen Gegenstücken, die sie bei der Erzeugung physischer Anfragepläne als Attribute übergeben bekommen.

5 Erzeugung physischer Anfragepläne

Um aus einer SPARQL-Datenstromanfrage-Zeichenkette einen ausführbaren Anfrageplan zu erhalten ist zunächst die Erzeugung eines logischen Anfrageplans notwendig. Dies wird dadurch realisiert, dass der von ARQ erzeugte AST rekursiv durchlaufen wird und alle Objekte der von `Element` abgeleiteten Klassen durch Objekte von entsprechenden Klassen der logischen Operatoren ersetzt werden. So wird bspw. ein Objekt der Klasse `ElementUnion` durch ein Objekt der Klasse `UnionAO` ersetzt. Anschließend werden zunächst die Ausgabeschemata der einzelnen Operatoren berechnet und dann für jeden logischen Operator eine mögliche physische Realisierung ausgewählt wird. Auch hier wird der logische Anfrageplan rekursiv durchlaufen und die logischen Operatoren werden durch die ausgewählten physischen Gegenstücke ersetzt. Bei der Ersetzung werden gleichzeitig die logischen Metadaten in den physischen Plan integriert. Die eigentliche Ausführung des Anfrageplans geschieht nach dem Open-Next-Close-Ansatz (ONC-Ansatz) (vgl. bspw. [Gra05]). Die in [Gra05] definierte Ausführungsengine ist dabei so generisch gehalten, dass mit den zuvor beschriebenen Klassenstrukturen, diese Engine ohne Anpassung übernommen werden kann und eine Anfrageausführung möglich wird.

6 Verwandte Arbeiten

Auf konzeptioneller Ebene ist diese Arbeit durch [Krä07] inspiriert worden. Insbesondere der Intervallansatz, der Gültigkeiten von Datenstromelementen über Zeitintervalle ausdrückt, und

[4] Von Odysseus bereitgestellte Klassen sind grau hinterlegt

die SweepArea wurden aus dieser Dissertation übernommen. Weiterhin basiert die Implementierung der Konzepte natürlich auf [Gra05], da das dort entwickelte DynaQuest-Framework die Grundlage für Odysseus darstellt. So konnte durch die Wiederverwendung bestehender Klassenhierarchien (`AlgebraOp`, `AbstractPlanOperator` der ONC-Ansatz aus [Gra05] übernommen werden. Außerdem wurden viele Ansätze von SPARQL [PS08] und ARQ [Sea07] übernommen.

7 Ausblick

Die SPARQL-Datenstrom-Erweiterung wurde vollständig in Odysseus implementiert. Es wurden bereits erste Ergebnisse mit einer Evaluierung von SPARQL-Datenstrom-Anfragen auf Simulationsdatenströmen erzielt, die jedoch noch nicht repräsentativ sind. Daher wird eine weitere Evaluierung auf Realdatenströmen, bspw. im Rahmen des Alpha-Ventus-Projektes[5], erfolgen. Weiterhin wird die SPARQL-Erweiterung um Aggregationsoperatoren angereichert und es werden zukünftig Alternativen zum Intervallansatz, wie bspw. der positiv-/negativ-Ansatz (vgl. [GHM+04]), untersucht werden.

Literatur

[ABW03] Arasu, Arvind ; Babu, Shivnath ; Widom, Jennifer: An Abstract Semantics and Concrete Language for Continuous Queries overStreams and Relations. In: *9th International Workshop on Database Programming Languages* Stanford University, 2003, S. 1 – 11

[BGJ08] Bolles, Andre ; Grawunder, Marco ; Jacobi, Jonas: Straming SPARQL - Extending SPARQL to process data streams. In: Bechhofer, Sean (Hrsg.) ; Hauswirth, Manfred (Hrsg.) ; Hoffmann, Joerg (Hrsg.) ; Koubarakis, Manolis (Hrsg.): *Proceedings of the 5th European Semantic Web Conference*, 2008. – Veröffentlichung im Juni 2008 erwartet

[GHM+04] Ghanem, Thanaa M. ; Hammad, Mustafa A. ; Mokbel, Mohamed F. ; G.Aref, Walid ; Elmagarmid, Ahmed K.: Query Processing using Negative Tuples in Stream Query Engines / Purdue University. 2004. – Forschungsbericht

[Gra05] Grawunder, Marco: *DYNAQUEST: Dynamische und adaptive Anfrageverarbeitung in virtuellen Datenbanksystemen*, Universität Oldenburg, Diss., 2005

[Int06] International Electrotechnical Commission Technical Commitee 88: 61400-25 Communications for monitoring and control of wind power plants / International Electrotechnical Commission. 2006. – Forschungsbericht

[Krä07] Krämer, Jürgen: *Continuous Queries over Data Streams - Semantics and Implementation*, Universität Marburg, Diss., 2007

[PS08] Prud'hommeaux, Eric ; Seaborne, Andy: SPARQL Query Language for RDF / World Wide Web Consortium. http://www.w3.org/TR/rdf-sparql-query/ letzer Zugriff: 21. Januar 2008, Januar 2008. – W3C Recommendation

[Sea07] Seaborne, Andy: *ARQ - A SPARQL Processor for Jena*. http://jena.sourceforge.net/ARQ/, letzter Zugriff: 15. Oktober 2007, 2007

[UG07] Uslar, Mathias ; Grüning, Fabian: Zur semantischen Interoperatibilität in der Energiebranche: CIM IEC 61970. In: *Wirtschaftsinformatik* 49 (2007), September, Nr. 4, S. 295 – 303

[5] http://www.alpha-ventus.de

Ein Aktionsmodell für die kooperative Bearbeitung von XML-Daten

Francis Gropengießer, Jens-Oliver Fischer
FG Datenbanken und Informationssysteme, TU Ilmenau
Fraunhofer-IDMT, Ilmenau
francis.gropengiesser@tu-ilmenau.de
fir@idmt.fraunhofer.de

Zusammenfassung

In vielen Anwendungsbereichen, wie z.B. im Design oder dem Medienproduktionsprozess, ist das kooperative Arbeiten mehrerer Nutzer auf einem zentralen Datenbestand unerlässlich geworden. Ein spezieller Anwendungsfall aus dem Bereich der Medienproduktion ist der Ausgangspunkt für die Betrachtungen in dieser Arbeit. Aus ihm lässt sich eine Reihe von Anforderungen ableiten. Für eine ausgewählte Menge existierender Transaktions-/Aktionsmodelle wird gezeigt, dass diese die Anforderungen nicht im vollen Maße erfüllen. In der Arbeit wird daher ein neues kooperatives Aktionsmodell vorgestellt. Die Vorteile des Modells sind die Ermöglichung eines kooperativen Arbeitsprozesses, ein voll automatisches Transaktionsrecovery, die einfache Behandlung von Konflikten, die infolge der fehlenden Serialisierbarkeit auftreten können, durch das Reset-Repeat-Prinzip sowie das Vermeiden eines nachträglichen Mischens verschiedener Versionen eines XML-Dokumentes.

1 Einleitung

Produktionsprozesse, z.B. Design- oder Medienproduktionsprozesse, erfordern ein kooperatives Arbeiten mehrerer Nutzer auf einem gemeinsamen Datenbestand bzw. an einem gemeinsamen Projekt. Nur so können die Ideen und Vorschläge jedes einzelnen Beteiligten berücksichtigt werden und in die Erstellung eines Gesamtergebnisses oder eines Endproduktes mit einfließen.

Einen speziellen Anwendungsfall aus dem Bereich der Medienproduktion stellt das am Fraunhofer-IDMT entwickelte neuartige Audiowiedergabesystem IOSONO [IOS] dar, das auf dem Prinzip der Wellenfeldsynthese [Ber88] beruht. Es ermöglicht die stabile Positionierung virtueller Schallquellen im dreidimensionalen Raum, wodurch ein realitätsgetreues Raumklangerlebnis für den Hörer entsteht. Die bei der Positionierung der Soundobjekte entstehenden Metainformationen werden in Form eines Szenegraphs dargestellt, der in einer XML-Datei abgespeichert wird.

Aus Sicht eines Datenbankentwicklers besteht die Kernaufgabe darin, das kooperative Arbeiten mehrerer Autoren auf den gleichen XML-Daten transaktionell zu ermöglichen. Unter Kooperation ist ein verhandelndes Arbeiten mehrerer gleichberechtigter Nutzer an einer gemeinsamen Aufgabe zu verstehen. Es findet ein Informationsfluss in beliebiger Richtung zwischen den Nutzern statt.

Allerdings birgt das kooperative Arbeiten mehrerer Autoren auf einer zentralen Datenbasis einige Probleme in sich. Die Transaktionen, welche die Designtätigkeiten der Autoren beinhalten, sind sehr lang. Dies führt einerseits dazu, dass Änderungen erst sehr spät sichtbar werden, was ein kooperatives Arbeiten verhindert, und andererseits dazu, dass im Falle eines Abbruchs der gesamte Arbeitsfortschritt verloren ist. Weiterhin erfordert die Ermöglichung eines kooperativen Arbeitsprozesses den Verzicht auf Serialisierbarkeit und damit die Aufweichung der Isolation.

Benötigt werden daher ein offen geschachteltes Transaktionskonzept, was ein frühes Sichtbarmachen von Änderungen erlaubt und die Atomarität im strengen Sinne aufweicht, ein Recoverykonzept, welches die Fehleratomarität gewährleistet, ein geeignetes Synchronisationsverfahren

sowie ein Konzept zur Behandlung von Konflikten, die infolge der fehlenden Serialisierbarkeit auftreten können.

2 Verwandte Arbeiten

Basis für diese Arbeit sind erweiterte Transaktionskonzepte. Diese stellen eine Möglichkeit dar, den Problemen langer Transaktionen zu begegnen. Allerdings sind sie nur eingeschränkt für den speziellen Anwendungsfall aus Abschnitt 1 geeignet, wie nachfolgend gezeigt wird.

Die geschlossen geschachtelten Transaktionen [Mos81, Kru97] fordern Isolation zwischen verschiedenen Transaktionsbäumen und zwischen Geschwistern innerhalb eines Transaktionsbaumes. Dadurch wird jegliche Form der Kooperation unterbunden.

Die offen geschachtelten Transaktionen [WS92, Kru97] erfüllen die Forderung nach Kooperation im vollen Umfang. Problematisch gestaltet sich jedoch die Fehlerbehandlung. Hier sind Kompensationstransaktionen notwendig, die eine Vielzahl von Problemen [GFJK03] in sich bergen.

Das Konzept der Sagas [GMS87, Moc95, Kru97] fordert Kommutativität der Subtransaktionen verschiedener Sagas, wodurch Kooperation unterbunden wird.

Das ConTract-Modell [WR92, Kru97] ermöglicht zwar die Kooperation verschiedener Autoren, bedarf jedoch wieder der Nutzung von Kompensationstransaktionen zur Fehlerbehandlung.

Das DOM-Transaktionsmodell [BOH$^+$92, Kru97] ermöglicht die Kooperation verschiedener Autoren. Allerdings basiert die Fehlerbehandlung auch in diesem Modell auf den Kompensationstransaktionen.

Das CONCORD-Modell [RMH$^+$94, Kru97] ermöglicht ebenfalls Kooperation. Allerdings ist in diesem Modell das Verschmelzen verschiedener Versionen eines XML-Dokumentes zu einer finalen erforderlich, wodurch ein zusätzlicher Arbeitsaufwand für die Autoren entsteht.

Die geschachtelten dynamischen Aktionen [NW94, Kru97] basieren auf den dynamischen Aktionen [NS92, Moc95], die ihrerseits die Dauerhaftigkeit garantieren und somit einen Verzicht auf Kompensationsaktionen ermöglichen. Allerdings fordern die geschachtelten dynamischen Aktionen Serialisierbarkeit für ganze Aktionsbäume, wodurch keine Kooperation zwischen verschiedenen Autoren möglich ist.

Die geschachtelten dynamischen Aktionen für kooperative Anwendungen [Kru97] heben die Serialisierbarkeit gezielt innerhalb von Kooperationsgruppen auf. Allerdings ist das Prinzip der Kooperationgruppen inpraktikabel für den speziellen Anwendungsfall aus Abschnitt 1. Weiterhin erfordert die Behandlung von Konflikten, die infolge der fehlenden Serialisierbarkeit auftreten können, ein sehr systemnahes Eingreifen der Autoren. Dies kann jedoch von einem Designer nicht erwartet werden.

Zusammenfassend kann festgehalten werden, dass keines der hier genannten Modelle als vollständig geeignet für den speziellen Anwendungsfall aus Abschnitt 1 angesehen werden kann. Somit ist die Entwicklung eines neuen Transaktions-/Aktionskonzepts erforderlich.

3 Das kooperative Aktionsmodell

3.1 Aufbau des Modells

Zunächst sollen die für die Arbeit an einem XML-Dokument notwendigen Operationen definiert werden. Read dient dem inhaltlichen, strukturellen oder holografischen Lesen eines Knotens oder Teilbaums. Inhaltliches Lesen dient dem Erfassen des Attributwerts oder des Textes eines Knotens, strukturelles Lesen dem Erfassen der Struktur eines XML-Dokumentes und holografisches Lesen dem Sehen bereits gelöschter Knoten. Mit Edit werden die Werte einzelner Attribut- oder Textknoten geändert. Delete dient dem Löschen von Knoten oder Teilbäume. Mit Insert kann eine Knotenmenge an ein XML-Element angefügt werden. Mit Hilfe der Operation Move wird

eine Knotenmenge verschoben. Dabei dürfen andere Operationen (Read, Edit, Delete, Move, Insert) auf dieser Knotenmenge ausgeführt werden. Mit Reset wird eine Folge von Operationen rückgängig gemacht. Repeat dient dem Wiederholen der durch ein Reset rückgängig gemachten Operationen. Das Reset-Repeat-Prinzip wird zur Behandlung von Konflikten genutzt, die infolge der fehlenden Serialisierbarkeit auftreten können.

Nachfolgend sollen nun die zulässigen Operationsfolgen definiert werden. Eine Operationsfolge besteht entweder aus einer Folge von Leseoperationen und höchstens einem Edit, Delete, Move, Insert, Reset oder Repeat oder zwei Folgen von Leseoperationen und genau einem Move. Mindestens eine Leseoperation ist Bestandteil jeder Operationsfolge. Jede weitere Leseoperation dient nur dem Hineinspringen in eine Verschiebung, da von außerhalb keine Änderungsoperationen (Edit, Delete, Move, Insert, Reset, Repeat) auf der sich in einer Verschiebung befindenden Knotenmenge ausgeführt werden dürfen.

Anhand eines Beispiels soll nun der Aufbau des Aktionsmodells dargestellt werden. Auf dem Beispielszenegraph aus Abbildung 1, wobei sich der Teilbaum *(Effekte, Knall)* in einer Verschiebung befindet, wird die Operationsfolge *Read(Szene, Dialog, Effekte, Knall) Read(Effekte, Knall) Delete(Knall)* ausgeführt. In der Abbildung 2 ist die Umsetzung dieser Operationsfolge auf das Aktionsmodell zu sehen. Dabei werden Operationen auf Teilbäumen in Operationen auf

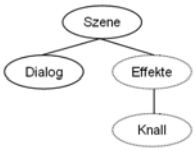

Abbildung 1: Beispielszenegraph

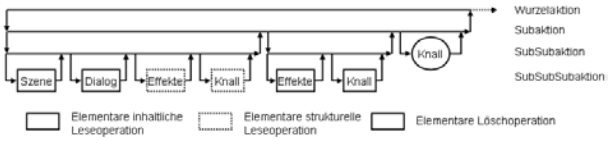

Abbildung 2: Aufbau des Aktionsmodells

einzelnen Knoten zerlegt. Dies ermöglicht das Abbrechen elementarer Operationen, ohne dass dies das Scheitern einer Operation auf einem Teilbaum oder einer ganzen Operationsfolge nach sich zieht.

3.2 Synchronisation

Die Synchronisation konkurrierender Operationen wird durch ein Sperrmodell realisiert. Hierbei wird berücksichtigt, dass es sich um Operationen auf XML-Bäumen handelt.

Zunächst sollen allen Operationen entsprechende Sperrtypen zugeordnet werden. CRL, SRL und HRL sind die Sperrtypen für das inhaltliche, strukturelle und holografische Lesen. EL wird der Operation Edit, DL dem Delete, IL der Operation Insert, ML dem Move und RRL dem Reset und Repeat zugewiesen.

Tabelle 1 zeigt die Sperren-Konfliktmatrix. Für die ML werden zwei Fälle unterschieden. In der vorletzten Zeile der Tabelle ist der Fall dargestellt, in dem sich der Autor außerhalb der Verschiebung befindet. Die letzte Zeile greift den Fall auf, in dem der Autor Teil der Verschiebung ist. ($\surd$ bedeutet vollständige Kompatibilität, $-$ keine Kompatibilität und $+$ bedeutet, dass die Sperren nur kompatibel sind, wenn nicht das Wurzelelement der Verschiebung betroffen ist.)

3.3 Transaktionsrecovery

Die Behandlung von Transaktionsabbrüchen basiert auf einem Objektversionensystem, wobei hier, im Gegensatz zur Multiversionen-Synchronisation [BG83], nur eine sequenzielle Historie von Objektversionen zugelassen wird. Jede Operation greift daher immer auf die aktuelle Version zu. Zur Protokollierung der Historie eines Knotens wird ein Log-Buch eingesetzt. Es hält in drei Dimensionen (Inhalt, Ort und Materialisierung) alle Informationen zu den verschiedenen Versionen eines Knotens fest. Wird eine Aktion abgebrochen, so müssen lediglich die entsprechenden Informationen aus den Dimensionen des Log-Buchs entfernt werden. Weiterhin ist durch die Verwendung dieses Fehlerbehandlungsmodells eine einfache Realisierung des Reset-Repeat-Prinzips

	SRL	CRL	HRL	EL	DL	IL	RRL	ML
SRL	√	√	√	√	−	√	−	
CRL	√	√	√	−	−	√	−	
HRL	√	√	√	√	√	√	√	
EL	√	−	√	−	−	−	−	
DL	−	−	√	−	−	−	−	
IL	√	√	√	−	−	√	−	
RRL	−	−	√	−	−	−	−	
ML	√	−	−	−	−	−	−	−
ML	√	√	√	√	+	√	−	+

Tabelle 1: Sperren-Konfliktmatrix

möglich. Um beispielsweise mehrere Operationen anderer Autoren rückgängig zu machen, muss ein Nutzer lediglich einen alten Zustand eines Objekts auswählen. Dieser wird dann durch den einmaligen Aufruf von Reset wiederhergestellt.

3.4 Eigenschaften

In diesem Abschnitt soll kurz auf einige Eigenschaften eingegangen werden, die durch das entwickelte Modell ermöglicht bzw. garantiert werden.

ACID Die Atomarität im strengen Sinne wird in diesem Modell stark aufgeweicht. Subaktionen können abbrechen, ohne dass dies das Scheitern der Wurzelaktion nach sich zieht. Die Fehleratomarität hingegen ist gesichert, da abgebrochene Aktionen keine Effekte auf die Datenbank haben.

Die einzige in diesem Modell definierte Integritätsbedingung ist die Wahrung der Struktur eines XML-Dokumentes. Diese wird durch die Operationen garantiert. Wird beispielsweise das Delete für einen Knoten aufgerufen, welcher Nachfolger hat, so wird auch für diese das Delete aufgerufen.

Die Isolation wird in dem Modell zwischen Wurzelaktionen aufgehoben. Wurde eine Subaktion beendet, werden die benutzten Sperren für die Allgemeinheit freigegeben.

Die Dauerhaftigkeit ist dadurch garantiert, dass das Modell auf den dynamischen Aktionen basiert, die ihrerseits die Dauerhaftigkeit gewährleisten.

Kooperation Die Kooperation wird in dem Modell dadurch ermöglicht, dass die Isolation zwischen Wurzelaktionen aufgehoben wird. Ordnet man jedem Autor eine Wurzelaktion zu, so kann ein Informationsfluss in beliebiger Richtung zwischen ihnen stattfinden.

Transaktionsrecovery Das vorgeschlagene Fehlerbehandlungsmodell ist einfach zu realisieren. Es ermöglicht eine vollständig automatische Behandlung von Aktionsfehlern und das Reset-Repeat-Prinzip zur Behandlung von Konflikten, die infolge der fehlenden Serialisierbarkeit auftreten können. Durch die streng sequenzielle Historie steht am Ende der Arbeiten der Autoren eine finale Version eines Dokumentes fest, die persistent gespeichert werden kann.

4 Zusammenfassung und Ausblick

Gegenstand dieser Arbeit war die Präsentation eines neu entwickelten Aktionsmodells. Aus einem speziellen Anwendungsfall, dem IOSONO-System, wurden einige Probleme und Aufgabenstellungen abgeleitet. Eine kurze Bewertung existierender Transaktions-/Aktionsmodelle hat ergeben, dass keines dieser Modelle als vollständig geeignet für den Einsatz in dem speziellen

Anwendungsfall angesehen werden kann. Daher wurde ein neues Modell auf Basis der dynamischen Aktionen für kooperative Anwendungen entwickelt und vorgestellt. Dieses zeichnet sich durch seine Unterstützung der Kooperation von Autoren, sein voll automatisches Transaktionsrecovery, die einfache Behandlung von Konflikten, die infolge der fehlenden Serialisierbarkeit auftreten können, durch das Reset-Repeat-Prinzip sowie das Vermeiden eines nachträglichen Mischens verschiedener Versionen eines XML-Dokumentes aus.

In zukünftigen Arbeiten gilt es herauszufinden, wie sich eine Realisierung dieses Modells in der Praxis bewährt. Des Weiteren ist die Entwicklung einer Recoverykomponente zur Behandlung von Systemausfällen notwendig.

Literatur

[Ber88] A. J. Berkhout. A holographic approach to acoustic control. In *Journal Audio Eng. Soc.*, volume 36, pages 977–995. 1988.

[BG83] Philip A. Bernstein and Nathan Goodman. Multiversion concurrency control - theory and algorithms. In *ACM Trans. Database Syst.*, volume 8, pages 465–483, 1983.

[BOH+92] Alejandro P. Buchmann, M. Tamer Ozsu, Mark Hornick, Dimitrios Georgakopoulos, and Frank A. Manola. A transaction model for active distributed object systems. In *Database Transaction Models for Advanced Applications*, pages 123–158, 1992.

[GFJK03] Paul Greenfield, Alan Fekete, Julian Jang, and Dean Kuo. Compensation is Not Enough. In *EDOC '03*, page 232, 2003.

[GMS87] Hector Garcia-Molina and Kenneth Salem. Sagas. In *SIGMOD '07*, pages 249–259, 1987.

[IOS] http://www.iosono-sound.com/.

[Kru97] Armin Kruse. Ein kooperatives Sperrverfahren für geschachtelte dynamische Aktionen — Diplomarbeit an der Universität Bonn. März 1997.

[Moc95] Michael Mock. *Aktionsunterstützung für verteilte, kooperative Anwendungen — Konzept und Realisierung.* GMD-Bericht Nr. 250, R. Oldenbourg Verlag, 1995.

[Mos81] J. Eliot B. Moss. *Nested Transactions: An Approach to Reliable Distributed Computing.* PhD thesis, 1981.

[NS92] Edgar Nett and Ralf Schumann. Supporting fault-tolerant distributed computations under real-time requirements. In *Comput. Commun.*, volume 15, pages 252–260, 1992.

[NW94] Edgar Nett and Beatrice Weiler. Nested dynamic actions - how to solve the fault containment problem in a cooperative action model. In *Symposium on Reliable Distributed Systems*, pages 106–115, 1994.

[RMH+94] Norbert Ritter, Bernhard Mitschang, Theo Härder, Michael Gesmann, and Harald Schöning. Capturing Design Dynamics the Concord Approach. In *ICDE*, pages 440–451, 1994.

[WR92] Helmut Wächter and Andreas Reuter. The contract model. In *Database transaction models for advanced applications*, pages 219–263, 1992.

[WS92] Gerhard Weikum and Hans-Jorg Schek. Concepts and applications of multilevel transactions and open nested transactions. In *Database Transaction Models for Advanced Applications*, pages 515–553. 1992.

Bestimmung interessanter zeitlicher Zusammenhänge für Temporal Data Mining

Tim Schlueter
Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
schlueter@cs.uni-duesseldorf.de

Zusammenfassung

Data Mining beschäftigt sich mit der Analyse großer Datenmengen mit dem Ziel, neues Wissen semi-automatisch aus bekannten Daten zu gewinnen. Der Bereich des Temporal Data Mining behandelt solche Datenmengen, bei denen das Attribut Zeit ausgezeichnet ist, um speziell zeitliche Zusammenhänge zu erkennen. Das grundlegende Problem ist dabei das Finden dieser Zusammenhänge und der zeitlichen Intervalle, in denen sie auftreten. Wir stellen ein Verfahren vor, das eine Transaktionsdatenbank reorganisiert, indem in einem einzigen Datenbank-Scan ein erweiterter P-Baum konstruiert wird, mit dessen Hilfe Zusammenhänge und die zugehörigen zeitlichen Intervalle effizient gefunden werden können.

1 Einleitung

In vielen Bereichen des Lebens fallen große Datenmengen an, die potentiell nützliche Information enthalten. Da eine manuelle Analyse der Daten auf Grund der Masse nicht sinnvoll ist, wurde in den letzten Jahren viel Zeit und Arbeit in das Forschungsgebiet *Knowledge Discovery in Databases* und speziell in den Bereich *Data Mining* investiert. Data Mining bezeichnet die semi-automatische Analyse großer Datenmengen, die das Ziel hat, bestimmte Muster und somit neues potentiell nützliches Wissen aus den Daten zu extrahieren. Die besondere Rolle, die Zeit in vielen Bereichen spielt, wird beim Temporal Data Mining berücksichtigt.

Bekannte Beispiel sind Datenbanken, die Satellitenbilder oder Einkaufsdaten enthalten. Satellitenbilder können bei der Erstellung eines Frühwarnsystems helfen, das z.B. vor ökologischen Katastrophen wie Ölpesten oder Vulkanausbrüchen warnen soll. Einkaufsdaten von Kunden können dazu benutzt werden, besondere Angebote zu machen, wie z.B. einen Sonderpreis werktags zwischen 7 und 9 Uhr für belegte Brötchen und Kaffee, um somit den Umsatz steigern.

Association Rule Mining (ARM) ist eine Teildisziplin des Data Mining, die bei der Analyse von Einkaufsdaten zum Einsatz kommt. Eine Assoziationsregel (AR) ist ein Implikation der Form $X \Rightarrow Y$, wobei X und Y eine Menge von *Items* (z.B. Waren) sind [AIS93]. Eine AR hat einen gewissen Support und eine gewissen Konfidenz, für das Beispiel mit Brötchen und Kaffee könnte z.B. die AR {Brötchen} $\Rightarrow$ {Kaffee} mit 30%-igem Support und 90%-iger Konfidenz gelten. Das würde bedeuten, dass 90% aller Kunden, die Brötchen gekauft haben, auch Kaffee gekauft haben, und das insgesamt 30% aller Kunden Brötchen und Kaffee zusammen gekauft haben. Das Beispiel ist das einer typisch zeitlichen AR, weil diese Regel wahrscheinlich nur zu gewissen Zeiten (z.B. morgens) mit ausreichendem Support gilt.

Beim ARM wird i.Allg. in zwei Schritten vorgegangen: zuerst bestimmt man die Menge aller häufig zusammen auftretenden Items, wobei häufig bedeutet, dass eine Minimum-Support-Bedingung erfüllen sein muss. Aus diesen häufigen Itemsets werden dann AR abgeleitet. Der Hauptaufwand ist bei großen Datenbanken der erste Schritt, bei dem immer wieder auf die Datenbank zugegriffen werden muss, damit die Häufigkeit der Itemsets bestimmt werden kann.

Algorithmen wie der bekannte Apriori-Algorithmus [AIS93] verringern durch Pruning-Schritte die Anzahl der Datenbankzugriffe. Eine Möglichkeit, das Bestimmen des Supports noch effizienter zu machen, ist eine Reorgansisation der Datenbank. In [GCL00] werden Algorithmen vorgestellt, die in einem einzigen Datenbank-Scan einen *P-Baum* (*Partial-Support-Tree*) aufbauen, mit dessen Hilfe der Support eines Itemset sehr effizient berechnet werden kann (siehe Kapitel 2 für Details).

Zwei gute Übersichts-Paper über den Bereich des Temporal Data Mining sind [AO01] und [LS06]. Eine der ersten Arbeiten im Bereich Temporal ARM ist [AS95], in der mehrere Algorithmen (u.a. Varianten des Apriori-Algorithmus) zum Finden von *sequential patterns* vorstellen werden. Ein Beispiel für sequential patterns ist, dass Kunden einer Videothek, die „Star Wars: Episode I“ und „Episode II“ gemietet haben, gewöhnlich auch „Episode III“ mieten. Ein anderer Ansatz ([AR00]) definiert neben dem „normalen“ Support eines Itemset X (der durch den Anteil der Transaktionen, die X enthalten, in der gesamten Datenbank definiert ist) den Begriff des zeitlichen Supports: der *temporal support* ist definiert als der Anteil der Transaktionen, die X enthalten, in der Teilmenge der Transaktionen, die in der *Lebensspanne* (z.B. in der Zeitspanne, in der ein Produkt verkauft wird) des Itemset auftreten. Ein weiterer Ansatz führt ein Framework für *calendar-based* Temporal ARM ein, das eine Kalender-Schema einführt und AR in Form von (r, e) darstellt, wobei r eine AR und e eine *calendar-based pattern* ist ([LNWJ01]). Der amerikanische Thanksgiving-Brauch würde sich z.B. in $(\{\text{Truthahn}\} \Rightarrow \{\text{Kürbis}\}, (*, 11, 2007))$ wiederspiegeln, wobei $(*, 11, 2007)$ für jeden Tag im November 2007 steht.

Das Bestimmen aller zeitlichen AR und der dazugehörigen Kalender-Schemata bzw. Intervalle ist leider sehr ineffizient, da bei den genannten Ansätzen die komplizierteren und somit interessanteren Kalender-Schemata iterativ aus einzelnen einfacheren aufgebaut werden, und dafür fast alle Möglichkeiten ausprobiert werden (abgesehen davon, dass einige Pruning-Schritte vorgestellt werden, die das Verfahren etwas effizienter machen). Dieses Paper greift die Idee der Reorganisation der Datenbank in Form eines P-Baums auf und erweitert sie derart, dass zeitliche Zusammenhänge erkannt werden können. Auf Basis eines erweiterten P-Baums, der in Kapitel 3 eingeführt wird, lassen sich wesentlich effizienter zeitliche Assoziationsregeln und die Intervalle, in denen sie auftreten, finden.

Das Paper ist folgendermaßen gegliedert: In Kapitel 2 werden die grundlegenden Begriffe definiert, Kapitel 3 beinhaltet den P-Baum und dessen Erweiterung um die zeitliche Komponente und Kapitel 4 fasst die Idee dieses Paper kurz zusammen und gibt einen abschließende Ausblick.

2 Definitionen

Die folgenden grundlegenden Definitionen sind aus [ES00] entnommen. $I = \{i_1, ..., i_m\}$ sei eine Menge von Literalen, den sogenannten *Items*. Eine Teilmenge $X \subseteq I$ wird *Itemset* genannt. Ein Itemset soll lexikographisch sortiert sein, d.h. ein Itemset X, das aus $i_1, ..., i_k$ besteht, kann als Tupel $(i_1, ..., i_k)$ geschrieben werden, wobei $x_1 \leq ... \leq x_k$ (für eine beliebige induzierte Ordnung auf I) gilt.

D sei eine Menge von Transaktionen T, wobei $T \subseteq I$. Jeder Transaktion T hat einen Zeitstempel t. Eine Transaktion T stellt z.B. die Menge der Artikel dar, die ein Kunde zu einem gewissen Zeitpunkt gekauft hat. Der *Support* eines Itemset $X \subseteq I$ ist definiert als der Anteil der Transaktionen in D, die X enthalten. Ein Itemset ist *häufig*, wenn sein Support größer als ein Schwellenwert `minsup` ist. Eine Assoziationsregel (AR) ist ein Implikation der Form $X \Rightarrow Y$, wobei X und Y Itemsets sind, für die $X \cap Y = \emptyset$ gilt. Der Support einer AR $X \Rightarrow Y$ ist der Support der Menge $X \cup Y$ in D. Als Maß der Interessantheit ist die *Konfidenz* einer AR $X \Rightarrow Y$ als der Anteil der Transaktionen in D definiert, die Y enthalten, in der Teilmenge der Transaktionen, die auch X enthalten.

Als Erweiterung für Daten mit zeitlicher Relevanz wurde der *temporal support* in [AR00] eingeführt. Die *Lebensspanne* eines Items i ist der Zeitraum $[t_1, t_2]$, in dem i in der Transaktionsda-

tenbank auftritt. Die Lebensspanne eines Itemset $X = \{i_1, ..., i_k\}$ ist der Zeitraum $[t, t']$, in dem das gesamte Itemset auftritt, also $t = \max\{t_1 \mid [t_1, t_2]$ ist die Lebensspanne eines Items i in $X\}$ und $t' = \min\{t_2 \mid [t_1, t_2]$ ist die Lebensspanne eines Items i in $X\}$. Der *temporal support* eines Itemset X ist definiert als der Anteil der Transaktionen aus D in der Lebensspanne von X, die das Itemset enthalten. Damit die Definition des temporal support nicht ad absurdum geführt wird, wird zusätzlich ein Schwellenwert `mintempsup` definiert. Ein Itemset ist dann *häufig* im Sinne des zeitlichen Supports, wenn dieser größer als `mintempsup` ist.

Ein *Kalender-Schema* [LNWJ01] ist definiert als $R = (G_n : W_n, ..., G_1 : W_1)$, wobei jedes Attribut G_i eine Granularität wie Jahr, Monat oder Tag darstellt und durch eine Funktion *valid* beschränkt ist. Jeder Wertebereich W_i ist eine endliche Teilmenge von $\mathbb{N}$. Die Funktion $valid : W_n \times ... \times W_1 \rightarrow \{0, 1\}$ ist eine Funktion, die angibt, ob ein gegebenes Schema gültig ist. Dabei kann *valid* zum einen bestimmte Zeiträume, wie z.B. Ferien, und zum anderen ungültige Attribute (z.B. für Monat Zahlen, die größer als 12 sind) ausschließen. Ein *calendar-based pattern* ist ein Tupel $\langle w_n, ..., w_1 \rangle$, wobei $w_i \in W_i$ oder $*$ ist ($*$ steht für jede Zahl des ensprechenden Attributs). Das Muster $\langle 2007, *, 1 \rangle$ stellt also zum Beispiel die Menge der Monatsersten im Jahr 2007 dar. Der Support eines Itemset X ist dann als der Anteil der Transaktionen aus D in dem Zeitraum, der durch ein Muster $\langle w_n, ..., w_1 \rangle$ angegeben ist, die X enthalten, definiert. Analog heißt eine Itemset *häufig*, wenn der Support größer als ein Schwellenwert ist. Wir führen für den Support und die Häufigkeit keine unterschiedlichen Begriffe ein, weil jeweils aus dem Zusammenhang klar ist, welche Definition gemeint ist.

Die grundlegende Aufgaben beim Association Rule Mining (ARM) besteht darin, alle Assoziationsregeln in einer Datenbank D zu finden, die ausreichenden Support und Konfidenz haben. Dabei bedeutet der Begriff „ausreichend“, dass die Werte größer oder gleich benutzerspezifischen Werten (`minsup` und `minconf`) sind. Diese Aufgabe kann in zwei Teilaufgaben aufgeteilt werden:

1. Das Finden aller Itemsets, für die Support $\geq$ `minsup` gilt.
2. Die Herleitung von AR, für die Konfidenz $\geq$ `minconf` gilt.

Aus den in Teil 1 gefundenen häufigen Itemsets können die Regeln leicht bestimmt werden, indem für jedes Itemset X und für jede Teilmenge $A \subset X$ die Konfidenz der Regel $A \Rightarrow (X - A)$ überprüft wird.

Der aufwendigere Teil ist das Finden der häufigen Itemsets. Der de facto Standard Algorithmus Apriori [AIS93] durchläuft dafür mehrfach die Datenbank, um häufige $k-1$-elementige Itemsets zu finden, und aus diesen dann Kandidaten für k-elementige Itemsets zu finden, die auf minimalen Support untersucht werden. Dabei nutzt der Algorithmus die Monotonie-Eigenschaft von häufig auftretenden Itemsets, die besagt, dass jede Teilmenge eines häufig auftretenden Itemset selbst auch häufig ist. Von einelementigen Itemsets an werden also immer größerer Itemsets untersucht, wobei für den Test auf minimalen Support immer wieder auf die Datenbank zugegriffen werden muss.

Diese Zugriffe auf die Datenbank sind gerade bei größeren Datenmengen sehr aufwendig, weshalb eine Reorganisation der Datenbank in [GCL00] vorgeschlagen wurden, um diesen Aufwand zu verringern. In einem einzigen Datenbank-Scan wird ein *P-Baum* (*Partial-Support-Baum*) erstellt, mit dessen Hilfe der Support von Itemsets wesentlich effizienter ermittelt werden kann.

Der P-Baum basiert auf dem Set-Enumeration-Framework von Ron Rymon [Rym92]. Es definiert den *SE-Baum*, eine Baumstruktur zur Darstellung von Elementen und ihreren Potenzmengen ist. Abbildung 1 zeigt die Potenzmenge $\mathcal{P}(\{A, B, C, D\})$ in Form eines SE-Baums. Die Darstellung als SE-Baum eignet sich besonders für Verfahren, die auf der Potenzmenge gewisser Elemente arbeiten, weil der SE-Baum eine geordnete irredundante Darstellung ermöglicht. In [GCL00] wird u.a. diese Eigenschaft genutzt, um den Support eines Itemset effizient zu bestimmen.

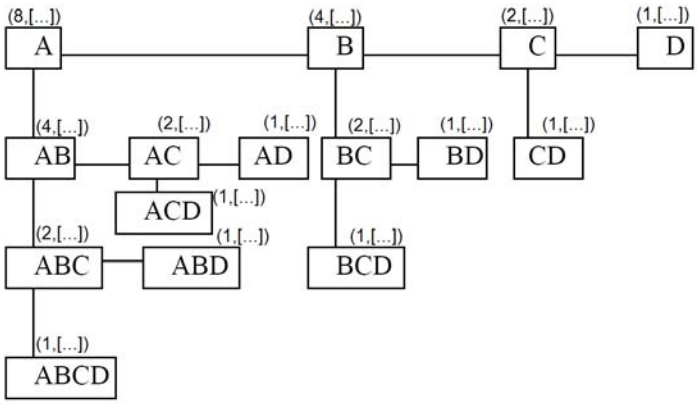

Abbildung 1: Darstellung der Teilmengen von $\{A, B, C, D\}$ als Baum [GCL00]

3 Der erweiterte P-Baum

Ein naiver Algorithmus zur Bestimmung des Supports aller möglicher Itemsets (also aller Teilmengen der Potenzmenge $\mathcal{P}(I)$) wäre, die Datenbank zu durchlaufen und für alle Teilmengen eines Datenbankeintrags einen Zähler für die entsprechende Teilmenge um 1 zu erhöhen. Bei diesem Verfahren würde z.B. das Auftreten des Itemset (A, B, D) zu der Erhöhung der Zähler für $(A, B, D), (A, B), (A, D), (B, D), (A), (B)$ und (D) führen. Eigentlich würde es aber reichen, nur den Zähler von (A, B, D) zu erhöhen, weil dies auch die Erhöhung der anderen Zähler impliziert.

Die Idee hinter der Bestimmung des Supports durch den Partial-Support-Baum (P-Baum) ist ein einmaliger Durchlauf durch die Datenbank, in der jede Transaktion (die aus der Menge der Items $X \in I$ und dem zugehörigen Zeitstempel t besteht) in den SE-Baum zu $\mathcal{P}(I)$ „eingetragen“ wird. Dafür hat jeder Knoten des SE-Baums zwei Einträge: ein Zähler Q_i, der initial auf Null gesetzt ist, und beim erweiterten P-Baum zusätzlich eine Liste, in der Zeitpunkte gespeichert werden, die ebenfalls initial leer ist. „Eintragen“ bedeutet dann, dass der (erweiterte) P-Baum auf dem durch die SE-Struktur festgelegten Weg von der Wurzel zu dem Knoten, der X entspricht, durchlaufen wird, und dass auf diesem Weg in jedem durchlaufenen Knoten der Zähler um 1 erhöht wird (und der Zeitpunkt t der Transaktion an die Liste angehängt wird).

Der *partielle Support* P_i eines Itemset $i \in I$ ist durch die Anzahl der Datenbankeinträge definiert, die mit i identisch sind. Der *totale Support* T_i kann dann durch die Formel

$$T_i = \sum_{j \supseteq i} P_j$$

berechnet werden.

Auf Grund der Struktur des P-Baums haben die Zähler nach dem Datenbank-Scan folgende Werte:

$$Q_i = \sum P_i \quad (\forall j, j \supseteq i \text{ und } j \text{ folgt } i \text{ in lexikographischer Ordnung})$$

Abbildung 1 zeigt den erweiterten P-Baum zu $\mathcal{P}(\{A, B, C, D\})$, bei dem über den Knoten jeweils Beispiel-Werte der Zähler und die angedeutete Listen eingezeichnet sind.

Der totale Support des Itemset kann durch

$$T_i = \sum Q_i + \sum P_j \quad (\forall j, j \supseteq i \text{ und } j \text{ geht } i \text{ in lexikographischer Ordnung voran})$$

berechnet werden. Bei dem erweiterten P-Baum kann analog dazu die vollständige Liste der Zeitpunkte zu einem Itemset erstellt werden, mit deren Hilfe dann relevante Zeiträume, in denen eine AR gilt, bestimmt werden können. Somit können zeitliche AR, die in bestimmten Zeitspannen

oder an einem bestimmten Kalender-Muster (siehe Kapitel 2) gelten, effizienter als bei [AR00] oder [LNWJ01] gefunden werden.

Da in der Datenbank sicher wesentlich weniger verschiedene Itemsets als $|\mathcal{P}(I)|$ auftreten, sollte der Baum dynamisch während des Datenbankdurchlaufs erstellt werden, damit der Bedarf an Speicher nicht exponentiell hoch ist. Algorithmen dafür werden in [GCL00] präsentiert, womit der Speicherbedarf des Baums auf linear in der größe der Datenbank sinkt. Damit wird das Verfahren praktikabel.

In [CGL04] wird darüber hinaus der *T-Baum* (*Target-Set-Tree*) eingeführt, der als Speicherstruktur bei der Berechnung des totalen Support fungiert, und der Algorithmus Apriori-TFP, eine Adaption des Apriori-Algorithmus, in dem der Support der Itemsets durch die eingeführte Baumstruktur effizient berechnet wird.

4 Zusammenfassung und Ausblick

In diesem Paper wurde ein bestehender Ansatz zur Reorganisation von Datenbanken vorgestellt, um den Hauptsaufwand beim ARM, die Anzahl der Datenbankzugriffe bzw. der häufigen Bestimmung des Supports, zu verringern. Ebenso wurden einige bestehende Ansätze im Bereich Temporal ARM vorgestellt. Der Beitrag dieses Paper ist die Kombination von Temporal ARM mit dem Ansatz zur Reorganisation der Datenbank. Dadurch können zeitliche Zusammenhänge und die Intervalle, in denen sie gelten, effizienter gefunden werden, als in den ursprünglichen Arbeiten [AR00] und [LNWJ01].

Eine mögliche Erweiterung ist die Anpassung des erweiterten P-Baums, so dass mit ihm sequential patterns [AS95] oder AR der Form $X \Rightarrow^T Y$ [DLM$^+$98] („wenn ein Käufer X kauft, wird er wahrscheinlich innerhalb des Zeitraums T auch Y kaufen“) gefunden werden können, sowie die Implementation und Evaluation eines Prototyp-Systems.

Literatur

[AIS93] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 1993.

[AO01] C. M. Antunes and A. L. Oliveira. Temporal data mining : An overview. *KDD Workshop on Temporal Data Mining*, August 2001.

[AR00] J. M. Ale and G. H. Rossi. An approach to discovering temporal association rules. In *SAC '00: Proceedings of the 2000 ACM symposium on Applied computing*, pages 294–300, 2000.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In Philip S. Yu and Arbee S. P. Chen, editors, *11th International Conference on Data Engineering*, pages 3–14. IEEE Computer Society Press, 1995.

[CGL04] F. Coenen, G. Goulbourne, and P. Leng. Tree structures for mining association rules. *Data Min. Knowl. Discov.*, 8:25–51, 2004.

[DLM$^+$98] G. Das, K. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *Knowledge Discovery and Data Mining*, pages 16–22, 1998.

[ES00] M. Ester and J. Sander. Knowledge Discovery in Databases - Techniken und Anwendungen, 2000.

[GCL00] G. Goulbourne, F. Coenen, and P. H. Leng. Algorithms for computing association rules using a partial-support tree. *Knowledge Based Systems*, 13(2-3):141–149, 2000.

[LNWJ01] Y. Li, P. Ning, X. S. Wang, and S. Jajodia. Discovering calendar-based temporal association rules. In *TIME*, pages 111–118, 2001.

[LS06] S. Laxman and P.S. Sastry. A Survey of Temporal Data Mining. In *Sadhana Vol. 31, Part 2*, pages 173–198, April 2006.

[Rym92] R. Rymon. Search through systematic set enumeration. Technical report, Department of Computer and Information Science, University of Pennsylvania, 1992.

Nutzerzentriertes maschinenbasiertes Lernen von Gewichten beim Multimedia-Retrieval

David Zellhöfer und Sebastian Lehrack
Institut für Informatik
Brandenburgische Techn. Universität Cottbus
david.zellhoefer@tu-cottbus.de
slehrack@informatik.tu-cottbus.de

Zusammenfassung

Die Suche in multimedialen Datenbeständen ist gekennzeichnet durch komplexe Anfragen, die sich verschiedener Suchparadigmen, wie der DB- oder Retrieval-Suche, bedienen. Während bei der klassischen DB-Suche ein exaktes Matching anhand einer Anfrage im Vordergrund steht, kann dieses beim Multimedia-Retrieval nicht gewährleistet werden. Hier werden primär ähnliche Ergebnisse gesucht.

Ähnlichkeit ist ein höchst subjektives Empfinden, was bereits frühzeitig durch psychologische Experimente gezeigt werden konnte. Folglich müssen Ähnlichkeit involvierende Anfrageteile immer entsprechend gewichtet werden, um diese Subjektivität abbilden zu können. Problematisch ist dabei, dass zu einer korrekten Gewichtung dieser Teile ein Wissen über deren Bedeutung beim Nutzer vorhanden sein muss.

Gerade das Zusammenwirken verschiedener Low-Level-Features multimedialer Objekte kann durch den Durchschnittsnutzer kaum vorhergesehen werden. Die vorliegende Arbeit stellt ein Verfahren vor, welches auf der interaktiven Angabe von Halbordnungen auf Ergebnismengen von Medienobjekten basiert, um das subjektive Ähnlichkeitsempfinden des Anwenders maschinell zu erlernen.

1 Einleitung

Die Suche in umfangreichen multimedialen Datenbeständen stellt noch immer ein großes Problem im Bereich der DB- und Retrieval-Systeme dar. Zum Auffinden solcher Daten sind verschiedene Suchparadigmen notwendig, die geeignet kombiniert werden müssen. So können beispielsweise Metadaten von Bildern mittels klassischer, relationaler DB-Operationen durchsucht werden, die ein attributsbasiertes, exaktes Matching auf Basis der Prädikatenlogik und Mengenlehre ermöglichen [5, 6]. Inhaltsbasierte Verfahren, wie die Suche in extrahierten Low-Level-Features[1] oder Texten, hingegen basieren auf der Modellierung von Ähnlichkeit gegenüber einem Anfrageobjekt. Ähnlichkeit kann dabei beispielsweise durch eine geordnete Liste abgebildet werden [1, 10].

Psychologische Forschungen [3, 16] und Nutzerexperimente im Bereich des Information-Retrievals [13] zeigen eine Besonderheit solcher auf Ähnlichkeit basierender Anfragen auf. Das Ähnlichkeitsempfinden eines Anwenders ist subjektiv und von verschiedenen Faktoren abhängig. Durch die Möglichkeit, verschiedene Teile einer Teilbedingungen einer Anfrage zu gewichten, kann diese Subjektivität modelliert werden, um ein höheres Maß an Nutzerzufriedenheit zu ermöglichen.

[1] z.B. Histogramme, dominante Farben etc.

2 Probleme des Multimedia-Retrievals

Ein wesentliches Problem des Multimedia-Retrievals (MMR) stellt die Kombination verschiedener Suchparadigmen [8, 11, 17, 4] und das „reasoning about uncertain data“ [9] dar. Klassische DB-Systeme basieren auf der boolschen Logik und Mengenlehre, während sich Retrieval-Systeme der Fuzzy-Logik [18] oder dem Vektorraummodell bedienen, um anhand der Ähnlichkeit geordnete Ergebnislisten zu einer Anfrage zu erzeugen. Dies hat zur Folge, dass Verfeinerungen von Anfragen unterschiedlich realisiert werden. Da es sich bei DB-Anfragen um logische Ausdrücke handelt, können diese entsprechend angepasst werden und z.B. Konjunktionen gegen Disjunktionen ersetzt werden. Die Verfeinerung bei Retrieval-Systemen hingegen wird durch das Setzen eines Schwellenwerts oder durch interaktive Techniken, wie dem Relevance Feedback, realisiert [12].

Schmitt entwirft in seiner Arbeit ein theoretisches Framework, welches auf der Quantenlogik [2] basiert und die genannten Ansätze erstmalig vereint [15]. Es kann außerdem gezeigt werden, dass der Ansatz mächtig genug ist, auch Gewichte in diese Logik zu integrieren [14]. Bisherige Ansätze, wie Fagin und Wimmers Gewichtung [7] verletzen die Gesetze der Logik[2], da sie oberhalb der Anfragesprache aufsetzen.

Obwohl durch dieses vereinheitlichende Framework bereits ein wesentlicher Beitrag zur Fortentwicklung von MMR-System geleistet wurde, muss das Setzen von Gewichten auf Bedingungen einer Anfrage gesondert diskutiert werden. Da Anfragegewichtungen einen wesentlichen Beitrag zur Erreichung eines hohen Maßes an Nutzerzufriedenheit haben (s.o.), muss sichergestellt werden, dass diese korrekt gesetzt werden. Problematisch ist hierbei, dass Gewichte in den komplexen Anfragen des MMR nur selten vollständig vom Nutzer gesetzt werden können. Dies hat unterschiedliche Gründe:

Unklarheit über das Suchziel: Dem Nutzer ist zu Beginn nicht klar, welche Kriterien ihm besonders wichtig sind, bzw. verändern sich diese, wenn ihm der Datenbestand bekannt wird.

Fehlannahmen über das konzeptionelle Modell: Dem Nutzer ist das konzeptionelle Modell des Systems unbekannt. Dies führt zu Fehlannahmen über die Auswirkung von Anfragegewichten bzw. einer Anpassung des Nutzers an das fehlerhaft prognostizierte Verhalten des Systems.

Komplexität der Gewichte: Die Anfrage ist so komplex, dass das Setzen von Gewichten kaum bewältigbar oder überschaubar wird. Dies umfasst auch Low-Level-Features, bei denen der Nutzer nur schwer abschätzen kann, wie sich diese verhalten bzw. welche Semantik sie haben.

Unerwartete Wechselwirkungen: Insbesondere bei komplexen Anfragen oder der Gewichtung von Low-Level-Features können sich Wechselwirkungen ergeben, die Ergebnisse erzeugen, die nicht mehr dem subjektiven Empfinden von Ähnlichkeit entsprechen.

Folglich ist es notwendig, den Nutzer beim Setzen von Gewichten zu unterstützen, da nicht erwartet werden kann, dass eine initiale Gewichtsetzung selbständig von ihm vorgenommen werden kann. Dies gilt insbesondere für Teile der Multimedia-Anfrage, die indirekt angegeben werden, wie Bilder mit den daraus abgeleiteten Low-Level-Features. Im Gegensatz dazu können Gewichte auf explizit angegebene Teile der Anfrage, wie DB-Bedingungen, eher durch den Anwender definiert werden.

3 Lernen von Gewichten

Aufgrund der Komplexität des diskutierten Problems der Angabe von Gewichten auf Anfrageteilen bietet es sich an, die Lösung dieses in den Prozess des Relevance Feedbacks zu integrieren. Gängige Retrieval-Systeme bedienen sich dieser Methode, um Bewertungen über die Ergebnismenge, die eine Anfrage liefert, durch Nutzereingaben zu gewinnen und die Anfrage dementsprechend zu modifizieren. Problematisch dabei ist, dass zur Steuerung oder Verfeinerung der Ergebnismenge möglichst eine Vielzahl von

[2] z.B. die Assoziativität

Ergebnisobjekten durch den Anwender bewertet werden muss. Dies steht im Gegensatz zu den Wünschen des Nutzers, der ohne viele manuelle Eingriffe ein akzeptables Ergebnis erhalten möchte.

Der vorgestellte Ansatz versucht Prinzipien des Relevance Feedbacks auf das maschinelle Lernen von Gewichten zu übertragen. Dabei wird im wesentlichen auf die Nutzerangabe von Halbordnungen über der Ergebnismenge zurückgegriffen, welche die subjektiv wahrgenommene Relvanzbewertung bezüglich der Anfrage ermöglicht, gleichzeitig aber auch eine Ordnung zwischen Ergebnisobjektpaaren zulässt. Die Angabe einer Halbordnung ist dabei sinnvoll, da es so dem Anwender erspart wird, die komplette Menge an Ergebnisobjekten zu bewerten.

Basis des Prozesses bilden Initialgewichte und eine Anfrage an das MMR-System. Die Festlegung der initialen Gewichte kann dabei auf vielerlei Arten geschehen; sie sollen hier aber nicht detailliert diskutiert werden. Denkbar sind Gewichtungen anhand des Eingabetyps der Bedingung[3], anhand von Such- bzw. Nutzerprofilen oder durch Direkteingabe sowie auf Basis neutraler oder zufälliger Werte.

Auf Grundlage dieser Parameter wird eine erste Ergebnismenge E_1 durch das MMR-System erzeugt, die durch die Angabe von Halbordnungen durch den Nutzer entsprechend seiner Wahrnehmung bewertet wird. Der Anwender kann dabei einzelne Elementpaare o_i und o_j aus E_1 mittels eines Vergleichsoperators in Beziehung setzen. Hierbei kann er zusätzlich eine Menge von Favoriten F definieren, die eine höhere Relevanz als alle anderen Elemente von E aufweisen. Als Besonderheit sind hier Elemente zu betrachten, die für den Nutzer als Favoriten erkannt werden, streng genommen jedoch nichts mit der Anfrage zu tun haben, sondern aus subjektiven Gründen ausgewählt werden. Zusätzlich kann eine Menge von irrelevanten Objekten I angegeben werden. Folglich ergibt sich eine Halbordnung über E_1, die auch F beinhaltet, und eine Teilmenge U von E_1, welche die vom Anwender abgelehnten Ergebniselemente beinhaltet. Diese bilden die Eingabeparameter für einen Lernalgorithmus, welcher die Bedeutung einzelner Anfrageteile und die damit notwendigen Gewichtungen zu erkennen versucht. Der Vorteil hierbei ist, dass der Nutzer durch das maschinelle Lernen unterstützt wird, da das System durch eine geringe Eingabe bereits Erkenntnisse über seine Ähnlichkeitsempfinden gewinnen kann. Dies wird beim nächsten Iterationsschritt deutlich, bei welchem die Anfrage entsprechend anpasst wird und erwartungskonformere und relevantere Ergebnisse zurückliefert.

Durch die Eingaben des Anwenders kann es jedoch zu Konflikten kommen, wenn die vorgegebene Anfrage aufgrund von Widersprüchen in der Halbordnung nicht erfüllt werden kann. Liegt ein solcher Konflikt vor, so müssen die ursächlichen Paare manuell entfernt werden. Eine automatische Entfernung ist nicht in jedem Fall sinnvoll, da sie das Ergebnis unter Umständen stark verzerrt. Die manuelle Entfernung zeigt dem Nutzer Problemstellen auf und überlässt ihm die Entscheidung, welche Kriterien erfüllt werden sollen. Es bietet sich an, dass diese Tests auf Widersprüche bereits während der Angabe der Halbordnung durch den Anwender ausgeführt werden, so dass er bereits zu einem frühen Zeitpunkt eventuelle Konflikte erkennt und beheben kann.

Wie bereits angesprochen ist es denkbar, dass der Nutzer während der Suche sein Suchziel ändert. Folglich ist es notwendig, dass der Lernalgorithmus dies erkennt und sich entsprechend anpasst. Würde der Lernprozess nach einer gewissen Zahl von Iterationsstufen abgebrochen werden, so entfernte sich die vom System vorgenommene Gewichtung erneut von der Nutzererwartung.

4 Ausblick

Die vorliegende Arbeit präsentiert einen Ansatz zum interaktiven Lernen nutzerbasierter Anfragegewichte für MMR-Systeme. Zukünftige Herausforderungen, welche im Rahmen dieser Arbeit nicht diskutiert wurden, liegen vor allem im Bereich der Benutzerschnittstellen und der Visualisierung. Um das System vollständig nutzbar zu machen, müssen geeignete Methoden gefunden werden, die dem Nutzer gefundene Konflikte nachvollziehbar vermitteln. Dies kann z.B. im Falle von Low-Level-Features durch eine Rückvisualisierung geschehen.

Ein weiteres Problem ist das Erlernen von Junktoren bzw. die Interpretation dieser, da nicht in allen

[3] explizit angegebene DB-Bedingungen könnten z.B. stärker gewichtet werden

Fällen davon ausgegangen werden kann, dass ein Anwender in einer komplexen Anfrage deren Zusammenspiel überblickt. Unter Umständen kann dieses Lernproblem als Spezialisierung des Gewichtungsproblems aufgefasst werden, wobei unterschiedliche Junktoren zuerst neutral gewichtet werden, um diese dann im Verlauf der Verwendung des Systems anzupassen.

Literatur

[1] BAEZA-YATES, R. und B. RIBEIRO-NETO: *Modern Information Retrieval*. ACM Press, Essex, England, 1999.

[2] BIRKHOFF, G. und J. VON NEUMANN: *The Logic of Quantum Mechanics*. Annals of Mathematics, 37:823–843, 1936.

[3] BRUCE, VICKI und PATRICK R. GREEN: *Visual Perception -physiology, psychology and ecology (2nd ed., reprinted)*. Lawrence Erlbaum Associates, Publishers, Hove and London, UK, 1993.

[4] CHAUDHURI, SURAJIT, RAGHU RAMAKRISHNAN und GERHARD WEIKUM: *Integrating DB and IR Technologies: What is the Sound of One Hand Clapping?* In: *CIDR*, Seiten 1–12, 2005.

[5] CODD, E. F.: *A Database Sublanguage Founded on the Relational Calculus*. In: *ACM SIGFIDET Workshop on Data Description, Access and Control*, Seiten 35–61, Nov 1971.

[6] DATE, C. J. und H. DARWEN: *A Guide to the SQL Standard*. Addison-Wesley, Reading, MA, 3 Auflage, 1993.

[7] FAGIN, R. und E. L. WIMMERS: *A Formula for Incorporating Weights into Scoring Rules*. Special Issue of Theoretical Computer Science, (239):309–338, 2000.

[8] LEW, MICHAEL S., NICU SEBE, CHABANE DJERABA und RAMESH JAIN: *Content-based multimedia information retrieval: State of the art and challenges*. ACM Trans. Multimedia Comput. Commun. Appl., 2(1):1–19, 2006.

[9] PARTICIPANTS OF THE LOWELL WORKSHOP: *The Lowell Database Research Self Assessment*. Technischer Bericht, Microsoft Research, 2003. http://research.microsoft.com/%7Egray/lowell/.

[10] RIJSBERGEN, C. J. VAN: *Information Retrieval*. Butterworths, London, 1979.

[11] ROWE, LAWRENCE A. und RAMESH JAIN: *ACM SIGMM retreat report on future directions in multimedia research*. ACM Trans. Multimedia Comput. Commun. Appl., 1(1):3–13, 2005.

[12] SALTON, GERARD und CHRIS BUCKLEY: *Improving Retrieval Performance by Relevance Feedback*. Technischer Bericht, Ithaca, NY, USA, 1988.

[13] SALTON, GERARD, EDWARD A. FOX und HARRY WU: *Extended Boolean Information Retrieval*. Commun. ACM, 26(11):1022–1036, 1983.

[14] SCHMITT, INGO: *Weighting in CQQL*. Computer Science Reports 4, Brandenburg University of Technology at Cottbus, Institute of Computer Science, November 2007.

[15] SCHMITT, INGO: *QQL: A DB&IR Query Language*. The VLDB Journal, 17(1):39–56, 2008.

[16] SELFRIDGE, O. G.: *Pandemonium. A paradigm for learning*. The mechanics of thought processes, 1959.

[17] WEIKUM, GERHARD: *DB&IR: both sides now*. In: *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, Seiten 25–30, New York, NY, USA, 2007. ACM.

[18] ZADEH, LOTFI A.: *Fuzzy Sets*. Information and Control, (8):338–353, 1965.

Entwicklung eines Frameworks für instanzbasiertes Ontologie-Matching

Katrin Zaiß
Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
D-40225 Düsseldorf, Deutschland
zaiss@cs.uni-duesseldorf.de

Zusammenfassung

Das Semantic Web ist die Vision eines Webs, in dem alle Daten semantisch annotiert und für Rechner semantisch verarbeitbar sind. Ontologien sind eine Möglichkeit einer solchen Annotation und können durch standardisierte Sprachen wie RDF oder OWL ausgedrückt werden. Im Falle einer globalen Anfragebearbeitung oder eines Informationsintegrations-Prozesses, muss ein Matching zwischen verschiedenen Ontologien durchgeführt werden. Dieser Prozess ist bei großen Ontologien sehr aufwendig und sollte deshalb weitesgehend automatisiert werden. Dabei können einige Schwierigkeiten wie z.B. die Verwendung von Synonymen oder unterschiedlichen Abstraktionsebenen auftreten. Dieser Beitrag soll die Probleme existierender Matching-Ansätze aufzeigen und ein neues Framework für instanzbasiertes Ontologie-Matching vorstellen.

1 Einleitung

Die Integration von Daten aus heterogenen Datenquellen ist ein gut erforschtes Gebiet in der Informatik. Die Art der Daten ist dabei vielfältig und geht von XML-Daten, über relationale Datenbankschemata bis hin zu Ontologien. Ziel der Integration ist immer die Zusammenfassung von Daten aus verschiedenen Quellen zu einem einheitlichen Datenbestand. Um die Integration durchführen zu können, muss zunächst ein Matching der verschiedenen Datenquellen durchgeführt werden, d.h. es müssen korrespondierende Klassen, Attribute oder Werte gefunden werden. Schon in den 80ern wurden zahlreiche Ansätze zum Matching von Datenbankschemata vorgestellt, die sich verschiedener Methoden bedienen. Zum einen können stringbasierte Methoden verwendet werden um ähnliche Tabellen/Attribute zu finden, andere Möglichkeiten sind die Betrachtung der Struktur der Tabellen oder die Einbeziehung der Instanzen usw. Viele der dort vorgestellten Methoden können für das Matching von Ontologien übernommen werden.

Ontologien sind ein in der Forschung viel diskutiertes Modell zur Repräsentation von Wissen. Besonders im Semantic Web können Ontologien eingesetzt werden um die Semantik von Webseiten für Rechner maschinell verarbeitbar zu machen. Auch in der Biologie werden viele Ontologien eingesetzt (zB. die GeneOntology) um ein standardisiertes Vokabular zu definieren oder aber auch um die Konzeptualisierung eines bestimmten Gebietes darzustellen. Das Matching von verschiedenen Ontologien ist dabei in vielen Szenarien sinnvoll. Möchten beispielsweise verschiedene Forschergruppen ihre Konzeptualisierung eines bestimmten Gebietes miteinander vergleichen oder die verschiedenen Ontologien zu einer vereinen, ist ein semi-automatischer Matching-Prozess zeitsparend und nützlich.

Das Matching von Ontologien oder anderen Datenquellen weist natürlich auch einige Probleme auf, die in Kapitel 2 in Verbindung mit einem kurzen Einblick in den Stand der Forschung erläutert werden sollen. Anschließend wird in Kapitel 3 kurz der im Rahmen einer Masterarbeit

entwickelte Prototyp eines Matching-Systems vorgestellt. In Kapitel 4 wird die Architektur eines neuen Systems zum Matchen von Ontologien vorgestellt. Abschließend gibt Kapitel 5 einen kurzen Ausblick auf zukünftige Entwicklungen.

2 Existierende Matching-Verfahren

In der Literatur findet man zahlreiche Systeme, die das semi-automatische Matching von Ontologien ermöglichen. Eine Klassifikation der einzelnen Ansätze kann man in [RB01] oder [ES07] finden. Hier sollen einige konkrete Systeme vorgestellt und ihre Nachteile kurz erläutert werden. Die Beschreibung weiterer Systeme findet man bspw. in [Zai08].

- *COMA++*(**CO**mbination of **Ma**tching algorithm, vgl. [DR02]) ist ein schema- und instanzbasiertes Matching-System.

  Die flexible Kombinierbarkeit von vielen, verschiedenen Matching-Methoden ist ein großer Vorteil des Systems. Es werden stringbasierte Methoden, wie z.B. die Edit-Distanz und Prefix/Suffix-Tests, verwendet, aber auch strukturbasierte Methoden, sowie Constraints und Synonymtabellen eingesetzt. Zusätzlich wurden instanzbasierte Methoden entwickelt (vgl.[EM07]), die constraint- und inhaltsbasierte Funktionen enthalten. Constraints sind Regeln, z.B. die durchschnittliche Länge oder verwendete Buchstaben/Symbole, die benutzt werden um Ähnlichkeiten zwischen zwei Konzepten zu bestimmen. Zusätzlich wird getestet, ob Instanzen einer Klasse einem vordefinierten Mustern folgen. Die inhaltsbasierten Funktionen betrachten die Instanzen paarweise und vergleichen diese mit konventionellen stringbasierten Vergleichsfunktionen. Ein Nachteil der instanzbasierten Matcher von COMA++ ist die Beschränkung auf den Abgleich mit vordefinierten Mustern, da dafür immer manuelle Vorarbeit notwendig und die Definition der passenden Muster sehr domänenabhängig ist.

- *QOM* (Quick Ontology Mapping) ist ebenfalls ein Hybridmatcher, der mit besonderem Hinblick auf eine geringe Laufzeit entwickelt wurde. Es gibt vordefinierte Matching-Regeln, die der Reihe nach für jedes Konzeptpaar (auch iterativ) getestet werden. Es gibt bspw. stringbasierte Label- und URI-Vergleiche, aber auch einen attributweisen Vergleich von Instanzmengen. Einge der verwendeten Methoden (SimSet, vgl. [ES04, S.8]) sind allerdings nicht sinnvoll, da Attribute mit Attributen desselben Konzepts verglichen werden.

- *GLUE* (vgl. [DMDH04])ist ein Matching-System, welches nur Instanzen zur Ähnlichkeitsberechnung verwendet. Dazu werden zwei Klassifikatoren, basierend auf Inhalt bzw. Namen der Instanzen, trainiert. Die Aneinanderreihung aller Konzeptnamen von der Wurzel bis zur Instanz ist in dem Fall der Instanzname. Der Klassifikator eines Konzepts wird nun auf die Instanzen des anderen Konzepts angewendet und so eine Wahrscheinlichkeitsverteilung berechnet. Durch die Anwendung der Jaccard-Funktion wird aus der Wahrscheinlichkeitsverteilung ein Ähnlichkeitswert berechnet. Mit Hilfe des Relaxation Labbeling und einige vordefinierten Bedingungen wird dann ein Mapping berechnet. Das Training der Klassifikatoren ist allerdings eher problematisch und führt nicht immer zu annehmbaren Ergebnissen (siehe auch [Zai07]).

3 Ansatz zum Matching von Ontologien

In [Zai07] wurde ein System entwickelt, dass konzept- und instanzbasierte Methoden verwendet um zwei Ontologien zu matchen. Der User lädt seine lokale Ontologie hoch um sie in eine schon vorhandene globale Ontologie einzufügen. Nachdem der User einige Einstellungen vorgenommen hat (Festlegung eines Thresholds, Gewichte für instanz- und konzeptbasierte Methoden,...), wird

der semi-automatische Matching-Prozess gestartet. Dabei wurden hauptsächlich Methoden von [DMDH04] und [ES04] verwendet, zusätzlich wurden aber auch einige Methoden selbst entwickelt. Das konzeptbasierte Matching umfasst Methoden wie den stringbasierten Vergleich der Label, die Verwendung des Trigram-Maßes, Prefix/Suffix Tests, der Vergleich der Attributmengen usw.

Die instanzbasierte Matching-Methode wird unabhängig von der konzeptbasierten ausgeführt. Hier werden die oben erläuterten Methoden von Glue verwendet, d.h. mit Hilfe der Instanzen eines Konzeptes wird ein Klassifikator trainiert, mit dessen Hilfe die Instanzen des anderen Konzeptes klassifiziert werden. So kann eine instanzbasierte Ähnlichkeit berechnet werden.

Konzept- und instanzbasierte Ähnlichkeiten werden nun aggregiert, die entsprechenden 1:1 Mapping-Vorschläge werden berechnet und dem Anwender zur Evaluation vorgelegt. Dieser kann die Vorschläge akzeptieren oder ablehnen und anschließend manuell weitere Zuordnungen hinzufügen.

Die in [Zai07] entwickelten Methoden wurden mit Hilfe von Testdaten evaluiert. Es war klar zu beobachten, dass die konzeptbasierte Methode alleine recht gute Ergebnisse liefert (F-Measure durchschnittlich 0,75). Auch die Zusammenarbeit mit der instanzbasierten Methode ist qualitativ zufriedenstellend, der F-Measure-Wert steigt sogar an. Verwendet man allerdings die instanzbasierte Methode alleine, können keine annehmbaren Resultate geliefert werden.

Aufgrund der in der Evaluation entdeckten Problematik der instanzbasierten Methode wurde deutlich, dass in diesem Bereich noch ein Forschungsbedarf besteht.

4 *IBOI*

Innerhalb eines neuen Systems zur Integration von Ontologien, *IBOI* (**I**nstance-**B**ased **O**ntology **I**ntegration), sollen nun neue instanzbasierte Methoden entwickelt und zusammen mit konzept- und strukturbasierten Methoden kombiniert werden. Zusätzlich soll der Anwender innerhalb einer Relevance-Feedback-Schleife die Mapping-Vorschläge bewerten, und so die iterative Berechnung der Vorschläge verbessern. Das System soll aus den Bewertungen des Anwenders auch für spätere Durchläufe lernen.

4.1 Referenz-Architektur

Eine vom User bestimmte lokale und die schon vorhandene globale Ontologie sind die Eingabe für den Matching-Prozess, der Mapping-Vorschläge berechnet und diese dem Nutzer zur Bewertung vorlegt. Der Nutzer bewertet anhand eines vorgegebenen Bewertungssystems, z.B. relevant vs. nicht relevant. Diese Angaben nutzt das System um den Matching-Prozess iterativ durchzuführen und die Ergebnisse zu verbessern. Damit dem Nutzer so wenig wie möglich Aufwand entsteht, ist die Anzahl solcher Feedbackschleifen zu begrenzen. Abschließend wird der Merging-Prozess durchgeführt, der die lokale Ontologie in die globale integriert. Nach diesem Schritt besteht wieder die Möglichkeit der Interaktion mit dem Anwender, der evtl. Verbesserungen am Merging-Ergebnis vornehmen kann. Abbildung 1 zeigt die Integration als Gesamtprozess.

Da der Matching-Prozess ein komplexer Vorgang ist, wird dessen Aufbau in Abbildung 2 im Detail dargestellt. Zunächst werden konventionelle konzeptbasierte Methoden (z.B. Stringvergleichen) verwendet um eine Vorauswahl zu treffen; den entsprechenden Konzepten sollten vor allem Instanzen zugeordnet sein. Basierend auf den Ergebnissen des instanzbasierten Matchings (siehe 4.2), werden struktur-(graph-)basierte Algortihmen verwendet um weitere Konzepte, besonders diejenigen ohne Instanzen, zu matchen. Als Ergebnis liefert der Matching-Prozess dann eine Menge von Mapping-Vorschlägen der Kardinalitäten 1:1 und 1:n.

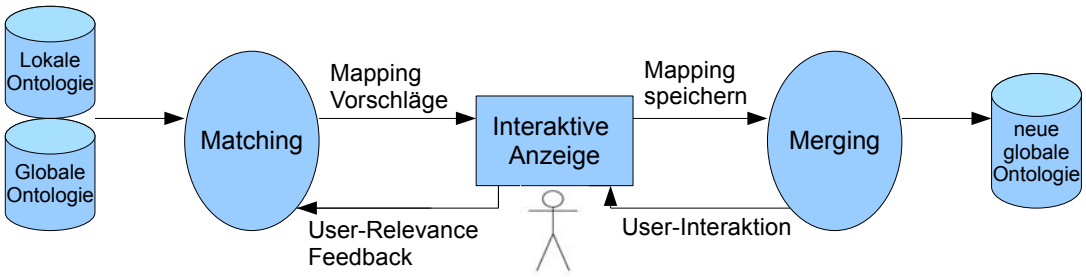

Abbildung 1: Die Referenz-Architektur von *IBOI*

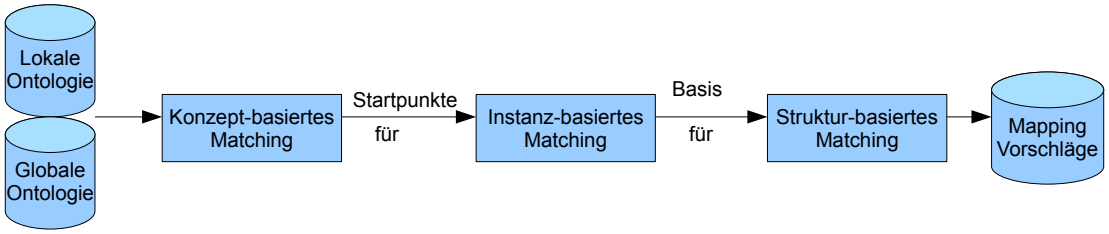

Abbildung 2: Der Matching-Prozess von *IBOI*

4.2 Neue instanzbasierte Methoden

Das Hauptziel von *IBOI* ist die Entwicklung von neuartigen instanzbasierten Methoden. Hierfür wurden schon einige Ansätze entwickelt, die im Folgenden kurz dargestellt werden sollen.

1. *Information-Retrieval-basierte Methoden:* Wenn man Instanzen paarweise miteinander vergleichen möchte, trifft man u.U. wieder auf das Problem von unterschiedlichen Sprachen, die Verwendung von Synonymen etc. Es scheint also sinnvoll zu sein Methoden zu entwickeln, die Instanzen mit Hilfe von Methoden des Information Retrieval manipulieren, wodurch sie besser vergleichbar werden. Instanzen können durch die Verwendung des Stemmings z.B. auf ihre Grundformen reduziert werden, oder man verwendet Thesauri und Wörterbucher um das Synonym/Homonym-Problem zu lösen. Zusätzlich könnte man bei längeren Beschreibungen Schlüsselwörter extrahieren und nur diese vergleichen. Der Vergleich der normalisierten/modifizierten Instanzen wird dann mit konventionellen String-Matching-Methoden durchgeführt.

2. *Feature Extraktion:* Die Extraktion von verschiedenen Merkmalen aus den vorhandenen Instanzen kann ebenfalls genutzt werden um Ähnlichkeiten zwischen Konzepten zu berechnen. Für stringbasierte Attribute kann man z.B. Merkmale wie Durchschnittslänge, die Verteilung der Zeichen, die Menge der verwendeten Sonderzeichen oder die Häufigkeitsverteilung verschiedener Worte oder Muster als reguläre Ausdrücke bestimmen.

   Für numerische Attribute lassen sich Merkmale wie z.B. der Durchschnittswert und die Durchschnittslänge, die Häufigkeitsverteilung der möglichen Werte, die Anzahl der Nachkommastellen oder der kleinste/größte Wert berechnen.

Die Berechnung der einzelnen Feature-Werte ist kein großer Aufwand, die Auswahl eines geeigneten Ähnlichkeitsmaßes dagegen ist schwieriger. Das zu verwendende Ähnlichkeitsmaß muss in der Lage sein, die Relevanz der einzelnen Feature mit einzubeziehen. Die Probleme sind z.B.: Was bedeutet es für die Ähnlichkeit zweier Konzepte, wenn der Durchschnittswert gleich ist? Was passiert, wenn die Attribute z.B. unterschiedliche Maßeinheiten verwenden, wodurch die Feature-Werte beeinflusst werden? Diese Fragestellungen müssen bei der Entwicklung einer geeigneten Ähnlichkeitsfunktion beachtet werden.

3. *Clustering-basierte Methoden:* Ein weiterer Ansatz besteht in der Verwendung von konventionellen Clustering-Algorithmen. Anhand der Instanzen der einzelnen Konzepte werden reguläre Ausdrücke gesucht, welche das Konzept am besten beschreiben, d.h. für jedes Attribut eines Konzeptes wird ein regulärer Ausdruck bestimmt. Diese werden in Vektoren zusammengefasst und mit Hilfe von (kategorischen) Clustering-Verfahren geclustert. Konzepte mit ähnlichen Vektor-Repräsentationen befinden sich in einen Cluster und sind demnach wahrscheinlich ähnlich, so dass man innerhalb der Cluster 1:1 bzw. 1:n Matches finden kann. Besonders zu beachten bei diesem Verfahren ist, dass es bei der Erstellung der regulären Ausdrücke zu keinem Overfitting kommt, dass die Besonderheiten der Instanzen zum Ausdruck kommen und man einzelne Ausreißer ignoriert.

4. *Stochastische Methoden:* Wie bspw. in [DMDH04] beschrieben, können auch stochastische Methoden für Matching-Verfahren verwendet werden. Mit den Instanzen eines Konzeptes wird ein Klassifikator trainiert, der die Instanzen eines anderen Konzeptes klassifiziert. Diese Methode war, wie in Kapitel 3 gezeigt, nicht zuverlässig, so dass über Verbesserungsmöglichkeiten nachgedacht werden soll. Man könnte die Klassifikatoren beispielsweise anhand von bereinigten Daten berechnen oder diese nur mit einer Teilmenge der Instanzen trainieren usw.

5 Ausblick

Die in Kapitel 4.2 vorgestellten instanzbasierten Methoden müssen detaillierter entwickelt und verbessert werden. Anschließend werden geeignete konzept- und strukturbasierte Methoden ausgewählt um den in Abbildung 2 dargestellten Prozess zu vervollständigen. Besondere Aufmerksamkeit muss auch auf die Entwicklung der Relevance-Feedback Mechanismen gelegt werden.

Literatur

[DMDH04] A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Ontology matching: A machine learning approach. In *Handbook on Ontologies*, pages 385–404. 2004.

[DR02] H. Do and E. Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In *VLDB 2002, Hong Kong, China*, pages 610–621, 2002.

[EM07] D. Engmann and S. Maßmann. Instance Matching with COMA++. In *BTW 2007, Workshop Proceedings, Aachen, Germany*, 2007.

[ES04] M. Ehrig and S. Staab. QOM - Quick Ontology Mapping. In *INFORMATIK 2004 - Informatik verbindet, Band 1, Ulm*, pages 356–361. GI, 2004.

[ES07] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.

[RB01] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, pages 334–350, 2001.

[Zai07] K. Zaiß. Query Reformulation and Ontology Integration in P2P Databases. Masterarbeit, Institut für Informatik, Heinrich-Heine-Universität Düsseldorf, 2007.

[Zai08] K. Zaiß. Ontologie-Matching: Überblick und Evaluation. *Datenbank-Spektrum*, (24):12–17, 2008.

Eine Datenbanklösung für Modellwiederverwendung in der Systembiologie

Dagmar Köhn[1], Andreas Heuer[1], Carsten Maus[2]
Universität Rostock, Institut für Informatik,
[1]Lehrstuhl Datenbank- und Informationssysteme,
[2]Lehrstuhl Modellierung und Simulation,
{dk103,ah,cm234}@informatik.uni-rostock.de

Zusammenfassung

Aus der steigenden Komplexität und Anzahl biologischer Modelle in der Modellierung und Simulation ist die Notwendigkeit entstanden, Modelle effizient zu speichern und zu verwalten. Biologische Prozesse werden modelliert und auf immer größere biologische Einheiten mit immer feinerer Granularität abgebildet. Daraus resultiert die Forderung nach effizienter Modellkopplung: Wie können ganze Modelle gekoppelt werden um Mehrebenen- und multi-granulare Modelle zu bauen? Im direkten Zusammenhang damit ist neben der Modellspeicherung ebenfalls eine Lösung für die Speicherung von Modell*komponenten* notwendig. Das folgende Papier gibt einen ersten Ausblick auf mögliche Antworten.

1 Einleitung & Motivation

Diese Forschungsarbeit ist Teil des vom DFG geförderten, integrierten Graduiertenkollegs (GRK) „**D**ie **i**ntegrative **E**ntwicklung von **Mo**dellierungs- und **Si**mulationsmethoden für **r**egenerat**i**ve **S**ysteme“ (dIEM oSiRiS, (16)). Das internationale Kolleg setzt sich aus Biologen, Medizinern und Informatikern zusammen. Ein Hauptanliegen des GRKs ist die Entwicklung von neuen Modellierungs- und Simulationsmethoden für die Beschreibung regenerativer Systeme. Als Anwendung für ein solches System dient der Wnt Signalweg bei der Differenzierung zu neuronalen Zellen. Im Rahmen von dIEM oSiRiS werden Experimentaldaten erfasst, welche dann zur Entwicklung von Modellen des Signalweges verwendet werden und schließlich zum besseren Verständnis des Systems und der Planung neuer Experimente führen. Die entwickelten Modelle sollen durch effiziente Speicherung wiederverwendbar gemacht werden. Außerdem steht im Fokus der Arbeit des GRKs die Entwicklung von Multi-Komponenten-Modellen, d. h. die Kopplung von Modellen, wobei die einzelnen Komponenten in verschiedenen Modellierungsformalismen beschrieben sein können. Zusätzlich sollen Informationen über die einzelnen Modellkomponenten abgelegt werden. Verdeutlicht wird das Szenario in Abbildung 1. Ziel ist es, zu bestehenden Modellen (oder Modellkomponenten) aus der Datenbank „passende“ Modelle (oder Modellkomponenten) zu erfragen. Die Anforderung ist neben der syntaktischen Kopplung mit Fragestellungen wie: „Kann die Ausgabe eines Moduls auf die Eingabe des nächsten abgebildet werden?“ auch die semantische Kopplung mit Fragestellungen wie: „Gelten für die Modellkomponenten gleiche biologische Randbedingungen?“. Die Entscheidung welche Modellkomponenten „passend“ sind soll über den Vergleich der Eigenschaften realisiert werden. Eine Voraussetzung für die Extraktion von Informationen aus Modellkomponenten ist somit eine einheitliche (syntaktische und semantische) Repräsentation der Modellierungsformalismen. Die Verwendung XML basierter Formate erscheint dafür sinnvoll. Da zur Modellierung bestimmter biologischer Systeme jedoch verschiedene Formalismen unterschiedlich gut geeignet sind, existiert kein universeller Modellierungsformalismus. Zwar hat sich in den letzten Jahren mit der XML basierten *Systems*

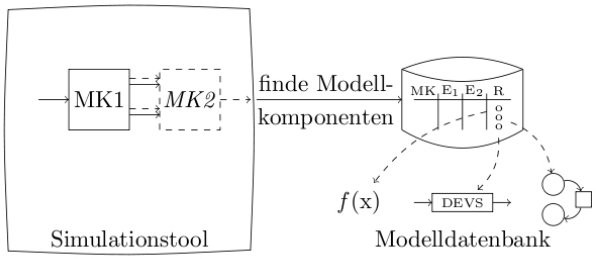

Abbildung 1: Speicherung und Anfrage von Modellen verschiedener Formalismen: *MK - Modellkomponente, E - Modelleigenschaft, R - Referenz auf die Modellkomponente*

Biology Markup Language (SBML, (4)) ein Quasi-Standard für die Beschreibung von kontinuierlichen *Differentialgleichungssystemen* (Ordinary Differenzial Equation, ODE) etabliert, dieser deckt jedoch nur einen stark eingegrenzten Bereich der Systembiologie ab. Insbesondere die Unterstützung diskret-ereignisorientierter Formalismen rückte in letzter Zeit verstärkt in den Fokus der Forschung (3). Beispielsweise eignen sich *Petrinetze* als visueller Modellierungsformalismus gut zur Beschreibung von Reaktionsnetzwerken. Eine standardisierte Darstellung von Petrinetzen ist die *Petri Net Markup Language* (PNML, (1)). Auch das *π-Kalkül* findet für die Modellierung biologischer Systeme immer mehr Anwendung. Es beschreibt ein System als eine Menge von Prozessen und Prozessinteraktionen, welche über (lokale und globale) Kanäle definiert sind. Eine mögliche Beschreibungsprache für das π-Kalkül ist die π Markup Language (πML, (8)). Als weitere diskret-ereignisorientierte Modellierungssprachen seien die zustandsbasierten Formalismen *State Charts* und *Discrete Event System Specification* (DEVS) genannt. Sie definieren ein Modell als Objekt, welches sich in verschiedenen Zuständen befinden kann und diesen aufgrund interner und externer Ereignisse wechselt. DEVS-Modelle sind modular hierarchisch aufgebaut, wobei zwischen gekoppelten und atomaren Modellen unterschieden wird. Die Kopplungen untereinander sind explizit über Ein- und Ausgabeports definiert. Während für State Charts ein Beschreibungsstandard existiert (State Chart XML, (SCXML) (17)), sind für DEVS lediglich erste Ansätze zur standardisierten Beschreibung veröffentlicht (5; 14).

2 Existierende Ansätze zur Modellspeicherung

Die Speicherung biologischer Modelle kann auf verschiedene Weisen geschehen. Als Vorbedingung für die Speicherung wurde die Speicherunterstützung auf die in Abschnitt 1 genannten Formalismen und Notationen beschränkt. Es ergeben sich daher folgende, im Anschluss ausführlich vorgestellte Speichermöglichkeiten:

1. Superformalismus *(Identifizierung eines allgemeinsten Formats und Transformation in dieses; einheitliche Speicherung)*
2. Native Speicherung *(Direkte Speicherung der XML-Struktur der Modelle)*
3. Black Box Speicherung *(Ganzheitliche Speicherung der Modelle und deren Semantik)*
4. Hybride Speicherung *(Speicherung der Modelle als BLOB; Extraktion der Formalismuskomponenten und Speicherung der XML-Anteile in einem XML-Datentyp)*

2.1 Superformalismus

Eine Speichermöglichkeit ist die Umwandlung der verschiedenen Formalismen in einen gemeinsamen Superformalismus. Dieser kann entweder durch einen der existierenden Formalismen re-

präsentiert werden oder aber die Vereinigungsmenge existierender Formalismen sein. Zur Transformation von Modellierungsformalismen ineinander gibt es bereits einige Vorschläge, so etwa für die Transformation von SBML in das π-Kalkül (2), oder von SBML in PNML (15). Bisher sind die existierenden Transformationen jedoch nicht vollständig. Somit sind die Transformationsergebnisse nicht genügend genau für eine Modellwiederverwendung. In (7) wurde der Versuch unternommen, XML - und OWL basierte Beschreibungssprachen für biologische Prozesse auf OWL-Ebene zu integrieren. Dies ist allerdings nur bedingt gelungen, da die syntaktischen Informationen in den Modellen nicht ausreichen um Aussagen über die Funktion und den (semantischen) Inhalt des Modells zu treffen.

2.2 Native XML-Speicherung

Da für den Großteil der in Abschnitt 1 vorgestellten Modellformalismen bereits standardisierte XML-Repräsentationen existieren, bzw. verstärkt an ihnen gearbeitet wird, erscheint die native Speicherung der Modelle in einer XML-Datenbank intuitiv. Es wird die syntaktische Angleichung der Formalismen erreicht und es kann auf einheitliche Anfrage- und Speichertechniken zurückgegriffen werden. Leider unterstützen aktuelle XML-Datenbanken noch nicht viele Suchfunkionen und sind im Vergleich zu relationalen Speicherungstechniken ineffizient. So unternahm die BioModels Datenbank (10) den Versuch der nativen Speicherung von SBML-Modellen basierend auf Xindice, welcher an schlechter Performanz gescheitert ist. Da native XML-Speicherung außerdem das Semantik-Problem noch nicht löst, soll vorerst von dieser Speichervariante abgesehen werden.

2.3 Black Box Speicherung und semantische Annotation

Bei der Black Box Speicherung wird das Modell als BLOB oder als Verweis auf das Dateisystem gespeichert. Um die Anfragbarkeit der Modelle zu unterstützen, werden diese zusätzlich mit semantischen Informationen annotiert. Das ist ein Lösungsansatz, der für SBML-Modelle bereits existiert: In der BioModels Datenbank werden die SBML-Modelle als Ganzes abgelegt und zusätzliche Informationen über die modellierten Reaktionen und Spezien, sowie über die biologische Publikation, die dem Modell zugrunde liegt usw. gespeichert. An einem Standard für die Annotation der Modelle, unter anderem mit ontologischem Wissen, wird derzeit gearbeitet (MIRIAM, (12)). So wurden im vergangenen Jahr Ontologien entwickelt, die die biologischen Modelle um Informationen über die genutzten biologischen Entitäten (SBO, (11)), über die Dynamik der Modelle (TeDDY, (13)), sowie um Informationen über die für die Simulation der Modelle verwendeten Algorithmen (KiSAO, (9)) erweitern.

2.4 Hybride Speicherung

Die hybride Speicherung von XML-Dokumenten ist in (6) beschrieben. Einige Teile des XML-Dokumentes werden hierbei in relationale Strukturen umgewandelt, während andere Teile in ihrer XML-Struktur belassen und somit direkt in die Datenbank gespeichert werden. Die XML-Dokumentteile können dann mit XML-Anfragetechnologien, wie z. B. XPath, angefragt werden, während für die relationalen Anteile SQL als Anfragesprache genutzt werden kann. Die Vorteile der hybriden Speicherung sind mehr Effizienz und Nutzung der ausgefeilten relationalen Speicherungstechniken für relational gespeicherte Dokumentteile und die gleichzeitige Realisierung von Strukturanfragen auf den XML-Dokumentteilen.

3 Hybride Speicherung Biologischer Modelle

Ein erster Versuch der effizienten Speicherung biologischer Modelle soll mit dem hybriden Ansatz unternommen werden. Dazu zeigt Abbildung 2 den Entwurf für eine mögliche Speicherstruktur:

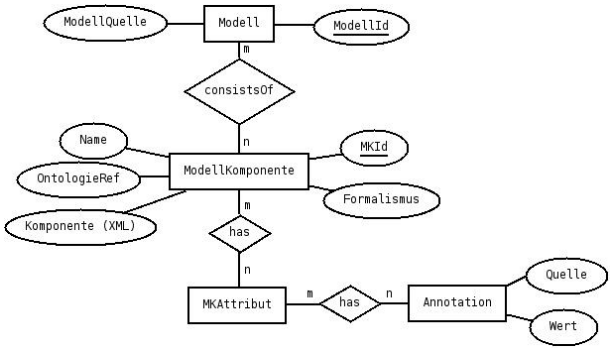

Abbildung 2: Speicherungsstruktur für die hybride Speicherung biologischer Modelle

Zusätzlich zu den eigentlichen Modellen mit deren ModellID werden die aus den Notationsformaten, z. B. PNML (siehe Abschnitt 1), extrahierbaren Formalismuskomponenten für spätere Anfragen nativ in XML gespeichert. Zu jeder Modellkomponente wird außerdem die ID und der Typ der Modellkomponente gespeichert. Der Typ der Modellkomponente ist besonders hilfreich für spätere Ranking-Funktionen, da die Kopplung verschiedener Modellkomponenten gleichen Formats der Kopplung von Modellkomponenten unterschiedlichen Formats vorgezogen wird. Optional kann für Modellkomponenten ein Name angegeben werden um besonders prägnante Teilmodelle zu kennzeichnen, z. B. „ATP-Produktion“.
Jede Modellkomponente ist durch bestimmte Formalismuskomponenten gekennzeichnet, die teilweise noch identifiziert werden müssen. Ein SBML-Modell setzt sich z. B. typischerweise aus einer Anzahl von Speziendefinitionen, Reaktionspartnern und -produkten, sowie einer oder mehrerer Definitionen von Kompartimenten (örtlich getrennten biologischen Räumen, z.B. Zellkern oder Zellmembran) zusammen. Vorschläge für extrahierbare Formalismuskomponenten diskreter-ereignisorientierter Formalismen werden beispielsweise in (3) gegeben. Alle identifizierten Modellkomponenten sollten später einzeln anfragbar sein um eine besonders effektive Modellkomponentenwiederverwendung zu gewährleisten. Die relationale Speicherung der Daten macht sie indexierbar und effizient anfragbar. Durch die Speicherung der einzelnen Komponenten in XML bleibt deren Struktur unverändert erhalten. Somit ist später eine Strukturanalyse von Modellkomponenten auf Basis von XPath/XQuery denkbar.
Des Weiteren sollen die semantischen Informationen, z.B. über das dynamische Verhalten des Modells, relational abgelegt werden. Dies ist insbesondere für die Modellkomposition von Bedeutung. Semantische Informationen über das Verhalten des Modells helfen bei der Suche nach und bei der Auswahl von passenden Modellkomponenten. Dazu ist z.B. eine Suche nach Modellen mit ähnlichem Verhalten denkbar, oder auch eine Suche nach Modellen, die mit demselben Simulationsverfahren simuliert werden können.

4 Ausblick

Im Verlauf der Doktorarbeit wird verstärkt an der Identifizierung und Evaluierung der Formalismuskomponenten gearbeitet, um eine ausreichende Indexierung und Dekomposition der Modelle auf Datenbankebene zu gewährleisten. Das in Abschnitt 3 vorgestellte Speicherungskonzept muss verfeinert und dann verifiziert und getestet werden. Zusätzlich muss evaluiert werden, ob die gespeicherten Meta-Informationen für das effiziente Retrieval von Modellen und Modellkomponenten ausreichend sind. Ziel der Arbeit ist schließlich die Unterstützung der Modellkopplung für die effiziente und flexible Ausführung von Simulationsmodellen in unterschiedlichen Modellformalismen.

Literatur

[1] Billington, J. ; Christensen, S. ; Hee, K. van u. a.: The Petri Net Markup Language: Concepts, Technology, and Tools. In: *Applications and Theory of Petri Nets 2003: 24th International Conference*. Eindhoven, Juni 2003, 1023–1024

[2] Eccher, C. ; Priami, C. : Design and implementation of a tool for translating SBML into the biochemical stochastic-calculus. In: *Bioinformatics* 22 (2006), Dezember, S. 3075–3081

[3] Ewald, R. ; Maus, C. ; Rolfs, A. ; Uhrmacher, A. : Discrete event modelling and simulation in systems biology. In: *Journal of Simulation* 1 (2007), S. 81–96

[4] Finney, A. ; Hucka, M. ; Le Novère, N. : *Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions*. 2003

[5] Janoušek, V. ; Polášek, P. ; Slavíček, P. : Towards DEVS Meta Language. In: *ISC Proceedings*, 2006, S. 69–73

[6] Klettke, M. ; Meyer, H. : *XML und Datenbanken*. Bd. 1. dunkt.verlag, 2003

[7] Köhn, D. : *A Schema Matching Architecture for the Bioinformatics Domain*, University of Linköping, Universität Rostock, Diplomarbeit, September 2006

[8] Köhn, D. ; John, M. : Making the π calculus exchangable – the π Markup Language (πML). In: *Computational Methods in Systems Biology, Edinburgh. Poster*, 2007

[9] Köhn, D. ; Le Novère, N. : *The KInetic Simulation Algorithm Ontology (KiSAO) - A Proposal for the Classification of Simulation Algorithms in Systems Biology*. Eingeladener Vortrag, SBGN Workshop, Japan, Januar 2008

[10] Le Novère, N. ; Bornstein, B. ; Broicher, A. u. a.: BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. In: *Nucleic Acids Res* 34 (2006), Januar. – Database issue

[11] Le Novère, N. ; Courtot, M. ; Laibe, C. : Adding semantics in kinetics models of biochemical pathways. In: *Proceedings of the 2nd International Symposium on experimental standard conditions of enzyme characterizations*, 2006

[12] Le Novère, N. ; Finney, A. ; Hucka, M. u. a.: Minimum information requested in the annotation of biochemical models (MIRIAM). In: *Nature Biotechnology* 23 (2005), Dezember, Nr. 12, S. 1509–1515

[13] Le Novère, N. ; Knüpfer, C. : TEDDY - A Terminology for the Description of the Dynamics of Bio-Models. In: *Foundations of Systems Biology in Engineering (FOSBE), Stuttgart. Poster*, 2007

[14] Röhl, M. ; Uhrmacher, A. : Flexible integration of XML into modeling and simulation systems. In: *Proceedings of the 38th Winter simulation Conference*, 1813–1820

[15] Shaw, O. ; Koelmans, A. ; Steggles, J. ; Wipat, A. : Applying Petri Nets to Systems Biology using XML Technologies. In: *Definition, Implementation and Application of a Standard Interchange Format for Petri Nets (ATPN), Bologna. Workshop*, 2004

[16] Uhrmacher, A. M. ; Rolfs, A. ; Frahm, J. : DFG Research Training Group 1387/1: dIEM oSiRiS - Integrative Development of Modelling and Simulation Methods for Regenerative Systems. In: *it - Information Technology* 49 (2007), Nr. 6, S. 388–395

[17] W3C: *State Chart XML (SCXML): State Machine Notation for Control Abstraction 1.0*. Working Draft, 2005. – `http://www.w3.org/TR/2005/WD-scxml-20050705/`

Database Caching: A Constraint-Based Approach Applied to XPath and XQuery Predicates

Mayce Ibrahim Ali, Theo Härder

University of Kaiserslautern, Postfach 3049, 67653 Kaiserslautern
{ali,haerder}@informatik.uni-kl.de

Abstract: Caching, in general, is important for databases to increase their scalability and performance behavior as well as to improve user-perceived latency (response time) and availability. The objective of this work is to design a caching scheme for XML databases supporting large distributed applications prevalent in the Web. For this reason, we introduce a new class of constraint-based database caching algorithms tailored to XML fragments. Those XML fragments are constructed from parameterized cache constraints, and their use is based on the concepts of *value-based subtree completeness* and *path completeness*. The cache constraints defined between or on XML fragments determine the elements and attributes of the corresponding document stored in the backend database server, which have to be kept in the cache. We describe in more detail how a client query is evaluated in the cache and how those constraints are exploited to guarantee the correctness and completeness of the query results.

1 Introduction

Database (DB) caching replicates a set of records stored and managed by the backend DB server in a so-called DB cache located close to the application in a frontend DB server to reduce the workload of the backend DB. Using declarative query languages such as SQL or XQuery, the goal is to enable query processing in the DB cache, which remains entirely transparent to the application. A class of caching techniques [1, 2, 6] maintains the cached records by so-called parameterized cache constraints. Specified by cache keys and referential cache constraints in [2], they control the filling of the cache with specific records when a reference to a parameter value – related to such cache constraints – causes a cache miss. Hence, they serve as a kind of semantic information or metadata concerning the cache contents and guarantee the complete execution of queries that are subsumed by such cache constraints. An XML database cache presented in [5] proposes a method for maintaining a semantic cache of materialized XPath views. The cached views include query results previously processed. This kind of XPath query/view mapping enables the reduction of tree operations to string operations for matching query/view pairs.

The remainder of this article is organized as follows: Section 2 describes the constraint-based DB cache used to specify the cache content and its completeness. Loading and evaluation of the cache is described in Section 3, while Section 4 concludes this contribution.

2 Constraint-based Database Caching

2.1 Definition of Constraints

The cache should be able to evaluate queries containing different types of XPath predicates. Those query predicates are simple path expressions with axes such as *parent, child, ancestor, descendant, …, name, and attribute test or wild-cards(*),* e.g. */bib/book[title= "XML"]*. Metadata delivers semantic information which enables the cache manager to correctly and completely answer client queries. In our approach, the cache includes metadata using a path synopsis, the information of which could be derived from the XML document or, if available, from its XML schema. The XML schema provides a means of defining structure, content, and semantics of the XML documents considered. In contrast, the path synopsis is a data structure that captures all paths of an XML document (and possibly statistical information) and represents them by

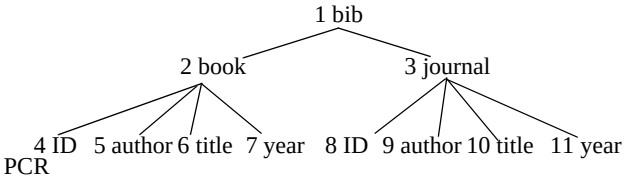

Figure 1: Sample path synopsis

path classes. Each path class of an XML document is a unique path in the path synopsis and stands for all paths from the root to the inner nodes or leaves having the same sequence of element/attribute names [3]. Each node t_n of a path synopsis has a unique number called path class reference (PCR) and can be reached (or denoted) by a path $p = e_1t_1e_2t_2\ldots e_nt_n$, where e_i refers to a child edge and t_i refers to an element or attribute node. Fig. 1 shows the path synopsis of a sample XML document; PCR=4 refers, e.g., to path class */bib/book/ID* .

We use a node labels based on a prefix-based labeling scheme [4], which efficiently supports navigational and declarative query processing. Furthermore, it enables dynamic insertions without relabeling [3]. Each document node has a unique Dewey number, and – given this number together with the PCR of its path class – the entire path of this document node can be reconstructed.

The cache stores subtrees (fragments) of the XML document considered where cache constraints control cache loading, query evaluation, and cache removal. We developed a class of constraints to guarantee the correct and complete evaluation of specific predicates *q*. The so-called predicate completeness means that all XML fragments required to process *p* are in the cache. The constraints specify the possible cache content and decide whether a predicate is evaluated in the cache or sent to the backend DB. Path constraints refer to paths and nodes defined in the path synopsis. When applied to the XML document, they specify sets of subtrees. To limit their number, we may add value-based restrictions which address the contents of an XML document and determine whether a subtree satisfying a path constraint qualifies. The standard option is to load entire subtrees qualified. If those subtrees are broad and most of their paths (data) are not needed for query evaluation and result construction, we may add an optional filter limiting the subtree size. The basic terms for constraint specification are defined as follows:

- **A Path Constraint** defined as a path class restricts the number of subtrees loaded in the cache. The path constraint guarantees the availability and completeness of the cached subtrees and comes in two forms:
  - **Absolute Path** is a path constraint starting from the document root to an arbitrary element node called subtree root. This path is the head part of the path class *p* and represented as $\sigma_{(e_1t_1e_2t_2\ldots e_it_i)\subseteq p}(XMLdoc)$, where e_j refers to a child or descendant edge, and t_j is an element node. t_1 specifies the document root and t_i the subtree root in the path synopsis. Hence, the identified subtree root determines which set of subtrees are loaded in the cache. The path */bib/book* is an absolute path as shown in Fig. 2.
  - **Relative Path** is a path constraint starting from an arbitrary element node called subtree root (except the document root) and ends at a leaf node. This path is the continuation of path class *p* and follows the absolute path. The relative path represented as $\sigma_{(e_it_ie_{i+1}t_{i+1}\ldots e_zt_z)\subseteq p}(XMLdoc)$, where e_i is a descendant edge, and (e_{i+1}, ..., e_z) refers to a child edge. The subtree root is represented by t_i and the leaf node by t_z (fill node). This path is the core path of the subtree and leads to a leaf node. *//book/author* is a relative path as illustrated in Fig. 2.
- **Fill Nodes** are specified for path constraints as leaf nodes at the end of a relative path. Each fill node may have several values stored in the backend database. These values specify the subtrees involved in cache loading (and removal) processes. In Fig. 2, *author* is the fill node of the path */bib/book/author*.
- **Candidate values** serve to restrict the domain of a fill node. They indicate subtrees to be potentially cached; hence, values of a fill node should be only specified as candidate values if they are frequently referred to by client queries. When referenced and not already in the cache, the cache manager loads all subtrees holding a currently referenced candidate value. As we see in Fig. 2, *Knuth* is a candidate value.

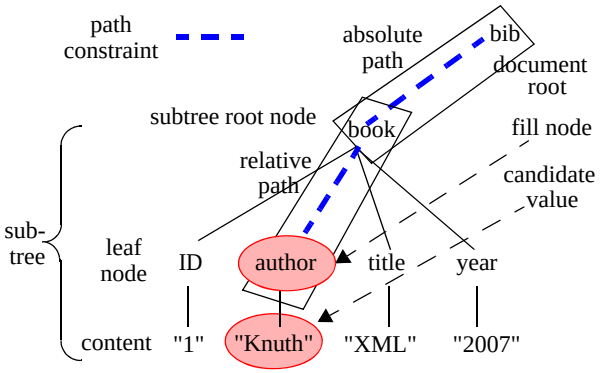

Figure 2: Ilustration of concepts

- **An XML fragment cache** is empty at the start of a processing period. The specified constraints control which subtrees will be loaded in the cache when a path constraint is activated and a candidate value is referenced by a query. The related path constraint determines the subtree root and, in turn, the subtrees of the XML document to be loaded in the cache.

2.2 Forms of Completeness

Predicate completeness for a query *q* has to be guaranteed in the cache to evaluate correct and complete query results for *q*. Having introduced the basic terminology, we are now able to define various specific forms of completeness which can be achieved by the cache manager.

2.2.1 Path Completeness

A path *p* is path complete, if all subtrees (in the backend document) needed to evaluate path *p* are in the cache. When the cache is empty or is not path complete for *p*, an incoming query, whose predicate is matching or subsumed by a path constraint *p*, triggers all subtrees satisfying *p* to be loaded in the cache. As a result, the cache can guarantee the completeness of this path.

Example 1.Consider path *p = /bib/book/author* is a path constraint. The cache is able to evaluate query *q* locally, if *q* matches with *p*. Then, all complete subtrees needed to evaluate *q* should be in the cache or should be loaded from the backend. For instance, query *q = /bib/book/author[.="Knuth"]* matches path *p = /bib/book/author* and, hence the cache can evaluate *q* locally. As a result, the cache returns all subtrees matching *p* and holding content *Knuth*.

In general, the cache is able to locally evaluate several types of queries subsumed by a path constraint. Consider as path constraint *c = /bib/book* and as cache content all subtrees whose subroots are *book*. Then, the cache can answer all queries subsumed by *c*, such as the following: */bib/book/author[.="Knuth"], /bib/book[author="Knuth"]/year[.="2007"*], */bib/book/title[.="XML"]* and */bib/book/author[.="Mueller"]*. Obviously, some constraints may cause enormous storage consumption in the cache, because all complete subtrees for the given constraints have to be loaded. Using pure path constraints (implicitly specifying the fill nodes' domains as candidate values), there is no way to restrict cache filling with qualified subtrees, even if the cache may not need all of them. This may produce unwanted overhead, which is augmented if updates on such subtrees occur in the backend. Then, the cache needs to update the subtrees affected to keep the states of cache and backend consistent (even if they are not used for query evaluation). For these reasons, we refine our approach to limit number of cached subtrees.

2.2.2 Value-based Subtree Completeness

Subtree load can be restricted when desired values for leaf elements are specified. Hence, we extend a path constraint definition with a value constraint to obtain a value-based path constraint *p*. When subtrees are loaded in the cache, only those subtrees are considered having value *w* in the specified leaf element *n*. This refinement is defined as follows:

Value-based subtree completeness: A value *w* for a leaf element *n* in a path constraint *p* is value-based subtree complete if all paths $\sigma_{(e_1t_1 \ldots e_it_ie_st_s[\ldots e_zt_z = "w"]) \subseteq p}(XMLdoc)$ are in the cache. Here e_j refers to a child or descendant edge, t_j to an element or attribute, t_1 to the document root, t_s to the subtree root node, and t_z to the leaf element *n*.

Example 2. Assume constraint *c* with path *p = /bib/book/author* and value *w = Knuth* for leaf element *n = author*. The cache can evaluate all queries that match with (or are subsumed by) *p*, *n* and *w*. For instance, query *q = /bib/book[author="Knuth"]* matches path constraint *p*, node *author* matches *n* and value *Knuth* matches *w*. If all related subtrees are in the cache, all queries subsumed by *c* can be evaluated as, for instance, the following examples: */bib/book/author[.="Knuth"]*, */bib/book[author="Knuth"]/title*, and */bib/book[author="Knuth"]/title[.="XML"]*.

For our path constraint specifications, we choose some path classes as path constraints thereby exactly defining their subtree node, fill node, and candidate values. Therefore, the path constraint is used as a general form to answer the queries that are subsumed by the path constraint. The cache can provide all the subtrees under the subtree root of a path constraint which satisfy the query. Let us go back to our example; if we have *q=/bib//author[.="Knuth"]* where the leaf node *author* appears in the document in a different sequences or paths. We see some path classes containing the leaf node *author*: */bib/journal/author* and */bib/book/author*. To enable the cache to answer all queries related to *author* when *author*

appears in several path classes, all subtrees holding *author* must be loaded. Hence, appropriate path constraints related to those path classes have to be specified. The path synopsis helps the cache to conclude and reconstruct the original paths of *author* and find the sequence of the nodes included in the subtree. Because of the restricted number of subtrees to be loaded, the cache will also need fewer update operations than the first approach.

2.2.3 Referential Constraints

Cache use will be considerably enhanced if the cache is able to also evaluate join predicates. Here, we introduce a referential constraint (RC) to specify subtrees to which join predicates are applied. By using a path synopsis, an RC is defined between two leaf nodes in disjoint subtrees.

Referential Constraint RC: A referential constraint is defined between leaf nodes of two subtrees in the cache belong to different path classes or constraints. The source subtree has a source node called *Node1*, which is an attribute (ID or IDREF) or element node and has a join partner, the target node *Node2* which is an attribute (ID or IDREF) or element node in the target subtree. An RC (*RC:Node*$\rightarrow$ *Node2)* is satisfied if and only if all the values *w* in *Node1* are value-based subtree complete in *Node2*.

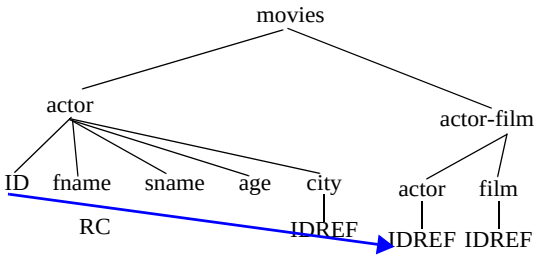

Figure 3: Referential constraint on a path synopsis

Example 3. *(/movies/actor/@ID)*$\rightarrow$*(/movies/actor-film/actor/@IDREF)* is an RC, refers to specific nodes (ID and IDREF) of a path synopsis, has the ID attribute as source node in the subtree *actor* and the IDREF attribute as target node in the subtree *actor-film* specifying the join partners as shown in Fig. 3. For example, if the attribute with the value ID="*w*" is in the cache, then all it's join partners with the attribute IDREF should be in the cache. Thereby the condition of the RC is satisfied. The RCs join the subtrees and can guarantee the completeness of joined values. According to the definition of the referential constraint which ensures whenever we find a value *w* in the content of *Node1*, we should also find all join partners of *w* in *Node2* satisfying *Node1=Node2*. Hence, we achieve completeness of subtrees.

2.2.4 Range predicates

Queries referring to a range of values using a range predicate can be evaluated in the cache, if the cache manager makes sure that all values in a specified range are value-based subtree complete. Such predicates must be restricted to a domain having ordered values. The closed interval [a,b] for such a domain is represented by two parameters a and b $-\infty \le a \le b \le +\infty$, or as a half-open interval [a,b) and (a,b] or an open interval (a,b). Then, the completeness condition is given by the following definition:

Interval Completeness: An interval *t* is said to be interval complete for a range of values of a cached node (*Node1*) if and only if all the subtrees of $\sigma_{(//Node1 \in t)}(XMLdoc)$ are in the cache. We determine the path class where *Node1* is a leaf node and load all subtrees holding *//Node1* $\in t$. All values in *t* are value-based subtree complete in the cache. A range predicate in a query can take several forms using the relationships $\Theta = \{>, <, \geq, \leq, =, \neq\}$.

Example 4. */bib/book[year>"2005"and year<"2007"]* represents a range-based path constraint.

3 Evaluation and Loading Process in the Cache

Before the cache starts to evaluate a query locally, the cache needs to check whether or not the value exists and satisfies the constraints (or making it complete). As we mentioned in Section 2, the cache depends on semantic information to find out which path is used as a path constraint and which path will be enforced by the constraints to be complete. The cache uses a simple query to check if the value exists in the cache; this process is called probing. If a probe query confirms that the value exists in a complete path (path constraint), we can guarantee that the value in the subtrees of this path is value-based subtree complete. For example, a probe query confirming the presence of value *Knuth in the cache* is *(fn:count (/bib/book/author[.="Knuth"])*. We describe the probing strategy by using different examples.

Example 5. The steps to evaluate query *q = //author[.="Knuth"]*:

- If the value *Knuth* exists in the cache:
  - First, the path is reconstructed using the path synopsis identifier (PCR) and the Dewey number. In this case, the path classes are */bib/book/author* and */bib/journal/author*.
  - The cache checks if this value is complete or not, by checking if the path is a path constraint, the node is a fill node and the value is a candidate value; then the value should be complete in the cache. The cache reconstructs the subtrees holding this value to provide the result of the query.
  - If the path is not a path constraint, the cache checks if the leaf node holding *Knuth* has an incoming RC from other nodes, when the leaf node is a target node. Then the cache checks if the value exists in the source node. If so, then it should be complete in the target node (RC definition).
- If the value is not in the cache, the path will be reconstructed from the path synopsis. Then the cache checks if the path matches the constraint. Then the cache will decide to load the subtrees into the cache or supply them to the client directly. If the path and value do not belong to the constraints, the cache will not load this value and supply the result directly from the backend to the client.

Example 6. The client sends the query */bib/book[author="Knuth"]/title=[.=”XML”]*. As we said before, the constraint enforces the loading of the candidate values of a fill node in the cache and also loads the needed sibling nodes of the fill node as a complete subtree. If a probe query confirms that the value *Knuth* exists in the cache, then the cache guarantees the completeness of this value *Knuth* and the sibling node *title*. The cache uses the equality predicate *(author="Knuth")* with the candidate value (*Knuth* is always complete) as an anchor, to restrict the number of the subtrees holding the element node (*title*). The complete values in the subtree are used as an entry point for the query in the cache. As we explained before section 2.1, the value *Knuth* is a candidate value in path constraint and is enforced to be complete.

4 Conclusion

We introduced special constraint types for the XML document in the cache. Those constraints guarantee correct evaluation of XPath and XQuery and safe loading of XML fragments when new instances are demanded. The constraints are adaptive, depending on the client’s usage. We used path classes to specify path constraints with their related fill nodes and candidate values. The cache manager loads all child- and descendant nodes under the subtree root having in a fill node a candidate value that has been referenced by a client query. The XML fragments can be joined together by using ID and IDREF attributes or type-compatible element values as defined in the corresponding XML Schema. Therefore, RCs based on the join between (ID, IDREF) or (element1, element2) pairs enforce the cache manager to load all counterpart values as soon as an ID, IDREF or element value enters the cache. Then, the cache manager is able to connect all join partners (subtrees) based on equal values of the RC attributes in the document. For some queries referring to a descendant axis, the cache needs to choose one or more path classes as path constrains to guarantee the complete result of those queries. As we mentioned before, to answer the query *q=//author*, the cache chooses all the paths classes holding *author* as path constraint and loads all the required XML fragments that satisfy *author*. Using these types of constraints, the cache can guarantee the correctness of several types of queries. In the future, we will focus on applying our constraints in the presence of delete and update operations in the cache and the backend.

References

[1] Altinel, M., Bornhövd, C., Krishnamurthy, S., Mohan, C., Pirahesh, H., Reinwald, B.: Cache Tables: Paving the Way for an Adaptive Database Cache. Proc. VLDB Conf. 2003: 718-729

[2] Härder, T., Bühmann, A.:Value Complete, Column Complete, Predicate Complete - Magic Words Driving the Design of Cache Groups, in: The VLDB Journal, 2008

[3] Härder, T., Mathis, C., Schmidt, K.: Comparison of Complete and Elementless Native Storage of XML Documents, Proc. IDEAS 2007, 102-113, Banff, Canada, 2007

[4] Haustein, M. P., Härder, T.: An Efficient Infrastructure for Native Transactional XML Processing, in: Data & Knowledge Engineering 61:3, 500-523, Elsevier, 2007

[5] Mandhani, B., Suciu, D.: Query Caching and View Selection for XML Databases. Proc. VLDB Conf. 2005: 469-480

[6] The TimesTen Team, Mid-tier caching: The TimesTen approach. Proc. SIGMOD Conf. 2002: 588-593

Adaptive Resource Management in a Native XDBMS

Yi Ou, Karsten Schmidt
{ou, kschmidt}@cs.uni-kl.de
University of Kaiserslautern, Germany

Abstract

The usual performance optimization efforts for DBMSs simply address transactional throughput only by analyzing the workload offline. Various resources (memory, CPUs, disks) are then allocated to specific workloads or environments. But today's DBMSs have to cope with dynamically changing and hardly predictable workloads. Configuration tuning done by administrators is exhausting and relatively slow regarding reaction time. Especially XML data processing leads to novel aspects when load balancing and autonomous configuring are addressed. We propose a lightweight framework adapted to XML databases which continuously monitors database state and resource consumption. A controller component triggered by the framework is used to adjust the database configuration online. Such a controller component may benefit from detected XML workload patterns and bottleneck analysis within the database system. Putting these approaches together should enable a native XML database management system (XDBMS) to instantly adapt its resources to momentary workload shifts or load peaks.

1 Introduction

Optimal performance of a computing system can not be achieved without maximized and well-balanced use of the available computing resources such as CPU and memory. In particular, exploring optimal utilization of resources is very important for database systems, which are becoming more and more resource-intensive. A commercial database system typically has hundreds of tunable parameters to be set correctly, depending on the working environment and available resources, to achieve optimal system performance. Yet such configuring efforts are exhausting, error-prone, and not "once and for all". There are many cases where the performance even of an initially optimal-configured database system (if this is possible at all) becomes suboptimal for various reasons such as:

- Shift of workload patterns (e.g. from DSS to OLTP) or growing number of users (e.g., drastic increase due to booming business);
- Change of available resources (e.g. hardware upgrade);
- Evolution of the database in terms of volume and schema.

Dealing with these problems manually, e.g., reconfiguring the system by DBAs, is often not acceptable with regard to the reaction time. Effective solutions for these problems require DBMSs to detect bottleneck resources and automatically adapt their usage. Software systems, such as DBMSs, can not decide how many resources are available, but only how allocated resources are used for their computing tasks. In [4], rules are presented for trading memory for I/O accesses and CPU time, to support economic operational decisions. Our basic idea is that scarce resources of one type can be traded for idle resources of a different type. We want to enable a system to

automatically respond to workload changes, so that all available resources are utilized to their full capacity – by trading surplus resources for resources which are currently in short supply (thus becoming the bottlenecks).

Similar problems are also faced by XDBMSs [8]. Moreover, XML data processing leads to novel aspects regarding resource usage [9, 7]. XTC (XML Transaction Coordinator) is a native XDBMS prototype [5] that supports all standard XML interfaces, such as XQuery, DOM, and SAX. XTC will be the testbed for the ideas discussed in this paper. The remainder is organized as follows: Section 2 discusses the potentials of optimizing XDBMSs by means of adaptive resource management; Section 3 introduces a framework that we implemented to check out our ideas; In Section 4, we summarize and outline perspectives for future research.

2 Potentials and Challenges

There are many possibilities for making trade-offs concerning the usage of resources. Here we consider the three most important resources and trade-off relations between them: CPU time, memory, and I/O bandwidth. A classical example is trading memory for saving I/O bandwidth by means of caching. However, the CPU overhead for managing the buffer pools was often ignored, which may become significant under certain circumstances, especially when the CPU is the bottleneck for the system performance. In the past, I/O bandwidth was often blamed to be the performance bottleneck in database systems. But this may not hold forever, because hardware technologies are evolving rapidly. Another example is the utilization of surplus CPU time for other concurrent tasks, when DBMS processing is delayed by I/O operations. But the additional memory costs for managing processes or threads should be taken into consideration. As an example, the triangle in Figure 1 illustrates these relationships, with arrows pointing to the direction of resource saving and the dash-lined arrows denoting the possible overhead.

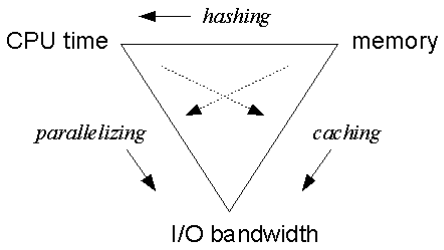

Figure 1: Resource Triangle

2.1 Query Processing

Similar to query processing in relational DBMSs, queries in XTC are internally processed by a set of physical operators. For a certain function, there are often several equivalent physical operators and, in turn, several variants based on alternative algorithms may exist for a single physical operator. These implementations may differ in their resource consumption, while providing the same functionality. For example, an operator only exploiting indexes might be substituted by an operator requiring partial document scans, thus leading to a different CPU load and memory consumption. Many alternatives, e.g., hash-based joins [7] and holistic twig joins [6], exist for the structural join operator, one of the most important physical operators in XTC. Hash-based joins might be superior to other algorithms in terms of CPU time, but they often consume much more memory[7].

We expect much better query processing performance, if we enhance our query engine in such a way that it always chooses the most suitable physical operators based on current resource usage and availability. This is especially true in multi-user systems, because their resource consumption is much more critical than that of single-user environments. Of course, when dealing with various alternative physical operators, traditional query optimization techniques based on cost models, size of intermediate results, and data characteristics must be considered, as well as the necessary preconditions for different algorithms (e.g., index, ordering, etc.).

2.2 Memory Management

Memory resources in modern DBMSs are often managed by various memory pools where each pool serves for a certain function area (e.g., page buffering, sorting, transactions management, etc.). Such a segmentation frequently leads to unbalanced memory usage. For example, the page buffer might be filled up while the memory pool for sorting is hardly used. In such cases, the size of memory pools could only be reconfigured offline. On the other hand, several memory pools with different characteristics are mandatory regarding page-based mapping and cost-based optimizations. Heuristics to provide suitable memory pool sizes are often more helpful than strictly assigned boundaries. Such hints can be built into the logic of a system-wide *memory controller* that assigns equal-sized memory portions to the requesting components (see Figure 2). For this reason, memory pools can be restricted by minimum and/or maximum size constraints to give meaningful hints for the *memory controller*. Furthermore, locked and fixed memory areas (currently used by a transaction) have to be considered when memory repartitioning takes place. Exceeding a certain threshold in a memory pool triggers the *memory controller* to adapt the memory partitioning. The most important design goal is to improve the average memory usage, i.e., increasing the used space of all memory pools. In addition, to support cost-based optimization, the memory controller may have the ability to swap out memory pages to disk, before the OS' swap mechanism is activated. Such a functionality may prevent wrong estimations for algorithmic costs.

Buffer and cache management should also be adjusted to actual workloads and resource consumptions. Let's consider concurrently running transactions blocking disk accesses or stressing CPU. A smart replacement strategy may wait to flush modified pages or prefers to deallocate unchanged pages, even if they are referenced, to free memory. Depending on the costs of flushing modified pages and discarding/loading read-only pages, cache management should take current CPU and I/O load into account. In addition, flexible commit strategies (e.g., group commit) may improve the cache system performance to the extent possible.

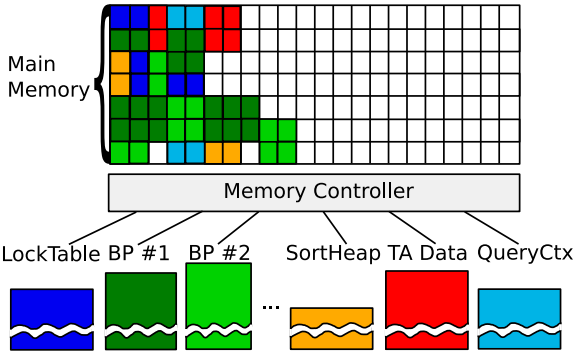

Figure 2: Memory Controller

2.3 Data Compression

Just as well as in traditional DBMSs, compression is also one of the most heavily used techniques in XTC (e.g., DeweyIDs [5]). Besides saving memory and disk space, the main objective of data compression is I/O reduction. The overhead of compression, i.e., the CPU time used for compression and decompression, are believed to be worthwhile for saving I/O bandwidth. However, if a higher degree of concurrency or parallelism is introduced, or if high-performance disk arrays are installed, CPU may become the bottleneck and I/O bandwidth may be, in turn, (partially) unused. It should be possible to switch compression on and off, depending on the current workload and resource consumption. But there is a "delayed side-effect" to be dealt with when using such an on-demand compression strategy, because the decision to store a piece of data uncompressed for saving CPU time may lead to a "punishment" of extra I/O costs in the future. To gain cost-effective compression behavior, additional influences resulting from memory usage, I/O, and CPU load must be considered, too.

2.4 Hardware Characteristics

Nowadays information systems are ubiquitous in a sense that enterprises without database-driven applications cannot be found. Nevertheless, there are hardly any identical deployments – due to varying hardware configurations and other conditions. An optimally performing database system must be individually configured by setting various parameters based on the existing hardware configuration. Although some deployment/configuration wizards are emerging, most database systems do not automatically respond to the changing hardware environment. Given the rapidly evolving hardware technology and the ever-changing user requirements, automatic adaptation becomes more and more urgent. The same is true for XML database systems, which might gain much more popularity soon.

An example is the emergence of large-volume flash disks, which can be used in several roles, e.g., as extended memory, as additional layer in the storage hierarchy between memory and I/O devices, as replacement or extension of traditional hard disks. That means, more complex hardware configurations will arrive in the near future. Flash disks are superior to traditional hard disks in terms of access times, energy consumption, etc. Database systems can benefit from these characteristics in many aspects. Nevertheless, getting the most out of these characteristics requires research in many functional DBMS areas, such as page movement, checkpoint processing, optimal page sizes, etc. as discussed in [3]. Furthermore, the number of processors and the sizes of L1 and L2 caches also add variation potential to hardware configurations, leading to an even higher complexity for deployment and performance tuning of database systems.

3 The Monitoring Framework

To build a resource-aware and self-adaptive XDBMS, we are following the monitor-diagnose-action paradigm for solving performance problems [1, 2, 8]. As the first step, we implemented an event-driven monitoring framework (XtcSam) for online monitoring of our native XDBMS.

We analyzed selected components of XTC and defined a set of metrics representing their runtime state. The architecture of XtcSam is shown in Figure 3. The XTC components under observation are instrumented so that they can generate events at runtime. A generated event is dispatched by the *EventDispatcher* to the listeners registered for this event. The listeners have to implement the *EventListener* interface and provide the core functions of XtcSam, such as updating of the various metrics and logging of events.

Besides the monitoring of XTC components, XtcSam also monitors the resource consumption of XTC with the help of Sun's JMX implementation. For example, XtcSam delivers CPU time and heap memory consumed by the XTC server. It also provides thread-level information such as the state of each thread (blocked, waiting, runnable, etc.) and its CPU usage.

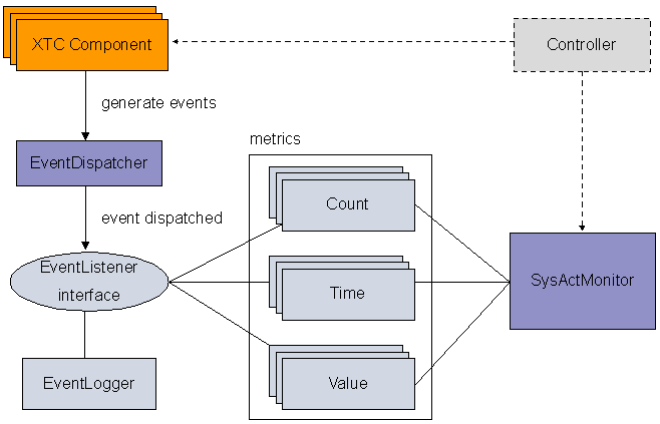

Figure 3: The Monitoring Framework XtcSam

The metrics and the *SysActMonitor* serve as the primary programming interface, which can be exploited by self-adaptive mechanisms, represented by the *Controller* component (in dashed lines). Proven by benchmark measurements, our implementation of the monitoring framework only causes an ignorable performance penalty [8].

4 Summary & Outlook

We argue that adaptive resource management is necessary for native XML processing and outlined several aspects that may lead to performance improvements for XML database systems. Some of the ideas discussed here, though in the context of XDBMSs, apply to traditional database systems as well. Although having various tuning parameters, current database systems do not pay enough attention to changeable hardware environments on their own. Adaptive resource management is highly desirable as well for broader application areas of database systems such as mobile devices, where the hardware configuration largely differs from that of a server environment, and in energy-saving mode, where computing resources may exhibit drastically different behavior.

Our XDBMS prototype provides an ideal platform for integrating and testing different approaches to adaptive resource management. However, the control of a DBMS over hardware resources is limited, because, to a large extent, they are managed by the OS. Furthermore, XTC is completely implemented in Java, which is another layer of abstraction between hardware and system and which provides its own mechanism for managing resources, e.g., via garbage collection. Major challenges we have to face are how accurate resource usage information for XTC can be extracted and how the influence of the platforms can be separated to derive correct estimations for the modeling of processing costs.

For the implementation of the self-adaptive mechanism, we will start with a *heuristic-based* approach, using simple metrics and rules, which can be derived with XtcSam and further experiments. The more heuristics are integrated into the system, the more complexity is added to compute the optimal configuration. Later, we will go further focusing on a *model-based* approach, e.g., using an abstract model for the memory hierarchy. Making trade-offs between various resources requires the alternatives to be quantitatively evaluated using proper metrics. The *benefit and overhead* of the adaptive mechanisms also have to be evaluated.

References

[1] Sanjay Agrawal, Nicolas Bruno, Surajit Chaudhuri, and Vivek R. Narasayya, *Autoadmin: Self-tuning database systems technology*, IEEE Data Eng. Bull **29** (2006), no. 3, 7–15.

[2] Nicolas Bruno and Surajit Chaudhuri, *To tune or not to tune? A lightweight physical design alerter*, Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), Seoul, Korea, September 12-15, 2006.

[3] Goetz Graefe, *The five-minute rule twenty years later, and how flash memory changes the rules*, Proceedings of the Third International Workshop on Data Management on New Hardware (DaMoN 2007), June 15, 2007, Beijing, China.

[4] Jim Gray and Franco Putzolu, *The 5 minute rule for trading memory for disc accesses and the 5 byte rule for trading memory for cpu time*, Tech. Report TR-86.1, HP Labs, May 22 1985.

[5] Michael Peter Haustein and Theo Härder, *An efficient infrastructure for native transactional xml processing*, Data Knowl. Eng **61** (2007), no. 3, 500–523.

[6] Haifeng Jiang, Wei Wang, Hongjun Lu, and Jeffrey Xu Yu, *Holistic twig joins on indexed XML documents*, (2003), 273–284.

[7] Christian Mathis and Theo Härder, *Hash-based structural join algorithms*, In Proc. EDBT Workshops, pp. 136-149, Munich, March 2006.

[8] Yi Ou, *Performance analysis and optimization of xml database systems exemplified by xtc*, Master's thesis, Technische Universität Kaiserslautern, Germany, 2008.

[9] Karsten Schmidt and Theo Härder, *An adaptive storage manager for XML documents*, Datenbanksysteme in Business, Technologie und Web (BTW 2007), Workshop Proceedings, 5.-6. März 2007, Aachen, Germany.

Processing and Optimizing Tree Pattern Queries in Native XML Database Management Systems

Sayyed Kamyar Izadi[1], Theo Härder[2], Mostafa S. Haghjoo[1]

[1]Iran University of Science and Technology, Tehran , Iran
{izadi, haghjoom}@iust.ac.ir
[2]University of Kaiserslautern, Postfach 3049, 67653 Kaiserslautern
haerder@informatik.uni-kl.de

Abstract: XML queries are based on path expressions where their elements are connected to each other in a tree pattern structure, called Query Tree Pattern (QTP). Therefore, the main operation in XML query processing is finding the node patterns in the document matching the given tree pattern. We have developed a novel method, called RG (result graph), which can selectively process the document nodes. In RG, unlike all previous methods, path expressions are not executed directly on the XML document, but a guidance structure, called *path synopsis*, is used to focus search and to avoid document access as far as possible. Enriched by information extracted from the *path synopsis*, a query execution plan, called SPSM, is generated by applying a path expression to the *path synopsis*. This structure enables RG, in contrast to existing methods, to limit access to document nodes that satisfy one of the leaf node conditions of the tree pattern and, furthermore, to those contributing to the final result.

1 Introduction

XML usage is increasing dramatically. There are many applications in areas such as science, biology, business, and, particularly, web information systems using XML as their data representation format. This growing trend towards XML confirms the need of XML database management systems (XDBMSs). Query processing is an essential functionality of any DBMS; this is especially challenging for XDBMSs, because XML documents combine tree structure with content. Both XPath and XQuery, the two most popular query languages in the XML domain, are based on path expressions. A so-called Query Tree Pattern (QTP) specifies a pattern of selection predicates addressing multiple elements in a path expression related by a tree structure. As a focal point of our discussion, these patterns include the most important query axes *parent-child* (P-C) and *ancestor-descendant* (A-D). To process XML queries, all fragments in the XML document identified by a given QTP have to be found, which is an expensive task, especially, when huge XML documents are involved.

2 State of the Art

The *Structural Join* is one of the first methods proposed to process XML path expressions [1]. By this method, path expressions are decomposed into several binary P-C, A-D relationships. For example, to process the XPath query: //A[./C//D]//B, it is decomposed into the three relationships (*A//B*, *A/C*, *C//D*). Each binary relationship is executed and its intermediate result is stored for further processing. The final result is formed by processing and combining these intermediate results.

Another famous XML query processing method is *TwigStack* [2]. In this method, instead of decomposing a query into its binary relationships like what *Structural Join* does, at first partial solutions to each root-to-leaf path in the main query (QTP) are found. Next, these single path results are merged together to form the final answer of the query. It is worth noticing that, in the first step, *TwigStack* outputs only those single path results which have a chance to be joined with other path results to form a complete match. As a consequence of these features, the amount of intermediate results derived by *TwigStack* is much lower than that of *Structural Join*.

TJFast [6] is another XML query processing method which minimizes its I/O requirements by only accessing those element indexes which are related to QTP leaves and also to inner nodes that have a predicate. To achieve this improvement, *TJFast* uses a refined version of the Dewey labeling method, which is able to encapsulate the complete ancestor path in the label of each node. Furthermore, *TJFast* uses a

finite state transducer to compute the complete path of a document node from its label. Thus, *TJFast,* which is inspired by *TwigStack,* can easily produce partial results of individual root-to-leaf paths of the query only by accessing the labels of nodes, which are related to the QTP leaves. These partial solutions are merged to form the query result in the second phase.

Twig²Stack [3] and *TwigList* [7] are two other XML query processing methods, which primarily aim at the elimination of the cost resulted from the merging phase that is used in *TwigStack* or *TJFast.* In these methods, partial solutions which are found during document processing (especially element indexes) are kept in such a way that subsequent merging is avoided and final matches are ready to be output only by applying a simple enumeration function. But these methods suffer from a main weakness: they have to load the entire document into main memory in the worst case.

In a nutshell, the above methods in conjunction with our proposed method could be compared based on three parameters: *Index access* that shows which indexes should be accessed, the number of elements which have to be read (# of *elements*), and the volume of *intermediate results* produced. Table 1 shows the comparison among the mentioned methods. All listed methods except TJFast and our proposed method have to access all element indexes related to the elements (QTP nodes) referred by the query. The number of elements which have to be read is minimal in our method compared to the other methods considered, even to *TJFast. Structural Join* produces the maximum amount of intermediate results, which are insignificant for our method.

Table 1 .Comparison of XML path expression processing methods

	StructJoin	TwigStack	TJFast	Twig²Stack	TwigList	our method
index access	all	all	leaves	all	all	leaves
# of elements	~ # of index accesses	~ # of index accesses	~ # of index accesses	~ # of index accesses	~ # of index accesses	minimum
intermediate results	large	~ # of results	~ # of results	~ # of results	~ # of results	insignificant

3 Key Ingredients

The power of our method is founded upon two key concepts: *SPLIDs* and *path synopsis*. It can be applied to XML documents with a complete or a virtualized structure part [5].

3.1 Node Labeling

An intensive comparison of labeling schemes and their empirical evaluation [4] led us to use a prefix-based scheme for the labeling of tree nodes based on the concept of Dewey order. As abstract properties of Dewey order encoding, each label consists of so-called *divisions* (separated by dots in the external format) and represents the path from the document's root to the node and the local order w. r. t. the parent node; in addition, optional sparse numbering facilitates node insertions and deletions. Refining this idea, a number of similar labeling schemes were proposed which differ in some aspects such as overflow technique for dynamically inserted nodes, attribute node labeling, or encoding mechanism. Examples of such schemes are DLN or Ordpath developed for Microsoft SQL Server™. Although similar to them, our mechanism is characterized by some distinguishing features and a label is denoted DeweyID [4]; it refines the Dewey order mapping: with a *dist* parameter used to increment division values, it leaves gaps in the numbering space between consecutive labels and introduces an overflow mechanism when gaps for new insertions are in short supply—a kind of adjustment to expected update frequencies. Because any prefix-based scheme is appropriate for our document storage, we use the term SPLID (Stable Path Labeling IDentifier) as synonym for all of them.

Existing SPLIDs are *immutable*, that is, they allow the assignment of new IDs without the need to reorganize the IDs of nodes present. Comparison of two SPLIDs allows *ordering* of the respective nodes in document order. Furthermore, SPLIDs easily provide the IDs of all ancestors, e.g., to enable intention locking of all nodes in the path up to the document root without any access to the document itself. For example, the ancestor IDs of 1.3.3.7.5.3 are 1.3.3.7.5, 1.3.3.7, 1.3.3, 1.3 and 1.

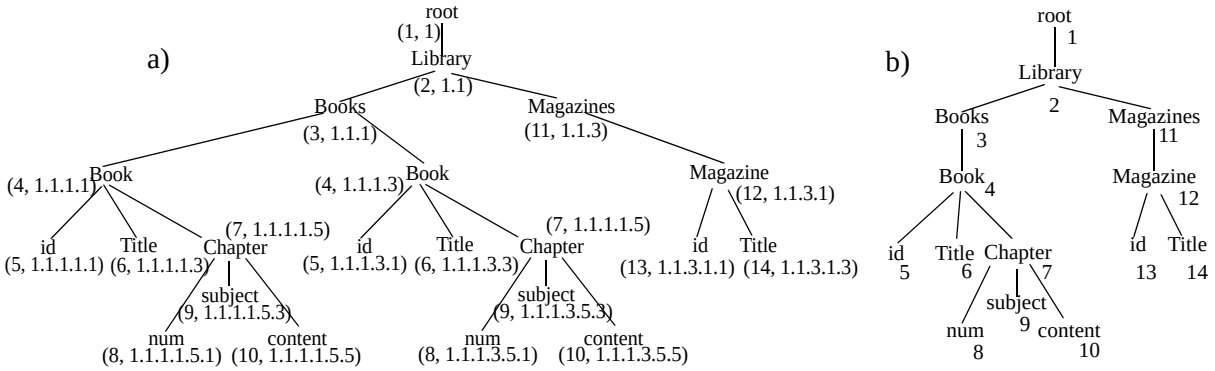

Figure 1: A sample XML document (labeled with SPLIDs) and its path synopsis

3.2 Path Synopsis

Our key idea is now to capture the structure of an XML document in a small data structure called path synopsis. All paths from the root to the leaves having the same sequence of element/attribute names form a *path class*. Thus, each path in the document can be assigned to one of the relatively few distinct path classes. As an example, Fig. 1a represents the structure part of an XML document for which the path synopsis is shown in Fig. 1b. Such a concise description of the document's structure is a prerequisite of effective virtualization of the structure, i. e., for an elementless storage of the document. When comparing them to the number of path instances, it becomes obvious that huge redundancy is introduced when all path instances are explicitly stored. In the popular *dblp* document, for example, one of the dominating path classes /bib/paper/author has ~570,000 instances.

Having such a separate structure, we can use it as a *query guide* to effectively govern the query optimization and evaluation process. Furthermore, we can remove and drop the entire structure part from the physically stored document and, nevertheless, are able to reconstruct each path or the entire document, whenever needed [5]. Note, by providing such an *on-demand option*, we don't want to sacrifice functionality, but only save substantial storage space. When we use structure virtualization, the complete path information consisting of the attribute/element names of all inner nodes and their structural relationships are not explicitly stored in the physical document representation. Their on-demand computation is "outsourced" to the *path synopsis*. Again, the *secret* enabling this structure virtualization is the SPLID mechanism with which each node carries a short-hand representation of its entire path to the root.

Cyclic-free XML schemata capture all information needed for the path synopsis; otherwise, this data structure can be constructed while the document is stored. Hence, when matching the right path class with a given SPLID, it is very easy to reconstruct the specific instance of this path class. In a sense, we must associate the value in a document leaf—whose unique position in the document is identified by its SPLID—with a space-saving reference to its path class. Furthermore, when document processing references an inner node—for example, by following an element index for *author* or by setting a lock on a particular *book* for some concurrency control task—, we must be able to rapidly derive the (sub-) path to the root in the virtualized structure. For this reason, by numbering all nodes in the path synopsis, we gain a simple and effective mechanism called path class reference (PCR). Such PCRs are used in the content nodes or in index structures together with SPLIDs serving as a path class encoding.

3.3 Elementless Document Storage

The sketched usage of the path synopsis indicates its central role as a repository to be used for all structural references and operations. Queries can often be completely evaluated using the path synopsis, specific indexes, and the SPLID mechanism, because entire document paths can be computed when SPLID and PCR are known for a node. However, when additional data are requested for the result generation, the stored document has to be accessed. In such cases, the form of the physical document representation strongly affects the query performance. Therefore, we have the option to store structure and content part of an XML document completely or to virtualize the structure part [5].

4 Processing Strategy of RG

Our proposed method has two main steps. In the first step, query is executed over the *path synopsis* of the document. The output of this step is a set of PSMs (*Path Synopsis Match*) called SPSM. Based on the features of path synopsis, we could claim that the set of PCRs of each combination of the document nodes matching the query is a member of SPSM. As a consequence, in the second step for each PSM, those nodes having the same PCR as PCRs of the PSM are fetched and processed to produce the final result of the query. Fig. 2 illustrates the interplay of the components when the RG method is applied.

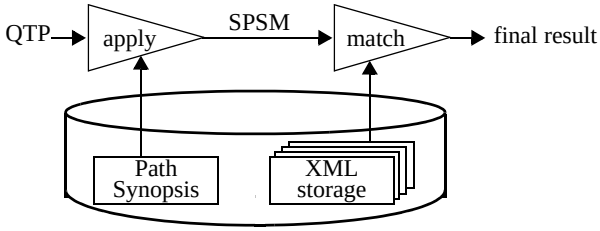

Figure 2: Overview of the RG method

4.1 Set of Path Synopsis Matches

The set of path synopsis matches (SPSM) acts as a query execution plan to provide a more focused way for accessing to the document nodes (indexes) to reduce I/O time. It also enables RG to process and form the matches in a more cost effective way. SPSM is a set of PSM and each PSM is itself a set of path synopsis nodes. In order to create the PSMs from a given query and *path synopsis*, the query is executed over the path synopsis (the logical structure of XML document and its *path synopsis* are the same). It is worth noting that in this step any twig query processing method like those described in the related work could be used. Because *path synopsis* is a small memory-resident object, the cost of SPSM construction is insignificant.

Fig. 3 shows an example of an artificial *path synopsis* (a) and a given twig query (b). The SPSM contains 5 members and reveals that *path synopsis* node D_8 is not involved in the SPSM. It means that we are sure that document nodes having PCR 8 can not participate in the final result, because any type node *D* should have a *C* node as its ancestor. Hence, the *path synopsis* confirms that there is not any possibility for *D* nodes with PCR 8 to participate in the query result. As a consequence, there is no need to access document nodes with PCR 8. This example shows why we can claim that the *path synopsis* enables more focused index accesses.

4.2 Matching Process

The matching process in RG uses the SPSM to produce the final matches. Each PSM is used to produce those final matches which their PCRs match the selected PSM. In order to execute any selected PSM, *path synopsis* nodes in the PSM related to the query leaves are extracted and then only those nodes from the document are accessed, which have the same PCRs as identified by these *path synopsis* nodes. This means that, during the entire process of matching, we only need to access the subset of nodes which are related to leaves of the query (QTP). For example, in order to execute the query whose structure is shown in Fig. 3b, none of the *A* and *C* elements are physically accessed and those *D* elements with PCR 8 are not accessed, too.

In the following, we describe the RG matching process by a hypothetical example. Consider path synopsis and query of Fig. 3. Execution of the query using the *path synopsis* leads us to an SPSM with 5 PSMs. In order to form the final result, each of these 5 PSMs should be processed. For example, consider

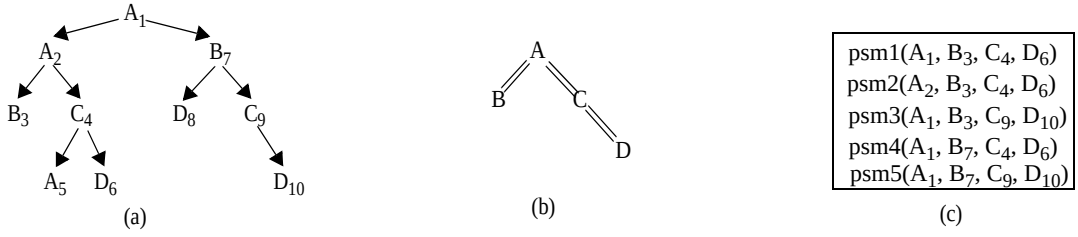

Figure 3: (a) Samle path synopsis, (b) sample QTP, (c) SPSM formed by appying the query to the path synopsis

the *psm2* and two given SPLIDs: $b_{3\text{-}1}$= *1.3.6* and $d_{6\text{-}1}$= *1.3.5.7*. With regard to the query and *psm2*, these two nodes should have a common ancestor with element type *A* and the SPLID of this node should indicate an allocation at level 2. The first two divisions in the $b_{3\text{-}1}$ and $d_{6\text{-}1}$ are the same (1.3). As a result, these two nodes have a common ancestor, where its SPLID indicates level 2. A look at the *path synopsis* confirms that this node is an *A* node and it is also clear that the node labeled $d_{6\text{-}1}$ has a *C* element as its ancestor with SPLID (*1.3.5*). Thus, $b_{3\text{-}1}$ and $d_{6\text{-}1}$ could match the query. Hence, the match obtains the following result: (1.3, 1.3.6, 1.3.5, 1.3.5.7).

To represent the algorithm more formally, Fig. 4 shows the pseudocode of the matching process for a single PSM with two leaves. It is straightforward to extend this algorithm in order to support general QTPs which have more than two leaves. The matching process in RG has two key features: first, the matching process is a simple one-phase algorithm when compared to previous methods like *TwigStack* and *TJFast* which need an extra phase to merge partial results produced in their matching process. The second important feature is the low memory consumption required by RG in comparison to the amount of memory needed to store intermediate results or data structures of previous methods.

```
 1: psn1 = psm.getLeaf(1);
 2: psn2 = psm.getLeaf(2);
 3: NCA = psm.getNCA(psn1, psn2).getLevel();
 4: L1 = list of sorted SPLIDs with PCRs equal to psn1.getPCR();
 5: L2 = list of sorted SPLIDs with PCRs equal to psn2.getPCR();
 6: slist1 = nodes from L1 which have the same prfix up to NCA level
 7: slist2 = nodes from L2 which have the same prfix up to NCA level
 8:    n1 = L1.head;
 9:    n2= L2.head;
10: while(!L1.end() and !L2.end())
11:   if (n1.prefix(NCA) == n2.prefix(NCA))
12:      for each combination of slist1 and slist2 produce a match
13:   elseif (n1.prefix(NCA) < n2.prefix(NCA))
14:      n1 = L1.head;
15:      slist1 = nodes from L1 which have the same prfix up to NCA le
16:   else
17:      n2= L2.head;
18:      slist2 = nodes from L2 which have the same prfix up to NCA le
19:   endif
20: end while
```

Figure 4: Pseudocode of the matching process

5 Conclusions

In this paper, we emphasized the power of SPLIDs and *path synopsis* as an auxiliary structure, which enables the virtualization of the structure part of XML documents resulting in an elementless document storage. Then, we described our method, RG, for efficient twig query processing and outlined how these structures help our method to produce matches in a cost-effective way regarding I/O and CPU time. A *path synopsis* not only enables focused index access to reduce the I/O cost as much as possible, but also facilitates the matching process by providing SPSM which reduced the matching process to a small set of SPLID comparisons. The low memory consumption of RG in conjunction with its low I/O and CPU costs makes it suitable to be used in real multi-user XML databases.

References

[1] Al-Khalifa, S., Jagadish, H.V., Koudas, N., Patel, J.M., Srivastava, D., Wu., Y.: Structural Joins: A Primitive for Efficient XML Query Pattern Matching. In *Proc. ICDE,* San Jose, California, USA, 141-152 (2002)

[2] Bruno, N., Koudas, N, Srivastava, D.: Holistic Twig Joins: Optimal XML Pattern Matching. In *Proc. SIGMOD,* Madison, Wisconsin, USA, 310–321 (2002)

[3] Chen, S., Li., H.G., Tatemura, J., Hsiung, W.P., Agrawal, D., Candan, K. S.: Twig^{2}Stack: Bottom-up Processing of Generalized-Tree-Pattern Queries over XML Documents. In *Proc. VLDB*, Seoul, Korea, 283-294 (2006)

[4] Härder, T., Haustein, M. P., Mathis, C., Wagner, M.: Node Labeling Schemes for Dynamic XML Documents Reconsidered. *Data & Knowledge Engineering* 60:1, 126-149, Elsevier (2007)

[5] Härder, T., Mathis, C., Schmidt, K.: Comparison of Complete and Elementless Native Storage of XML Documents. In *Proc. IDEAS,* Banff, Canada, 102-113 (2007)

[6] Lu. J., Ling, T. W., Chan, C. Y., Chen, T.: From Region Encoding To Extended Dewey: On Efficient Processing of XML Twig Pattern Matching. In *Proc VLDB*, Trondheim, Norway, 193–204 (2005)

[7] Qin., Lu, Yu, J.X., Ding, B.: TwigList: Make Twig Pattern Matching Fast. In *Proc. DASFAA*, Bangkok, Thailand, 850-862 (2007)

Coordinating Activities in Emergency Situations

Birgitta König-Ries, Aygul Gabdulkhakova

Heinz-Nixdorf Endowed Chair for Practical Informatics
Institute of Computer Science
Friedrich-Schiller-University Jena
Jena, Germany
{ koenig, aygul}@informatik.uni-jena.de

Abstract

When a disaster happens, emergency management personnel, equipment, and volunteers have to be coordinated to provide help in the most efficient way. This coordination is a complex task: it needs to be done dynamically, based on incomplete information in an ever-changing situation. In this paper we describe our proposed solution to this problem.

1. Introduction

Emergency events like natural disasters and large scale accidents pose a number of challenges to handle. In this paper, we concentrate on one of these challenges, the requirement to coordinate a wide range of organizations and activities, public and private to provide efficient help for the victims. When a disaster occurs, there is typically an immediate and large outpour of offers to help by volunteers but many of these offers are not used efficiently. For instance, after hurricane Katrina, there were numerous incidents of skilled volunteers and useful equipment being turned down or not called upon [1]. Similar reports are known from other events. This lack of coordination has been one of the major points of critique regarding the performance of rescue work.

In order to tap the potential of volunteers and emergency personnel, and to efficiently use available equipment, a certain amount of preplanning (and in particular training) is necessary. On the other hand, the assignment needs to be done on-the-fly once the disaster has occurred: While for some disasters (e.g., a fire in a football stadium), rather detailed emergency plans can be developed beforehand, there are also numerous disasters, where this is not the case. Consider, e.g., hurricanes or train accidents. These disasters can occur anywhere and can vary widely in severity. Depending on the exact location, access to the accident site will differ, the number, experience, and equipment of available emergency personnel and volunteers will also depend on the location. Finally, the help needed will depend on the type of train (e.g., a train transporting people vs. a train transporting toxic waste or chemicals), etc. Often, only limited information about the accident will be available in the beginning. More information about the situation will become known over time.

Thus, a system is needed that can efficiently assign emergency personnel, equipment, and volunteers to the necessary tasks according to their abilities based on the limited information available at any given time. Furthermore, this system needs to function locally without relying on the existence of a fixed infrastructure. It is unrealistic to assume, that after a disaster such infrastructures will be available to use. Therefore, we need a system for the coordination of volunteer efforts that works independent of an existing infrastructure that is in a decentralized and ad-hoc manner.

Let us consider a scenario to explain the desired functionality of the system in more detail: Assume that a train has derailed close to a village. Anna, a certified nurse, and her friend Beth, a preschool teacher, happen to drive by the accident site. They volunteer to help and are directed to Bob, who is coordinating volunteers. Bob has a copy of our system running on his computer. He transmits the necessary files for installation via Bluetooth to Anna's and Beth's cell phones. Anna, being a medical professional, is already registered with the system. She identifies herself and is directed to the triage area to help with treating the injured. Beth first needs to register and enters her skills into the system.

She has some basic first aid knowledge, a driver's license, speaks English, French, and Russian, is qualified to work with children, and is willing to help with whatever comes up. Meanwhile, Tom, an officer with the local fire department is assessing part of the accident scene. Right now, he is standing in front of an upturned coach. The passengers have already succeeded in smashing one of the windows. Most of them appear to be only slightly injured; however, there are two unconscious people still in the coach. One of them is a woman which seems to be the mother of two children. Tom tries to talk to the children, but is informed by one of the passengers that the kids speak Russian only. Tom enters into his system that he needs two unconscious passengers to be taken care of, that there are ten lightly injured people to attend to and that a Russian speaking volunteer is needed to take care of the children.

Based on this information, the system should now assign suitable personnel and equipment. In a first step, it needs to determine that in order to take care of unconscious passengers they need to be transported to the triage area. To do so, paramedics and stretchers are needed. Similarly, the appropriate treatment of the lightly injured people (a volunteer, blankets, and something to drink) is determined. Then, appropriate resources are allocated. Maybe Beth gets assigned to take care of the children.

So, the basic idea is to develop a system that will allow volunteers to register with their abilities and to be then matched to current needs. That means, that we are solving a time and resource scheduling problem, but it is different from standard situations because we have a lot of additional conditions and requirements, such as: only partial knowledge about the situation, information may be wrong, the situation will change all the time; no precise preplanning possible (e.g. train may be derail anywhere); high heterogeneity (personnel from different organisations; equipment from different organisations; volunteers have different qualifications). Additionally, we want the system to be able to break complex tasks down to atomic ones which can be solved by assigning resources.

It should be obvious, that in order for such a system to work, some preparation ahead of time is needed: First, it would make sense to register qualified personnel beforehand. Second, common tasks and their decomposition should already be modelled at a planning stage.

In the following, we describe a system architecture and the most important components of this architecture, which will be able to perform this task: Section 2 introduces the system architecture and explaining the role of agents in the offered system and Section 3 provides a summary and an outlook to future work.

2. System Architecture

As mentioned above, due to the lack of fixed infrastructures, a decentralized solution is needed. Our proposal is to have copies of the system as depicted in Figure 1 run on each device. Depending on the role of the owner of the device (volunteer vs. commander of the fire brigade), and the capabilities of the device different parts of the system may be activated or not. Resources (both human and technical) are modelled as agents in this system

As you can see in Figure 1, the system consists of two main parts:

1. A distributed reputation system which allows managing information about agents, estimating their reliability, and their ability to execute different tasks. The reputation system should also contain information about the performance of "teams". In particular, it should be able to determine which combination of personnel and equipment works best (i.e., it might make sense to provide people with tools they are used to instead of those of another brand) [2];

2. A kind of workflow management system which allows managing information about tasks, decomposing them, assigning resources to them and controlling their execution.

Both parts of the system must exchange information with one another: information from the reputation system will help to correctly assign agents to tasks. The other way round, information gained from the execution control part of the workflow system serves as input for the reputation system and helps to determine the ability of agents and teams.

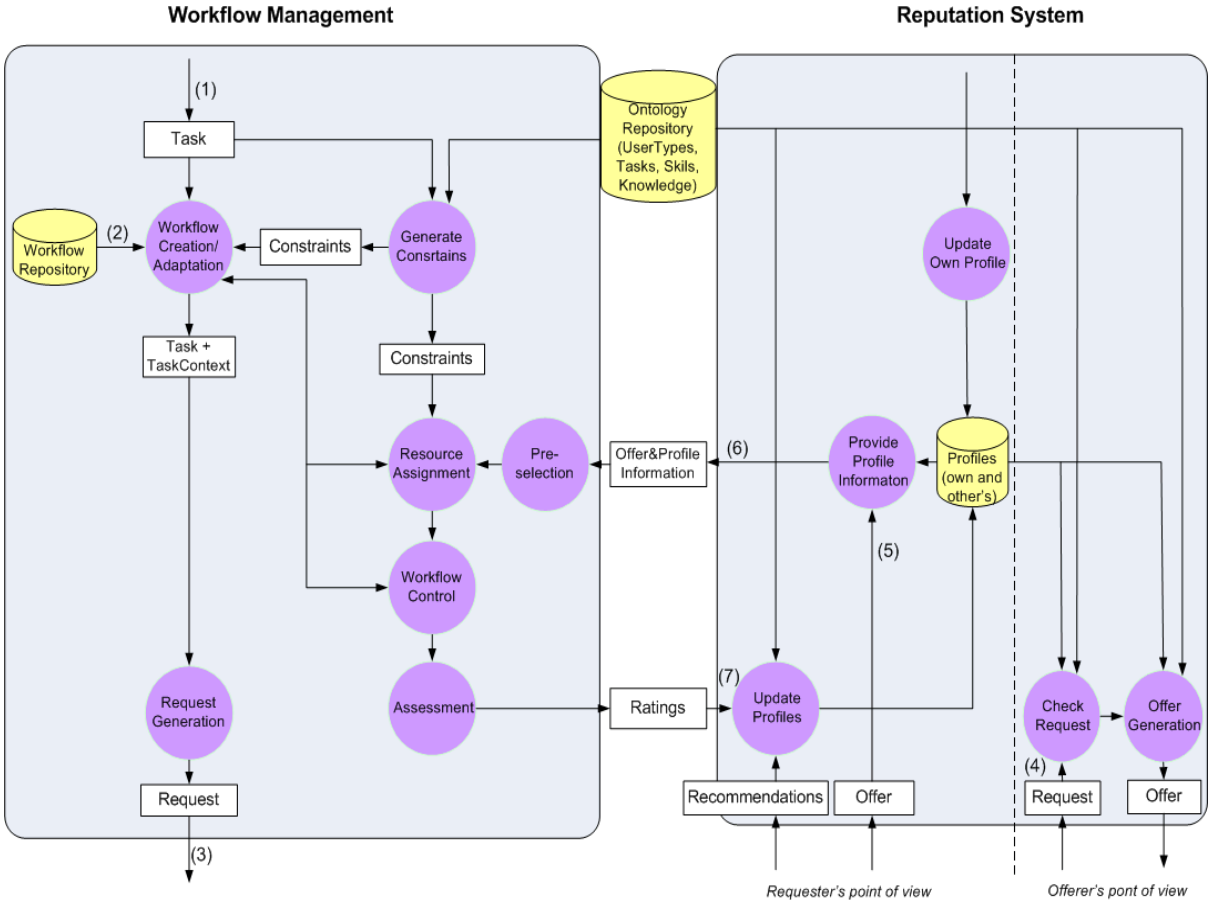

Figure 1: System Architecture

Before explaining the data and control flow in the system, let us take closer look at some of the central concepts:

- Agent is a human or technical resource with the ability to perform certain actions (skills).
- A task is a problem that needs to be solved. Tasks can be decomposed into (sub-)tasks. Eventually, atomic subtasks will need resources to be worked out. An example for a task could be the evacuation of an injured person.
- A request describes the need for a resource with certain abilities. It is formalized as a record, including a request name and type (a type of task to be completed) and a vector of skill requirements, describing what skills are necessary for this task in which “amount”. An example could be a person capable of driving an ambulance.
- An offer describes an agent. It is formalized as a record, including an offer name and type, and a vector of skills, describing what skills has the owner has to which degree.
- The Workflow repository contained executable descriptions of how to solve known tasks. The evacuation mentioned above could be done by using an ambulance with a driver and an additional person, or with helicopter, a pilot, and someone else with medical training.

Basically, tasks are decomposed into subtasks. This decomposition is stored in a task ontology [3].for each atomic subtasks this ontology also contains descriptions of how these tasks can be performed, i.e. which resource with which skill set are needed. The latter can be viewed as web services. Accordingly, we plan to use semantic web services descriptions here [4].

- The Profile repository stores information about available agents (both human and technical) such as
  - qualification (characteristics)
  - synergetic effects of working groups
  - credibility of the offer
  - authorization of the offer
  - trustworthiness of the offer
  - legal and organizational restrictions

- The Ontology repository contains a base set of terms that is used for the tasks, skills and knowledge description.

The system aims at find a match between a request and an offer, thus finding an agent to perform a task. An agent might be thought to be matching, if he has all skills necessary to perform a task, and may be even more.

Let us analyze the main building blocks of the two parts of the system and their interplay:
Assume that the workflow management system receives a new task (1). This serves as the beginning for the development of a workflow. The workflow repository is checked to see whether such a task has been considered during pre-planning. If this is not the case, the decomposition has to be done manually. If it is the case, then a workflow (consisting of atomic subtasks and their data and control flow) is created and launched (2). Whenever one of the subtasks needs to be carried out, a request for the necessary resources is sent to the reputation system (both the local and remote copies that are reachable) to find agents that might be able to provide them (3, 4). Each copy of the reputation system checks whether the offer of its owner matches the request (5). If this is the case an answer is being to the workflow management. This answer contains the profile information of the offer (6). An answer is also produced if the system has stored relevant reputation information, i.e., information about another agent that claim to be able to perform the tasks. Based on this information the workflow management system assigns agents to the tasks and starts execution. Once agents have performed their tasks the requester and its subrequesters of that tasks provide a single value rate (given some scale) reflecting the quality of the task execution and if needed combine it to a rate for (a part of) the overall workflow (7).

3. Conclusion

For coordination of emergency personnel and resources in case of a disaster is necessary to have the operative information on the current situation and development of its factors. Effective distribution of resources and time plays the main role. In this paper we have proposed an IT-based assistance for coordination of relief forces and have presented system architecture. The proposed solution combines approaches from multiagent systems, workflow management system, artificial intelligence and reputation to provide a flexible and effective solution. In our future work, we plan to extend on this basic idea, further work out the details and to implement a prototype of our system.

References

1. http://www.whitehouse.gov/reports/katrina-lessons-learned/chapter5.html

2. Philipp Obreiter, Birgitta König-Ries: A New View on Normativeness in Distributed Reputation Systems - Beyond Behavioral Beliefs. Proceedings of the Forth International Workshop on Agents and Peer-to-Peer Computing (AP2PC\'05), Utrecht, Netherlands July 2005

3. Munehiko Sasajima, Yoshinobu Kitamura, Takefumi Naganuma, Shoji Kurakake, Riichiro Mizoguchi: Task Ontology-Based Framework for Modeling Users' Activities for Mobile Service Navigation, ESWC 2006, Budva, Montenegro, June 2006

4. Michael Klein, Birgitta König-Ries, Michael Müssig: What is needed for Semantic Service Descriptions - A Proposal for Suitable Language Constructs International Jounal on Web and Grid Services 2005 - Vol. 1, No. 3/4, pages 328-364. October 2005

ODYSSEUS: Ein flexibles Framework zum Erstellen anwendungsspezifischer Datenstrommanagementsysteme

Jonas Jacobi, Marco Grawunder
Universität Oldenburg
Department für Informatik, Informationssysteme
[jonas.jacobi|marco.grawunder]@uni-oldenburg.de

11. April 2008

Zusammenfassung

Auf dem Markt existieren eine Reihe von Systemen und Forschungsprototypen für Datenstrommanagementsysteme. Diese meist universellen Produkte sind selten applikationsspezifisch (beispielsweise für die Überwachung von Windparks) optimierbar. Mit ODYSSEUS entwickeln wir ein flexibles Framework, welches die Erstellung von anwendungsbezogenen Datenstrommanagementsystemen unterstützen soll. ODYSSEUS erlaubt es gezielt, bestimmte Aspekte (wie die Anpassung einer Scheduling Strategie) in den Vordergrund zu stellen, oder verschiedenartige Datenstrukturen einfach zu integrieren.

Im Rahmen dieser Arbeit wollen wir die wesentlichen Basiskonzepte des Frameworks kurz vorstellen. Dies sind im wesentlichen ein sehr flexibles Operatorenkonzept, welches unterschiedliche Arten der Verarbeitung (push, pull) erlaubt, ein generischer Scheduler, der angepasste Strategien für die Ausführung von Operatorenplänen erlaubt, die Basismechanismen für die Übersetzung und Ausführung von Anfragen, sowie Mechanismen zur Überwachung der Anfrageausführung.

Wir haben unser Framework bisher in der relationalen Welt und mit RDF (inkl. einer Erweiterung von SPARQL um die Möglichkeit der Stromverarbeit) eingesetzt. In Planung befindet sich die Betrachtung der Manufacturing Message Specification, die als Basis für die Überwachung von Windparks dienen soll.

1 Einleitung

In einer Vielzahl von Anwendungsszenarien spielt die Überwachung oder Analyse von Daten eine Rolle, die kontinuierlich von einer aktiven Quelle versandt werden. Weisen solche Datenströme hohe Datenraten/-volumina auf und sollen zeitnah verarbeitet werden, geraten klassische Ansätze auf Grundlage von traditionellen Datenbankmanagementsystemen an ihre Grenzen. Bis vor einigen Jahren bedeutete dies, dass spezieller Progammcode (häufig in C) geschrieben werden musste, der die Datenströme verarbeitet. Die Forschungs- und Entwicklungsergebnisse auf dem Gebiet des Datenstrommanagements der letzten Jahre machen es möglich an dessen Stelle ein Datenstrommanagementsystem[1] (DSMS) einzusetzen, an das – wie an eine Datenbank – (deklarative) Anfragen gestellt werden können, die kontinuierlich auf den eingehenden Daten ausgewertet werden (z. B. liefere einen Alarm, wenn der durchschnittliche Messwert eines Sensors innerhalb der letzten zehn Minuten größer als x ist). Dies vereinfacht die (Weiter-)Entwicklung solcher Applikationen ungemein, da nur noch ein Satz von Anfragen erstellt werden muss – die optimierte Ausführung übernimmt das System.

[1] Neben Forschungsprototypen wie Aurora [ACc+03]/Borealis oder YFilter gibt es inzwischen auch einige kommerzielle Anbieter wie Coral8 (`http://www.coral8.com`) und StreamBase Systems (`http://www.streambase.com`)

Bisherige DSMS sind als Universalsysteme ausgelegt, die nur eingeschränkt anpassbar sind. Zumeist ist ein Datenmodell vorgegeben (zumeist relational oder aber XML) und interne Komponenten wie der Scheduler oder das Speichermanagement können nicht ohne weiteres verändert werden. Eine Sonderstellung nimmt PIPES [KS04] ein, das als Softwarebibliothek vielseitige Anpassungen möglich macht. Anwendungsszenarien haben unter Umständen aber besondere Anforderungen an die Verarbeitung der Daten, die mit solch einem Universalsystem nicht ohne weiteres erfüllt werden können. Als Beispiel sei hier die priorisierte Verarbeitung von Alarmmeldungen im Kontext der Überwachung dezentraler Energieerzeuger wie Windenergieanlagen genannt. Die Beschränkung auf relationale Daten macht eine Konvertierung komplexerer Strukturen nötig, welche Performanceeinbußen mit sich bringt und die Anfragen u. U. unnötig verkompliziert (analog zur Abbildung von Objekten auf Relationen in der Datenbankwelt). Aus diesem Grund entwickeln wir mit dem Oldenburger DynaQuest Datastream Query System (ODYSSEUS) ein Framework, das es mit möglichst geringem Aufwand erlaubt einzelne Aspekte der Verarbeitung anzupassen und einfach um die native Unterstützung nichtrelationaler Datenmodelle erweitert werden kann.

2 Architektur

Die Architektur von ODYSSEUS ist in Abbildung 2 dargestellt und gliedert sich grob in drei Teile: Anfrageübersetzung, Anfrageausführung und Überwachung.

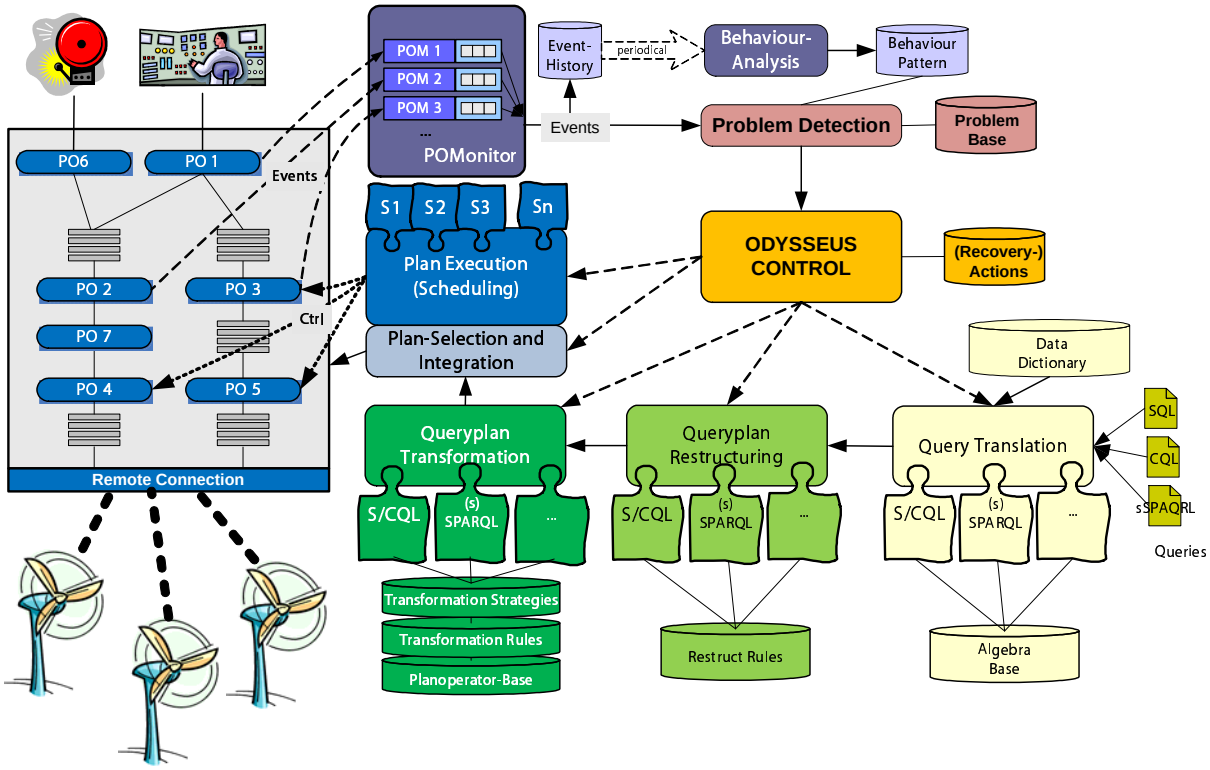

Abbildung 1: ODYSSEUS Architektur

Die Anfrageübersetzung findet analog zu klassischen Datenbanksystemen statt (vgl. [Gra93]). Zunächst wird eine Anfrage an das System formuliert. Dies kann entweder über eine (deklarative) Anfragesprache oder prozedural mittels der Definition eines Anfragebaumes geschehen. Zur Anfrage an Datenströme stehen im Moment zwei deklarative Anfragesprachen zur Verfügung. Für

relationale Datenstromanfragen ist dies eine StreamSQL[2] ähnliche SQL-Erweiterung. Anfragen an RDF-Datenströme können mittels Streaming-SPARQL[BG08] formuliert werden. Eine Anfrage wird in einen logischen Anfrageplan überführt, der in der Restrukturierungsphase algebraisch optimiert wird. Aus dem optimierten Algebraplan werden verschiedene mögliche Varianten physischer (ausführbarer) Anfragepläne erzeugt.

Zur Anfrageausführung wird nun aus den verschiedenen Möglichkeiten ein Anfrageplan ausgewählt. Dieser wird in einen Anfragegraphen, der die Operatoren aller Anfragen enthält, integriert. Die physischen Operatoren können untereinander mittels Warteschlangen verbunden werden, in die die Ausgaben eines Operators geschrieben und von den nächsten gelesen werden. Dadurch können Operatoren unabhängig voneinder ausgeführt werden (solange eine Warteschlange nicht voll ist), was den Einsatz verschiedener Scheduling-Strategien zu ihrer Ausführung ermöglicht.

Während der Ausführung generieren die Operatoren Ereignisse. Diese Ereignisse werden von Monitorobjekten überwacht, welche daraus abstraktere Ereignisse ableiten – so können z. B. Blockierungen oder Bursts in den Datenlieferraten erkannt werden. Zur nachträglichen Analyse können Ereignisse in einer History gespeichert werden. Auf Basis der Ereignisse kann eine Verhaltensanalyse erfolgen, die beispielsweise Muster im Datenlieferverhalten von Quellen aufdecken kann [JG07], welche wiederum zur Optimierung des Systems genutzt werden können. Zur Laufzeit des Systems können außerdem weitere Metadaten zu verschiedenen Komponenten des Systems wie Planoperatoren/Anfragepläne, Quellen oder Warteschlangen erfasst werden. Diese können dazu genutzt werden Parameter des Systems beziehungsweise die Ausführung der Operatoren zu beeinflussen.

3 Operatoren-Konzept

Eine wesentliche Grundlage der Flexibilität unseres Frameworks ist die Umsetzung des Operatorenkonzepts. Die (physischen) Operatoren folgen dem Open-Next-Close (ONC) Protokoll und können Daten sowohl daten- (push) als auch anfragegetrieben (pull) verarbeiten.

Es wurde beim Entwurf der Operatoren insbesondere auf die leichte Erweiterbarkeit und Wiederverwendbarkeit existierenden Codes geachtet. Das ONC Protokoll sowie die Verarbeitungsart, sind in der vorhandenen Operatorklassenhierarchie gekapselt und für den Entwickler transparent.

Als Beispiel für die einfache Erweiterbarkeit von ODYSSEUS wird im Folgenden beschrieben, wie eine neue Sprache mit einem neuen Datenmodell eingeführt wird. Zunächst muss eine Überführung der Sprache in einen algebraischen Anfrageplan implementiert werden. Es existiert eine Klassenhierarchie von algebraischen Operatoren, die im Wesentlichen unabhängig vom verwendeten Datenmodell sind. Diese kann bei der Einführung wiederverwendet und leicht um eventuell noch nicht vorhandene erweitert werden. Bei der Einführung von SPARQL/RDF mussten beispielsweise nur noch wenige SPARQL spezifische Operatoren wie `Ask` (zeigt an, ob eine Anfrage ein Ergebnis liefert) oder `Describe` (liefert beschreibende Elemente zu einer Ressource) implementiert werden. Der manuell zu erstellende Code wird durch die Verwendung vorhandener abstrakter Basisklassen auf ein Minimum reduziert.

Ebenso steht für die Implementierung physischer Operatoren ein Klassengerüst bereit, das durch konsequente Anwendung der Strategie- und Template-Methoden-Entwurfsmuster sehr einfach an neue Datenmodelle anpassbar ist. Als einfaches Beispiel ist in Abbildung 3 der für einen Selektionsoperator relevante Auschnitt aus der Klassenhierarchie zu sehen.

Soll ein Selektionsoperator für einen neuen Datentyp entstehen muss nur noch eine Implementierung der `SelectAction`-Schnittstelle für diesen Datentyp erfolgen, bei der die Methode `evalute` überschrieben wird, ein Operator selbst muss nicht mehr programmiert wer-

[2] `www.streamsql.org`

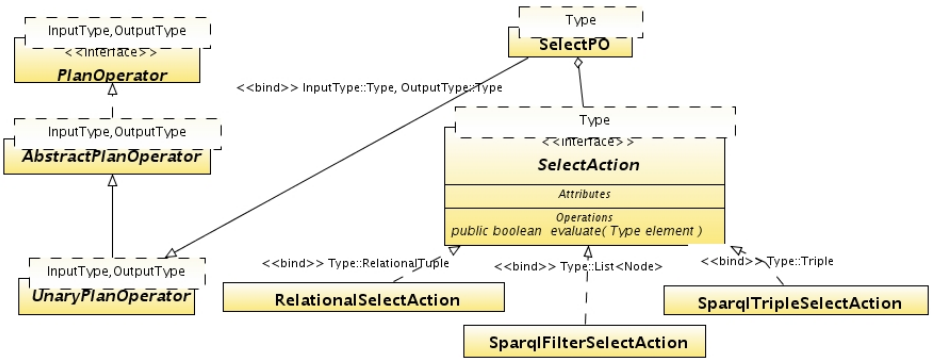

Abbildung 2: Klassenhierarchie für die Selektionsoperation

den. Auch das erstellen neuer Operatorklassen wird durch vorhandene abstrakte Basisklassen im Normalfall auf die Implementierung einer einzigen Methode beschränkt (`AusgabeTyp processNext(EingabeTyp in)`), der das jeweils nächste zu verarbeitende Element übergeben wird.

Als letzter Schritt müssen noch Regeln definiert werden, welche Algebraoperatoren in passenden physischen Gegenstücke überführen.

4 Scheduling

Die Ausführung der Operatoren wird von einem Scheduler gesteuert. Es stehen jeweils ein bis mehrere (System-)Threads zur Verfügung, innerhalb derer Operatoren ausgeführt werden können. Der Scheduler ist dafür zuständig für einen Thread den auszuführenden Operator auszuwählen. In ODYSSEUS ist es möglich abhänging von Optimierungszielen wie Durchsatzmaximierung oder Latenzminimierung unterschiedliche Scheduling-Strategien zu verwenden. Zum jetzigen Zeitpunkt stehen neun verschiedene Strategien zur Verfügung, die in Tabelle 1 kurz erläutert werden. Weitere Strategien können leicht hinzugefügt werden. Durch Entfernen von Warteschlangen zwischen Operatoren können mehrere zu einem Verbund zusammengeführt werden und damit innerhalb des Schedulings als ein einzelner virtueller Operator betrachtet werden.

5 Ausblick

ODYSSEUS wird u. a. im Rahmen von Dissertationen und Diplomarbeiten aktiv weiterentwickelt. Aktuell laufen Arbeiten zur Multi-Anfragen-Optimierung, priorisierter Verarbeitung besonderer Elemente/Anfragen (z. B. von Alarmmeldungen im Überwachungskontext), zur Visualisierung ausgeführter Pläne und ihrer Metadaten, sowie zur Integration eines regelverarbeitenden Systems für die Restrukturierung, Plantransformation und dynamische Anpassung von Laufzeitparametern des Systems.

Geplant ist weiterhin eine Vereinigung von Techniken des Complex Event Processing (CEP) und des „traditionellen Datenstrommanagements“ in einem System, sowie ein deutlicher Ausbau der Funktionalität im Bereich Laufzeitadaption des Systems.

Da wir planen ODYSSEUS zur Überwachung dezentraler Energieerzeugern wie Windenergieanlagen einzusetzen, ist außerdem eine direkte Unterstützung der in dem Kontext eingesetzten Nachrichten auf Basis der Manufactoring-Message-Specification als Datenmodell geplant.

Strategie	Beschreibung
Zufall	Der auszuführende Operator/Operatorverbund wird per Zufall ausgewählt
Round Robin	Auswahl erfolgt per Round Robin verfahren
FIFO	Ein Datum wird solange wie möglich durch den Plan propagiert. Die Operatoren werden dementsprechend ausgewählt.
Prioritätsbasiert	Die Operatoren bekommen eine feste Priorität zugewiesen, nach denen der Scheduler die Ausführung richtet
Greedy	Ebenfalls prioritätsbasiert, allerdings wird die Operatorpriorität zur Laufzeit anhand der Selektivität (σ) und benötigten Verarbeitungszeit pro Element (e) berechnet ($\frac{1-\sigma}{e}$)
Multi-Threaded	Jeder Operator wird in einem eigenen Thread ausgeführt
Hybrid Multi-Threaded Scheduling	Aus dem PIPES Projekt stammende Strategie, in der der Operatorplan partitioniert wird und die einzelnen Partitionen als Threads ausgeführt werden. Innerhalb einer Partition kann theoretisch wiederum eine beliebige Strategie zur Auswahl des Operators gewählt werden, sollte die Partition aus mehr als einem (virtuellen) Operator bestehen.
Aurora Scheduling	Auswahl erfolgt nach dem im Aurora Projekt entwickelten Verfahren, das entweder versucht den Durchsatz, die Verzögerung oder den Speicherverbrauch zu optimieren
Chain Scheduling	Auswahl erfolgt nach dem im STREAM Projekt entwickelten Chain-Verfahren [BBDM03], das versucht die Größe der Interoperatorwarteschlangen zu minimieren

Tabelle 1: Mögliche Scheduling-Strategien im ODYSSEUS-Framework

Literatur

[ACc+03] Abadi, Daniel J. ; Carney, Donald ; Çetintemel, Ugur ; Cherniack, Mitch ; Convey, Christian ; Lee, Sangdon ; Stonebraker, Michael ; Tatbul, Nesime ; Zdonik, Stanley B.: Aurora: a new model and architecture for data stream management. In: *VLDB J.* 12 (2003), Nr. 2, S. 120–139

[BBDM03] Babcock, Brian ; Babu, Shivnath ; Datar, Mayur ; Motwani, Rajeev: Chain: Operator Scheduling for Memory Minimization in Data Stream Systems. In: Halevy, Alon Y. (Hrsg.) ; Ives, Zachary G. (Hrsg.) ; Doan, AnHai (Hrsg.): *SIGMOD Conference*, ACM, 2003. – ISBN 1–58113–634–X, S. 253–264

[BG08] Bolles, André ; Grawunder, Marco: Implementierung einer RDF-Datenstromverarbeitung mit SPARQL. (2008)

[Gra93] Graefe, Gotz: Query Evaluation Techniques for Large Databases. In: *ACM Computing Surveys* 25 (1993), Nr. 2, S. 73–170

[JG07] Jacobi, Jonas ; Grawunder, Marco: Vorhersage des Antwortverhaltens von Quellen auf Grundlage gelernter Muster im DynaQuest Anfrageprozess. In: Höpfner, Hagen (Hrsg.) ; Klan, Friederike (Hrsg.): *Grundlagen von Datenbanken* Bd. 02/2007, School of Information Technology, International University in Germany, 2007 (Technical Report), S. 67–71

[KS04] Krämer, Jürgen ; Seeger, Bernhard: PIPES: a public infrastructure for processing and exploring streams. In: *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. New York, NY, USA : ACM, 2004, S. 925–926

Verbesserung der Suche nach Webbildern durch automatische Annotationen

Sadet Alčić
Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
D-40225 Düsseldorf, Deutschland
alcic@cs.uni-duesseldorf.de

Zusammenfassung

Die inhaltsorientierte Suche in Bilddatenbeständen stellt existierende Bild-Retrieval Systeme aufgrund der *semantischen Lücke* zwischen der internen Darstellung der Bilder und den tatsächlich abgebildeten Objekten vor eine große Herausforderung.

Um die semantische Lücke zu umgehen, wird in diesem Beitrag die Bildmenge auf Webbilder eingeschränkt. Webbilder treten in Webseiten im Zusammenhang mit Textinhalten auf, die wiederum semantische Informationen zu den Bildern liefern können. Diese Informationen lassen sich mit Hilfe von Textmining-Verfahren aus den Textinhalten in Form von Stichworten gewinnen. Das Voranschalten einer stichwortbasierten Suche vor der inhaltbasierten Suche filtert semantisch nicht relevante Bilder aus der Ergebnismenge heraus. Somit ist eine höhere Suchqualität in einem auf diesem Ansatz basierenden System zu erwarten, was in den Implementierungen [Alc07] bestätigt wird.

1 Einleitung

Mit der rasanten Entwicklung des Internets und der Erschwinglichkeit breitbandiger Internet-Zugänge für Heimanwender wächst die Flut an abrufbaren Informationen explosionsartig. Waren vor einigen Jahren nur textuelle Informationen zugänglich, so werden nun auch Bilder, Audios und Videos der breiten Masse verfügbar gemacht. Umso wichtiger wird dabei die effiziente Verwaltung dieser Daten, da sonst die Suche nach Informationen und insbesondere der Zugriff auf informationentragende Medienobjekte unmöglich ist.

Bisherige *Infomation-Retrieval*-Verfahren sind lediglich auf Textdokumente anwendbar und können die Suchmodelle nicht ohne Weiteres auf die implizite Semantik der "Neuen Medienobjekte" übertragen. Bei Rasterbildern beispielsweise liegt uns lediglich eine rechteckige Matrix von Pixelwerten vor, die im Zusammenhang gesehen von einem Menschen zu Objekten, Personen und Umgebungen interpretiert wird. Anders als bei Texten, bei denen die einzelnen Wörter bereits eine gewisse Semantik durch die Sprache ausdrücken, existieren in Bildern explizit keine Semantik tragenden Bestandteile. Es sind also neue Verfahren nötig, die auf die besondere Darstellung der Medienobjekte zugeschnitten sind.

Speziell für Bilddaten bestehen bereits Verfahren, die eine Suche ermöglichen; hier werden zwei Verfahrensweisen, die oft in der Literatur erwähnt werden, grob vorgestellt. Ein bekannter Ansatz ist die *manuelle Annotation* von Bildern, wobei ein sogenannter Redakteur ein Bild betrachtet, es interpretiert und mit beschreibenden Stichworten versieht. Die Suche erfolgt dann, ähnlich wie im Information Retrieval, auf Basis dieser Stichworte. Dieser Ansatz hat jedoch einige Nachteile, wie der hohe manuelle Aufwand und die Subjektivität der Annotationen. Nichtsdestotrotz ist es ein gängiges Verfahren in vielen neuen Online-Communities, wie *Flickr* oder *YouTube*, wo alle Benutzer sich den Annotationsaufwand teilen.

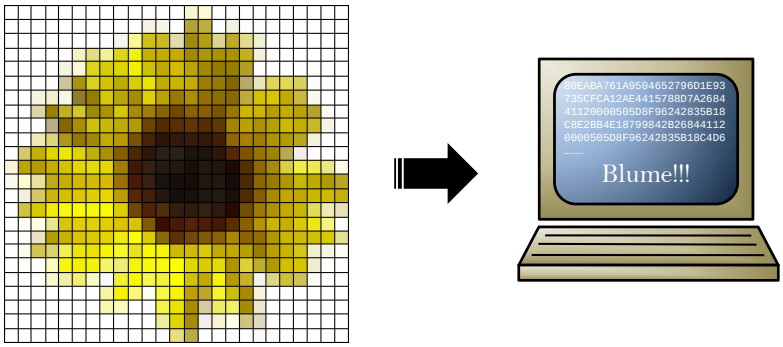

Abbildung 1: Die Herausforderung für Bildverwaltungssysteme

Ein weiterer Ansatz, der automatisch aus den Pixeldaten eines Bildes Merkmale berechnet und somit keinen manuellen Aufwand erfordert, ist bekannt unter dem Namen *Content-based Image Retrieval* (CBIR) und wurde bereits als zusätzliche Komponente in viele kommerzielle Datenbanksysteme [AFH+95, Ora99] integriert. Die Merkmale (Farb-, Textur- und Formeigenschaften) werden mit Hilfe von Verfahren aus der Bildverarbeitung extrahiert und stellen die *low-level* Beschreibung des Bildes dar, auf welcher die Suche basiert. Die Anfrage kann durch ein Beispielbild oder eine Spezifikation von bestimmten Bildeigenschaften festgelegt werden und im Ergebnis werden alle Datenbankbilder, nach der durch ein *Ähnlichkeitsmaß* berechneten Relevanz zur Anfrage sortiert. Wir haben in [Alc05] bereits ein eigenes CBIR-System vorgestellt. Aber auch dieses Verfahren hat erhebliche Defizite, wenn man beispielsweise nach semantischen Inhalten, wie Personen, Objekten oder Umgebungen suchen möchte, da diese aus der Pixeldarstellung nicht ohne zusätzliches Wissen ermittelt werden können. Diese Problemstellung, auch als *Semantische Lücke* (Semantic Gap) bekannt, wird in Abbildung 1 am Beispiel eines Bildes dargestellt.

2 Ein Ansatz zur automatischen Annotation von Bildern

Zur automatischen Annotation von Bildern, wird die Bildmenge auf Webbilder eingeschränkt. Als Webbilder werden Grafiken bezeichnet, die in Webseiten eingebettet sind. Eine Automatisierung des Annotationsprozesses ist hier möglich, da neben den Bildern auch weitere Informationen auf einer Webseite existieren, die semantische Informationen liefern können.

Die Annotation von Webbildern mit Stichworten aus dem zugehörigen Webdokument ist prinzipiell kein neues Verfahren, da dies inzwischen eine gängige Methode ist, die kommerzielle Bildsuchmaschinen wie *Google Bilder* und *Altavista* zur Bildindexierung verwenden. Auch in der Forschung existieren zahlreiche Arbeiten, die von diesem Ansatz profitieren. Dabei werden etwa in [CCS+04, OBMCP00] die kompletten Textinhalte der Webseite zur Annotation verwendet und die einzelnen Stichworte nach bestimmten Kriterien gewichtet, um deren Relevanz zum Inhalt des Bildes auszudrücken. In anderen Forschungsarbeiten, wie z.B. [HWLL05, FSA96, SC97, RLL+07], werden nur bestimmte Textelemente einer Webseite in Betracht gezogen. Hierzu existieren Regeln, nach denen die Auswahl der Elemente erfolgt. Zum Beispiel wird der Alternativtext der Bilder, der Seitentitel, unmittelbare Überschriften, der Linktext und die Wörter in einer bestimmten Umgebung des Bildes im Quelltext zur Annotation herangezogen. Der Nachteil dieser Regeln ist, dass viele für ein Bild nicht relevante Wörter in der Analyse berücksichtigt werden, aber auch möglicherweise relevante Wörter ausgelassen werden. Betrachten wir beispielsweise die Nachrichtenseite *SPIEGEL Online*, dann steht im Seitentitel meistens nur der Name der Webseite, der nichts mit dem Inhalt der Bilder gemein hat.

Die grundlegende Idee in diesem Beitrag besteht darin, Webseiten zu analysieren und den

dort vorkommenden Bildern möglichst passende Texte zuzuordnen. Dadurch fließen nur die Worte des Artikels, zu dem das Bild gehört, in die Annotation ein.

Um diese Aufgabe zu bewältigen, wurde ein Parser implementiert, welcher als Eingabe eine URL erhält und den durch diese Adresse erreichbaren Quelltext zunächst nach Bildelementen durchsucht. Auf Webseiten können viele Bilder existieren, von denen die meisten Seitenlayoutelemente oder Werbungen darstellen. Diese Bilder müssen bei der Quelltextanalyse als solche erkannt und rausgefiltert werden, da sie keinen Beitrag für die eigentlichen Informationen der Webseite tragen. Durch eine Definition von Regeln, die auf die Bildgröße, Seitenverhältnis und Grafikformat beruhen, kann ein effizienter Filter realisiert werden.

Den als relevant eingestuften Bildern soll im nächsten Schritt eine Textpassage zugeordnet werden. Dazu werden zunächst alle Textinhalte zu Passagen gruppiert. Ein Passage besteht zum Beispiel aus einer Überschrift, einer kurzen Zusammenfassung und einem längeren (verglichen mit der Länge der Zusammenfassung) Text. Der entwickelte Algorithmus zur Textzuordnung betrachtet Textpassagen in der Umgebung jedes Bildes. Diese werden als Kandidatenpassagen gekennzeichnet und zu einer Kandidatenmenge gruppiert. Im weiteren Verlauf werden sukzessive Kandidatenpassagen eines Bildes eliminiert. Dazu verwenden wir folgende zwei Regeln:

R1: Wenn in der Kandidatenmenge Textpassagen mit Überschrift, Zusammenfassung und Text vorhanden sind, dann entferne alle Kandidatenpassagen *ohne* diese drei Elemente.

R2: Entferne sukzessiv die im Quelltext am weitesten vom Bild entfernte Passage.

Die Abbruchbedingung für Regel **R2** ist erreicht, wenn nur noch eine Textpassage übrig ist. Diese wird dann dem Bild zugeordnet und einer weiteren Analyse unterzogen, welche aus dem Text gewichtete Indexterme ermittelt.

3 Textanalyse

Die einem Bild zugeordneten Texte müssen einer Textanalyse unterzogen werden, um für die Annotation geeignete Stichworte zu liefern. Die Aufgaben der Textanalyse sind bereits aus dem Information Retrieval bekannt; dort existieren auch effiziente Ansätze zur Lösung der Problemstellung. Der allgemeine Ablauf lässt sich der Abbildung 2 entnehmen.

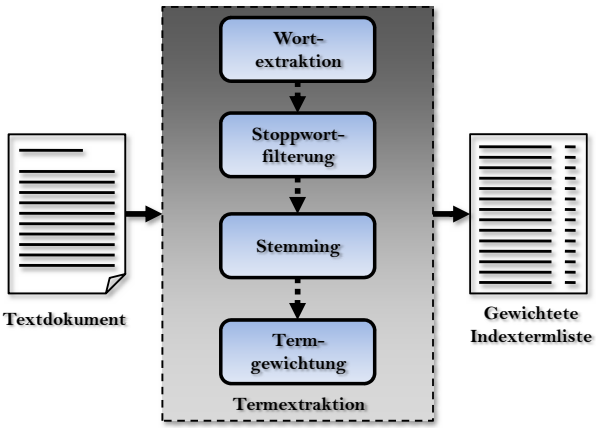

Abbildung 2: Textanalyseprozess

Die Texte werden zunächst in die einzelnen Wörter gespalten und Bilden eine Wortmenge. Diese Aufteilung ermöglicht eine Untersuchung der einzelnen Begriffe, die im weiteren Verlauf unbedigt notwendig ist.

Im nächsten Schritt erfolgt eine Eleminierung der Stoppwörter. Als Stoppwörter werden Wörter einer Sprache bezeichnet, die im Prinzip keinen semantischen Inhalt tragen und somit für das Bild irrelevant sind. Dazu wird eine bereits angefertigte Stoppwortliste herangezogen und es werden alle darin vorkommenden Wörter aus der Wortmenge entfernt.

Im Text kommen Wörter vor, die einen gleichen Stamm haben, jedoch unterschiedliche Flexionen dieses Stammwortes darstellen. Die Zurückführung von Wörtern auf ihre Grundform wird als Grundformreduktion oder Stemming bezeichnet. Hierzu gibt es sowohl regelbasierte Verfahren, welche versuchen aufgrund von bestimmten Endungen auf die Grundform zu schließen, als auch lexikonbasierte Verfahren, bei denen die Grundform im Lexikon nachgeschlagen wird. Einen regelbasierten Ansatz für die englische Sprache liefert Rainer Kuhlen in [Kuh77].

Zum Schluss werden den Termen Gewichtungen zugeordnet, um deren Relevanz zum Bild auszudrücken. Dabei unterscheidet man zwischen lokalen Gewichtungsfaktoren, die im Wesentlichen von dem Textauschnitt selbst abhängen, und globalen Gewichtungsfaktoren, welche die gesamte Textmenge betreffen. Ein lokaler Gewichtungsfaktor ist beispielsweise die Termfrequenz, d.h. die Häufigkeit des Terms im betrachteten Dokument. Als globalen Gewichtungsfaktor verwendet man die Dokumentenfrequenz, welche die Anzahl der Dokumente angibt, in denen der Term vorkommt.

4 Bildanalyse

Während die Textanalyse eine Bildindexierung durch gewichtete Indexterme realisiert, ist es die Aufgabe der Bildanalyse eine visuelle Beschreibung der Bilder zu liefern. Wie bereits in der Einleitung erwähnt werden mit Hilfe von statistischen Methoden aus der Bildverarbeitung sogenannte Featurewerte berechnet, die für bestimmte Eigenschaften repräsentativ sind. Man unterteilt diese Werte in Farb-, Textur und Formmerkmale. Abbildung 3 zeigt beispielhaft ein Bild und dessen Farbhistogramm.

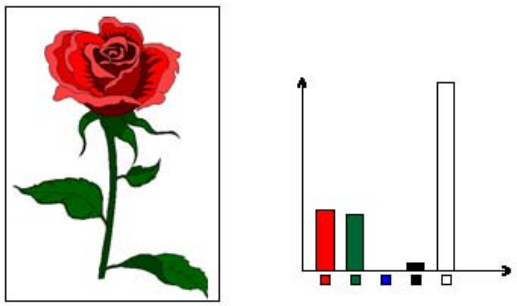

Abbildung 3: Farbhistogramm eines Bildes

Die Ermittlung des Farbhistogramms erfolgt unmittelbar aus der physischen Darstellung des Bildes. Zunächst wird der gesamte Farbbereich in eine vorgegebene Anzahl von Referenzfarben unterteilt. Dann erfolgt ein Durchlauf durch die gesamte Pixelmatrix, wobei die Pixelwerte der nächsten Referenzfarbe zugeordnet werden. Der Wert dieser Referenzfarbe wird dementsprechend um eins erhöht. Schließlich gibt das Histogramm also die Pixelhäufigkeiten von Referenzfarben im Bild an. Im Prototypen [Alc07] haben wir weit über 100 Merkmalswerte zu ungefähr 15 verschiedenen Merkmalen. Alle berechneten Werte werden zu einem Merkmalsvektor zusammengefasst und repräsentieren das Bild in einem mehrdimensionalen Vektorraum. Der Merkmalsvektor wird in der Datenbank abgelegt und steht für die Suche zur Verfügung.

5 Kombination der stichwort- und inhaltsbasierten Suche

Die inhaltsbasierte Suche kann durch Voranschalten der stichwortbasierten Suche enorm verbessert werden. Die wesentlichen Nachteile des inhaltsbasierten Ansatzes betreffen die mangelnde Semantik in den berechneten Featurewerten. Diese betreffen sowohl die Anfrageformulierung, als auch die Ergebnisaufbereitung in inhaltsbasierten Systemen.

Die Semantik von Webbildern lässt sich nun durch die ermittelten Annotationen ausdrücken. Damit kann ein Benutzer bei der Anfrageformulierung seine Anforderungen durch Stichworte festlegen und benötigt kein Anfragebild, wie es in CBIR-Systemen der Fall ist. Zugleich erfolgt die Anfrageverarbeitung basierend auf den Indextermen und den zugehörigen Gewichtungen. Im Ergebnis werden die Bilder sortiert nach Relevanz ihrer Indexterme zu den Anfragetermen dem Benutzer präsentiert.

An dieser Stelle ist es möglich die inhaltsbasierte Suche anzusetzen. Dazu kann der Benutzer eines der Ergebnisbilder auswählen und eine *Query by Example* starten, welche in diesem Fall nicht auf den gesamten Datenbestand ausgeführt wird, sondern lediglich die Ergebnisbilder der Stichwortsuche berücksichtigt. Dadurch sind die fehlerhaften Ergebnisse, die durch mangelnde Interpretationsfähigkeit der Semantik im traditionellen CBIR auftreten, beseitigt und es werden nur Bilder mit geforderter Semantik angezeigt. Zum Schluss entsprechen die Ergebnisse sogar hinsichtlich der Farben, Texturen und Formen den Anforderungen des Benutzers.

6 Ausblick

In dieser Arbeit wird ein Ansatz zur automatischen Annotation von Webbildern vorgestellt. Die Annotationsstichpunkte werden dabei aus dem zum einem Bild gehörenden Artikeltext generiert. Die Ermittlung dieses ausgezeichneten Textausschnitts aus dem Quelltext erfolgt mit dem in Abschnitt 2 erläuterten Verfahren. Das Verfahren funktioniert für viele Webseiten zuverlässig, jedoch kann es bei einem "ausgefallenen" Programmierstil eines Seitenautors zu Fehlern kommen. Die Erweiterung des Zuordnungsverfahrens um DOM (Document Object Model)-spezifische Konzepte ist als nächstes Forschungsziel anzusehen.

Die Suche nach Bildern erfolgt für den Benutzer in zwei Phasen. Während er in der ersten Phase seinen Bedarf mittels Stichworten formulieren muss, erfordert die zweite Phase ein weiteres Feedback, bei dem der Benutzer nochmals ein Anfragebild spezifiziert. Prinzipiell wäre es auch möglich durch Untersuchung der Ergebnisse der ersten Phase, beispielsweise durch Clustering, automatisch ein Bild als Anfragebild für die zweite Phase zu bestimmen und somit den zusätzlichen Aufwand für den Benutzer zu reduzieren.

Literatur

[AFH+95] Jonathan Ashley, Myron Flickner, James L. Hafner, Denis Lee, Wayne Niblack, and Dragutin Petkovic. The query by image content (qbic) system. In Michael J. Carey and Donovan A. Schneider, editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995*, page 475. ACM Press, 1995.

[Alc05] Sadet Alcic. Konzeption und Implementierung einer Datenbank zur Speicherung, Verwaltung und Retrieval von multimedialen Daten. Bachelorarbeit, Heinrich-Heine-Universität Düsseldorf, 2005.

[Alc07] Sadet Alcic. Improvement of Content-based Image Retrieval Quality using automatic Annotation. Masterarbeit, Heinrich-Heine-Universität Düsseldorf, 2007.

[CCS+04] Tatiana Almeida Souza Coelho, Pável Pereira Calado, Lamarque Vieira Souza, Berthier Ribeiro-Neto, and Richard Muntz. Image retrieval using multiple evidence ranking. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):408–417, 2004.

[FSA96] Charles Frankel, Michael J Swain, and Vassilis Athitsos. Webseer: An image search engine for the world wide web. Technical report, Chicago, IL, USA, 1996.

[HWLL05] Zhigang Hua, Xiang-Jun Wang, Qingshan Liu, and Hanqing Lu. Semantic knowledge extraction and annotation for web images. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 467–470, New York, NY, USA, 2005. ACM.

[Kuh77] Rainer Kuhlen. *Experimentelle Morphologie in der Informationswissenschaft.* Dokumentation München, 1977.

[OBMCP00] Michael Ortega-Binderberger, S. Mehrotra, K. Chakrabarti, and K. Porkaew. Webmars: A multimedia search engine for the world wide web. In *In Proceedings of the SPIE Electronic Imaging 2000: Internet Imaging*, San Jose, CA, 2000.

[Ora99] Oracle Corporation. Oracle8i iterMedia Audio, Image, and Video User's Guide and Reference. Release 8.1.5 (A67299-01), 1999.

[RLL+07] Xiaoguang Rui, Mingjing Li, Zhiwei Li, Wei-Ying Ma, and Nenghai Yu. Bipartite graph reinforcement model for web image annotation. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 585–594, New York, NY, USA, 2007. ACM.

[SC97] John R. Smith and Shih-Fu Chang. Image and Video Search Engine for the World Wide Web. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 84–95, 1997.

Cost-Effective Usage of Bitmap-Indexes in DS-Systems

Andreas Lübcke
Otto-von-Guericke-University Magdeburg
Department of Computer Science
Institute for Technical and Business Information Systems
D-39016 Magdeburg, Germany
P.O. Box 4120
andreas.luebcke@ovgu.de

1 Introduction

Index structures are a widely used function of Database Management Systems (DBMS) in order to tune them for a special purpose. Finding the right index configuration is on the one hand extremely complex (NP-problem [6]) and a variation of the Knapsack Problem [9] but on the other hand manual configuration requires high administrative effort by cost-intensive experts (like DBAs).
Regarding these facts several approaches like Feedback Control Loop [14] or MAPE from IBM [7] were introduced in current research of self-tuning techniques. These proposals should reduce costs for human resources and bring us closer to more efficient autonomous DBS.
Based on these perceptions we analysed the state of the art in the field of Data Warehousing and Online Analytical Processing (OLAP) systems. In this area it is noticeable that Bitmap-Indexes have often a marginal use in spite of crucial advantages in the given scenarios of DSS. Many case studies are available for the Oracle DBMS which has integrated a support for Bitmap-Indexes [8]. Instead we could find several approaches, which were based on heuristics of experts and DBAs like Join-Indexes [12, 4]. With the aid of these approaches it is possible to evaluate design tools like Database Tuning Advisor for Microsoft SQL Server 2005 [1, 2] or DB2 Design Advisor [15] and give online advise to tune a DBS, but until now it is not possible to give an online cost estimate for Bitmap-Indexes. Hence, a self-tuning approach like those for B-Trees [3, 11] is not available yet.

2 An Approach and Ideas on Bitmap-Index Tuning

2.1 Capability of Bitmap-Indexes

A Bitmap-Index is composed of a number of bit vectors where every existing value of the indexed attribute is represented by a single bit vector. The structure of a simple Bitmap-Index is shown in **Figure 1**. We know from experience that Bitmap-Indexes are particularly suitable for low cardinality attributes.
This fact leads us to the point where we must consider that only predicates from the *WHERE*-Clause are relevant to Bitmap-Indexes on a relation $r(R)$. Furthermore these predicates have to be in the following form:

$$A = const.\ with\ A \in R$$

We assume that attribute A , which should be used for index candidates, has to satisfy the following condition:

$$card(dom(A))/card(r(R)) < maxSelectivity \quad (1)$$

TID	B	F	O
$rowid_1$	0	0	1
$rowid_2$	0	1	0
$rowid_3$	0	0	1
$rowid_4$	1	0	0
$rowid_5$	1	0	0
$rowid_6$	0	1	0
$rowid_7$	0	0	1
⋮	⋮	⋮	⋮

Figure 1: Structure of a Sample-Bitmap-Index

We will name **Formula (1)** as Attribute Value Cardinality (**AVC**) in the following considerations. Some case studies show, that 1/10000 is a practicable threshold for the AVC, e.g. in Oracle 9i Date Warehousing Guide [10].
Other important facts according to Bitmap-Indexes are the low building cost and humble required space compared to other index structures.

2.2 Underlying Self-Tuning Concepts

The following explanation for the tuning procedure is based on the Feedback Control Loop [13]. The three steps of the control cycle will be clarified through **Figure 2**. First step is the

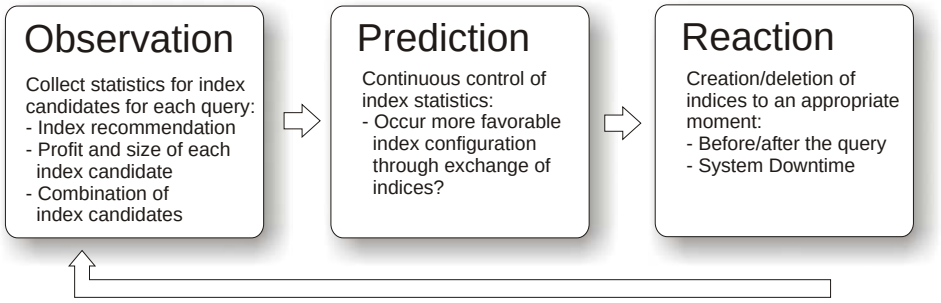

Figure 2: Control cycle according Weikum et al.

Observation where the system collects statistics about the data, existing index configuration and workload. In the second step we monitor continuously our statistics and the altering conditions, e.g. the load on one special relation. Hence, we can decide, when a new index configuration could be better than the old one. After all we execute several operations[1] to tune the index configuration.
But first of all we have to find algorithms to weigh the cost and benefit of an index candidate and also an index configuration. For the further views we take a number of queries $Q_1, ..., Q_m$ and set of index candidates $I_1, ..., I_n$ for granted. We come to the conclusion that the benefit of an index candidate I_i for a query Q_k is the maximum of difference between the execution time of query Q_k without index candidate I_i and with index candidate I_i. Hence, we will annotate in **Formula (2)**:

$$profit\,(Q_k, I_i) = max\,\{0, cost\,(Q_k) - cost\,(Q_k, I_i)\} \tag{2}$$

Furthermore, we must consider the involved administration costs to our profit calculation which we establish as $mcost\,(I_i)$ for index I_i. Assuming an assessment for one index candidate is not

[1] Create or delete indexes

practicable, we adopt an index configuration $C \subseteq I_1, ..., I_j$. So we make use of **Formula (2)** and account for maximisation of our profit in **Formula (3)**:

$$\sum_{i=1}^{m} max\left\{profit\left(Q_i, I_j : I_j \in C\right)\right\} - \sum_{I_j \in C} mcost\left(I_j\right) \tag{3}$$

In addition we have to consider that we have a restricted amount of space S in our index pool. From this we conclude to **Formula (4)**:

$$\sum_{I_j \in C} size\left(I_j\right) \leq S \tag{4}$$

Thus we can estimate after analyses of **Formula 1-4** [11], when a bitmap-index configuration C is cost-efficient. But we do not attend to the peculiarities of Bitmap-Indexes, hence, we can not compare Bitmap-Indexes performance clearly with other index structures.

2.3 Singularity of Bitmap-Indexes according to Cost-Models

Section 2.2 presents a generally admitted cost-model which is suitable for **B-Trees**. However, the cost-model does not fit for Bitmap-Indexes, because it does not consider the extreme high update costs ($O\left(n^2\right)$) for a Bitmap-Index. Beyond we have no possibility in our established cost-model to include the advantage of logical connectability for Bitmap-Indexes.
To the first fact, we assume that the update-problem of Bitmap-Indexes is a facile problem for DSS, because updates are in these scenarios rare. Mostly the updates will be performed after the load process, before the system goes online. Furthermore, the indexes will not updated because of the huge amount of data, but rather they will be recreated. An other technique is to mark deleted tuple in an extra bit vector, to avoid the reorganisation of the bitmap-index every time. But in OLAP systems both assumptions will not afford a cost-effective index configuration.
Thus, a cost-model extension is required. Supposed that the system has all statistics about the data, indexes and workload, we can estimate the behaviour (e.g. the workload) with help of a probability density function (Ψ) of a probability. We assume our probability density function (Ψ) could be an Exponential distribution to simulate a workload. That means:

$$f_\lambda(x) = \lambda e^{-\lambda x} \; for \; x \geq 0 \; and \; f_\lambda(x) = 0 \; for \; x < 0$$

Hence, we describe the update behaviour in our example with the parameter $\alpha = 1 - \Psi$, which is the negation of our probability density function (Ψ). In this case an attribute, which has not been updated for longer time, would have a high probability (theoretic) get updated in the near future. The conclusion is that the profit of a bitmap-index I_i have to be reduced, if it is updated very frequently. In other words the adapted cost-model is defined as follows:

$$profit(Q_k, I_i) = \alpha * max\left\{0, cost(Q_k) - cost(Q_k, I_i)\right\} \tag{5}$$

The second aspect, we have to regard, is the logical connectability for Bitmap-Indexes. From this can be concluded that a Bitmap-Index can not only be used for queries in which the single indexed attribute is used but also as prefix or suffix in complex queries like B-Trees too. Hence, the multi-usage of an index candidate have to be observed and should be included into the cost-model. The huge advantage of Bitmap-Indexes is the logical connection in any order between different Bitmap-Indexes to use the index candidates for complex queries. Hence, we assume that we did not have to create Multi-Value-Bitmap-Indexes, because they can build by their components (indexes). That saves space in the restricted index pool and gives us more variability for index configurations, because the attributes have not be multi indexed in different orders or combinations. Considering these facts, we suggest a solution which follows our proposal to the

Update-Problem. For this purpose we take any probability density function (Θ) for granted. But also we assume that probability density functions like the exponential distribution are not adequate for our intention. Normal curve of distribution or Student distribution are a more appropriate real world section in this case. In our example we choose the Student distribution because this distribution has been proven that it is suitable for sample analysis.
Under the same conditions as before, the Student distribution is defined as follows:

$$f_n(x) = \frac{\Gamma(\frac{n+1}{2})}{\sqrt{n\pi}\Gamma\left(\frac{n}{2}\right)}(1+\frac{x^2}{n})^{-\frac{n+1}{2}} \; for \; -\infty < x < +\infty$$

$$with \; \Gamma(x) = \int_0^{+\infty} t^{x-1}e^{-t}dt$$

Furthermore, we introduce a new parameter $\omega = 1 + \Theta$ to raise the benefit. However, we have to choose the enhancement not too severe, because the probability for re-use should not be estimated too high. But if the probability for re-use is zero, we get with aid of our parameter ω a minimum factor of one. These observations lead to a second new cost-model formula:

$$profit(Q_k, I_i) = \omega * max\left\{0, cost(Q_k) - cost(Q_k, I_i)\right\} \tag{6}$$

We have noticed the special characteristics of Bitmap-Indexes with **Formula 5** and **6**. Both formulas together form an improved new cost-model for Bitmap-Indexes:

$$profit(Q_k, I_i) = \alpha(\omega * max\left\{0, cost(Q_k) - cost(Q_k, I_i)\right\}) \tag{7}$$

The cost-model in **Formula 7** allows a comparison of Bitmap-Index configurations with other index structures. Additionally, we can describe the peculiarities for Bitmap-Indexes better than before.

2.4 Possibilities and Variants of Bitmap-Indexes

First, we will present the different variants of Bitmap-Indexes and why we do not observe them. One variant is the Interval-Coordinated-Bitmap-Index which is a special case and would get us away from a general approach for Bitmap-Indexes. Furthermore, these indexes are very complex and more difficult to predict. Hence, we should find first a solution for non-specialised Bitmap-Indexes. A second variant is the Multi-Value-Bitmap-Index which is more complex in his structure. This is one reason why they are such difficult to estimate. Additional complexity restricts our search domain to find a minimal and optimal[2] index configuration [5]. Furthermore, the Multi-Value-Bitmap-Indexes are not as flexible as desired for our targets. At last these indexes could build together by the single indexed attributes.
Beyond we will discuss the possibility to use Bitmap-Indexes together with other index structures. The difficulty lies in the assuredness that only one variety of index structure can be used in an index pool. The question we point to ourselves, how we could conquer or avoid this problem of index pool management. One idea could be to get a set of index pools, e.g. one for every index structure. Hence, we have to change the optimizer of the DBMS and all other parts which work with the indexes. But is it possible to resolve such a complex problem with all its variants and peculiarities? An other idea is to have a set of optimizer and management instances, one for each index structure. But we have the problem how these different optimizer can communicate with each other. And they have to collaborate with each other because we have to avoid multi-indexed attributes or relations. This solution could impose a huge overhead for optimization. These are challenging and interesting question which we want to answer.

[2] As far as possible because of the complexity of the search problem

3 Conclusion

We have shown different aspects in the field of Bitmap-Indexes. We pointed out the advantages and disadvantages of Bitmap-Indexes compared with other index structures. Furthermore, we developed an improved cost-model for Bitmap-Indexes according the characteristics of these indexes. The cost-model build the foundation to use index-self-tuning techniques including Bitmap-Indexes. There are many open problems and new approaches are needed, e.g. in the field of comparability and combined usage with other types of index structures.

References

[1] S. Agrawal, S. Chaudhuri, L. Kollár, A. P. Marathe, V. R. Narasayya, and M. Syamala. Database Tuning Advisor for Microsoft SQL Server 2005. In *VLDB '04*, pages 1110–1121, 2004.

[2] N. Bruno and S. Chaudhuri. To Tune or not to Tune? A Lightweight Physical Design Alerter. In *VLDB '06*, pages 499–510, 2006.

[3] N. Bruno and S. Chaudhuri. An online approach to physical design tuning. In *ICDE*, pages 826–835. IEEE, 2007.

[4] D. Burleson. How to use oracle9i bitmap join indexes. http://www.dba-oracle.com/art-builder-bitmap-join-idx.htm, November 2002. Visited Novmeber 2007.

[5] C.-Y. Chan and Y. E. Ioannidis. Bitmap index design and evaluation. pages 355–366, 1998.

[6] D. Comer. The Difficulty of Optimum Index Selection. *ACM Transactions on Database Systems*, 3(4):440–445, 1978.

[7] I. Corporation. An architectural blueprint for autonomic computing. White Paper, June 2005. Third Edition.

[8] A. B. Danchenkov and D. K. Burleson. *Oracle Tuning: The Definitive Reference*. Rampant Techpress, 2006.

[9] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer-Verlag, Berlin, Heidelberg, 2004.

[10] P. Lane and V. Schupmann. *Oracle9i Data Warehousing Guide, Release 2 (9.2)*, pages 122–125. Oracle Corporation, 2002.

[11] M. Lühring, K.-U. Sattler, E. Schallehn, and K. Schmidt. Autonomes index tuning - dbms-integrierte verwaltung von soft indexen. In *BTW*, pages 152–171, 2007.

[12] P. Valduriez. Join indices. *ACM Trans. Database Syst.*, 12(2):218–246, 1987.

[13] G. Weikum, C. Hasse, A. Moenkeberg, and P. Zabback. The COMFORT Automatic Tuning Project, Invited Project Review. *Information Systems*, 19(5):381–432, 1994.

[14] G. Weikum, A. Mönkeberg, C. Hasse, and P. Zabback. Self-tuning Database Technology and Information Services: from Wishful Thinking to Viable Engineering. In *VLDB '02*, pages 20 – 31, 2002.

[15] D. C. Zilio, J. Rao, S. Lightstone, G. M. Lohman, A. Storm, C. Garcia-Arellano, and S. Fadden. DB2 Design Advisor: Integrated Automatic Physical Database Design. In *VLDB '04*, pages 1087–1097, 2004.

Effiziente Textsuche mit Positionsinformation

Andreas Broschart Ralf Schenkel

Universität des Saarlandes

Zusammenfassung

Top-k-Aggregationsverfahren zur effizienten Bestimmung der besten Ergebnisse einer Anfrage sind ein wichtiger Baustein moderner Anfrageverarbeitung. Mit der Familie der Threshold-Algorithmen haben sich dabei in den letzten Jahren Standardverfahren etabliert, die sich in vielen Anwendungsfällen einsetzen lassen. Für konkrete Anwendungen können dagegen spezielle Lösungen bessere Performance bieten, die Anwendungswissen ausnutzen. Dieses Paper diskutiert, wie Textsuche durch Integration von Positionsinformation in Scoring und Anfrageverarbeitung sowie den Einsatz von anwendungsspezifischen Indexstrukturen um mindestens eine Größenordnung gegenüber den besten Thresholdverfahren beschleunigt werden kann. Anhand von Experimenten auf einer sehr großen Textsammlung zeigen wir außerdem, dass sich die Ergebnisgüte dabei nicht verschlechtert, sondern sogar leicht verbessert.

1 Motivation

Textsuche früherer Jahre verwendete zum Berechnen der Menge von top-k Ergebnissen für eine Query meist Modelle, die Termhäufigkeiten (tf) und inverse Dokumentfrequenzen (idf) der Queryterme einsetzten, um ein Relevanzmaß zu berechnen. Bekannte Modelle dieser Art sind tf*idf, Okapi BM25 und statistische Sprachmodelle [2]. Proximityinformation in Dokumenten wurde hierbei nicht berücksichtigt. Unter Termproximityinformation verstehen wir dabei die Positionen von Querytermen in Dokumenten (n-tes Wort im Dokument) und wie nahe die Querytermauftreten in einem Dokument zueinander stehen. Um zu zeigen, warum die Einbeziehung von Proximityinformation sinnvoll ist, betrachten wir die Query "*surface area* of *rectangular pyramids*". Scoringmodelle, die keine Proximityinformation verwenden, geben mit hoher Wahrscheinlichkeit allgemeine mathematische Dokumente zurück, die jeden der vier Terme *surface, area, rectangular* und *pyramid*, oft im Dokument enthalten. Jedoch werden nicht notwendigerweise Dokumente zurückgeliefert, die den Informationsbedarf des Nutzers befriedigen: es könnte sein, dass in einem Ergebnisdokument die Berechnung des Volumens von Pyramiden sowie die Berechnung der Grundfläche rechteckiger Prismen diskutiert wird. Eine Möglichkeit dieses Problem zu umgehen, ist die Formulierung von Phrasenqueries, die es wahrscheinlich machen, dass gefundene Dokumente die gewünschte Information enthalten. Allerdings werden so auch viele relevante Dokumente aus der Ergebnismenge ausgeschlossen, z.B. welche, die über "*surface area* of *pyramids* having a *rectangular base*" reden. An diese Stelle können proximitybasierte Scoringmodelle treten, die eine Art weiches Phrasenquerying erlauben. In erster Näherung erhalten dabei Dokumente mit dicht beieinander stehenden Querytermauftreten größere Scores als Dokumente mit Querytermauftreten, die von vielen Nicht-Querytermen getrennt werden. Proximitybasierte Scoringmodelle existieren seit den 90er Jahren, sind aber auch aktuell auf großen IR-Konferenzen vertreten. Einige dieser Ansätze klassifizieren wir in [3]. Während es viele Vorschläge für effektive Scoringfunktionen mit Verwendung von Proximityinformation gab, haben nicht viele Arbeiten den Effizienzaspekt untersucht. In [3] haben wir gezeigt, dass das Ausnutzen von Proximityinformation nicht nur die Ergebnisgüte gegenüber aktuellen Inhaltsscores verbessern kann, sondern auch die Effizienz der Anfrageverarbeitung um ein bis zwei Größenordnungen steigern kann.

2 Scoring

Für ein gegebenes Dokument d der Länge l bezeichnen wir den Term, der an Position i von d auftritt, als $p_i(d)$ oder p_i, wenn aus dem Kontext klar wird, welches Dokument gemeint ist. Für einen Term t bezeichnen wir mit $P_d(t) \subseteq \{1, \ldots, l\}$ die Menge der Positionen im Dokument d, an denen t vorkommt oder wir schreiben kurz $P(t)$. Gegeben eine Query $q = \{t_1, \ldots, t_n\}$ schreiben wir $P_d(q) := \cup_{t_i \in q} P_d(t_i)$ für die Positionen von Querytermen im Dokument d oder $P(q)$, wenn d aus dem Kontext gegeben ist. Wir bezeichnen Positionspaare unterschiedlicher Queryterme in Dokument d mit

$$Q_d(q) := \{(i,j) \in P_d(q) \times P_d(q) \mid i < j \wedge p_i \neq p_j\}$$

Büttchers Scoringmodell [1] ist eine Linearkombination aus BM25 Content-Scoringfunktion und einem im Einfluss beschränkten Proximityscore, der die Form eines BM25-Scores übernimmt und die $tf_d(t)$-Werte durch einen akkumulierten Proximityscore $acc_d(t)$ ersetzt:

$$score_{\text{Büttcher}}(d,q) = score_{\text{BM25}}(d,q) + \sum_{t \in q} min\{1, idf(t)\} \frac{acc_d(t) \cdot (k_1 + 1)}{acc_d(t) + K}$$

K ist wie im BM25-Score als $K = k \cdot [(1-b) + b \cdot \frac{|d|}{avgdl}]$ definiert, mit $|d|$ als Dokumentlänge und $avgdl$ als durchschnittliche Dokumentlänge in der Kollektion. b, k_1 sowie k sind konfigurierbare Parameter, die auf $b = 0,5$ und $k = k_1 = 1,2$ gesetzt werden [1].

Unser Scoringmodell basiert auf Büttchers Scoringmodell, das für hohe Ergebnisgüte steht, aber nicht für die Indexvorberechnung geeignet ist, da es bei der Berechnung von $acc_d(t)$ nur aufeinanderfolgende Querytermpaare in Dokumenten betrachtet. Wir modifizieren es derart, dass die Ergebnisgüte kaum beeinflusst (das haben wir experimentell gezeigt), aber Indexvorberechnung ermöglicht wird: wir ignorieren die Dokumentenlänge, indem wir $b = 0$ setzen und zur Berechnung der acc-Werte *jedes* Querytermauftreten betrachten statt ausschließlich *adjazente* Queryterme. Die modifizierte Akkumulationsfunktion acc' definieren wir als

$$acc'_d(t_k) = \sum_{(i,j) \in Q_d(q) : p_i = t_k} \frac{idf(p_j)}{(i-j)^2} + \sum_{(i,j) \in Q_d(q) : p_j = t_k} \frac{idf(p_i)}{(i-j)^2} \tag{1}$$

Da der Wert von $acc'_d(t_k)$ nicht nur von d und t_k abhängt, sondern auch von den anderen Querytermen, können wir diesen Wert immer noch nicht unabhängig von der Query berechnen. Jedoch können wir die Definition von $acc'_d(t_k)$ wie folgt neu formulieren:

$$acc'_d(t_k) = \sum_{t \in q} idf(t) \underbrace{\left(\sum_{\substack{(i,j) \in Q_d(q) : \\ p_i = t_k, p_j = t}} \frac{1}{(i-j)^2} + \sum_{\substack{(i,j) \in Q_d(q) : \\ p_i = t, p_j = t_k}} \frac{1}{(i-j)^2} \right)}_{:= acc_d(t_k, t)} \tag{2}$$

$$= \sum_{t \in q} idf(t) \cdot acc_d(t_k, t) \tag{3}$$

Damit wurde $acc'_d(t_k)$ als monotone Kombination der Querytermpaar-Scores $acc_d(t_k, t)$ repräsentiert, die Querytermreihenfolge ist dabei irrelevant, d.h. $acc_d(t_k, t) = acc_d(t, t_k)$. Wir können diese Paarscores für alle Termpaare, die in Dokumenten vorkommen, vorberechnen und sie in nach absteigendem Score geordneten Indexlisten anordnen. Indem unser Algorithmus auch auf diesen Listen sequenzielle Zugriffe ausführt, können wir in Analogie zu den Queryterm-Listen auch leicht obere Schranken für $acc'_d(t_k)$ berechnen.

3 Query Processing

3.1 Indexstrukturen

Wir materialisieren zwei Arten von Indizes, die in erster Linie sequenziell zugegriffen werden, aber dennoch auch wahlfreien Zugriff erlauben. Für jeden Term t_i materialisieren wir eine `Text-Indexliste(TL)`, eine Liste der Form $(d_k, score_{BM25}(d_k, t_i))$, sortiert nach absteigendem BM25-Score. Weiterhin halten wir für jedes ungeordnete Paar $\{t_i, t_j\}$ von Termen mit $t_i < t_j$ (lexikographische Ordnung) eine `kombinierte Indexliste(KL)` vor, welche die Form $(d_k, acc_{d_k}(t_i, t_j), score_{BM25}(d_k, t_i), score_{BM25}(d_k, t_j))$ besitzt und nach absteigendem acc-Score sortiert ist. Analog gibt es weiterhin für $\{t_i, t_j\}$ eine `Proximity-Indexliste(PXL)` der Form $(d_k, acc_{d_k}(t_i, t_j))$ gleicher Sortierung. Neben diesen vollständigen Indizes bauen wir auch unterschiedliche Varianten dieser Indizes auf, die nur einen Teil der TL- und KL-Einträge materialisieren. Alle sind im Gegensatz zu den vollständigen Indizes nicht nach absteigendem Score, sondern nach aufsteigenden Dokument-IDs sortiert.

Wir untersuchen zunächst die Größe unserer Indizes bei unterschiedlichen Kürzungsniveaus (ohne Stemming, weil die Indexgröße ohne Stemming eine obere Schranke für die Indexgröße mit Stemming liefert). Da es zu viel Plattenplatz erfordert, alle kombinierten Listen zu materialisieren, haben wir zufällig 1,5 Mio. Termpaare, die in mindestens 10 Dokumenten der Kollektion auftreten, ausgewählt. Unter diesen Termpaaren besaßen etwa 1,2% eine nichtleere kombinierte Liste. Tabelle 1 zeigt die Indexgrößen (Anzahl der Listeneinträge) für Text- (genaue Werte) und kombinierte Listen (geschätzte Werte) mit unterschiedlicher maximaler Anzahl von Einträgen. Sie werden je nach Art der gespeicherten Daten, die in Abschnitt 3 beschrieben wurden, berechnet bzw. geschätzt. Wir nehmen an, dass Dokumenten-IDs und Scores jeweils 8 Bytes groß sind. Folglich benötigt ein `TL`-Eintrag 16 Bytes, da er aus einer Dokumenten-ID und BM25 Score besteht. Ein `KL`-Eintrag ist mit 32 Bytes doppelt so groß, weil er Dokumenten-ID, acc-Score und zwei BM25 Scores umfasst. Es ist offenkundig, dass es - selbst mit Längenbeschränkung - nicht machbar ist, alle kombinierten Listen auf Festplatte vorzuhalten. Allerdings kann die Indexgröße durch Entfernen aller Einträge mit Scores unter 0,01 (Approximation für Fenstergrößen von 10) um einen Faktor von 8 bis 15 gegenüber dem ungekürzten Index merklich reduziert werden. Weil Plattenplatz heutzutage günstig ist, kann das toleriert werden. Die Größe der TL-Indizes ist relativ unproblematisch, weil sie ungekürzt nur 47,5 GB benötigen und weiter durch Vorgabe maximaler Listenlängen größenreduziert werden können. Weitaus problematischer sind KL-Indizes, die mit geschätzten 41 TB untragbar groß sind. Durch Beschränken der Listenlängen kann die Situation verbessert, aber nicht gelöst werden. Durch Einführung von Mindestscores von 0,01 auf KL-Indizes erhalten wir tolerierbare Größen zwischen 860 GB und 1,3 TB für KL-Indizes. Wie wir später zeigen werden (Tabelle 2), können exzellente Ergebnisse bereits durch Beschränkung der Listengröße auf maximal 2.000 Einträge erreicht werden. Dazu benötigen wir etwas mehr als 1 TB Plattenplatz, um `TL+KL`(2.000, score≥0,01) auf einer Dokumentenkollektion mit 426 GB Daten ausführen zu können.

Index/max. Länge	1.000		2.000		3.000		ungekürzt	
TL	5,3 GB	[355]	6,6 GB	[442]	7,4 GB	[496]	47,5 GB	[3.191]
KL (geschätzt)	12,7 TB	[435.326]	15,0 TB	[515.079]	16,5 TB	[566.277]	41,0 TB	[1.410.238]
KL, score $\geq$ 0,01 (gesch.)	860 GB	[28.855]	1,1 TB	[38.985]	1,3 TB	[45.186]	2,5 TB	[87.049]

Tabelle 1: Indexgrößen(Plattenplatz[Mio. Tupel]), variierte max. Längen, mit/ohne Mindestscore

3.2 Merge-Join basierte Anfrageverarbeitung

In [3] haben wir gezeigt, dass mit Hilfe dieser gekürzten Indexlisten vergleichbare Präzisionswerte wie auf ungekürzten Indexlisten erzielt werden können. Die Messungen wurden damals mittels

Algorithmen aus der Familie der Threshold-Algorithmen in der Open Source TopX-Suchmaschine[1] durchgeführt. Da Threshold-Algorithmen bei der Ausführung einen gewissen Overhead bei der Verwaltung von Kandidatendokumenten verursachen, verwenden wir hier die gekürzten Indexlisten, um Queryprocessing mit Hilfe von Merge Joins durchzuführen. Dazu werden für jeden Queryterm zunächst die entsprechende gekürzte TL aus der Datenbank in den Hauptspeicher geladen und je nach Modus gegebenenfalls für jedes Querytermpaar eine KL. Auf diesen kurzen, nach Dokumenten-IDs sortierten Listen wird nun im Hauptspeicher ein n-ärer Merge-Join ausgeführt (mit n=Anzahl der Listen im Hauptspeicher). In jedem Verarbeitungsschritt betrachten wir das Dokument mit der höchsten Dokument-ID, dessen Score noch nicht berechnet wurde. Dabei können wir, dem Merge Join-Ansatz entsprechend, auf wahlfreie Zugriffe verzichten und die Listen sequenziell lesen. Die kurzen Resultatslisten werden anschließend nach Score sortiert und die top-k Dokumente mit den höchsten Scores zurückgegeben. Die leichtgewichtige Implementierung unseres Ansatzes verzichtet wegen der Kürze der Listen auf Parallelisierung. Wir evaluieren Kombinationen von Pruning mit Mindestscore, maximalen Listenlängen und der Pruningtechnik von Soffer et al. [4] für k=10.

3.3 TopX mit vollständigen Listen

Um die Effizienz unseres Merge-Join basierten Ansatzes zu demonstrieren, vergleichen wir ihn mit einer Evaluation mit Hilfe von TopX auf vollständigen Listen. TopX verwendet dabei für das Processing jeder einzelnen Liste einen separaten Thread und nutzt zur Beschleunigung des Queryprocessings Wissen über paarweises Auftreten von Querytermen im gleichen Dokument aus. Der Algorithmus terminiert, typischerweise lange bevor die Listen vollständig gelesen wurden. Um die Auswertung weiter zu beschleunigen, verwendet TopX zusätzlich wahlfreie Zugriffe.

4 Evaluation

4.1 Setup

Unsere Experimente wurden auf dem TREC Terabyte Korpus, welcher etwa 25 Millionen Dokumente umfasst und etwa 426 GB groß ist, durchgeführt. Wir haben unsere Methoden mit den 100 Adhoc Anfragen der TREC Terabyte Tracks von 2004 und 2005 geprüft. Weil wir an den top-k Ergebnissen interessiert sind, haben wir Präzisionswerte für verschiedene Listenlängen gemessen. Zur Messung der Effizienz geben wir die durchschnittlichen Laufzeiten der 100 Queries an. Wir zeigen außerdem die durchschnittlichen Präzisionswerte für diese Queries nach k Ergebnissen. Wir haben die Dokumente mit dem Indexer des TopX-Systems bei eingeschalteter Stopwortentfernung indiziert und die Proximitylisten mit einem weiteren Tool berechnet. Im Okapi BM25 Modell haben wir die Parameter $k = k_1 = 1.2$ und $b = 0,5$ verwendet. Für die Berechnung der gekürzten Listen haben wir ein zusätzliches Tool verwendet, das die Einträge von Dokumenten mit den höchsten BM25 Scores in getrennten Tabellen, je nach Intensität der gewünschten Listenkürzungen, materialisiert hat. Gekürzte kombinierte Listen für Querytermpaare, die Proximityscore und BM25 Scores für beide Queryterme enthalten, wurden mit einem anderen Tool aufgebaut. Für die Durchführung unserer Experimente haben wir einen Server mit Microsoft Windows 2003 Enterprise 64-bit Edition verwendet, der über einen Dual Core AMD Opteron Prozessor mit 2,6 GHz und 32 GB RAM verfügt. Sowohl TopX als auch der Merge-Join basierte Ansatz wurden in einer Sun Java 1.6 VM ausgeführt, die maximal 4 GB Hauptspeicher verwenden durfte. (Der tatsächliche Speicherbedarf liegt allerdings weit darunter.) Die Baseline der Experimente wurde mit TopX im `RR-LAST` Modus bei einer Batchgröße von 100.000 berechnet, d.h. sequenzielle Zugriffe reihum in den Indexlisten in Batches von 100.000 Einträgen und Verschieben der wahlfreien Zugriffe ans Ende des Queryprocessings.

[1] http://topx.sourceforge.net

Indexlisten wurden in einem Oracle 10g DBMS abgespeichert. Tabelle 2 zeigt durchschnittliche Präzisionswerte und Laufzeiten (in ms) für verschiedene TopX- und Merge Join-basierte Läufe. Dabei sind die TL+KL-Läufe schneller als die TL+PXL-Läufe bei gleicher Präzision, TL-Läufe weisen eine geringere Präzision als TL+KL und TL-PXL Läufe auf. Begründungen und Information über die Anzahl von sequenziellen und wahlfreien Zugriffen derartiger TopX-Läufe findet man in Abschnitt 6.2 in [3]. Während die Laufzeit der TopX-Läufe in der Regel mit steigendem k wächst, ist die Laufzeit der Merge Join-Läufe von k unabhängig, da die Listen unabhängig von k vollständig sequentiell gelesen werden, während TopX die Queryauswertung abhängig von der Anzahl der gewünschten Ergebnisse frühzeitig terminieren kann. Die Laufzeit verhält sich im Falle der Merge Join-Implementierung linear proportional zu den Längen der gelesenen Listen. Die Merge Join-Implementierung mit den gekürzten TL+KL-Listen kann die hervorragenden Präzisionswerte der TopX-Runs mit ungekürzten Listen aufrecht erhalten und gleichzeitig die Laufzeit um ein bis zwei Größenordnungen verringern.

	Run	k=10		k=30		k=50		k=70		k=100	
		P@k	t[ms]	P@k	t[ms]	P@k	t[ms]	P@k	t[ms]	P@k	t[ms]
TopX	TL	0,57	5.220	0,49	6.129	0,45	6.868	0,42	7.974	0,38	8.827
	TL+PXL	0,61	8.476	0,52	12.811	0,48	15.753	0,45	14.485	0,40	15.524
	TL+KL	0,61	2.565	0,52	4.688	0,48	7.045	0,45	7.913	0,40	10.083
Merge Join	TL(1.000 Tupel)	0,30	35	0,24	36	0,22	35	0,20	34	0,18	35
	TL(2.000 Tupel)	0,34	68	0,27	71	0,24	64	0,22	67	0,20	67
	TL(3.000 Tupel)	0,37	101	0,29	103	0,27	100	0,25	99	0,22	100
	TL+KL(1.000 Tupel)	0,60	78	0,51	80	0,46	80	0,43	81	0,39	79
	TL+KL(2.000 Tupel)	0,61	156	0,51	155	0,47	153	0,44	155	0,40	158
	TL+KL(3.000 Tupel)	0,61	225	0,52	230	0,47	224	0,44	223	0,40	232
	TL+KL(1.000; $\epsilon = 0,025$)	0,60	79	0,51	80	0,46	80	0,43	81	0,39	81
	TL+KL(2.000; $\epsilon = 0,025$)	0,61	154	0,51	153	0,47	153	0,44	155	0,40	153
	TL+KL(3.000; $\epsilon = 0,025$)	0,61	223	0,52	223	0,47	225	0,44	225	0,40	224
	TL+KL(1.000; score$\geq$0,01)	0,62	70	0,51	73	0,46	70	0,42	71	0,38	76
	TL+KL(2.000; score$\geq$0,01)	0,62	137	0,51	135	0,47	134	0,44	135	0,39	134
	TL+KL(3.000; score$\geq$0,01)	0,62	193	0,51	199	0,47	191	0,44	191	0,40	191

Tabelle 2: Vergleich: TopX auf ungekürzten Listen mit Mergejoin auf gekürzten Listen

5 Ergebnis

Dieses Paper präsentierte Algorithmen und Implementierungstechniken zur effizienten Evaluation von top-k Anfragen auf Textdaten unter Ausnutzung von Positionsinformation der Queryterme in den Ergebnissen. Wir haben gezeigt, dass wir durch Einsatz eines leichtgewichtigen n-ären Merge Joins in Kombination mit gekürzten und nach Dokument-ID sortierten Indexlisten substantielle Vorteile erzielen können. So sparen wir nicht nur viel Plattenplatz ein, sondern können gleichzeitig auch die Evaluation um ein bis zwei Größenordnungen gegenüber der Auswertung mit TopX auf vollständigen Listen beschleunigen.

Literatur

[1] Stefan Büttcher and Charles L. A. Clarke. Efficiency vs. effectiveness in terabyte-scale information retrieval. In *TREC*, 2005.

[2] David A. Grossman and Opher Frieder. *Information Retrieval*. Springer, 2005.

[3] Ralf Schenkel, Andreas Broschart, Seung won Hwang, Martin Theobald, and Gerhard Weikum. Efficient text proximity search. In Nivio Ziviani and Ricardo A. Baeza-Yates, editors, *SPIRE*, volume 4726 of *Lecture Notes in Computer Science*, pages 287–299. Springer, 2007.

[4] Aya Soffer et al. Static index pruning for information retrieval systems. In *SIGIR*, pages 43–50, 2001.

Die Verwendung von Ontologien für die semantische Integration von Daten und Werkzeugen

Dennis Heimann[1], Jens Nieschulze[1], Birgitta König-Ries[2]

[1]Max-Planck-Institut für Biogeochemie, Jena
{dheimann, jniesch}@bgc-jena.mpg.de

[2]Friedrich-Schiller-Universität, Jena
koenig@informatik.uni-jena.de

Zusammenfassung

Domänenübergreifende und nachhaltige Interpretierbarkeit von Daten und damit verbundene Auswertungs- und Analysemöglichkeiten gewinnen im Rahmen wissenschaftlicher Forschungsprojekte immer mehr an Bedeutung. Trotz der zunehmenden Anreicherung der Daten mit Metainformationen ist eine fachbereichsübergreifende Auswertung oft nicht möglich, da sich die Semantik der Daten nicht ohne Weiteres erschließen läßt. Dies hat zur Folge, dass semantisch ähnliche oder gleiche Datenbestandteile aufgrund ihrer strukturellen oder syntaktischen Unterschiede in der Regel nicht automatisch verknüpfbar sind. Schwierigkeiten liegen hier neben der Verknüpfung der heterogenen Daten untereinander, außerdem in der Herstellung von Beziehungen zu geeigneten Analyse-Werkzeugen, um beispielsweise statistische Auswertungen zu ermöglichen. Die hier gezeigten Lösungsansätze basieren auf der Verwendung von Ontologien, mit deren Hilfe die notwendigen semantischen Beziehungen abgebildet werden können. Es werden aktuelle Entwicklungen diskutiert und bestehende Herausforderungen analysiert.

1 Einleitung

Heutige wissenschaftliche Großprojekte haben zunehmend den Anspruch, erhobene Forschungsdaten nicht nur einfach in einer Datenbank abzulegen, sondern diese nachhaltig, d.h. vor allem langfristig und domänenübergreifend interpretierbar, in einem geeigneten Informationssystem zu speichern und der Allgemeinheit zugänglich zu machen. Solche Systeme verfolgen dabei unter anderem zwei Aspekte, die Daten- und die Werkzeug-Integration, welche z.B. auch in den Anforderungen an das Informationsystem für die Biodiversitäts-Exploratorien[1] (BE) wiederzufinden sind.

Die BE werden von der DFG[2] gefördert und untersuchen auf drei Langzeituntersuchungsgebieten in Deutschland (Schwäbische Alb, Schorfheide-Chorin und Hainich) den Einfluss von Landnutzungsänderungen auf die Biodiversität und deren Auswirkungen auf Ökosystemfunktionen und -dienstleistungen. Hierzu werden über voraussichtlich 12 Jahre in über 30+ Teilprojekten mit über 150 Mitarbeitern auf insgesamt mehr als 3000 Feldplots die Zusammenhänge zwischen Landnutzung, Biodiversität und Ökosystemprozessen untersucht.

Alle in den BE erhobenen Daten sollen in einer Datenbank gespeichert werden und zentral zugreifbar sein. Neben den unterschiedlichen Sichtweisen der Fachbereiche auf die Projektziele, muss unter anderem berücksichtigt werden, dass jedes Teilprojekt, möglicherweise auch jeder

[1]http://www.biodiversity-exploratories.de/
[2]Deutsche Forschungsgemeinschaft http://www.dfg.de/

Wissenschaftler, seine eigenen Formate, Bezeichnungen und Schemata für die Erhebung der Daten verwendet.

Daraus und aus zusätzlichen Vorgaben der DFG ergeben sich für das Datenmanagement eine Reihe von Herausforderungen und Zielstellungen.

Ziel ist zum einen, die Daten auch dann noch auswerten zu können, wenn deren Erheber nicht mehr zur Verfügung stehen, weil diese beispielsweise bereits aus dem Projekt ausgeschieden sind. Zum anderen besteht ein interdisziplinärer Anspruch darin, die Auswertung und Analyse der vielen heterogenen Daten über Projektgrenzen hinaus zu gewährleisten, um eventuelle, fachbereichsübergreifende Zusammenhänge zu entdecken.

Gerade fachbereichsübergreifende Auswertungen scheitern häufig daran, dass sich die Bedeutung eines Datensatzes, trotz Metainformationen, nicht ohne weiteres erschließen läßt, wodurch sich semantisch ähnliche oder gleiche Datenbestandteile aufgrund ihrer strukturellen oder syntaktischen Unterschiede in der Regel nicht automatisch verknüpfen lassen.

Für Auswertungs- und Analysezwecke wäre es außerdem sinnvoll, Daten aus externen Quellen, sowie externe Anwendungen in das System zu integrieren und semantisch mit bestehenden Daten zu verknüpfen. So könnten Berechnungen oder auch Visualisierungen direkt auf den Daten durchgeführt werden.

Bereits existierende Lösungsansätze verwenden für die reine Datenintegration oft Ansätze, die auf einem Einsatz von Ontologien basieren [WVV+01]. Bei der Integration von Analyse-Werkzeugen bauen dagegen bisher nur wenige Entwicklungen auf Ontologien auf, was sie oft relativ unflexibel bezüglich dynamischer Werkzeug-Integration macht [Rif07].

Diese Arbeit verschafft einen Überblick über den Einsatz von Ontologien bei der Daten- und Werkzeug-Integration in wissenschaftlichen Informationssystemen, und untersucht, ob bestehende Ansätze der langfristigen Datenhaltung und Interpretierbarkeit genügen. Dazu wird in den Abschnitten 2 und 3 zunächst die prinzipielle Verwendung von Ontologien zur Daten- und Werkzeug-Integration beschrieben. Anschließend werden die Ansätze in Abschnitt 4 hinsichtlich der beschriebenen Problemstellung diskutiert und schließlich in Abschnitt 5 zusammengefasst.

2 Daten-Integration

Eine Ontologie ist eine explizite formale Spezifikation einer Konzeptualisierung [Gru93]. Mit Hilfe von Ontologien kann die Semantik zu integrierender Datenquellen beschrieben werden. Hierbei können sowohl eine globale Ontologie, welche die Datenquellen über ein gemeinsames Vokabular verknüpft, als auch mehrere verschiedene Ontologien, die sich möglicherweise zum Teil inhaltlich überschneiden und entweder auf verschiedenen oder dem gleichen Vokabular basieren, eingesetzt werden.

Jeder Ansatz bietet dabei verschiedene Vor- und Nachteile. So hängt es vor allem vom abgedeckten Anwendungsbereich der zu integrierenden Datenquellen ab, welcher Ansatz schließlich Verwendung findet. Werden beispielsweise nur Quellen eines Bereichs mit gleicher Terminologie integriert, bietet sich eine einzelne globale Ontologie an [WVV+01]. Sollen dagegen Datenquellen aus verschiedenen Bereichen integriert werden, ist es sinnvoller eine möglichst erweiterbare Ontologie als globale Ontologie zu verwenden, welche gemeinsame Basis-Konzepte über verschiedene Domänen hinweg ausdrücken kann. Eine solche Ontologie wird dann als *Core Ontology* bezeichnet. Dazu kommen dann sogenannte *Domain Ontologies* für die Quellen, welche die gemeinsamen Basis-Konzepte der Core Ontology entsprechend ihrer Domäne erweitern [DHL03].

Bei der Integration werden die lokalen Schemata der Datenquellen als Konzepte, Rollen und Attribute der Ontologie(n) modelliert und so mit entsprechenden Teilen der Ontologie(n) verknüpft. Auch die Definition von Generalisierungen, Spezialisierungen oder Axiomen ist in diesem Zusammenhang möglich [LN07].

Die formale Beschreibung von Ontologien kann unter anderem durch Sprachen, wie RDF-

Schema[3] oder OWL[4] bzw. deren Vorgänger DAML+OIL[5] erfolgen. Diese auf XML basierenden Sprachen erlauben die Modellierung einer Ontologie in Form von Beschreibungslogik, welche den entscheidbaren Teil der Prädikatenlogik umfasst, und somit automatisches Schlussfolgern (Reasoning) erlaubt [DvL05].

3 Werkzeug-Integration

Mit der Integration von Werkzeugen soll den Anwendern z.B. ermöglicht werden, triviale, immer wiederkehrende Analysen auf Daten durchzuführen, die Daten zu verknüpfen, statistisch auszuwerten oder zu visualisieren. Aber auch komplexere Anwendungen sind denkbar und wurden in einigen Arbeiten schon umgesetzt [Rif07].

Das Problem in existierenden Entwicklungen besteht darin, dass größtenteils nur auf vorher fest definierte Werkzeuge und dazu passende Daten zugegriffen werden kann. In anderen Fällen werden Datensätze nur an externe Anwendungen übergeben und nach der Bearbeitung wieder importiert. Damit wird natürlich die Flexibilität des Systems eingeschränkt und Erweiterungen sind nicht ohne weiteres möglich [Rif07].

Einen Ansatz, der eine flexible Einbindung von Werkzeugen ermöglichen soll, bietet SWAMI[6]. SWAMI stellt sich die Aufgabe ein integriertes Informationssystem, à la "one-stop-shop", für mikrobiologische Daten und auch Analyse-Werkzeuge zur Verfügung zu stellen. Neue Werkzeuge und Daten sollen ohne oder mit nur geringen Änderungen am Programmcode eingebunden werden können. Hier handelt es sich allerdings nicht um einen reinen Ontologieansatz, da keine wirkliche Ontologie für die Integration verwendet wird, sondern ein Konzept-gesteuertes Framework, das im Moment durch Konfigurationsdateien in XML-Form realisiert wird, aber in Zukunft auch von einer formalen Ontologie gesteuert werden können soll. Die Konfigurationsdateien werden während eines Registrierungsprozesses erstellt und beinhalten alle erforderlichen Informationen bezüglich der Werkzeuge, wie Namen, Funktionen, Input-/Output-Parameter und Ergebnisformat. Mit Hilfe von semantischen Annotationen an Daten und Werkzeugen können automatische Überprüfungen und Konvertierungen von Daten zu Input-Parametern vorgenommen werden [Rif07].

Gerade die Konvertierung der Daten in Input-Parameter ist ein wichtiger Punkt. In [BL04] wird ein Framework zur Ontologie-basierten Datentransformation vorgestellt. Hier definieren semantische Annotationen konzeptionelle Informationen für den Input und Output (Ports) eines Werkzeuges. Auf diese Art werden semantische Vor- und Nachbedingungen des Werkzeuges beschrieben, die unabhängig von der Struktur der Ports sind. Es wird so zwischen *Semantic Types* zur Beschreibung der Semantik und *Structural Types* zur Beschreibung der Struktur von Ports und Daten unterschieden. Bei der Übergabe von Daten an ein Werkzeug wird zunächst überprüft, ob die Semantic Types der Daten mit dem Semantic Type des Ports kompatibel sind. Ist dies der Fall, werden die Structural Types verglichen und die Daten gegebenenfalls mit Hilfe von vorher definierten Abbildungsvorschriften in das richtige Format konvertiert (siehe Abbildung 1). So können Inkompatibilitäten vor der Ausführung erkannt werden.

4 Diskussion

Bezüglich der in Abschnitt 1 geschilderten Anforderungen an moderne Informationssysteme für wissenschaftliche Forschungsprojekte ist festzustellen, dass viele gute Ansätze für die Integration von Daten und Werkzeugen mittels Ontologien existieren, so dass domänenübergreifende, Werkzeug-basierte Datenauswertungen prinzipiell realisierbar erscheinen.

[3] RDF Vocabulary Description Language, http://www.w3.org/TR/rdf-schema/
[4] Web Ontology Language, http://www.w3.org/2004/OWL/
[5] http://www.w3.org/TR/daml+oil-reference
[6] The Next Generation Workbench, http://www.ngbw.org/

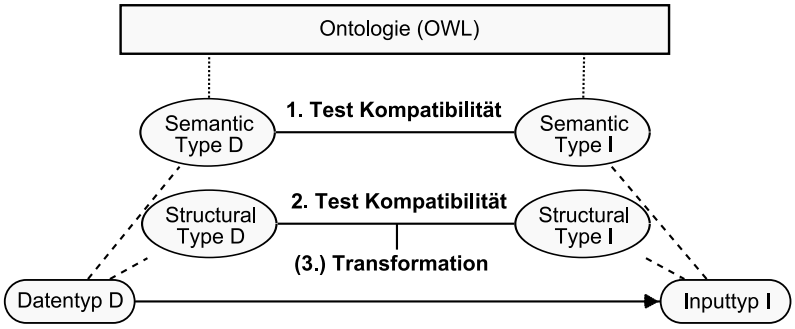

Abbildung 1: Schematische Darstellung der Ontologie-basierten Datentransformation in [BL04]

Der Weg zur vollautomatischen, und vor allem semantischen Integration ist allerdings noch weit. So ist es zwar möglich, Semantik über Ontologien in integrierte Informationssysteme einzubinden und somit beispielsweise Daten aus unterschiedlichen Quellen gemeinsamen Konzepten zuzuweisen und sie vergleichbar zu machen. Eine langfristige Interpretierbarkeit kann dadurch allerdings nicht garantiert werden.

Ein Grund dafür liegt vor allem in möglichen Veränderungen an der den Ontologien zugrunde liegenden Beschreibung von Domänen. Ändert sich diese Beschreibung, so müssen sowohl die korrespondierende Ontologie, als auch alle abhängigen Datenquellen-Beschreibungen oder Anwendungen entsprechend angepasst werden. Dieser *Ontology Evolution*-Prozess ist daher sehr komplex und kann viele Inkonsistenzen hervorrufen [SSH02]. Da es außerdem an ausreichenden Formalismen zur Beschreibung des Ontology Evolution-Prozesses fehlt, fällt eine Automatisierung schwer, obwohl bereits viele Werkzeuge für diesen Zweck entwickelt wurden [FPA06].

Im Bereich der Werkzeug-Integration gibt es bisher nur wenige, auf Ontologien basierende Ansätze. SWAMI und das in [BL04] beschriebene Konzept zur Datentransformation von Input-Parametern deuten allerdings bereits an, welche Vorteile eine Ontologie-basierte Integration bietet. So können Werkzeuge ohne Anpassungen am Programmcode des Informationssystems, nur durch entsprechende Konfigurationsdateien, dynamisch im System registriert und so mit einer Ontologie verknüpft werden. Dadurch ergibt sich die Möglichkeit entsprechende Daten, eine ausreichende semantische Beschreibung vorausgesetzt, direkt oder über Konvertierungsroutinen als Input-Parameter an das Werkzeug zu übergeben.

Aber auch in diesem Bereich sind bei weitem nicht alle Probleme gelöst. Es stellt sich z.B. die Frage, wie während der Datentransformation mit fehlerhaften oder unvollständigen Abbildungsvorschriften umgegangen werden soll oder wie semantische Konvertierungen zwischen korrespondierenden Abbildungsvorschriften automatisiert werden können [BL04].

Ein weiteres Problem, sowohl bei der Daten-, als auch bei der Werkzeug-Integration, stellt neben den erwähnten Schwierigkeiten, insbesondere die aufwendige und komplizierte Entwicklung und Verwendung der Ontologien dar. Viele Systeme stellen zwar Werkzeuge für die Entwicklung zur Verfügung, die Methodik der Entwicklung deckt jedoch oft nur Ontologien für einen bestimmten Zweck ab, der vom Informationssystem vorgegeben wird und nicht allgemeingültig ist. Die auf diese Weise entwickelten Ontologien sind daher in den meisten Fällen in anderen Projekten nicht wiederverwendbar [WVV+01].

Eine Möglichkeit dieses Problem zu umgehen wäre die Verwendung einer standardisierten *Upper Ontology*, wie sie beispielsweise in [NP01] beschrieben wird. Diese stellt allgemeine Definitionen für Terme bereit, aus welchen spezifischere Domänen-Ontologien abgeleitet werden können. Die so entwickelten Domänen-Ontologien sind aufgrund der verwendeten Basis-Konstrukte aus der Upper Ontologie wiederverwendbar.

5 Zusammenfassung und Ausblick

Die technischen Voraussetzungen bzw. Ansätze für eine langfristige Speicherung und Interpretierbarkeit, sowie domänenübergreifende Auswertungen von Forschungsdaten mit Hilfe integrierter Werkzeuge werden durch Ontologien zu einem großen Teil geschaffen.

Als problematisch zu bewerten ist allerdings, dass viele der Schritte für die Integration und der Verknüpfung mit Ontologien nur durch Anwenderinteraktion halbautomatisch oder vollständig manuell erfolgen und daher oft nur durch Domänenexperten umsetzbar sind.

Schließlich behindert auch der Mangel an standardisierten Ontologien, die als Upper Ontologies zur Ableitung von Domänen-Ontologien dienen, die Entwicklung wiederverwendbarer Lösungen.

In Anlehnung an bestehende Probleme existierender Entwicklungen soll sich unsere zukünftige Arbeit auf die Erforschung von verbesserten Techniken zur Auswertung und Analyse von heterogenen Projektdaten konzentrieren. Dazu gehört speziell die Suche nach Methoden zur automatischen Datentransformation, zum einen für domänenübergreifende Auswertungen und zum anderen für die flexible, datenunabhängige Einbindung von Analyse-Werkzeugen. Des weiteren sollen Möglichkeiten gefunden werden, die eine Automatisierung des Integrationsprozesses mittels Ontologien vorantreiben.

Literatur

[BL04] Shawn Bowers and Bertram Ludäscher. *Data Integration in the Life Sciences*, chapter An Ontology-Driven Framework for Data Transformation in Scientific Workflows, pages 1–16. Springer-Verlag, 2004.

[BLL04] Shawn Bowers, Kai Lin, and Bertram Ludäscher. On integrating scientific resources through semantic registration. In *SSDBM '04: Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM'04)*, page 349, Washington, DC, USA, 2004. IEEE Computer Society.

[DHL03] M. Doerr, J. Hunter, and C. Lagoze. Towards a core ontology for information integration. *Journal of Digital Information*, 4(1):169, 2003.

[DvL05] Gerrit Diederichs and Kai von Luck. Bausteine für semantic web anwendungen. Hochschule für Angewandte Wissenschaften Hamburg, 2005.

[FPA06] G. Flouris, D. Plexousakis, and G. Antoniou. Evolving ontology evolution. *SOFSEM 2006: Theory and Practice of Computer Science: 32nd Conference on Current Trends in Theory and Practice of Computer Science, Měřín, Czech Republic, January 21-27, 2006: Proceedings*, 2006.

[Gru93] T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[LN07] Ulf Leser and Felix Naumann. *Informationsintegration - Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen*. dpunkt.verlag, 2007.

[NP01] I. Niles and A. Pease. Towards a standard upper ontology. *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 2–9, 2001.

[Rif07] R. Rifaieh. Swami: Integrating biological databases and analysis tools within user friendly environment. In *Data Integration in the Life Sciences*, volume 4544/2007, pages 48–58. Springer-Verlag Berlin Heidelberg, doi:10.1007/978-3-540-73255-6_7, 2007.

[SSH02] L. Stojanovic, N. Stojanovic, and S. Handschuh. Evolution of the metadata in the ontology-based knowledge management systems. *Proceedings of the 1st German Workshop on on Experience Management: Sharing Experiences about the Sharing of Experience*, pages 65–77, 2002.

[WVV+01] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based integration of information-a survey of existing approaches. *IJCAI-01 Workshop: Ontologies and Information Sharing*, 2001:108–117, 2001.

Management von Ontologien in den Lebenswissenschaften

Michael Hartung

Interdisziplinäres Zentrum für Bioinformatik
Universität Leipzig
hartung@izbi.uni-leipzig.de

Zusammenfassung

Die Bedeutung bzw. der praktische Nutzen von Ontologien zeigt sich insbesondere in den Lebenswissenschaften. Durch die gestiegene Akzeptanz und Anwendung von Ontologien stellt deren Management mehr und mehr ein wichtiges Problem dar. Aus diesen Grund werden die ständige Weiterentwicklung (Evolution) von Ontologien und die starke Heterogenität bezüglich ihrer Formate in diesem Beitrag näher betrachtet. Im speziellen versucht ein erster Ansatz verschiedene Ontologien über eine Middleware in Gridumgebungen zu integrieren, um Gridnutzern bzw. Anwendungen im Grid einen einheitlichen und einfachen Zugang zu Ontologieinformationen zu ermöglichen. Eine weitere Untersuchung beschäftigt sich primär mit der Evolution von Ontologien. Auf Basis eines Frameworks wurde eine quantitative Analyse der Evolution von 16 aktuell verfügbaren, biomedizinischen Ontologien durchgeführt.

1 Einführung

Ontologien erleben insbesondere in den Lebenswissenschaften eine stetig steigende Anwendung und Bedeutung. Sie beinhalten gewöhnlich ein kontrolliertes Vokabular und strukturieren damit spezielle Domänen, bspw. molekulare Funktionen für Proteine oder anatomische Strukturen verschiedener Spezies. Das Vokabular selbst enthält Konzepte, welche Entitäten der Domäne repräsentieren und durch Relationen, insbesondere is-a und part-of, in einem Graph oder Baum miteinander verbunden sind.

Durch die wachsende Popularität kommt dem Management von Ontologien eine immer wichtigere Rolle zu. Einerseits müssen für eine einheitliche Nutzung von Ontologien, z.B. in einem Grid, Heterogenitäten zwischen unterschiedlich entwickelten Ontologien aufgelöst werden. Hierzu zählen insbesondere die verschiedenen Formate und Speicherformen für Ontologien, z.B. relationale Datenbanken, XML, textbasierte Beschreibungen oder Standardisierungen wie OBO [6] in den Lebenswissenschaften. Auf der anderen Seite unterliegen Ontologien einer ständigen Weiterentwicklung (Evolution), da neue Forschungsergebnisse bzw. erweiterte Erkenntnisse in deren Entwicklung einfließen. Um die Evolution von Ontologien besser zu verstehen und deren Auswirkungen auf andere Strukturen (z.B. Annotationen, Ontologiemappings) zu ermitteln, erscheint es hilfreich Analysen über die Evolution von Ontologien durchzuführen. Aus diesen Gründen widmet sich dieser Beitrag zwei Arbeiten im Bereich Ontologiemanagement in den Lebenswissenschaften:

- Vorstellung einer Middleware zur Integration heterogener Ontologien in Gridumgebungen, insbesondere die Integration von Ontologien in MediGRID[1]
- Beschreibung eines Frameworks zur quantitativen Analyse von Ontologieevolution inklusive der vergleichenden Evolutionsanalyse von 16 Ontologien aus den Lebenswissenschaften

Der Rest des Beitrags gliedert sich wie folgt. Das zweite Kapitel beschreibt eine Middleware zur Integration von Ontologien in Grids und deren Anwendung in MediGRID. Kapitel 3 stellt ein Framework zur quantitativen Analyse von Ontologieevolution vor und berichtet über ermittelte Ergebnisse. Ein Ausblick auf weitere Arbeiten bildet den Abschluss des Beitrags.

[1] http://www.medigrid.de

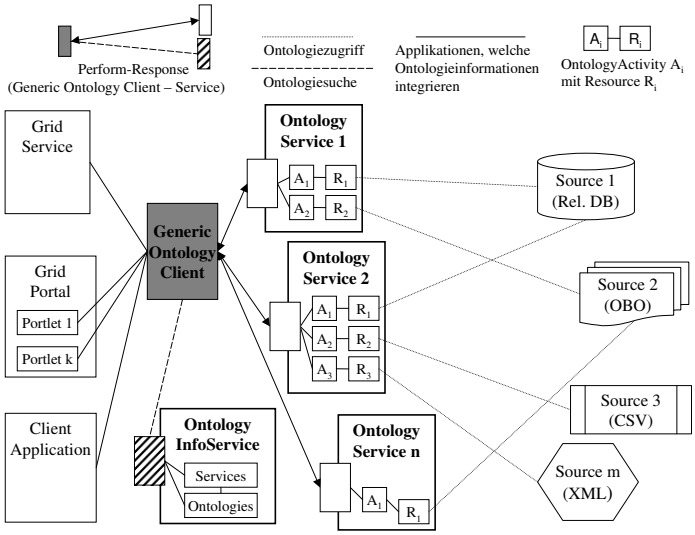

Abbildung 1: Architektur der Middleware

2 Integration von Ontologien in Grids

Um Ontologien verschiedener Formate und Herkunft einheitlich in Gridumgebungen integrieren zu können, wurde eine Middleware zur Integration von Ontologien in Grids entwickelt [3]. Die Middleware, sowie deren Nutzung in MediGRID werden in den beiden folgenden Abschnitten erläutert.

2.1 Architektur

Die in Abb. 1 dargestellte Architektur vermittelt einen Überblick über die Middleware, welche die folgenden Komponenten beinhaltet: (1) *Ontology Sources* als Ontologiequellen, (2) *Ontology Services* und der *OntologyInfoService* sowie (3) ein *Generic Ontology Client* als API für Gridanwendungen. Ontology Sources bezeichnen die ursprünglichen Ontologiequellen, welche Ontologieinformationen über Konzepte bzw. Beziehungen zwischen Konzepten enthalten. In den Lebenswissenschaften existieren die unterschiedlichsten Formate und Speicherformen für Ontologien. Bspw. wird die stark verbreitete GeneOntology (GO) [8] als relationale Datenbank veröffentlicht. Darüber hinaus existieren Standardisierungsbemühungen, wie z.B. das OBO-Format, um Ontologien in einem einheitlichen Format zu entwickeln. Weitere textbasierte Formate reichen von einfachen CSV Dateien bis zu den aus dem Semantic Web bekannten Formaten RDF bzw. OWL.

Im Kern der Middleware befinden sich Ontology Services, welche einen einheitlichen Zugriff auf eine bestimmte Anzahl zugeordneter Ontology Sources ermöglichen. Die Ontology Services basieren auf DataServices des OGSA-DAI Frameworks [4], welches in vielen Gridprojekten als Quasi-Standard für den Zugriff auf strukturierte Daten verwendet wird. Die Zuordnung von Ontology Services zu Ontology Sources ist $m : n$, d.h. ein Ontology Service kann auf mehrere unterschiedliche Ontology Sources zugreifen, umgekehrt kann eine Ontology Source von mehreren Ontology Services integriert werden.

Um diverse Ontologien einheitlich integrieren zu können, werden einerseits in jedem Ontology Service Ressourcenbeschreibungen für die zu integrierenden Ontologien verwaltet. Andererseits wird die im OGSA-DAI Framework verfügbare Activity Komponente verwendet, um spezielle Zugriffsroutinen (OntologyActivities) für die Integration von Ontologieinformationen zu realisieren. Die Kombination aus Ressourcenbeschreibung und OntologyActivity ermöglicht es einheitliche Gridschnittstellen für den Zugriff auf Ontologieinformationen bereitzustellen. Die Gridschnittstellen beinhalten typische Ontologieabfragen, wie z.B. Suche in Ontologien, Ermittlung von Konzeptattributen (Name, Definition, Synonyme, Referenzen) oder Ermittlung von Relationen zwischen Konzepten (Super- bzw. Subkonzepte).

Um die verteilten Ontology Services und ihre zugehörigen Ontology Sources im Grid lokalisieren zu können, wurde zudem ein zentraler OntologyInfoService realisiert. Der Service

dient vorzugsweise dem Discovery erreichbarer Ontologien im Grid und ist somit Ausgangspunkt für die Interaktion mit den Ontology Services. Die clientseitige Interaktion mit den jeweiligen Services übernimmt eine zentrale API, der Generic Ontology Client. Die Client API kann in unterschiedlichen Clients, z.B. Portlets in Portalen, Grid-Services im Grid oder eigenständigen Applikationen, eingesetzt werden. Die bereitgestellten Schnittstellen des Clients ermöglichen ein Interaktion, d.h. sie erlauben den Zugang zu Ontologieinformationen, ohne Kenntnis über technische Hintergründe, wie bspw. Speicherort einer Ontologie im Grid oder deren Format.

2.2 Anwendung in MediGRID

Die zuvor beschriebene Middleware wurde in MediGRID für die Integration diverser Ontologien aus den Lebenswissenschaften verwendet. MediGRID verwendet als zentralen Zugang zu seinem Grid ein Applikationsportal in welchem die unterschiedlichen Anwendungen (Bioinformatik, Bildverarbeitung, Klinische Forschung) einfach und web-basiert über Portlets erreichbar sind. Die Integration der Ontologien in das MediGRID Portal geschieht über zwei verschiedene Wege. Einerseits existiert ein zentraler Ontology LookUp Service, welcher Such- und Browsingfunktionalitäten für die integrierten Ontologien anbietet. Andererseits können MediGRID-Applikationen, wie bspw. AUGUSTUS [7] oder SNP Selection [1] ihre Daten unter Verwendung des Generic Ontology Clients mit Ontologieinformationen verknüpfen.

Aktuell wurden insgesamt 14 verschiedene Ontologien aus den Lebenswissenschaften mit Hilfe der Middleware in das MediGRID integriert. Hierzu zählen einerseits die stark verbreitete GeneOntology und der am National Cancer Institute (NCI) verwendete NCIThesaurus [5]. Weiterhin wurde mit RadLex ein Lexikon mit Begriffen aus dem radiologischen Bereich verfügbar gemacht. Weitere eher domänenspezifische Ontologien der OBO Initiative wurden ebenfalls integriert und werden in Anwendungen verwendet (z.B. SequenceOntology).

3 Quantitative Analyse der Evolution von Ontologien

Neben der Integration von Ontologien in bestimmte Umgebungen, hier am Beispiel Grid erläutert, spielt deren Evolution, d.h. deren Weiterentwicklung basierend auf neuen Erkenntnissen, im Ontologiemanagement eine wichtige Rolle. Die folgenden Abschnitte beschreiben Teile eines Frameworks zur quantitativen Evolutionsanalyse von Ontologien und angrenzenden Strukturen (z.B. Annotations- oder Ontologiemappings) [2]. Des weiteren werden Ergebnisse der Evolutionsanalyse für 16 Ontologien aus den Lebenswissenschaften präsentiert.

3.1 Ontologie- und Evolutionsmodell

Eine Ontologie in einer bestimmten Version v zu einem Zeitpunkt t wird durch folgendes Tupel charakterisiert: $ON_v = (C, R, t)$. Einerseits besteht eine Ontologie aus einer Menge von Konzepten C, welche über binäre Relationen R miteinander verbunden sind. C und R bilden gemeinsam mit den *roots* (Konzepte die keine Vorgänger besitzen) einen azyklisch gerichteten Graph (DAG). Konzepte selbst beinhalten diverse Attribute, in diesem Beitrag werden insbesondere zwei Attribute, das *accession* Attribut sowie die *obsolete* Information über den Status eines Konzepts verwendet. Das *accession* Attribut dient der eindeutigen Identifikation eines Ontologiekonzepts innerhalb einer Ontologie, während andererseits der Status eines Konzepts darüber informiert, ob ein Konzept aktiv ist (noObsolete) oder ob es veraltet ist und nicht mehr verwendet werden soll (obsolete).

Das Evolutionsmodell des Frameworks unterscheidet zwischen 3 grundlegenden Evolutionsoperatoren: *add*, *delete* und *toObsolete*. Während add die Einfügung neuer Objekte erfasst, werden durch delete Löschungen von Objekten beschrieben. Der insbesondere bei Ontologien angewendete toObsolete Operator erfasst Objekte, welche ihren Status von noObsolete auf obsolete ändern. Für die quantitative Evolutionsanalyse werden analog zu den Operatoren Evolutionsmengen berechnet. Als Eingabe dienen Objektmengen einer Version v_i (O_{vi}) und der geänderten Version v_j (O_{vj}) einer Quelle. Um veraltete von aktiven Objekten unterscheiden zu können, werden Submengen $O_{vi,obs}$ für veraltete und $O_{vi,nonObs}$ für aktive Objekte gebildet. Im Fall der Ontologieevolution können Objekte Konzepte C oder Relationen

R darstellen. Die Evolutionsmengen werden dann wie folgt berechnet: $add_{vi,vj} = O_{vj}/O_{vi}$, $del_{vi,vj} = O_{vi}/O_{vj}$ und $toObs_{vi,vj} = O_{vj,obs} \bigcap O_{vi,nonObs}$. Die Berechnung der Mengen erfolgt auf Basis eines Vergleichs von IDs (accessions) der beteiligten Objekte. Existiert bspw. ein Objekt in Version v_j und nicht in v_i, so wird dieses Objekt $add_{vi,vj}$ zugeordnet, umgekehrt würde das Objekt in $del_{vi,vj}$ eingefügt werden.

3.2 Metriken

Um die Evolution quantitativ erfassen zu können, werden Metriken definiert. Aus Platzgründen werden in diesem Beitrag nur einige relevante Metriken beschrieben.

Auf Basis der Objektmengen werden die folgenden Basismetriken definiert:

- $|O_{vi}|$: Anzahl von Objekten in Version v_i; $O \in$ {Konzepte C, Relationen R}
- $|C_{obs}|$, $|C_{nonObs}|$: Anzahl veralteter (obsolete) bzw. aktiver Konzepte

Aufbauend auf den Evolutionsmengen werden folgende Metriken zur Einschätzung der Evolution definiert:

- $Add_{vi,vj} = |add_{vi,vj}|$: Anzahl eingefügter Objekte zwischen v_i und v_j
- $Del_{vi,vj} = |del_{vi,vj}|$: Anzahl gelöschter Objekte zwischen v_i und v_j
- $Obs_{vi,vj} = |toObs_{vi,vj}|$: Anzahl zu obsolete markierter Objekte zwischen v_i und v_j
- $Add_{p,t}$, $Del_{p,t}$, $Obs_{p,t}$: durchschnittliche Anzahl veränderter Objekte pro Intervall t in einem Zeitraum p

Einerseits werden Versionsübergänge direkt quantifiziert, andererseits ermöglichen erweiterte Metriken die Analyse der Evolution innerhalb eines Zeitraums p für ein bestimmtes Änderungsintervall t. Bspw. können monatliche oder jährliche Änderungsraten (t) für einen gewissen Zeitraum (p) berechnet werden. Um nicht nur absolute Änderungen zu erfassen, werden auf deren Basis relative Änderungen wie auch eine add-delete ratio (adr) definiert: $adr_{vi,vj} = Add_{vi,vj}/(Del_{vi,vj} + Obs_{vi,vj})$, $add - frac_{vi,vj} = Add_{vi,vj}/|O_{vj}|$, $del - frac_{vj,vi} = Del_{vi,vj}/|O_{vi}|$, $obs - frac_{vj,vi} = Obs_{vi,vj}/|O_{vi}|$. Die relativen Änderungen sind ebenfalls auf beliebige Zeitintervalle und Perioden anwendbar: $add{-}frac_{p,t}$, $del{-}frac_{p,t}$ und $obs{-}frac_{p,t}$.

3.3 Ergebnisse der Evolutionsanalyse

Für die Analyse wurden 16 Ontologien mit insgesamt 386 Versionen in ein zentrales Repository integriert. Die verschiedenen Versionen stammen aus dem Zeitintervall [Mai 2004,Feb 2008], also einem Beobachtungszeitraum von 45 Monaten. Im Durchschnitt stieg die Anzahl von Ontologiekonzepten $|C|$ um 60%, wobei GeneOntology und NCIThesaurus ein Wachstum von 50% bzw. 78% aufweisen. Mithilfe der beschriebenen Metriken wurden die durchschnittlichen Änderungsraten pro Monat (absolut wie relativ) für den ganzen Beobachtungszeitraum

	Anzahl von Konzepten |C|		Mai 04 - Feb 08							Feb 07 - Feb 08		
Ontologie	*Start*	*Feb 08*	*Add*	*Del*	*Obs*	*adr*	*add-frac*	*del-frac*	*obs-frac*	*Add*	*Del*	*Obs*
NCI Thesaurus	35.814	63.924	627	2	12	42,4	1,3%	0,0%	0,0%	416	0	5
GeneOntology	17.368	25.995	200	12	4	12,2	0,9%	0,1%	0,0%	222	20	5
-- Biological Process	8.625	15.001	146	7	2	16,2	1,2%	0,1%	0,0%	133	10	2
-- Molecular Function	7.336	8.818	36	3	2	6,8	0,4%	0,0%	0,0%	69	7	3
-- Cellular Components	1.407	2.176	18	2	0	8,9	1,0%	0,1%	0,0%	19	3	0
ChemicalEntities	10.236	18.007	256	62	0	4,1	1,8%	0,5%	0,0%	384	67	0
FlyAnatomy	6.090	6.222	5	1	1	3,3	0,1%	0,0%	0,0%	6	0	0
MammalianPhenotype	4.175	6.077	65	2	9	6,0	1,2%	0,0%	0,2%	74	2	3
AdultMouseAnatomy	2.416	2.745	11	0	0	30,9	0,4%	0,0%	0,0%	1	0	0
ZebrafishAnatomy	1.389	2.172	33	5	1	5,5	1,8%	0,3%	0,1%	45	2	1
Sequence	981	1.463	19	3	2	4,1	1,5%	0,3%	0,2%	19	0	0
ProteinModification	1.074	1.128	5	2	1	1,5	0,4%	0,2%	0,1%	7	0	2
CellType	687	857	5	1	0	2,8	0,7%	0,2%	0,1%	1	0	0
PlantStructure	681	835	5	0	1	6,1	0,7%	0,0%	0,1%	3	0	0
ProteinProteinInteraction	194	819	21	0	0	41,7	2,7%	0,0%	0,2%	4	0	0
FlyBaseCV	658	693	1	0	1	2,1	0,2%	0,0%	0,1%	0	0	0
Pathway	427	593	7	1	0	7,9	1,3%	0,2%	0,0%	6	2	0

Abbildung 2: Änderungen in analysierten Ontologien (sortiert nach $|C|$ absteigend)

als auch für das letzte Jahr ermittelt (Abb. 2). Die Analyse ergab, dass die großen Ontologien ($|C| > 10000$, dunkelgrau hinterlegt) ebenfalls die größten Änderungsraten erfahren: 360 (25) Einfügungen (Löschungen) pro Monat im Vergleich zu 86 (6) Einfügungen (Löschungen) in allen untersuchten Ontologien. Weiterhin kann man erkennen, dass Einfügungen in allen Ontologien dominieren, jedoch existieren einige Ontologien, z.B. ChemicalEntities, die ebenfalls hohe Löschraten bzw. obsolete Markierungen aufweisen. Die *adr* Metrik zeigt, dass bspw. NCIThesaurus im Schnitt 42 mal mehr hinzufügt als löscht bzw. obsolete markiert. Hingegen weist ChemicalEntities hier einen Wert von 4 auf, was im Durchschnitt einen Anteil von 20% Löschungen an den Gesamtänderungen bedeutet. Die relativen Änderungsraten deuten an, dass einige mittelgroße bzw. kleine Ontologien in Bezug auf ihre Größe durchaus hohe Evolutionsraten besitzen, z.B. ZebrafishAnatomy und ProteinProteinInteraction.

Interessant erscheint die Verwendung von obsolete zur Markierung veralteter Konzepte in den analysierten Ontologien. Während wenige Ontologien, z.B. ChemicalEntities und AdultMouseAnatomy, auf obsolete Markierungen komplett verzichten, nutzen 13 der 16 Ontologien einen hybriden Ansatz (gleichzeitige Löschungen und obsolete Markierungen). Eine weitere Analyse vergleicht die Evolution im gesamten Beobachtungszeitraum mit der im letzten Jahr. Hier zeigen sich 3 verschiedene Szenarien. Einerseits existieren Ontologien, welche in beiden Zeiträumen hohe Änderungsraten aufweisen: NCIThesaurus oder GeneOntology. Andere Ontologien wiederum wurden im letzten Jahr verstärkt weiterentwickelt, d.h. die Domänen bzw. Forschungsfelder dieser Ontologien erscheinen derzeit hoch aktuell, bspw. ChemicalEntities oder GO MolecularFunction. Die letzte Gruppe umfasst Ontologien, welche im letzten Jahr gegenüber dem Gesamtzeitraum keine bzw. wenige Änderungen aufzeigen. Hierzu zählen unter anderem AdultMouseAnatomy oder CellTypeOntology.

4 Zusammenfassung und Ausblick

Dieser Beitrag berichtete anhand zweier ausgewählter Beispiele über das Ontologiemanagement in den Lebenswissenschaften: (1) Integration heterogener Ontologien in Grids und (2) Quantitative Evolutionsanalyse von Ontologien. Möglichkeiten für künftige Arbeiten umfassen bspw. den Ausbau des Frameworks zur Evolutionsanalyse um weitere Evolutionstypen (z.B. Änderung an Attributen). Weiterhin ist die Verwaltung der unterschiedlichen Ontologieversionen im zentralen Repository effizienter gestaltbar. Zudem könnten Analyseergebnisse der Öffentlichkeit (z.B. Ontologiedesignern) zugänglich gemacht werden.

Acknowledgements Diese Arbeit wurde vom BMBF im Rahmen des Projektes “MediGRID - Ressourcenfusion für Medizin und Lebenswissenschaften” (01AK803E) gefördert.

Literatur

[1] J. Hampe, S. Schreiber, and M. Krawczak. Entropy-based SNP selection for genetic association studies. *Human Genetics*, 144:36–43, 2003.

[2] M. Hartung, T. Kirsten, and E. Rahm. Analyzing the Evolution of Life Science Ontologies and Mappings. *To appear in Proc. of DILS 2008*, 2008.

[3] M. Hartung and E. Rahm. A Grid Middleware for Ontology Access. *1st German eScience Conference*, 2007.

[4] K. Karasavvas et al. Introduction to OGSA-DAI Services. In *LNCS*, volume 3458, pages 1–12, 2005.

[5] N. Sioutos et al. NCI Thesaurus: A semantic model integrating cancer-related clinical and molecular information. *Journal of Web Semantics*, 40:30–43, 2007.

[6] B. Smith et al. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25:1251–1255, 2007.

[7] M. Stanke et al. AUGUSTUS: ab initio prediction of alternative transcripts. *Nucleic Acids Res*, 34(Web Server issue):435–439, 2006.

[8] The Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Res*, 32:258–261, 2004.

Verbesserung der Nutzbarkeit von Webinhalten für sehgeschädigte Computernutzer durch Strukturanalyse und Aufgabenmodelle

Mathias Köhnke[1], Martina Weicht[2], Temenushka Ignatova[1] und Ilvio Bruder[2]

[1]Universität Rostock, [2]IT Science Center Rügen
mathias.koehnke@gmail.com,temi@informatik.uni-rostock.de
{weicht,bruder}@it-science-center.de

Zusammenfassung

Auch wenn sogenannte Screenreader sehgeschädigten Computernutzern heute schon eine gute Zugänglichkeit zu Anwendungsprogrammen und Inhalten unter den verschiedenen Betriebssystemen bieten, so stellen gerade zu Zeiten des Web2.0 umfangreiche Webseiten eine große Hürde dar. Der Aufwand für das erstmalige Erkunden einer Webseite und das Erfassen der bentigten Informationen ist sehr gro und führt besonders bei schlecht ausgezeichneten Seiten zu Frustration. Die in diesem Beitrag beschriebene semantische Strukturanalyse soll relevante Bereiche einer Webseite in Bezug auf ein bestimmtes Nutzungsszenario auffinden und für den (sehgeschädigten) Nutzer leichter zugänglich machen. Dazu wurde ein Mapping zwischen den Elementen einer Web-Seite und einer Menge semantischer Konstrukte (dargestellt in RDF) entwickelt. Die Strukturanalyse wird mittels so genannter Aufgabenmodelle gesteuert und ist mit Hilfe des UIMA-Frameworks von IBM umgesetzt.

1 Motivation

Eine erhebliche Einschränkung des klassischen *Information Retrieval* als Basis für heutige Websuchmaschinen besteht darin, dass semantische Kontextdaten nicht in den Retrieval–Prozess einbezogen werden. Ungenaue Suchergebnisse, überladene Webseiten und irreführende User Interfaces führen dazu, dass besonders sehgeschädigte Nutzer diese Webangebote meiden. Um die Zugänglichkeit und Nutzbarkeit solcher Webseiten zu erhöhen, sollten Technologien eingesetzt werden, die die semantische Struktur von Webseiten in die Analyse einbeziehen. So werden relevante Bereiche und Elemente dann besser aufgefunden, wenn sie in Zusammenhang mit einer mit Hilfe der Webpräsentation zu erledigenden Aufgabe stehen. Dabei bilden verschiedene Kombinationen dieser Bereiche und Elemente, also unterschiedliche syntaktische Konstrukte, semantische Einheiten, die ihrerseits konkreten, vordefinierten (Teil–) Aufgaben innerhalb eines Aufgabenmodells zugeordnet sind. Ein Aufgabenmodell entspricht hier einer Art Workflow, der den Nutzer bei einer komplexen Aufgabe, etwa dem Online–Einkauf, unterstützt und begleitet.

Die beschriebenen Technologien werden in einen am IT Science Center in Putbus auf Rügen entwickelten Screenreader eingebunden, der grafische Bildschirminhalte für sehgeschädigte Computernutzer aufbereitet und alternativ über Sprache und/oder eine Braillezeile ausgibt. Durch die Ausstattung mit verschiedenen Erweiterungen soll die Nutzbarkeit des Screenreaders erhöht und die Arbeit mit demselben spürbar verbessert werden.

2 Semantische Konstrukte in Webdokumenten

Techniken mit deren Hilfe sinnvolle Informationen in Webdokumenten aufgespürt werden, entstammen hauptsächlich den Gebieten des Webmining [7] und des User Interface Design. Viele Ansätze, die sich mit dem Erkennen semantischer Strukturen in Dokumenten befassen, teilen die (Web–)Quelldokumente in Strukturblöcke und analysieren beispielsweise das Dokumentenbild [2], setzen *Wrapper learning*–Methoden ein [6] oder werten die DOM–Repräsentation des HTML–Quellcodes aus [1, 3]. Dabei beziehen sie jedoch keine spezifischen Strukturen ein, sondern unterteilen die Webdokumente in Bereiche und klassifizieren sie entsprechend bestimmter Kriterien.

Das Basis-Konzept zur Beschreibung semantischer Strukturen in Webseiten bilden bei unserem Ansatz sogenannte semantische Konstrukte, zum Beispiel Navigationsmenüs, Login-Felder oder Suchdialoge. Diese semantischen Konstrukte repräsentieren Bausteine eines Webdokumentes, die jeweils einer bestimmten Aktivität innerhalb des Nutzungsszenarios der Webseite entsprechen. Letzteres wird mit Hilfe eines Aufgabenmodells modelliert und leitet den Nutzer in der richtigen Reihenfolge durch die einzelnen Teilaufgaben. Das Konzept der Aufgabenmodelle entstammt ursprünglich der modellbasierten Software–Entwicklung und wird hauptsächlich zum Design von User Interfaces eingesetzt [4]. Die Aufgabenmodelle bilden die Ausgangsbasis in unserem Ansatz, in dem ihre Teilaufgaben jeweils verschiedenen semantischen Konstrukten in Webdokumenten in einer Tabelle zugeordnet werden. Wird nun durch eine Webseite navigiert, die sich einer speziellen Aufgabe widmet, so können konkrete semantische Konstrukte aufgefunden und ihre Position innerhalb des Dokumentes dem Benutzer angezeigt werden. Das dafür benötigte Mapping illustriert die **Abbildung 1**. Dadurch entfällt das umständliche Navigieren zum Beispiel mit der Tab-Taste, das den Nutzer unter Umständen nie zu dem gewünschten semantischen Konstrukt führt.

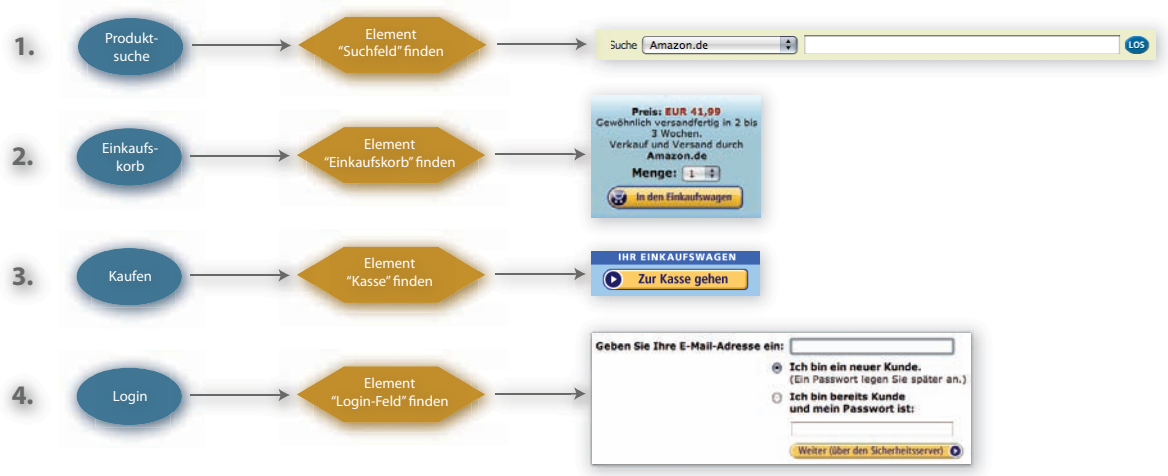

Abbildung 1: Mapping einer "Online Shopping"–Aufgabe auf Strukturen der Amazon–Webseite

Das semantische Konstrukt besteht seinerseits aus einer Kombination syntaktischer Elemente, die sich aus der Analyse verschiedener Webseiten, die ein solches semantisches Konstrukt enthalten, ergibt. Dabei kann es für ein semantisches Konstrukt verschiedene Kombinationen syntaktischer Elemente geben. Ein Suchdialog kann sich beispielsweise aus einem Eingabefeld, einem Label und einem Button zusammensetzen, die anhand ihrer Charakteristiken oder *Features* beschrieben werden. Alle semantischen Konstrukte bzw. ihre Features werden mithilfe einer Menge von RDF[1]–Aussagen beschrieben, wobei eine solche Aussage immer aus einem Subjekt

[1]Resource Description Framework

(Ressource), einem Prädikat (Eigenschaft) und einem Objekt (Statement) besteht. Diese RDF–Tripel beschreiben in unserem Zusammenhang jeweils das Vorkommen eines Features als Teil eines syntaktischen Konstruktes. Das RDF–Schema, das diese Beschreibungen enthält, umfasst Klassen für alle syntaktischen Elemente, die innerhalb von HTML–Dokumenten vorkommen können, und wird als Wörterbuch für die semantische Strukturanalyse verwendet. Jede Klasse beschreibt die Eigenschaften, die ein bestimmtes syntaktisches Element besitzen sollte oder könnte.

Aus dem RDF–Schema wird ein RDF–Dokument, also eine Instanz für jedes semantische Konstrukt generiert. Diese Aufgabe sollte ein erfahrener Webentwickler übernehmen, der eine ausreichende Anzahl von Webdokumenten analysiert und daraus generische Beschreibungen der syntaktischen Elemente von semantischen Konstrukten ableitet. Das RDF–Dokument enthält damit je eine Instanz aller für das semantische Konstrukt relevanten syntaktischen Elemente und ihrer entsprechenden Eigenschaften. Dabei erlaubt die RDF–Syntax auch, Hierachien oder alternative syntaktische Elemente festzulegen. Das semantische Konstrukt "Suchfeld" beispielsweise ist sehr häufig auf Webseiten anzutreffen, seine syntaktischen Elemente können dabei jedoch unterschiedlich kombiniert sein. Oft ist das Konstrukt in ein <form>–Element eingebettet (#**SearchForm**), weshalb seine RDF–Beschreibung hierarchisch strukturiert ist. Wenigstens zwei der enthaltenen Elemente, ein Suchtextfeld (#**SearchInputText**) und ein Suchbutton, der ebenfalls unterschiedlich realisiert sein kann (#**SearchInputSubmit, #SearchInputImage** etc.), müssen in dem Formular vorhanden sein, um dieses als Suchfeld zu identifizieren. Diese Bedingung wird durch den Mengentyp **rdf:Bag** im RDF–Dokument ausgedrückt, während die verschiedenen Button–Alternativen durch den **rdf:Alt**–Typ beschrieben werden (Siehe **Abbildung 2**).

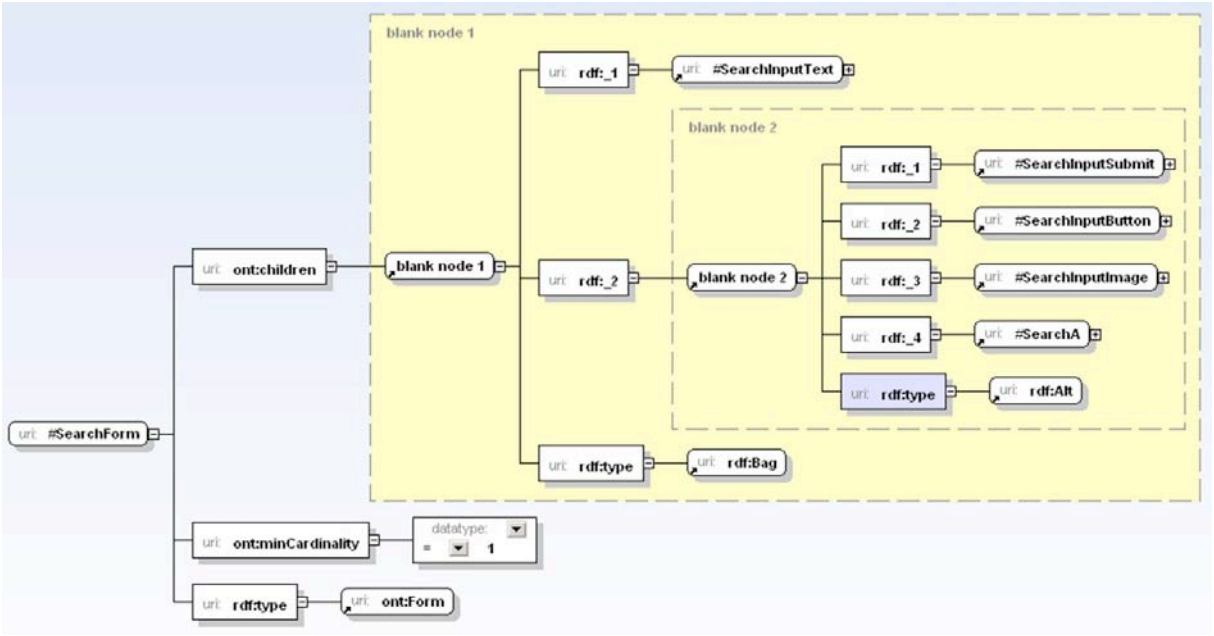

Abbildung 2: RDF–Schema für das semantische Konstrukt "Suchfeld"

3 Identifikation und Interpretation semantischer Konstrukte

Zu jeder Aufgabe im Aufgabenmodell soll nun ein entsprechendes semantisches Konstrukt innerhalb der Webseite gefunden werden. In einem ersten Schritt werden dazu die RDF–Aussagen, die dieses Konstrukt beschreiben, mit den syntaktischen Elementen im DOM–Baum des Webbdokumentes in vordefinierter Reihenfolge verglichen und entsprechend mit wahr oder falsch bewertet.

Jedes RDF–Statement kann dabei auf verschiedenen Abschnitten des DOM–Baumes ausgewertet werden, die als Kandidaten bezeichnet werden. Der Wahrheitswert für einen Kandidaten errechnet sich aus der Summe aller Wahrheitswerte der ausgewerteten Aussagen geteilt durch die Anzahl der Attribute. Der Kandidat mit der höchsten Wahrscheinlichkeit wird zur weiteren Auswertung des semantischen Konstruktes herangezogen.

Auf Basis der gefundenen syntaktischen Elemente wird nun entschieden, ob das jeweilige semantische Konstrukt in der Webseite enthalten ist. Dazu kann ein empirisch bestimmter Schwellwert auf alle Wahrscheinlichkeiten angewendet werden. Problematisch hierbei ist jedoch, dass verschiedene RDF–Aussagen für diese Entscheidung von unterschiedlicher Relevanz sind. Daher setzen wir Klassifikationsmethoden ein, die die Gewichte der einzelnen RDF–Statements in einer Trainingsphase berechnen können. In diesem Paper werden Neuronale Netze als Klassifikationsmethode eingesetzt, wobei die einzelnen Wahrscheinlichkeitswerte der syntaktischen Elemente als Eingabewerte für das Netz verwendet werden. Letzteres berechnet dann die Wahrscheinlichkeit mit der das fragliche semantische Konstrukt auf der Webseite gefunden wurde. Trifft dies zu, so werden seine Positionsdaten an die Screenreader–Anwendung übermittelt.
In der Trainingsphase wird für jedes semantische Konstrukt ein Neuronales Netz mit Hilfe sogenannter Positiv– und Negativlisten, also Listen mit URLs von Webseiten, auf denen das semantische Konstrukt entweder enthalten oder eben nicht enthalten ist, erstellt. Für jede dieser URLs werden die entsprechenden RDF–Dokumente für das semantische Konstrukt ausgewertet und die Ergebnisse als Trainingsdaten für das Neuronale Netz verwendet. Der Vorteil dieses Ansatzes besteht darin, dass Neuronale Netze besonders gut mit “unscharfen” Werten umgehen können. Nachteilig auf die Evaluierung kann sich jedoch die Tatsache auswirken, dass Neuronale Netze auch “auswendig” lernen können. Alternativ könnten die Daten deshalb in Entscheidungsbäumen ausgewertet werden. Es sind jedoch große Mengen von Trainingsdaten erforderlich um einen Entscheidungsbaum aufzubauen, der zudem jedes Mal, wenn sich die Trainingsdaten ändern, neu erstellt werden muss.

4 Implementierung

Um einen hohen Grad an Flexibilität bei der semantischen Strukturanalyse zu gewährleisten wurde das entwickelte Konzept mit Hilfe des UIMA[2], einem plattformunabhängigen Java–Frameworks zum einfachen Erstellen von Anwendungen zur Extraktion relevanten Wissens aus unstrukturierten Informationen [8] implementiert.

Ein Algorithmus, der die gewünschte Aufgabe erfüllt, sowie seine Übergabeparameter und Rückgabewerte werden mit Hilfe einer Komponentenbeschreibung (Component Descriptor) in UIMA integriert, die aus diesen Vorgaben eine UIM–Applikation erstellt. Die Analyse–Ergebnisse werden in strukturierter Form wieder ausgegeben und können in einer Datenbank erfolgen (z.B. als Vorverarbeitung eines Data-Mining-Prozesses) oder als Indizes für (semantische) Suchmaschinen gespeichert werden. In welcher Form die strukturierte Ausgabe der Daten erfolgt, liegt dabei also in der Hand des Entwicklers. In das UIMA–Framework können auf sehr einfache Art und Weise verschiedene Komponenten zu einem Analyseprozess hinzugefügt oder aus ihm entfernt werden. Die Analysekomponenten für die hier vorgestellte Anwendung sind alle von drei UIMA–Hauptkomponenten, *Collection Reader*, *Analysis Engine* und *CAS–Consumer*, abgeleitet worden, auf deren Basis weitere Komponenten implementiert werden können, wenn neue Konstrukte in die Analyse von Webdokumenten einbezogen werden sollen [5].

Die semantische Strukturanalyse wurde in Java implementiert und umfasst verschiedene Verarbeitungsschritte. Am Beispiel unseres Online–Shopping–Modells werden alle Aufgaben analysiert und eine Abarbeitungsreihenfolge bestimmt. Das Aufgabenmodell wird dabei in einem XML–Dokument beschrieben. Einen Weg, um ein Aufgabenmodell zu beschreiben, bietet CTTE,

[2]Unstructured Information Management Architecture

die Concurrent Task Tree Environment [4], mit der eine grafische Repräsentation eines Aufgabenmodells erstellt und anschließend in XML exportiert wird. Aus einer internen Zuordnungstabelle geht hervor, welche RDF–Dokumente die Beschreibungen der den Aufgaben zugeordneten semantischen Konstrukte enthalten. Diese Dokumente werden anschließend mit dem RdfFile-Parser eingelesen. Anschließend werden diese mit Hilfe von Jena[3], einem Java–Framework für Semantic–Web–Anwendungen, geparst und verifiziert. Danach wird das Quelldokument eingelesen, validiert und in das Document Object Model überführt, auf der die Analyse durchgeführt wird. Alle RDF–Statements werden in der vorbestimmten Reihenfolge abgearbeitet und liefern jeweils einen Wahrscheinlichkeitswert zurück, aus deren Menge eine Gesamtwahrscheinlichkeit berechnet wird. Diese muss einen bestimmten Schwellwert überschreiten, damit ein semantisches Konstrukt dem Quelldokument zugeordnet werden kann. In unserem Fall werden die Positionsdaten desselben angezeigt und dem Screenreader übergeben.

5 Fazit

Client–seitige Analysetechniken zur Verbesserung der Zugänglichkeit und besseren Nutzbarkeit von Webseiten können Sehgeschädigten helfen, sich besser im Web zurechtzufinden. In diesem Paper stellen wir einen Ansatz vor, mit dem die Semantik von Interaktionselementen in Webseiten automatisch erkannt werden kann. Dabei können semantische Domänen mit Hilfe von Aufgabenmodellen eingeschränkt, syntaktische auf semantische Konstrukte abgebildet und die Zuverlässigkeit der erkannten semantischen Konstrukte gewichtet werden.
Unsere Lösung bietet somit eine alternative Webseiten–Navigation für vordefinierte Aufgaben. Für ihre nachhaltige Nutzung muss jedoch eine Community aufgebaut werden, die neue Aufgabenmodelle und entsprechende RDF–Beschreibungen hinzufügt bzw. bestehende Modelle und Beschreibungen aktualisiert. Für eine gründlichere Analyse können zusätzlich grafische Elemente in die Analyse einbezogen werden, aus denen durch *Optical Character Recognition*–Anwendungen textuelle Informationen extrahiert werden. Auch der Einsatz linguistischer Verfahren kann die Ergebnisse noch weiter verbessern.

Literatur

[1] Feng, J., Haffner, P., Gilbert, M.: A Learning Approach to Discovering Web Page Semantic Structures, In: Proc. of the 2005 8th Conference on Document Analysis and Recognition, Seoul, (2005).

[2] Berardi, M., Lapi M., Malerba, D.: An Integrated Approach for Automatic Semantic Structure Extraction in Document Images. Document Analysis Systems, Florence, (2004).

[3] Mukherjee, S., Yang, G., Tan, W., Ramakrishnan, I. V.: Automatic Discovery of Semantic Structures in HTML Documents In: Proc. of the 7th Int. Conf. on Document Analysis and Recognition, Edinburgh, (2003).

[4] Mori, G., Paterno, F., Santoro, C.: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. In IEEE Transactions on Software Engineering, Vol. 28, No. 8, (2002).

[5] Köhnke, M.: Erkennung von Struktur und semantischen Beziehungen in Web-Dokumenten zur Unterstützung der Mensch-Maschine-Interaktion für sehgeschädigte Nutzer. Universität Rostock. Diplomarbeit, (2008).

[6] Cohen, W. W.: Learning and Discovering Structure in Web Pages IEEE Data Engineering Bulletin, Volume 26, (2003).

[7] Etzioni, O.: The world wide web: Quagmire or gold mine. Communications of the ACM, 39(11), (1996).

[8] Ferrucci et al.: IBM Research Report. Towards an Interoperability Standard for Text and Multi-Model Analytics. Technical Report. IBM, (2006).

[3] `http://jena.sourceforge.net`

Kombiniertes Mining von strukturellen und relationalen Daten

Frank Eichinger Klemens Böhm

Institut für Programmstrukturen und Datenorganisation (IPD),
Universität Karlsruhe (TH), Deutschland, {eichinger, boehm}@ipd.uka.de

Zusammenfassung

Data Mining Techniken wie Klassifikation, Regression und Clusteranalyse finden heutzutage eine weite Verbreitung. Entsprechende relationale Daten liegen in vielen Anwendungsdomänen vor, und effiziente Data Mining Algorithmen sind in kommerzielle Werkzeuge sowie in Datenbank Management Systeme integriert. In den letzten Jahren wurden aber auch verschiedene strukturelle Data Mining Techniken entwickelt, die z. B. mit Graph-basierten Daten arbeiten. Solche Techniken erschließen neue Anwendungsgebiete, bieten aber auch das Potential, bisherige Techniken zu ergänzen. Oft können durch Kombination bisherige Ergebnisse verbessert werden. In diesem Beitrag präsentieren wir Arbeiten aus dem Bereich der Vorhersage von Kundenverhalten und der Fehlersuche in Software, in denen strukturelle und relationale Data Mining Techniken erfolgreich kombiniert wurden. Schließlich geben wir einen Ausblick auf weitere Anwendungsgebiete und zukünftige Herausforderungen.

1 Einleitung

Viele Data Mining Techniken wie Klassifikation und Regression zur Vorhersage von Zielgrößen sind inzwischen weit verbreitet. Entsprechende effiziente Algorithmen sind in kommerziellen Data Mining Anwendungen wie auch in Datenbank Management Systemen (DBMS) integriert. Auch Explorationstechniken wie die Cluster- und Assoziationsregelanalyse, die in der Lage sind, Zusammenhänge in Datensammlungen aufzudecken, finden weite Verbreitung. Solche Data Mining Techniken arbeiten üblicherweise mit Daten, die relational vorliegen. Jede Instanz (z. B. jeder Kunde im Fall von Kundendaten) wird mit einem Tupel von numerischen und kategorischen Werten beschrieben. Im Fall von Kundendaten sind dies z. B. „Alter“, „Umsatz im letzten Jahr“ und „Dauer des Kundenverhältnisses“ als numerische Bestandteile und „Kundengruppe“ und „Wohngegend“ als kategorische Größen. Entsprechende Daten liegen neben Kunden-Datenbanken in den unterschiedlichsten Anwendungsdomänen vor. Dies hat zu einer weiten Verwendung von Data Mining Techniken geführt. In bestimmten Domänen hingegen können Techniken, die auf relationalen Daten operieren, nicht angewendet werden. Dies ist beispielsweise bei der Analyse von Molekülstrukturen in der Pharmaforschung der Fall. In einigen Domänen, in denen Data Mining inzwischen etabliert ist (z. B. bei der weit verbreiteten Analyse von relationalen Kundendaten), wünschen sich Anwender jedoch oft deutlich bessere Genauigkeiten von Vorhersagen, als sie allein mit den bisherigen Techniken möglich sind. Neben der Weiterentwicklung von bisherigen Techniken zur Analyse und Datenvorverarbeitung, sowie der Integration solcher Techniken in verbreitete Produkte wie DBMS, gibt es einen Bedarf an fortgeschrittenen Technologien, die über das bisher übliche hinausgehen.

In den letzten Jahren wurde eine Reihe von strukturellen Data Mining Techniken entwickelt. Dazu gehört vor allem *Sequenz Mining* [1], *Tree Mining* [2] und *Graph Mining* [3]. Diese Techniken werden zunächst zur Exploration und Wissensentdeckung eingesetzt. Sie sind in der Lage, automatisch strukturelle Muster wie Teilsequenzen und Teilgraphen zu finden, die eine bestimmte Häufigkeit in einer Datenbank von Instanzen haben. Mit diesen strukturellen Techniken wird eine Reihe von neuen Analysemöglichkeiten erschlossen. Beispielsweise werden mit Se-

quenz Mining Einkaufsgewohnheiten von Kunden über längere Zeiträume analysiert und Graph Mining, wie bereits erwähnt, in der Pharmaforschung auf Moleküldaten angewendet. So kann genau entdeckt werden, welche Teilstruktur für die Wirkung eines bestimmten Medikaments verantwortlich ist, und nicht nur, dass Moleküle mit bestimmten gemessenen Eigenschaften diese Wirkung erzielen. Es wird also direkt mit Strukturen gearbeitet und nicht mit beispielsweise numerischen Daten, die eine bestimmte Struktur in aggregierter Form beschreiben.

Strukturelles Data Mining stellt aber keineswegs eine Ablösung von etablierten Techniken dar, sondern eine Ergänzung. Interessant sind vor allem Kombinationen von herkömmlichen und strukturellen Techniken, da sich so die Stärken von beiden Herangehensweisen gegenseitig ergänzen können. So kann die Einbeziehung von struktureller Information in herkömmliche Verfahren zur Steigerung der Ergebnisqualität beitragen, bzw. können etablierte Techniken strukturelles Data Mining ergänzen, um so neue Anwendungsgebiete zu erschließen. Verbesserte Ergebnisse sind dabei vor allem deshalb möglich, weil etablierte Techniken oft mit numerischen Daten arbeiten, die Strukturen beschreiben. Diese können hingegen mit strukturellen Techniken unmittelbar und ohne Verluste betrachtet werden, die oft durch die Reduktion auf numerische Größen entstehen.

Im Folgenden stellen wir Arbeiten vor, bei denen wir durch Integration von Sequenz Mining und Klassifikation die Qualität von Voraussagen von Kundenverhalten verbessern konnten und es durch die Kombination mit Feature Selection Algorithmen ermöglichst haben, beim Graph Mining auch numerische Kantengewichte mit einzubeziehen. Des Weiteren werden wir einen Ausblick auf weitere Arbeiten geben, bei denen wir durch Kombination von herkömmlichen und strukturellen Techniken verbesserte Ergebnisse erwarten.

2 Voraussagen von Kundenverhalten

In der Telekommunikationsbranche gehört es zu einer der wichtigsten Fragen, Voraussagen darüber machen zu können, welche Kunden mit hoher Wahrscheinlichkeit kündigen werden, um den Anbieter zu wechseln. Mit verschiedenen Data Mining Techniken werden dazu Voraussagen getroffen [4], die eine wichtige Information für Marketing und Geschäftsführung darstellen. Die Analyse von Kundendaten ist in aller Regel schwierig, so dass Voraussagen meist nicht außerordentlich gut sind, allerdings oft besser als gar keine.

Zu den Kundendaten gehört neben den beschriebenen relationalen Daten auch die Interaktions-Historie mit dem Anbieter. Hierzu gehören z. B. Anrufe bei der Service-Line des Anbieters sowie Reparaturen, Störungsmeldungen, Bestellungen etc. Diese Daten müssen für die Anwendung von herkömmlichen Techniken aggregiert werden. Das führt zu einer Vielzahl von dünn-besetzten Attributen wie der Anzahl der Reparaturen im letzten Monat, Jahr und insgesamt. In [5] haben wir Sequenz Mining eingesetzt, um häufige Sequenzen von Kundenverhalten aufzudecken. Das Ergebnis ist, dass zwar interessante Sequenzen gefunden wurden, dass diese alleine aber nicht geeignet sind, das zukünftige Kundenverhalten vorauszusagen.

Aus diesem Grund haben wir ein kombiniertes Verfahren entwickelt: In einem ersten Schritt suchen wir häufige Sequenzen in einem Kundendatensatz. Davon ausgehend identifizieren wir für jede dieser Sequenzen die Kunden, die die jeweilige Sequenz von Interaktionen in ihrer Historie aufweisen. Ausgehend von der strukturellen Information, welche Sequenzen ein Kunde enthält, werden die Kunden also in Gruppen eingeteilt. Für jede dieser Kundengruppen lernen wir dann einen Entscheidungsbaum, der neben Daten zu den jeweiligen Interaktionen auch den allgemeinen Kundendatensatz mit einbezieht. Um dann neue Kunden zu klassifizieren und Voraussagen zu machen, überprüfen wir zunächst, welche Sequenzen in der Historie eines Kunden enthalten sind. Davon ausgehend wählen wir die spezialisierten Entscheidungsbäume aus, mit denen der Kunde mit seinem spezifischen Interaktionsverhalten klassifiziert werden kann. Liegen mehrere unterschiedliche Klassifikationen für einen Kunden vor, so kombinieren wir diese mit Ensemble-Techniken [6]. Der Vorteil von diesem Vorgehen liegt im Wesentlichen darin, dass jeder Entschei-

dungsbaum nicht einfach auf alle Kunden angewendet wird, sondern auf homogenere Gruppen. Das führt zu besseren Ergebnissen. Auf diesem Weg können auch die Informationen betrachtet werden, die zu den einzelnen Interaktions-Events gehören, beispielsweise die Kosten einer Reparatur. Nach herkömmlichem Vorgehen hätte solche Information nur in aggregierter Form analysiert werden können. Das bedeutet im Allgemeinen einen Verlust an potentiell nützlicher Information.

Mit dem beschriebenen Verfahren konnten in einem Sample von einem echten Kundendatensatz 19,5% aller Kunden im Voraus entdeckt werden, die dann tatsächlich gekündigt haben. Die False-Positive-Rate lag dabei bei lediglich 2,6%. Diese Ergebnisse sind etwas besser als die Ergebnisse mit bisherigen nicht-strukturellen Verfahren auf derselben Datenbasis.

3 Fehlerlokalisierung in der Softwaretechnik

Auch in intensiv getesteter Software kommt es immer wieder zu Fehlfunktionen nach der Auslieferung, die mit hohen Kosten einhergehen. Aus diesem Grund spielen Werkzeuge, die in der Lage sind, Fehler zu entdecken, eine wichtige Rolle. Besonders schwer zu finden sind die so genannten *noncrashing bugs*, bei denen es nicht zum Abbruch des Programms kommt, sondern zu einem Fehlverhalten wie der Berechnung eines falschen Ergebnisses. Bei solchen Fehlern werden keine *stack traces* ausgegeben, die dem Entwickler einen Hinweis geben, wo sich ein Fehler versteckt.

In der Literatur gibt es vielfältige Ansätze, Fehler mit Hilfe von Data Mining zu finden. In [7] werden beispielsweise Quellcode-Metriken aus der Softwaretechnik mit Daten aus einer Bug-Datenbank zusammengeführt und mit Regressionstechniken analysiert. Für die Metrik *cyclomatic complexity* [8] wird hier z. B. ein starker Zusammenhang zur Fehleranfälligkeit von Software festgestellt. Diese basiert auf den Kontrollflussgraphen einzelner Funktionen und dem Grad der Verzweigungen dieser. Ein Kontrollflussgraph umfasst die sogenannten Basic Blocks mit mehreren Nicht-Sprung-Befehlen als Knoten und die verschiedenen Ausführungswege als Kanten. Die cyclomatic complexity ist aber dennoch nur eine Kenngröße, die aus Graphen abgeleitet wird, die genaueren Strukturen selbst können in ihr nicht abgebildet werden.

Es gibt aber auch Arbeiten [9, 10], die die *Call-Graphen* von fehlerfreien und fehlerhaften Programmausführungen mit Graph Mining Techniken analysieren und auf diesem Weg Fehler lokalisieren. Solche Call-Graphen spiegeln die Aufrufstruktur von Methoden bzw. Funktionen einer Programmausführung wieder. Ausgehend von der `main()`-Methode werden weitere Methoden aufgerufen, die als Kindsknoten dargestellt werden und selbst ggf. weitere Methoden aufrufen (s. Abb. 1(a)). Im Allgemeinen werden solche Call-Graphen sehr groß. Das liegt vor allem daran, dass durch Schleifen bestimmte Teilstrukturen sehr oft hintereinander ausgeführt werden. Da Graph Mining Algorithmen im Allgemeinen nicht gut skalieren, müssen die Call-Graphen zunächst reduziert werden. In [9] wird der Call-Graph derart reduziert, dass jede Methode nur einmal vorkommt (*totale Reduktion*). In [10] dagegen werden nur Iterationen betrachtet: Taucht eine Teilstruktur mehr als zweimal auf, werden alle darüber hinausgehenden Teilstrukturen aus dem Graphen entfernt. Beispiele für die verschiedenen Reduktionsstufen finden sich in Abb. 1.

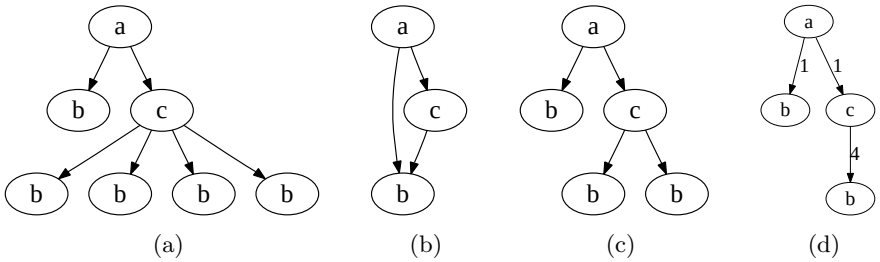

Abbildung 1: **(a)** Ein unbearbeiteter Call-Graph, **(b)** der gleiche Graph total reduziert wie in [9], **(c)** dargestellt wie in [10] und **(d)** reduziert mit Kantengewichten.

Wir sind der Überzeugung, dass bei der totalen Reduktion [9] zu viel potentiell wichtige Information verloren geht. Auch die Reduktionstechnik in [10] ist problembehaftet. Es kann z. B. passieren, dass bei Fehlern in Schleifenbedingungen eine Teilstruktur in fehlerfreien Ausführungen einige wenige Male ausgeführt wird, in fehlerhaften Fall aber einige hunderte Male. In beiden genannten Reduktionsverfahren resultiert das in identischen Graphen im fehlerfreien und fehlerhaften Fall. Aus diesem Grund schlagen wir vor, Call-Graphen derart zu reduzieren, dass Iterationen strukturell zu einer zusammengefasst werden, die Ausführungshäufigkeit jedoch als Kantengewicht notiert wird. Auf diese Art können wir die Größe der Graphen signifikant verringern und verlieren nur ein Minimum an Information (s. Abb. 1(d) für ein Beispiel).

Nachdem wir Programme ausgeführt haben, entscheiden wir durch Vergleich des Ergebnisses mit einem Referenz-Ergebnis, ob die Ausführung als fehlerfrei oder fehlerhaft zu klassifizieren ist. Dann reduzieren wir die Call-Graphen wie oben beschrieben, wobei Kantengewichte entstehen. Die so gewonnene Datenbank von Call-Graphen analysieren wir nun mit dem Graph Mining Algorithmus *CloseGraph* [11] – die Kantengewichte werden dabei zunächst außen vor gelassen. Dies resultiert in einer Menge von häufigen Teilgraphen. Da wir uns auf die Analyse von Kantengewichten konzentrieren wollen, betrachten wir nur die Teilgraphen, die sowohl in fehlerfreien als auch in fehlerhaften Ausführungen auftreten. Ausgehend von solchen Graphen betrachten wir die Kantengewichte, mit dem Ziel herauszufinden, welche Gewichte einen besonders großen Beitrag leisten, fehlerfreie von fehlerhaften Ausführungen zu unterscheiden. Dazu verwenden wir etablierte Feature Selection Algorithmen, die auf Entropie (Informationsgewinn) basieren. Auf diesem Weg gelangen wir zu einem Ranking, welche Methodenaufrufe eines Programms am wahrscheinlichsten für fehlerhafte Ausführungen verantwortlich sind.

Da unterschiedliche Arten von Fehlern unterschiedlich gut mit verschiedenen Techniken entdeckt werden können, kombinieren wir die Ergebnisse unseres auf Kantengewichten basierenden Verfahrens mit dem rein strukturell arbeitenden Verfahren aus [10]. Als Ergebnis wird eine kombinierte Liste von Methoden ausgegeben, die nach der Reihenfolge der Wahrscheinlichkeiten sortiert ist, einen Fehler zu enthalten. Eine solche Liste kann dann einem Software-Entwickler gegeben werden, der so ganz gezielt einige Methoden noch einmal überprüfen kann. Erste Experimente zeigen, dass auf diesem Weg einerseits Fehler lokalisiert werden können, die mit dem Verfahren aus [10] nicht gefunden werden konnten. Hierbei handelt es sich z. B. um Fehler in Schleifenbedingungen, die sich in unterschiedlichen Kantengewichten niederschlagen. Andererseits tritt im Mittel aller anderen Fälle, durch die Kombination beider Verfahren, eine Verdoppelung der Lokalisierungsgenauigkeit gegenüber der rein strukturellen Analyse in [10] ein.

4 Auswahl von Hardware-Architekturen

Ein interessantes Problem in der Rechnerarchitektur ist es, die Architektur vorherzusagen, die am Besten für ein bestimmtes Programm geeignet ist, auf dem dieses also am schnellsten ausgeführt wird. Dieses Auswahlproblem lässt sich als Klassifikationsproblem formulieren und findet seine Motivation in der zukünftigen Verfügbarkeit von rekonfigurierbarer Hardware und Multi-Core-Prozessoren mit mehreren spezialisierten Kernen. Hier möchte man gerne möglichst zur Laufzeit Aussagen darüber machen können, auf welcher Hardware(-Konfiguration) ein gegebenes Programm am Besten ausgeführt werden kann. Als Eingabedaten können prinzipiell alle Informationen dienen, die ein Programm beschreiben, beispielsweise verschiedene Code-Metriken aus der Softwaretechnik. Eine vielversprechende Herangehensweise ist auch die Analyse der Kontrollflussgraphen der einzelnen Funktionen, die aus einem Programm statisch generiert werden können.

Durch Experimente mit der SPEC-Benchmark-Suite und einer Reihe von Quellcode-basierten Metriken aus der Softwaretechnik haben wir bisher herausgefunden, dass durch die Verwendung von herkömmlichen Data Mining Techniken bereits gute Vorhersageergebnisse werden können. Diese haben bisher eine Genauigkeit von 72%. In ersten Tests mit einer Reihe von Metriken, die

auf Kontrollflussgraphen basieren, haben wir zudem gezeigt, dass diese die bisherigen Ergebnisse stak verbessern können. In Zukunft wollen wir auch in dieser Anwendungsdomäne mit Graph Mining Techniken häufige Substrukturen entdecken, mit denen Software-Artefakte noch präziser beschrieben werden können. So wollen wir Strukturen extrahieren und Informationen über ihr Auftreten in Kombination mit weiteren Metriken mit Standard-Techniken analysieren.

5 Ausblick

In den vorgestellten Arbeiten haben wir das Potential gezeigt, mit unterschiedlichen Kombinationstechniken von strukturellem und relationalem Data Mining zu verbesserten Ergebnissen zu gelangen und dass weitere Forschung in diese Richtung vielversprechend ist. In Zukunft wollen wir weitere Studien in diese Richtung unternehmen. Bei der Fehlerlokalisierung im Gebiet der Softwaretechnik analysieren wir die Kantengewichte zur Zeit in einem dem Graph Mining nachgelagertem Schritt. Diesem nachgelagertem Vorgehen wollen wir einen Ansatz in der Vorverarbeitung (z. B. Diskretisierung) gegenüberzustellen und Möglichkeiten der Behandlung von numerischer Information innerhalb des Graph Mining Algorithmus erforschen.

Vor allem aber wollen wir eine systematische Klassifikation von Kombinationsmöglichkeiten von strukturellen und herkömmlichen Techniken entwickeln, in die sowohl bisherige als auch zukünftige Arbeiten eingeordnet werden können. Daraus kann ein Rahmen entstehen, mit dem die identifizierten Kombinationsmöglichkeiten leicht realisiert werden können.

Durch weitere Studien, auch in anderen Gebieten wie der Transportlogistik, wollen wir einerseits durch kombinierte Methoden zu guten Ergebnissen kommen. Andererseits wollen wir aber auch an neuen Kombinationstechniken sowie an Herausforderungen im Bereich des Graph Minings selbst arbeiten.

Literatur

[1] Rakesh Agrawal and Ramakrishnan Srikant. Mining Sequential Patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE)*, 1995.

[2] Y. Chi, R. Muntz, S. Nijssen, and J. Kok. Frequent Subtree Mining – An Overview. *Fundamenta Informaticae*, 66(1–2):161–198, 2005.

[3] D.J. Cook and L.B. Holder, editors. *Mining Graph Data*. John Wiley & Sons, 2006.

[4] Scott A. Neslin, Sunil Gupta, Wagner Kamakura, Junxiang Lu, and Charlotte H. Mason. Defection Detection: Measuring and Understanding the Predictive Accuracy of Customer Churn Models. *Journal of Marketing Research*, 43(2):204–211, 2006.

[5] Frank Eichinger, Detlef D. Nauck, and Frank Klawonn. Sequence Mining for Customer Behaviour Predictions in Telecommunications. In *Proceedings of the Workshop on Practical Data Mining at ECML/PKDD*, 2006.

[6] L.I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, 2004.

[7] Nachiappan Nagappan, Thomas Ball, and Andreas Zeller. Mining Metrics to Predict Component Failures. In *Proceedings of the 28th International Conference on Software Engineering (ICSE)*, 2006.

[8] T.J. McCabe. A Complexity Measure. *Transactions on Software Engineering*, 2(4):308–320, 1976.

[9] Chao Liu, Xifeng Yan, Hwanjo Yu, Jiawei Han, and Philip S. Yu. Mining Behavior Graphs for "Backtrace" of Noncrashing Bugs. In *Proceedings of the 5th SIAM International Conference on Data Mining (SDM)*, 2005.

[10] Giuseppe Di Fatta, Stefan Leue, and Evghenia Stegantova. Discriminative Pattern Mining in Software Fault Detection. In *Proceedings of the 3rd International Workshop on Software Quality Assurance (SOQUA)*, 2006.

[11] Xifeng Yan and Jiawei Han. CloseGraph: Mining Closed Frequent Graph Patterns. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD)*, 2003.

Flash-Disks - Neue Speicherhierarchieen für Datenbankmanagementsysteme?

Michael Höding
FH Brandenburg, Magdeburger Straße 50, 18770 Brandenburg/Havel
hoeding@fh-brandenburg.de

Zusammenfassung

In diesem Beitrag wird der Einsatz von Flash-Disks für DBMS-basierte Systeme untersucht. Nach der Vorstellung der technologischen Grundlagen wird auf das Gebiet Benchmarking eingegangen. Erste Messungen demonstrieren das Verhalten von Datenbankanwendungen bei der Nutzung von Flash-Disks.

1 Einführung und Motivation

Datenbankmanagementsysteme müssen zwei widersprüchliche Anforderungen erfüllen. Zum einen sollen sie für viele Benutzer eine hohe Transaktionsleistung erbringen. Zum anderen muss die Datensicherheit gewährleistet sein. Zur sicheren Datenspeicherung kommen dabei Festplattenspeicher in verschieden Konfigurationen und Anschlussarten zum Einsatz, die als sicher angesehen werden. Die als Speicherlücke bekannte langsamere Schreibleistung von Festplatten (gegenüber dem flüchtigen Hauptspeicher) macht für die Erfüllung der hohen Performanceanforderungen eine ausgeklügelte DBMS-Architektur notwendig. Allgemein nutzen DBMS intensiv die Techniken Pufferung und Transaktionslogs, verbunden mit Anfrageoptimierung.

Derzeit beginnen Flash-Disks (bzw. Solid-State-Disks) hinsichtlich Speichervolumen den traditionellen Festplatten Konkurrenz zu machen. Grundsätzlich ermöglichen sie aufgrund ihrer Technologie einen deutlich schnelleren wahlfreien Zugriff, auch wenn sie die Geschwindigkeit des schnellen Hauptspeichers aufgrund der Speichertechnologie und der Anbindung nicht erreichen. Denkbar ist jedoch eine Entwicklung, in der durch den Einsatz sehr schneller nichtflüchtiger Speicher auf Flash-Basis eine einfache DBMS-Architektur möglich wird.

In diesem Beitrag sollen nach einer kurzen Einführung in die Grundlagen Potenziale des Einsatzes von Flash-Disk für DBMS diskutiert werden. Zur Überprüfung der Thesen wird kurz auf Datenbank-Benchmarks eingegangen, deren typisches Lastverhalten für erste Messungen genutzt wird. Die Messungen untersuchen in einem Negativszenario die Speichergeschwindigkeiten einer exemplarischen Flash-Disk.

2 Architekturen

Ein grundsätzliches Problem bei der Datenverarbeitung ist die Speicherlücke [5]. Um die Möglichkeiten des sehr schnellen Primärspeichers auszunutzen müssen die benötigten Daten im Hauptspeicher gepuffert werden. Gleichzeitig muss ein schneller schreibender Zugriff gewährleistet sein, ohne dabei Abstriche an der Datensicherheit zu machen. Auch ist zu beachten, dass der Hauptspeicher in der Regel um ein bis zwei Größenordnungen kleiner ist als der Sekundärspeicher. Vor diesem Hintergrund wurden für DBMS eine Reihe von Mechanismen zur Performance-Verbesserung entwickelt, u.a. Zugriffspfade (Indexe), regelbasierte Anfrageoptimierung, statistikbasierte Anfrageoptimierung, Pufferung oder schnelle Transaktionen durch Log-Dateien [5]. Diese Mechanismen und Verfahren sind ausgereift und trotz ihrer Kompliziertheit sehr verlässlich. Jedoch muss beachtet werden, dass für eine Optimierung ein gewisser zusätzlicher Zeitaufwand notwendig ist, der deutlich unter dem Gewinn durch die Optimierung liegen muss.

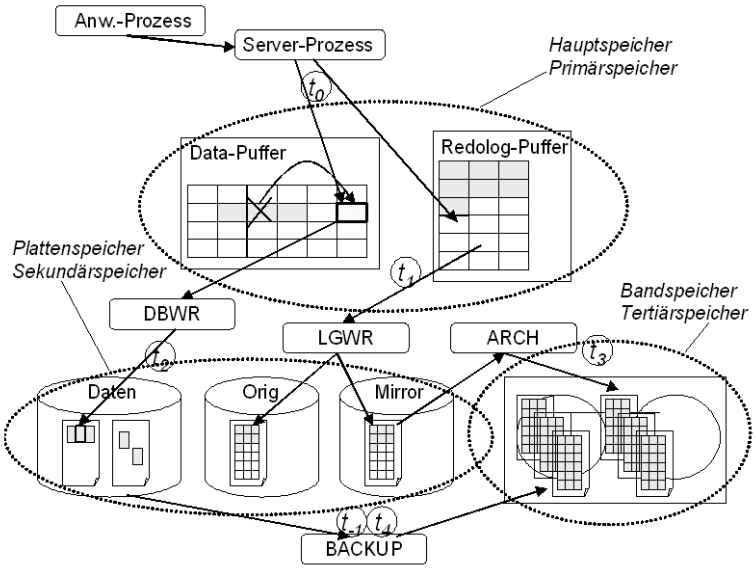

Abbildung 1: Speichervorgänge am Beispiel von Oracle

Abbildung 1 veranschaulicht die Nutzung des Log-Mechanismus für schreibende Transaktionen [1]. Eine zum Zeitpunkt t0 initiierte Änderung wird nach einem commit zum Zeitpunkt t1 persistent gemacht, indem die Log-Einträge in das Transaktionslog geschrieben werden. Zu einem späteren Zeitpunkt werden dann auch die Datenblöcke in die Daten-Dateien eingefügt. Der Gewinn resultiert aus dem schnellen Schreiben in die sequentiellen Log-Dateien.

Datenbankanwendungen nutzen eine mehrschichtige Client-Server-Architektur. In der klassischen dreischichtigen Architektur, wie sie zum Beispiel durch SAP R/3 weit verbreitet ist, greifen viele Clients auf einen bzw. eine Gruppe von Applikationsservern zu, die wiederum als Clients ein DBMS nutzen. Hierbei werden im Applikationsserver in einer mehr oder weniger sauberen Weise Applikationsobjekte (z.b. betriebswirtschaftliche Objekte wie Plan oder Fertigungsauftrag) betrachtet. Das Datenbankmanagementsystem dient als sicherer, schneller und komfortabler Tabellenspeicher. In traditionellen Systemen wird kaum auf die in modernen objektrelationalen DBMS ausgeprägte Objektorientierung zurückgegriffen.

In diesem Zusammenhang gibt es auch auf der Applikationsebene Transaktionsmechanismen, die Pufferung und Logging benutzen. Hier wird möglicherweise doppelt gepuffert. Betrachtet man das unter dem DBMS wirkende Betriebssystem, bzw. genauer das Dateisystem, so sind auch hier Pufferungs- und Loggingmechanismen ausgeprägt. Zu nennen sind Journaling-File-Systeme wie ReiserFS, ext3 oder NTFS. In diesem Sinne ist das eigentliche verteilte Schreiben von Datenbankblöcken durch einen Database-Writer-Prozess ein sequentielles Schreiben, analog dem Transaktionslog. Dieser Aspekt soll hier aber nicht weiter vertieft werden.

3 Solid-State-Disks

Flash-Speicher erfreuen sich durch zahlreiche Anwendungen im mobilen Bereich großer Beliebtheit. In diesem Zusammenhang ist ein überdurchschnittliches Wachsen der verfügbaren Speichergrößen und ein Sinken der Preise zu beobachten. Flash-Speicher sind ausgesprochen robust und sparsam. Sie haben deshalb in vielen Bereichen CDs und DVD verdrängt. Zusammengefasst zu Solid-State-Disks beginnen sie als Festplatte im Notebook-Bereich Fuss zu fassen.

Der Zugriff auf Flash-Speicher unterscheidet sich vom Zugriff auf RAM und Plattenspeicher. Grundsätzlich ist ein wahlfreier Zugriff wie beim RAM ohne das mechanische und damit zeitaufwendige Positionieren eines Schreib-Lese-Kopfes möglich. Jedoch ist das Schreiben deutlich aufwendiger und langsamer. Das Setzen eines Bit im Flash-Speicher erfolgt durch eine hohe

Spannung. Ausgesprochen aufwendig ist jedoch das Rücksetzen eines Bits, das nur für einen ganzen Block erfolgen kann. Hier werden durch eine negative Spannung alle Bits eines Blocks auf 0 gesetzt. Anschließend muss dann wieder der Orginalzustand hergestellt werden. Nach [2] ergibt sich folgendes Bild:

- Hinsichtlich der Lesegeschwindigkeit sind Flash-Speicher mit 60 MBytes/s ähnlich schnell wie Notebook-Festplatten
- Beim Schreiben sind Flash-Speicher halb so schnell.
- Hinsichtlich der Zugriffszeit ist der Flash-Speicher um ca. zwei Größenordnungen der Festplatte überlegen. (Damit allerdings drei Größenordnungen unter RAM.)

Durch die Anordnung von einzelnen Flash-Chips zu Solid-State-Disks in Form von RAID-0 lassen sich Verbesserungen erreichen. Des weiteren sind die Blockgrößen, Schnittstellen und die Art des Zugriffs entscheidend.

4 Benchmarking

Benchmarking dient dem Vergleich von Systemen bzw. Systemkomponenten. Die Anwendungen sind vielfältig. Mit den populären Datenbankbenchmarks der TPC [3] vergleichen einerseits Hardware-Hersteller ihre Serversysteme am Beispiel eine bestimmten DBMS. Andererseits vergleichen DBMS-Hersteller ihre Systeme unter Nutzung einer bestimmten Hardware. Interessant ist, dass der TPC-Benchmark auch solche Aussagen liefert wie "Transaktionen pro Dollar". Für das Benchmarking von Flash für die DBMS-Nutzung gibt es verschiedene Ansätze:

- *Nutzung von Festplattenbenchmarks:* Für die Vermessung von Festplatten stehen eine Reihe von Programmen zur Verfügung. Erste Anhaltspunkte liefert die Linux-Kommand `hdparm -t /dev/hda`. Hier wird allerdings nur das sequentielle Lesen simuliert. Sehr viel detailliertere Ergebnisse liefern Benchmarks wie `hdbench`, die verschiedene Zugriffsarten in Zusammenhang mit verschiedenen Blockgrößen testen. Bei Nutzung von Festplattenbenchmarks muss man sehr genaues Wissen über den physischen Datenzugriff des DBMS haben. Sie bieten dann einen guten Ansatzpunkt um über eine Optimierung der Architektur nachzudenken und diese zu testen.
- *Proprietäre Benchmarks:* Diese Benchmarks prüfen anhand einzelner SQL-Operationen die System-Performance für ein konkretes DBMS. Ein populärer Vertreter ist sql-bench in Zusammenhang mit mysql. Hier sind Kenntnisse über den SQL-Mix notwendig.
- *Anwendungs-Benchmarks:* Sie simulieren anhand einer Beispieldatenbank eine übliche Transaktionslast.
  - TPC [3]: Die Benchmarks des *Transaction Processing Performance Council* liefern auf Basis einer genauen Spezifikationen der Datenbanklast gute Vergleichswerte und sind allgemein anerkannt. Aus Datenbanksicht sind besonders der TPC-C für OLTP und der TCP-H für OLAP interessant.
  - DBT2: Der DBT2-Benchmark wurde von *Open Source Development Labs* (OSDL) als offizieller Vergleichswert für Datenbanken in Anlehnung an den TPC-C entwickelt [4]. Für ein großes Handelsunternehmen mit mehreren Lagern wird eine Datenbank erzeugt, gegen die dann ein typischer Lastmix von Aufträgen, Anfragen und Lieferung gefahren wird.

  Diese Art Benchmark liefert einen allgemeinen Überlick der Systemperformance aus Anwendersicht für den praktischen Einsatz.

Zusammenfassend ist zu sagen, dass die Wahl der Benchmarkart von der Aufgabenstellung abhängt. Das letztendliche Ziel ist die Verbesserung der Performance aus Anwendungssicht.

Davon abgeleitet kann man als Ziel auch die Suche nach besonderen Einsatzgebieten für Flash-basierte DBMS betrachten. Zur Erreichung dieser Anwendungsziele können nun die anderen Benchmarkarten hinzugezogen werden. Sie helfen Schwachstellen zu entdecken und ggf. durch die Optimierung des Systems bis hin zur Architektur zu beseitigen.

Benchmarking ist in der Regel sehr aufwendig und sollte einer Systematik folgen. Nach der Wahl eines Benchmarks muss zunächst die Umgebung eingerichtet werden. Die ersten Benchmark-Läufe dienen dem Kennenlernen des Benchmarks und der Anpassung der Umgebung. Im nächsten Schritt sollte das Experiment definiert und Vorbetrachtungen über die erwarteten Ergebnisse angestellt werden. Die anschließende Durchführung der Messungen benötigt Zeit und ist fehleranfällig. Beispielsweise konnte bei den durchgeführten Messungen die Zahl der Clients nicht beliebig erhöht werden, da die erlaubte Prozesszahl im Betriebssystem zu niedrig war. Der Benchmark bricht dann ab. Aus diesem Grund ist es sinnvoll erst den zweiten erfolgreichen Lauf aus Messlauf zu nutzen. Abschließend müssen die Ergebnisse aufbereitet und mit den Vorbetrachtungen verglichen werden.

5 Erste Messungen und Ergebnisse

Um die Technologie Flash-Disk für den praktischen Einsatz zu untersuchen wurde der folgende Ansatz gewählt. Eine über USB angebundene 32GB-Solid-State-Disk von Transcend wurde in einem PostgreSQL betrieben. Als Benchmark wurde DBT2 eingesetzt. Insbesondere die Nutzung der USB-Schnittstelle erscheint kritisch, da der Zugriff hier laut [2] um den Faktor 4 beim Lesen und den Faktor 2 beim Schreiben unter der Anbindung über IDE-SATA liegt. Erreicht nun die Konfiguration mit Flash eine bessere Leistung als mit Festplatte so ist die Annahme, dass Flash dir DBMS-Performance verbessert, bestätigt. Als Versuchsumgebung wurde ein PC mit AMD64-3,2 GHz Prozessor und 1 GB RAM genutzt. Es wurden als Betriebssystem OpenSuse10.3 und als DBMS PostgreSQL 8.2.4 eingesetzt.

Messung mit Festplatte

Als Parameter wurde eine Dauer von 20 Minuten, 10 Lager, 10 Clients (Software zum Erzeugen der Last) und 50 Terminals (also 500 Benutzer) festgelegt.

```
Command line: ./run_workload.sh -d 1200 -w 10 -c 10 -t 50 Scale
Factor: 10 warehouses
                      Response Time (s)
 Transaction      %    Average :    90th %        Total        Rollbacks      %
------------  -----  ----------------------  -----------  ---------------  -----
    Delivery   2.93      0.045 :     0.068          788                0   0.00
   New Order  60.71      0.174 :     0.314        16322             7753  90.48
 Order Status  2.87      0.034 :     0.016          772                0   0.00
     Payment  30.69      0.025 :     0.022         8251                0   0.00
 Stock Level   2.79      0.009 :     0.003          750                0   0.00
------------  -----  ----------------------  -----------  ---------------  -----
60.67 new-order transactions per minute (NOTPM)
```

Messung mit Flash-Disk

Hier wurde der selbe Parametersatz genutzt. Für die Messung wurde das Verzeichnis mit den Datenbankdaten (PGDATA) mit eine Softlink auf die Flash-Disk gesetzt.

```
 Transaction      %    Average :    90th %        Total        Rollbacks      %
------------  -----  ----------------------  -----------  ---------------  -----
    Delivery   3.38    150.438 :   200.414          122               10   8.93
   New Order  58.34    118.688 :   209.662         2105              965  84.65
 Order Status  3.63    148.659 :   206.331          131               11   9.17
     Payment  31.65    158.613 :   215.051         1142               92   8.76
 Stock Level   2.99    156.886 :   210.447          108                8   8.00
------------  -----  ----------------------  -----------  ---------------  -----
13.01 new-order transactions per minute (NOTPM)
```

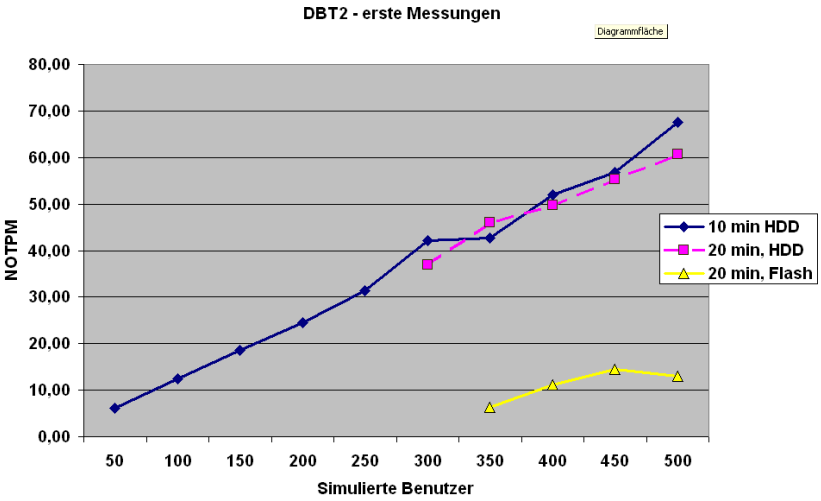

Abbildung 2: Erste Messergebnisse mit DBT2

Die ersten Ergebnisse sind ernüchternd. Der NOTPM-Wert der Flash-Disk liegt um eine Größenordnung unter dem Zugriff bei Nutzung der normalen Festplatte. Die Annahme, dass Flash-Disk unmittelbar Performance-Gewinn bringen, konnte nicht bestätigt werden.

Für weitere Untersuchungen sind Messungen mit veränderten Parametersätzen vorzunehmen (vgl. Abb. 2). Hier wurde für verschiedene Benutzerzahlen (Terminals) der NOTPM Wert gemessen. Insbesondere die Zahl die Terminals muss erhöht werden um das System in den Grenzbereich zu bringen. Die Laufzeit des Test musste von 10 auf 20 Minuten für eine große Zahl an Terminals erhöht werden, da sonst die Startzeit der Testumgebung die Testzeit übersteigt. Die Ergebnisse scheinen für die beiden Laufzeiten ähnlich zu sein. Deutlich ist der Performance-Nachteil des Flash-basierten Systems zu erkennen.

6 Diskussion und Ausblick

Die ersten Messungen mittels eines Anwendungsbenchmarks können keine Überlegenheit der Flash-Technologie für den Einsatz in normalen Datenbanken belegen. Die Ursachen sind dabei vielfältig. Zunächst ist die langsame Anbindung über USB ein Problem. Des weiteren sind heute verfügbare Flash-Speicher teilweise recht langsam - hier wird es sicher eine Entwicklung geben. Auch sind die modernen Datenbanksysteme für die Nutzung von Festplatten optimiert. Möglicherweise wirkt sich deshalb der Vorteil des wahlfreien Zugriffs nicht entsprechend aus. Denkbar ist z.B., dass die Nutzung von Indexen die Zugriffsgeschwindigkeit bremst. Die Untersuchung dieser und weiterer Fragestellungen bietet Ansätze für zukünftige Arbeiten. In diesem Sinne kann das Fragezeichen des Titels bestehen bleiben. Allerdings lassen die ersten Untersuchungen vermuten, dass hinsichtlich der Speicherhierachie optimiert werden muss um die Eigenschaften von Flash-Disk für ein bessere DBMS-Performance zu nutzen.

Literatur

[1] Andre Faustmann, Michael Höding, Gunnar Klein und Ronny Zimmermann: *Oracle-Datenbankadministration für SAP.* Galileo-Verlag, 2007.

[2] Benz, Benjamin und Boi Feddern: *Festplatten ade.* c't Magazin für Computer und Technik, 25(21):100–105, 2007.

[3] Gray, Jim (Herausgeber): *The Benchmark Handbook for Database and Transaction Systems (2nd Edition).* Morgan Kaufmann, 1993.

[4] Labs, Open Source Development: *DBT2 - Database Test Suite.* http://osdldbt.sourceforge.net/, 2006.

[5] Saake, Gunter und Andreas Heuer: *Datenbanken: Implementierungstechniken.* MITP-Verlag, 1999.

InGVer - Intelligente Gefahrgutverfolgung

Die Unterstützung der logistischen Kette durch den Einsatz von Ontologien

Matthias Virgin, Ilvio Bruder, Andreas Heuer
Universität Rostock
E-Mail: {mv, ilr, heuer}@informatik.uni-rostock.de

Abstract

Die Beförderung gefährlicher Güter auf der Straße, der Schiene, dem Wasser und zu Luft nimmt stetig und im starken Maße zu. Damit einhergehend steigt die Gefährdung von Personen, Gegenständen und der Umwelt ebenfalls kontinuierlich an. Zahlreiche Gefahrgutunfälle in den letzten Jahren, die teils zu schweren Umweltkatastrophen führten, zeigen, dass trotz aller Bemühungen und Sicherheitsmaßnahmen noch viele Restrisiken vorhanden sind. Die Quellen dieser Risiken lassen sich neben Anderem in fehlenden und fehlerhaften Kommunikationswegen, in unzureichend vorliegenden Informationen während des Transports von Gefahrgütern sowie in der Komplexität und hohen Aktualisierungsintensität der Vorschriften und Richtlinien für den Gefahrguttransport wiederfinden. Ein Hauptziel des Projektvorhabens „InGVer“ ist die Entwicklung verschiedener Konzepte zur intelligenten Steuerung, Überwachung und Dokumentation der komplexen Gefahrgutprozesse. Ein Weg zur Realisierung dieser Ziele liegt in der Nutzung von speziellem Wissen, mit dessen Hilfe z.B. vorhandene IT-Systeme eingebunden und Rechtsvorschriften beherrschbar gemacht werden können. Mit der Umsetzung dieser Lösungsansätze sollen vorhandene fehlerhafte Gefahrgut-Dokumentationen und lückenbehaftete Informationsflüsse vermieden werden. In dieser Arbeit werden erste Überlegungen und Aspekte zum Einsatz wissensbasierter Unterstützung in den verschiedensten Prozessen des Gefahrguttransportes erfolgen.

1 Einleitung

Die Gefahrgutverfolgung ist ein bisher aufwendiges und über lange Prozess- bzw. Nachrichtenketten protokolliertes System. Der Transport und diverse Informationen über diesen Transport sind zwei unterschiedlich voneinander ablaufende Prozesse. Während der Transport lediglich von Lagerungen unterbrochen wird, kann es in der Informationskette zu kompletten Lücken kommen (siehe Abb. 1 – Informationskette). Die Heterogenität der IT-Systeme innerhalb der Logistikkette macht es teilweise unmöglich, Informationen weiter zu verwenden, um somit nachgelagerte IT-Systeme zu füllen.

Hinzu kommen für jeden Verkehrsträger verschiedene gesetzliche kennzeichnungs-, dokumentations-, transport- und verpackungstechnische Regularien, die gerade für gefährliche Güter sehr ausgeprägt sind. Die Fülle dieser Richtlinien und Ausnahmen sowie Sondervorschriften erschweren damit das effiziente Arbeiten mit Gefahrgutsendungen und verlangen gut ausgebildete Mitarbeiter, die mit diesen komplexen Gesetzestexten umgehen können müssen. Der Aufwand, der dabei entsteht, kann oftmals nicht durch die finanziellen Einnahmen beglichen werden.

Diese Missstände in den einzelnen Bereichen des Gefahrgutsektors können somit zu Fehlersituationen führen, wie z.B. falsche bzw. ungenügende Kennzeichnung von Gefahrgut. Dies kann im besten Fall zur Beförderungsablehnung der Güter im weiteren Transportprozess führen und im schlimmsten Fall während einer Havarie zum Einsatz falscher Bekämpfungsmaßnahmen und daraus resultierend zu Unfällen kommen.

Weiterhin entsteht durch den Mangel an Informationen innerhalb der logistischen Kette für die beteiligten Unternehmen und Behörden, als auch für die Kunden eine nicht transparente Abwicklung von Geschäftsvorgängen. Und gerade der Bereich der Gefahrgutlogistik stellt hinsichtlich Kundenbindung ein sehr sensibles Umfeld dar. Deshalb wäre hier neben weiteren Gründen eine durchgängige Versorgung mit Informationen sehr wichtig und sinnvoll.

2 Realisierungsansätze

Anforderungen an das abzubildende Wissen

Der Umfang und die Komplexität der in der Einleitung geschilderten Gefahrguttransporte und das Einbeziehen von verschiedensten Daten-, Informations- und Wissensbasen aus unternehmensübergreifender Sicht verlangen im Vorfeld eine Anforderungsanalyse, die gewisse Aspekte über das einzusetzende Wissen betrachtet.

Zum einen sind aus der unternehmensübergreifenden Informationsgewinnung zur Abwicklung der Gefahrguttransporte unterschiedliche Unternehmensbarrieren zu überwinden. Das heißt, dass bestimmte erforderliche Informationen durch beteiligte Logistikpartner nicht ohne Weiteres für nachgelagerte Prozesse zur Verfügung gestellt werden. Somit müssen Wissensbasen, die in diese Richtung eingesetzt werden, eine gewisse Form der Diskretion bzw. Geheimhaltung erfüllen, damit Barrieren anonym und sicher überwunden werden können.
Zum anderen existiert eine hohe Fluktuation der Daten und Informationen in einigen Bereichen des Gefahrgutsektors. Umfassende und durch umfangreiche Verknüpfung der Wissensbasen untereinander entstehende komplexe Update-Funktionalitäten sind zu berücksichtigen.

Weiterhin ist ein Augenmerk auf das Einbinden der Wissensbasen zu setzen, da wir uns in einem dezentralen Anwendungsszenario befinden, in dem verschiedenste Abläufe, ausgehend von heterogenen Systemen, zu harmonisieren sind. Somit ist der Einsatz übergeordneter Steuerungsmechanismen, wie z.B. Meta-Workflows, denkbar.

Einsatzgebiete wissensbasierter Werkzeuge

In den weitläufigen Prozessen und Situationen des Gefahrguttransportes können an mehreren Stellen Schwachpunkte und Fehlersituationen auftreten, die oftmals durch menschliches Versagen und ungenügende Datenbereitstellung durch Informationssysteme begründet werden können. Aus einer groben Ausgangssituation heraus, stehen hier folgende Prozesse und Situationen, in denen wissensbasierte Werkzeuge einen Vorteil gegenüber üblicher Vorgehensweisen[1] erbringen können.

Diese Prozesse und Situationen sind im Einzelnen:

1. der Einbezug heterogener Daten und Informationen aus ebenso heterogenen IT-Systemen
2. die Klassifizierung der Gefahrgüter für den Transport
3. die Umkodierung der Gefahrgüter bei Wechsel der Verkehrsträger

1. Einbezug heterogener Daten und Informationen aus ebenso heterogenen IT-Systemen

An dem Transport beteiligte Auftraggeber und –nehmer besitzen zumeist eigene IT-Systeme und Formalismen zur Abwicklung des Transportes. Dabei treten oftmals folgende Probleme auf:

- ungenügende bzw. keine Weitergabe wichtiger Informationen im weiteren Transportprozess
- Übermittlungsfehler durch manuelle Übergabe von Daten
- kaum vorhandene bzw. unstandardisierte Schnittstellen zum Datenaustausch
- aufwendige Zusammenarbeit durch bestehende unternehmensphilosophische Barrieren

Diese Probleme treten überwiegend bei Schnittstellenwechseln im Transportprozess auf, also beim Wechsel der Verkehrsträger und beim Wechsel der verantwortlichen IT-Systeme (siehe Abb. 1 – Informationskette). Um diesen Problemen entgegenzuwirken, könnten Schnittstellendefinitionen (soweit vorhanden) in Ontologien abgebildet werden, um den Übergang von Daten in nachgelagerte Systeme zu erleichtern. Dabei ist es unerheblich, ob die Daten über Web-Services oder anderen Schnittstellen zur Verfügung gestellt werden. Diese Schnittstellen werden in OWL Dateien gehalten. Für die

[1] Übliche Vorgehensweisen: manuelle Behandlung, Einsatz herkömmlicher Abbildungsverfahren

Schnittstellenbeschreibung von Web-Services werden hier OWL-S[2] Konstrukte eingesetzt werden (siehe Beispiel in Listing 1).

```
...
   <service:Service rdf:ID="Logistics_Ltd_RequestAgent">
     <service:describedBy>
       <process:CompositeProcess rdf:ID="Logitics_Ltd_GoodRequest">
         <process:hasInput rdf:resource="#CodeOfGood"/>
         <process:hasOutput rdf:resource="#TargetOfGood"/>
       </process:CompositeProcess>
     </service:describedBy>
     <service:providedBy rdf:resource="http://www.logistics-ltd.de"/>
     <service:supports>
       <grounding:WsdlGrounding rdf:ID="Logistics_Ltd_Grounding">
         ...
          <grounding:wsdlMessagePart rdf:datatype="XMLSchema#string">
            http://www.logisticsltd.de/wsdl/logistics.wsdl#TargetOfGood
          </grounding:wsdlMessagePart>
          ...
       </grounding:WsdlGrounding>
     </service:supports>
     <service:presents rdf:resource="#Logistics_Ltd_Profile"/>
   </service:Service>
...
```

Listing 1: OWL-S Ausschnitt für eine Anfrage nach dem Ziel des Transportgutes

Die Verarbeitung der benötigten Informationen wird hier als Blackbox-Verfahren gegenüber jedem Logistikpartner erfolgen, sodass in der Wissensbasis auch sensible Informationen abgebildet werden können. Auch für Situationen, in denen keine IT-gestützten Systeme verwendet werden, müssen Informationen wie Telefaxnummern oder E-Mail-Adressen als Übergabeverfahren hinterlegt werden. Zwar können dabei auch Fehler durch falsche Dateneingabe erfolgen, aber diese Fehler werden durch die generelle Erhöhung der Automation so gering wie möglich gehalten. Hier kann aber auch eine Teil-Verifikation der Gefahrgutdaten Abhilfe schaffen. An einem späteren Transportzeitpunkt vorhandene Daten können mit solchen abgeglichen werden, die sich in einem System am Anfang der Transportkette befinden. Somit kann anonym die Richtigkeit der statischen Gefahrgutdaten erfolgen. Und durch die Verifikation der korrekten Gefahrgutdokumentation wird die Sicherheit für den weiteren Transportprozess erhöht.

2. Klassifizierung der Gefahrgüter für den Transport

Für den Transport von Gefahrgütern muss im Vorfeld eine Klassifizierung bzw. Kennzeichnung vorgenommen werden. Verschiedene Labels an den Gütern und am Transportmittel repräsentieren diese Klassifizierung und zusätzlich wird eine schriftliche, gesetzlich vorgeschriebene Gefahrgutdokumentation benötigt. Diese Dokumentationen und Kennzeichnungen werden meist per Hand durchgeführt. Dabei unterlaufen den Gefahrgutbeauftragten oftmals Fehler. Sei es nur die falsche Reihenfolge von Einträgen oder eine vollkommen falsche Kennzeichnung. Im weiteren Transportprozess würde das zur Ablehnung der Güter für den Weitertransport führen.
Die für jeden Verkehrsträger vorhandenen unterschiedlichen Reglements gewährleisten, dargestellt in Ontologien (siehe auch Abb. 1), eine automatische korrekte Kodierung oder stellen zumindest eine zuverlässige Assistenzfunktion dar. Dabei kann alles, von der richtigen Klassifizierung über die Kennzeichnung bis hin zur korrekten Auswahl der Verpackungseinheiten, abgebildet werden. An mehreren Schnittstellen während des Transportes kann somit helfend in den Transportprozess eingegriffen werden (Abb. 1 – Transportkette).

3. Umkodierung der Gefahrgüter bei Wechsel der Verkehrsträger

Der Wechsel der Gefahrgüter auf andere Verkehrsträger erfordert oftmals eine Umkodierung der Klassifizierung der Güter. Für alle Verkehrsträger gelten andere Kennzeichnungs- und Dokumentationspflichten, die wiederum oft per Hand vorgenommen werden. Zusätzlich besteht eine

[2] OWL-S: Web Ontology Language for Web Services, http://www.w3.org/Submission/OWL-S/

zusätzliche Schwierigkeit wenn die Gefahrgüter die Landesgrenzen überschreiten. Zwar sind die meisten Gefahrgutreglements international gültig, aber es existieren eine Reihe von Ausnahmen,

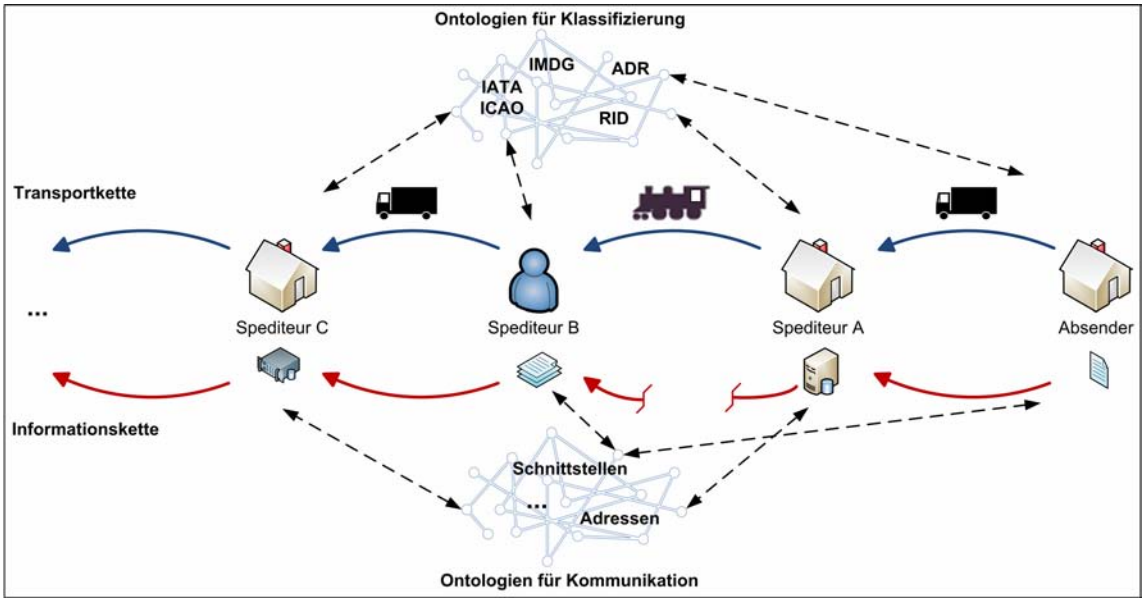

Abbildung 1: Unterstützung durch Ontologien

Sondervorschriften und Ergänzungsschriften. Diese Komplexität wird durch die Verwendung von Ontologien beherrschbar gemacht. Gerade die umfangreichen Vernetzungen und Zusammenhänge zwischen den Gesetzesschriften können durch die Beschreibungslogik der Ontologien abgebildet und effizient angefragt werden. Durch diese Form der Abbildung lassen sich „Todkodierungen“ von Gefahrgütern vermeiden. Damit ist die mehrfache Nacheinander-Umkodierung der Gefahrgutklassifikation gemeint, bei der es dazu kommen kann, dass in besonderen Fällen das Gefahrgut am Ende der Kodierungsreihe „kein Gefahrgut“ mehr ist. Deswegen ist hier darauf zu achten, dass die Klassifizierung der Gefahrgüter dem Assoziativ- und dem Kommutativgesetz entspricht.

Assoziativgesetz: Die Reihenfolge der Ausführung spielt keine Rolle (× = Umkodierung).

- RID × (IMDG × ADR) = (RID × IMDG) × ADR

Kommutativgesetz: Die Reihenfolge der Argumente ändert nichts am Ergebnis

- RID × IMDG = IMDG × RID

Das an dieser Stelle bestehende Problem besteht darin, dass einige Regionen der einzelnen Domain-Ontologien mit den anderen nicht immer in einer 1zu1-Beziehung stehen. Das heißt, dass sich entweder die Bezeichnungen der Konzepte und/oder die Menge der Relationen zwischen den Konzepten unterscheiden. Es kann aber auch sein, dass einige Konzepte der Ontologie A in keiner Form in der Ontologie B vorhanden sind. Um diese Probleme behandelbar zu machen, haben sich drei wesentliche Ansätze entwickelt.

- **Aligning**: Hier werden semantische Beziehungen zwischen den Konzepten der Ausgangsontologien erstellt, sogenannte Alignment-Regeln. Die Ontologie als solches wird beibehalten und ermöglicht durch Verbundrelation das Bezugnehmen auf die andere Ontologie.
- **Merging**: Durch Mischen der Ontologien wird eine neue Ontologie geschaffen, die Begriffe und Relationen der ursprünglichen Ontologien enthält, aber auch neue Begriffe und Bezeichnungen.
- **Verwenden von Upper-Ontologien**: Aufnahme von generalisierten Konzepten in eine Ontologie, die allgemein und abstrakt genug sind, um für eine Vielzahl von Anwendungsgebieten zu gelten.

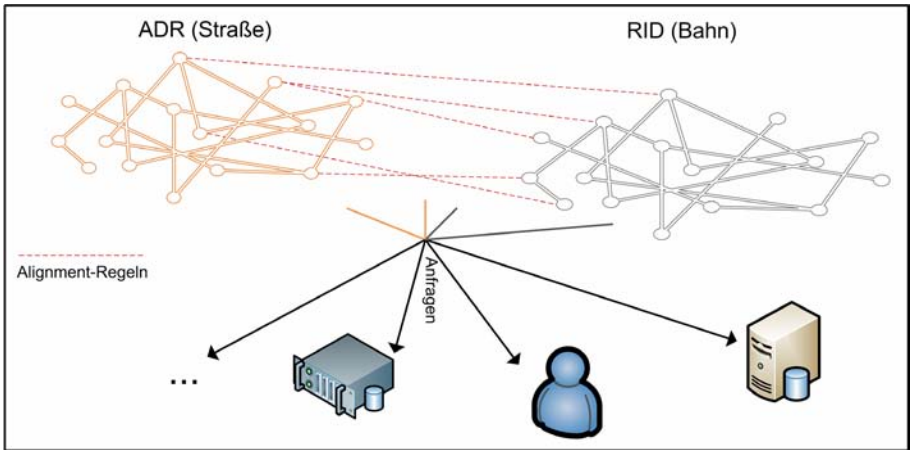

Abbildung 2: Aligning zweier Ontologien der Klassifikation

Das Aligning erhöht die Datenmengen in den jeweiligen Ontologien, kann aber den Abfrageprozess effizient unterstützen und führt somit zu guten Ergebnissen. Allerdings ist hier der Aufwand durch das Vorhandensein mehrerer Ontologien abzuschätzen. Das Merging hingegen führt durch die vorhandene Menge der Reglements zu einer großen Ontologie. Anfrage- und Updateprozesse auf einzelne Reglements innerhalb einer vereinigten Ontologie lassen auf einen erhöhten Aufwand schließen, der mit dem Ergebnis in keiner guten Relation zueinander zu stehen scheint. Eine Upper-Ontologie hingegen verspricht wiederum die Komplexität der Verknüpfungen und Abhängigkeiten der Klassifizierungsontologien auf ein niedrigeres Level herunter zu holen. Alle drei Konzepte müssen allerdings nach Hinzunahme neuer Datenbestände die Konzepte und Relationen auf ihre bisherige Gültigkeit hin überprüfen.

3 Zusammenfassung und Ausblick

Die Ergebnisse der zu realisierenden Ziele innerhalb des InGVer-Projektes dienen für eine lückenlose Unterstützung während des gesamten logistischen Prozesses des Gefahrguttransportes. Auf Grundlage der Nutzung diverser Konzepte und Technologien aus interdisziplinärer Sicht werden unterschiedliche Lösungsansätze entwickelt, die auf ihren Nutzen hin im Gesamtszenario untersucht werden können. Die zu schaffende Unterstützung in den weitläufigen Gefahrgutprozessen soll dabei mehr als eine Art Assistenzunterstützung anzusehen sein, als das Ersetzen vorhandener Systeme. Der Einsatz von Werkzeugen des Wissensmanagement an verschiedenen Stellen im Gefahrguttransportprozess verspricht eine Entlastung aller beteiligten Partner und kann die Sicherheit und Informatisierung der Umwelt innerhalb der gesamten logistischen Kette erhöhen. Allerdings kann noch nichts über die Güte der Einsatzfähigkeit der Werkzeuge gesagt werden. Gerade in der Harmonisierung der Ontologien zur Klassifikation versprechen vorhandene Konzepte noch keine umfassende Behandlungsmöglichkeit der Probleme.

4 Relevante Literatur

Fridman Noy, N. ; Musen, M.: *SMART: Automated Support for Ontology Merging and Aligning*. In: Twelfth Banff Workshop on Knowledge Acquisition, Modeling and Management, 1999

Fridman Noy, N. ; Musen, M.: *PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment*. 2002

Noy, N.F. ; Klein, M.: *Ontology evolution: Not the same as schema evolution*. Knowledge and Information Systems. 2003

Schützel, Andrea: *Ontologiegetriebene Anfrageerweiterung in wissensbasierten Informationssystemen*. Diplomarbeit, 2006, Universität Rostock, Institut für Informatik, LS DBIS

Semantische Fehler in Entity-Relationship-Diagrammen

Martin Herzberg
Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg
martin.herzberg@informatik.uni-halle.de

Zusammenfassung

Fehler im konzeptionellen Datenbankdesign sollten rechtzeitig vor den weiteren Entwicklungsschritten erkannt und behoben werden. Viele aktuelle ER-Entwicklungswerkzeuge wie z.B. Oracle Designer bieten jedoch kaum Möglichkeiten zur Fehlererkennung. Wir entwickeln ein vom später verwendeten Datenbanksystem unabhängiges Werkzeug für den Datenbank-Entwurf ("Database Designer"). Mit diesem Werkzeug sollen dann Möglichkeiten zur Fehlererkennung und Vermeidung getestet werden. Um geeignete Ansätze und Methoden zur Erkennung und Verhinderung solcher Fehler zu finden, wird zunächst eine Liste möglicher Fehler erarbeitet und das Auftreten der ermittelten Fehler in der Praxis an von Studenten erarbeiteten Datenbankdesignvorschlägen in Klausuren und Übungen untersucht. Der vorliegende Beitrag zeigt den aktuellen Stand dieser Untersuchungen.

1 Einleitung

Es wurde als erster Schritt eine Klassifizierung dieser Fehler nach verschiedenen Kriterien wie z.B. Ursachen, betroffenen Objekten und Auswirkung vorgeschlagen. Darauf basierend wird eine umfassende Liste möglicher Fehler sowie deren Auswirkungen erarbeitet. Fehler in ER-Diagrammen kann man zunächst in syntaktische, semantische und stilistische Fehler klassifizieren. Syntaktische Fehler verletzen die Regeln für den Aufbau des Diagramms mit den zur Verfügung stehenden Konstrukten. Da die Regeln für den syntaktischen Aufbau bekannt sind, ist es kein Problem, solche Fehler zu erkennen und zu vermeiden. Bei der Benutzung einer Entwicklungsumgebung wie z.B. Oracle Designer, CA ERWin oder Sybase PowerDesigner werden viele Fehler, die im Entwurf per Hand entstehen können, ausgeschlossen.

Ein semantischer Fehler in einem ER-Modell liegt vor, wenn für ein bestimmtes Problem aus der realen Welt ein syntaktisch korrektes Schema konstruiert wurde, das aber nicht den beabsichtigten Teil der Realität abbildet. Das Modell ist zu allgemein (nicht korrekt) oder zu eingeschränkt (nicht vollständig).

Stilistische Probleme sind eigentlich keine Fehler, aber möglicherweise kann man den betroffenen Sachverhalt einfacher ausdrücken. Zu umständliche Modelle sind auch fehleranfälliger.

Basierend auf dieser Klassifizierung und der Untersuchung von Studenten erarbeiteter ER-Entwürfe wurde folgende Liste von Fehlern erstellt.

2 Syntaxfehler

Syntaxfehler können und sollten eigentlich immer vermieden werden. Manchmal ist es jedoch auch sinnvoll, bestimmte Syntaxfehler als Zwischenzustand während der Entwicklung des Modells zu erlauben. Es werden daher zwei Gruppen von Syntaxfehlern betrachtet:

Unzulässige Syntaxfehler

Diese Fehler können nie einen sinnvollen Zwischenzustand während der Entwicklung des Diagramms darstellen. Beispiele wären fehlende Verbindungslinien, fehlende Namen oder an nicht

zulässigen Stellen platzierte Objekte. Formal ausgedrückt enthält diese Gruppe Fehler, die nicht durch Hinzufügen weiterer korrekter Konstrukte zum Modell korrigiert werden.

Temporär zulässige Syntaxfehler

Diese Syntaxfehler entstehen bei der normalen Vorgehensweise beim Aufbau eines Diagramms und könnten während des Designs - zur Vereinfachung der Arbeit - kurzzeitig zugelassen werden:

- Spezialisierung ohne Subentities für einen Top-Down-Entwurf bzw. Spezialisierung ohne Super-Entity, wenn ein Bottom-Up-Entwurf gewählt wurde.
- Strukturiertes Attribut ohne elementare Attribute (falls es in der verwendeten Notation für ein strukturiertes Attribut ein eigenes Symbol gibt)
- Arc (Exklusivitätsconstraint) enthält weniger als 2 Relationships
- Im *Database Designer* werden Schlüssel als benannte Integritätsbedingungen definiert, die Attribute und Relationships enthalten. Ein solcher Schlüssel darf nicht leer sein.

Das temporäre Zulassen dieser Fehler ermöglicht also beim Entwurf der Datenbank ein konsequentes Beibehalten des Top-Down- bzw. Bottom-Up-Entwurfs. Aufgrund der Existenz verschiedener Notationen mit unterschiedlichen Konstrukten gibt es Fehler, die nur in einigen Notationen möglich sind. Diese Unterschiede wurden detailliert in [9] untersucht.

3 Semantische Fehler

Zu jedem Fehler wird, sofern dieser in der verwendeten Notation möglich war, die absolute Anzahl in den 87 untersuchten Arbeiten und die Anzahl der betroffenen Arbeiten angegeben.

3.1 Strukturelle Integrität

Strukturfehler führen zu einem inkonsistenten Modell und sind unabhängig von der konkreten Anwendung als Fehler erkennbar.

Zyklen und inkonsistente Abhängigkeiten

- Zyklus aus Relationships mit inkonsistenten Kardinalitäten (3/3)
- Zyklus aus Spezialisierungen
- Zyklus aus Weak-Entity-Beziehungen (0/0)

Domains: Inkonsistente Domaindefinition z.B. Minimum > Maximum (für Numeric-Typen)

Entities

- Eindeutige Namen für Entities (1/1)
- Ein Entity muss mindestens ein Attribut enthalten (17/13)
- Mehrere automatisch generierte Attribute
- Schlüssel nicht definiert (19/14)
- Mehrfachvererbung vom gleichen Super-Entity

Ein interessanter Fall sind mit dem restlichen Diagramm nicht verbundene Entities. Chen hat ursprünglich vorgeschlagen, dass jedes Entity mindestens eine Relationship und ein Attribut haben muss [1]. Ein solches Entity könnte jedoch als eine Art “globale Variable” trotzdem sinnvoll sein.

Zusammenfassend kann man folgende Regel für die *strukturelle Korrektheit* eines Entities betrachten: Ein Entity muss mindestens ein Attribut oder ein Superentity oder >2 Relationships haben, um einen Zweck im Schema erfüllen zu können. Das heißt, ein Entity, das ≤2

Relationships und keine Attribute hat und nicht Subtyp in einer Spezialisierung ist, hat sehr wahrscheinlich keine Wirkung im Modell und wäre redundant und damit ein Fehler. Ein Entity, das keine Attribute hat und nicht Subtyp einer Spezialisierung ist, kann als leeres Association-Entity zur Simulation einer Relationship höheren Grades verwendet werden, falls die verwendete Notation (Krähenfuß-Notation) solche Relationships nicht unterstützt.

Attribute von Entities: Strukturiertes Attribut mit nur einem Subelement

Schlüssel

- Nichtminimaler Schlüssel (Identifier inclusion)
- Subattribut in strukturiertem Attribut als Schlüssel definiert
- Optionales Attribut als Schlüssel definiert

Relationships: Rekursive verpflichtende Relationship (11/11) | Rekursive Schlüsselbeziehung

Exklusivitätsconstraints (Arcs)

- Arc umfasst Relationships mit unterschiedlichen Minimalkardinalitäten: entspricht ausschließlich optionalen Relationships. Es wird höchstens eine der Relationships realisiert.
- Arc umfasst eine Relationship, die Teil eines Schlüssels ist. In diesem Fall muss jede andere im Bogen enthaltene Relationship ebenfalls ein (alternativer) Schlüssel sein.

Spezialisierung

- Totale Spezialisierung mit nur einem Subtyp (9/9)
- Überlappende Spezialisierung mit nur einem Subtyp
- Redundante Spezialisierung
- Implizite Spezialisierung (definiert durch Integritätsbedingungen) (9/7)
- Gleiche Relationships und Arc zu allen Subtypen statt Relationship zum Supertyp (3/3)

Weak Entities Diese Fehler gelten analog für Association Entities.

- Kardinalität der Schlüsselbeziehung ist nicht (1,1) (38/20)
- Schlüssel nicht erweitert in Weak Entity (2/1)
- Relationship als Schlüssel für beide Entities definiert (Bijective Dependency) (10/7)

3.2 Semantische Integrität

Semantische Fehler sind Konstrukte, die für eine bestimmte Anwendung korrekt sein können, für eine andere Anwendung hingegen falsch sind, da ihre Semantik nicht dem zu modellierenden Teil der Realität entspricht. Theoretisch könnte man hier für jedes Konstrukt des Entity-Relationship-Modells Anwendungen finden, auf die es gerade nicht zutrifft. Daher beschränkt sich die Liste zunächst auf eine Auswahl der wichtigsten Fälle.

Eingeschränktes Schema

- Attribut verpflichtend statt optional (9/8)
- Kardinalitäten von Relationships zu eingeschränkt (194/60)
- Beziehung mit Attribut statt Association Entity mit partiellem Schlüssel
- Attribut auf 1-Seite einer Relationship (28/19)
- Zu allgemeiner Schlüssel schränkt Menge der möglichen Entities ein (26/20)
- Normale (1,1)-Relationship statt Weak-Entity-Beziehung (9/6)
- Grad einer Relationship zu hoch
- Relationship zum Sub- statt zum Supertyp (10/6)

Zu allgemeines Schema

- Attribut optional statt mandatory (2/2)
- Kardinalitäten von Beziehungen zu frei definiert (82/34)
- Zu spezieller Schlüssel (58/35)
- Relationship zum Super- statt Subtyp (7/6)
- Attribut auf N-Seite einer Relationship (9/7)
- Exklusivitätsconstraint (Arc) fehlt (13/13)

Information nicht darstellbar Ein Konstrukt im Modell fehlt.

- Attribut fehlt (63/38) | Beziehung (32/27) | beliebiges Objekt fehlt z.B. Subtyp (43/14)
- Grad einer Relationship zu niedrig
- Implizite Spezialisierung (45/26)

Redundante Informationen

- Attribut redundant dargestellt: Fremdschlüssel (49/29), vererbte Attribute
- Relationship Shortcut (10/8)
- Redundante Attribute statt Spezialisierung (26/12)
- Redundantes Attribut - Denormalisierung durch implizites Entity (5/5)
- Spezialisierung mittels Indikatorattribut (attribute-defined specialization) (8/8)
- Gleiches Attribut in allen Subtypen (4/3)| in einer Teilmenge der Subtypen (2/2)
- Konstrukt mehrfach modelliert (37/22) | nicht benötigte Konstrukte modelliert (83/44)

Relationship höheren Grades: Attribut an n-ärer Relationship

Ungewöhnliche Kardinalitäten

- Rekursive Relationships: z.B. verpflichtend 1:1, N:1 mit verpflichtender 1-Richtung (8/8)
- 1:1 Relationships: Hinweis auf eine implizite Spezialisierung (78/35)

Ungewöhnliche Anwendung von Exklusivitätsconstraints

- Arc umfasst Relationships mit unterschiedlichen Maximalkardinalitäten
- Arc an rekursiver Relationship

Fehler mit schwachen Entities: Schlüsselbeziehung als verpflichtend 1:1 deklariert (30/18)

Probleme mit Attributen und Schlüsseln

- Subattribut in strukturiertem Attribut als Schlüssel definiert
- Subattribute als Werte

4 Fehler in der Praxis

Als Grundlage für die weitere Vorgehensweise wurde das Auftreten der zuvor beschriebenen Fehler in der Praxis untersucht. Zu diesem Zweck wurden Klausuren und Übungsaufgaben hinsichtlich der darin auftretenden Fehler analysiert. Die Ergebnisse entsprachen in weiten Teilen den Erwartungen. Sie lieferten jedoch auf einzelnen Gebieten, wie z.B. Modellierung ableitbarer und redundanter Informationen, einige unerwartete Erkenntnisse und damit möglicherweise Ansätze für eine verbesserte Unterstützung des Designers bei der Arbeit mit CASE-Tools.

Insgesamt wurden 87 Arbeiten ausgewertet. Die folgende Tabelle zeigt zusammengefasst das Ergebnis dieser Untersuchung. Detaillierte Ergebnisse sowie die Aufgabenstellungen für die untersuchten Modelle findet man unter `http://www.informatik.uni-halle.de/~herzberg/ER`.

Fehlertyp	Anzahl	Arbeiten	Fehlertyp	Anzahl	Arbeiten
Syntax	156	47			
Struktur			**Semantik**		
Zyklen	4	4	eingeschränktes Schema	317	76
Entity	37	18	zu allgemeines Schema	205	60
Relationship	11	11	Information nicht darstellbar	187	57
Arcs	-	-	Redundanz	236	63
Spezialisierung	21	13	Kardinalitäten	86	39
Weak Entity	50	23	Exklusivität	-	-
			schwache Entities	30	18

5 Verwandte Arbeiten und Ausblick

Einige konkrete Fehler wurden bereits detailliert untersucht. Batra, Antony und Santhanam sowie Batra und Zanakis untersuchen in ihren Arbeiten wie z.B. [2], [4], [3] die Ursachen logischer Fehler im Datenbankdesign. Sie analysieren die Vorgehensweise des Designers und die Einflüsse, die zu Fehlern führen können. Dey et al. untersuchen u.a. in [5] Probleme mit Relationships höheren Grades. Lenzerini und Calvanese sowie Hartmann beschäftigen sich u.a. in [6], [7] bzw. [8] mit Inkonsistenzen von Integritätsbedingungen.

Bisher haben wir nur relativ wenige Diagramme in der Krähenfuß-Notation untersucht. Für eine umfassende Analyse der in der Praxis und in der Lehre auftretenden Fehler sollten weitere Diagramme in anderen Notationen und für stärker differenzierte Aufgaben analysiert werden. Es folgt eine prototypische Implementierung der Fehlererkennung im Database Designer. Diese soll als Grundlage für die weitere Untersuchung und möglicherweise Erweiterung der bekannten und in [10] vorgeschlagenen Erkennungsmethoden dienen.

Literatur

[1] Peter Pin-Shan Chen, “The Entity-Relationship Model - Toward a Unified View of Data”, *ACM Transactions on Database Systems, Vol. 1, No. 1*, March 1976, 9-36

[2] Dinesh Batra und Solomon R. Antony, “Novice errors in database design”, *European Journal of Information Systems, Vol.3, No.1*, 1994, 57-69

[3] Dinesh Batra, S.H. Zanakis, “A conceptual database design approach based on rules and heuristics”, *European Journal of Information Systems, Vol.3, No.3*, 1994, 228-239

[4] Dinesh Batra, Solomon R. Antony, “Consulting Support during conceptual database design in the presence of redundancy in requirements specifications: an empirical study”, *Int. Journal Human-Computer Studies No.54*, 2001, 25-51

[5] Debabrata Dey, Veda C. Storey, Terence M. Barron, “Improving Database Design through the Analysis of Relationships”, *ACM Transactions on Database Systems, Vol. 24, No. 4*, December 1999, 453-486

[6] Maurizio Lenzerini, Paolo Nobili, “ON THE SATISFIABILITY OF DEPENDENCY CONSTRAINTS IN ENTITY-RELATIONSHIP SCHEMATA”, *Proceedings of the 13th VLDB Conference*, 1987, 147-154

[7] Diego Calvanese, Maurizio Lenzerini, “On the Interaction Between ISA and Cardinality Constraints”, *Proceedings of the Tenth IEEE International Conference on Data Engineering (ICDE’94)*, 1994, 204-213

[8] Sven Hartmann, “Coping with inconsistent constraint specifications”, *Proceedings: Conceptual Modeling - ER 2001*, November 2001, 241-255

[9] Tobias Reichelt, “Konzeptioneller Datenbank-Entwurf: verlustfreie Transformation zwischen verschiedenen graphischen Notationen”, Diplomarbeit an der Martin-Luther-Universität in Halle, 2007

[10] Martin Herzberg, “Klassifizierung logischer Fehler in Entity-Relationship-Diagrammen”, Diplomarbeit an der Martin-Luther-Universität in Halle, 2007

Liste der Teilnehmer

Sadet Alčić
Institut für Informatik
Datenbanken und Informationssysteme
Heinrich-Heine-Universität Düsseldorf
Universitätsstr. 1
40225 Düsseldorf
Tel.: +49 211 81 10657
Fax.: +49 211 81 13463
alcic@cs.uni-duesseldorf.de

Stefan Audersch
Zentrum für Graphische Datenverarbeitung e.V.
Rostock
Joachim-Jungius-Str. 11
18059 Rostock
Tel.: +49 381 4024168
Fax.: +49 381 446088
audersch@rostock.zgdv.de

Prof. Dr. Wolf-Tilo Balke
Carl-Friedrich-Gauß-Fakultät
Institut für Informationssysteme
Technische Universität Braunschweig
Müpfordtstraße 23
38106 Braunschweig
Tel.: +49 531 391 74 42
Fax.: +49 531 391 32 98
balke@ifis.cs.tu-bs.de

André Bolles
Praktische Informatik
Abteilung Informationssysteme
OFFIS
Escherweg 2
26121 Oldenburg
Tel.: +49 441 9722 220
Fax.: +49 441 9722 202
andre.bolles@uni-oldenburg.de

Andreas Broschart
Databases and Information Systems
Max-Planck-Institut für Informatik
Campus E1.4
66123 Saarbrücken
Tel.: +49 681 9325 512
Fax.: +49 681 9325 599
abrosch@mpi-inf.mpg.de

Prof. Dr. Stefan Conrad
Institut für Informatik
Datenbanken und Informationssysteme
Heinrich-Heine-Universität Düsseldorf
Universitätsstr. 1
40225 Düsseldorf
Tel.: +49 211 81 14088
Fax.: +49 211 81 13463
conrad@computer.org

Frank Eichinger
Institut für Programmstrukturen und Datenorganisation (IPD)
Universität Karlsruhe (TH)
Am Fasanengarten 5
76131 Karlsruhe
Tel.: +49 721 608 7336
Fax.: +49 721 608 7343
eichinger@ipd.uka.de

Dr. Marco Grawunder
Fakultät II
Department für Informatik
Universität Oldenburg
Escherweg 2
26121 Oldenburg
Tel.: +49 441 9722 220
Fax.: +49 441 9722 202
Marco.Grawunder@uni-oldenburg.de

Francis Gropengießer
Fakultät für Informatik und Automatisierung
FG Datenbanken und Informationssysteme
Technische Universität Ilmenau
Postfach 100 565
98684 Ilmenau
Tel.: +49 3677 69 4559
Fax.: +49 3677 69 4541
francis.gropengiesser@tu-ilmenau.de

Michael Hartung
Interdisziplinäres Zentrum für Bioinformatik (IZBI)
Universität Leipzig
Härtelstraße 16 - 18
04107 Leipzig
Tel.: +49 341 97 32240
Fax.: +49 341 97 32209
hartung@izbi.uni-leipzig.de

Martin Herzberg
Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg
Von-Seckendorff-Platz 1
06120 Halle (Saale)
Tel.: +49 345 55 24731
Fax.: +49 345 55 27333
martin.herzberg@informatik.uni-halle.de

Alexander Holupirek
Dept. of Computer & Information Science
Database & Information Systems Group
University of Konstanz
Box D 188
78457 Konstanz
Tel.: +49 7531 88 4449
Fax.: +49 7531 88 3577
alexander.holupirek@uni-konstanz.de

Prof. Dr. Michael Höding
Fachbereich Wirtschaft
Netzbasierte Anwend. für den Handel/E-Business
Fachhochschule Brandenburg
Postfach 2132
14737 Brandenburg
Tel.: +49 3381 355 243
Fax.: +49 3381 355 199
hoeding@fh-brandenburg.de

Dr.-Ing. Hagen Höpfner
School of Information Technology
DB&IS Research Group
International University in Germany
Campus 3
76646 Bruchsal
Tel.: +49 7251 700 239
Fax.: +49 7251 700 250
hoepfner@acm.org

Mayce Ibrahim Ali
Department of Computer Science
AG DBIS
University of Kaiserslautern
P.O. Box 3049
67653 Kaiserslautern
Tel.: +49 631 205 3274
Fax.: +49 631 205 3299
ali@informatik.uni-kl.de

Jonas Jacobi
Fakultät II
Department für Informatik
Universität Oldenburg
Escherweg 2
26121 Oldenburg
Tel.: +49 441 9722 207
Fax.: +49 441 9722 202
jonas.jacobi@uni-oldenburg.de

Friederike Klan
Institut für Informatik
Friedrich-Schiller-Universität Jena
Ernst-Abbe-Platz 2
07743 Jena
Tel.: +49 3641 9 46 413
Fax.: +49 3641 9 46 302
voyager@minet.uni-jena.de

Dagmar Köhn
Institut für Informatik
Datenbank- und Informationssysteme
Universität Rostock
Albert-Einstein-Straße 21
18059 Rostock
Tel.: +49 381 498 7440
Fax.: +49 381 498 7592
dk103@informatik.uni-rostock.de

Prof. Dr. Birgitta König-Ries
Institut für Informatik
Heinz-Nixdorf-Stifungsprofessur für Prak. Inf.
Friedrich-Schiller-Universität Jena
Ernst-Abbe-Platz 2
07743 Jena
Tel.: +49 3641 9 46 430
Fax.: +49 3641 9 46 302
koenig@informatik.uni-jena.de

Sebastian Lehrack
Institut für Informatik
Database and Information Systems
Brandenburg University of Technology Cottbus
Walther-Pauer-Str. 1
03046 Cottbus
Tel.: +49 355 69 3499
Fax.: +49 355 69 3315
slehrack@informatik.tu-cottbus.de

Andreas Lübcke
Department of Computer Science
Institute for Technical and Business IS
Otto-von-Guericke-University Magdeburg
P.O. Box 4120
39016 Magdeburg
Tel.: +49 391 67 12845
Fax.: +49 391 67 12020
andreas.luebcke@ovgu.de

Yi Ou
Department of Computer Science
AG DBIS
University of Kaiserslautern
P.O. Box 3049
67653 Kaiserslautern
Tel.: +49 631 205 3326
Fax.: +49 631 205 3299
ou@cs.uni-kl.de

André Peters
Institut für Informatik
Datenbank- und Informationssysteme
Universität Rostock
Albert-Einstein-Straße 21
18059 Rostock
Tel.: +49 381 498 7442
Fax.: +49 381 498 7592
andre.peters@uni-rostock.de

Jan Pretzel
GECKO mbH Rostock

Deutsche Med Haus 2
18055 Rostock
Tel.: +49 381 454 88 0
Fax.: +49 381 454 88 50
jpr@gecko.de

Prof. Dr.-Ing. habil. Kai-Uwe Sattler
Fakultät für Informatik und Automatisierung
FG Datenbanken und Informationssysteme
Technische Universität Ilmenau
Postfach 100 565
98684 Ilmenau
Tel.: +49 3677 69 4579
Fax.: +49 3677 69 4541
kus@tu-ilmenau.de

Dr.-Ing. Eike Schallehn
Department of Computer Science
Institute for Technical and Business IS
Otto-von-Guericke-University Magdeburg
P.O. Box 4120
39016 Magdeburg
Tel.: +49 391 67 12845
Fax.: +49 391 67 12020
eike@iti.cs.uni-magdeburg.de

Dr.-Ing. Ralf Schenkel
Databases and Information Systems
Max-Planck-Institut für Informatik
Campus E1.4
66123 Saarbrücken
Tel.: +49 681 9325 504
Fax.: +49 681 9325 599
schenkel@mpi-inf.mpg.de

Tim Schlüter
Institut für Informatik
Datenbanken und Informationssysteme
Heinrich-Heine-Universität Düsseldorf
Universitätsstr. 1
40225 Düsseldorf
Tel.: +49 211 81 14087
Fax.: +49 211 81 13463
schlueter@cs.uni-duesseldorf.de

Prof. Dr. Ingo Schmitt
Institut für Informatik
Database and Information Systems
Brandenburg University of Technology Cottbus
Walther-Pauer-Str. 1
03046 Cottbus
Tel.: +49 355 69 2039
Fax.: +49 355 69 3315
schmitt@tu-cottbus.de

Prof. Dr. Bernhard Seeger
Department of Mathematics and Computer Science
Research Group Database Systems
Philipps-University Marburg
Hans-Meerwein-Str.
35032 Marburg
Tel.: +49 6421 28 21526
Fax.: +49 6421 28 21573
seeger@informatik.uni-marburg.de

Matthias Virgin
Institut für Informatik
Datenbank- und Informationssysteme
Universität Rostock
Albert-Einstein-Straße 21
18059 Rostock
Tel.: +49 381 498 7601
Fax.: +49 381 498 7592
mv@informatik.uni-rostock.de

Martina Weicht
IT Science Center Rügen gGmbH
Circus 14
18581 Putbus / Rügen
Tel.: +49 38301 8829 65
Fax.: +49 38301 8829 59
weicht@it-science-center.de

Katrin Zaiß
Institut für Informatik
Datenbanken und Informationssysteme
Heinrich-Heine-Universität Düsseldorf
Universitätsstr. 1
40225 Düsseldorf
Tel.: +49 211 81 10657
Fax.: +49 211 81 13463
zaiss@cs.uni-duesseldorf.de

David Zellhöfer
Institut für Informatik
Database and Information Systems
Brandenburg University of Technology Cottbus
Walther-Pauer-Str. 1
03046 Cottbus
Tel.: +49 355 69 2028
Fax.: +49 355 69 3315
david.zellhoefer@tu-cottbus.de